

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.mo. Mario VENTO

XXXVII cycle - Academic year 2021/2022

CONTENTS

Overview	2
1 Background	3
1.1 Introduction	3
1.2 State-of-the-Art	7
1.2.1 Problem Definition	7
1.2.2 Source of Demonstration	8
1.2.3 Methods	9
1.2.4 Summary	27
2 Research Plan	28
2.1 Research Gaps	28
2.2 Proposed Research Activity	31
3 Other Activities	33
3.1 Attended Courses	33
3.1.1 Compulsary Courses	33
3.1.2 Additional Courses	33
Bibliography	35

OVERVIEW

This 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 3 mandatory chapters, Background 1, Research Plan 2, and Other Activities 3.

The first chapter contains the Introduction (Section 1.1) and the State-of-the-Art (Section 1.2) sections. The former introduces to the topic of *Learning from Demonstration*, that is the main topic of this report, pointing out the importance of the field and the existing knowledge. The latter presents the literature review. This section analyzes from a methodological and application perspective the main approaches by which the problem is solved, highlighting the pros and cons of each method.

The second chapter contains the Research Gaps (Section 2.1) and the Research Plan (Section 2.2) sections. The first presents the gaps that can be extrapolated from the State-of-the-Art, with respect to the context of interest. The second reports the proposed research activity, going on to highlight the connection between the proposal and the gaps presented earlier, the importance of the proposal with respect to the context of interest, and the methodological and experimental procedures that will be followed to track and evaluate progress.

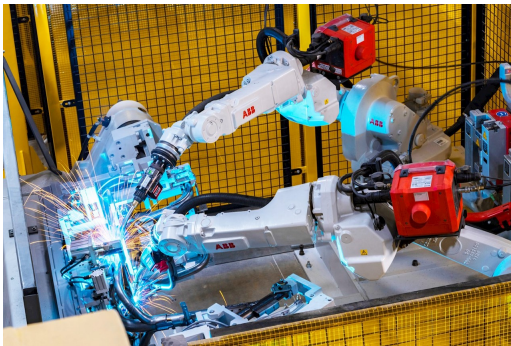
The third chapter shows all the activities carried out during the year, including courses attended at the university, both mandatory and not.

CHAPTER 1

BACKGROUND

1.1 Introduction

Robot 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, robot 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) Adaptability to new conditions**, i.e., the system must be able to easily adapt to dynamic change in system and environmental conditions (e.g. motor failure or a different ground's coefficient of friction for a quadruped robot [3]), performing “*intelligent*” behaviors to handle these new scenarios and solve the desired task; **(b) Adaptability to new tasks**, i.e., the system must be able to easily adapt to both new variations of a known task (e.g., new objects and different initial conditions [4]) and completely new tasks [5] by exploiting past experience to infer actions and solve them; **(c) Reliability**, i.e., the proposed system must be reliable in terms of success rate (tasks must be solved with a success rate that is reasonable for a real-world application), and safety (actions must not harm the environment, people, or the robot itself); **(d) Computation efficiency**, i.e., the system must meet all the constraints above and run in real-time on real-world robot platforms as efficiently as possible.

These requirements, especially **(a)-(b)**, can be difficult to obtain with typical robot 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 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 is 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, and the only used past experience is related to the last

policy rollout. Generally, this approach can lead to data inefficiency problems.

- **Off-policy** Reinforcement Learning (Figure 1.2b), in this setting there are two policies, the behavior policy used to collect data, and the learned target policy. The latter is improved during the training by leveraging past experience contained in the dynamic buffer \mathcal{D} , which is updated after each iteration by adding new samples collected through the behavior policy.
- **Offline** Reinforcement Learning (Figure 1.2c), this setting is strictly related to the classic *supervised learning approach*. The target policy is trained by exploiting a *static dataset* D , collected by an unknown policy π_β . Unlike previous methods, the training procedure requires no further interaction with the environment, and despite strong theoretical results, which can lead to prefer offline RL over behavioral cloning [15], when these algorithms are applied in robot manipulation tasks with sparse-reward and high-dimensional high-level state representation (e.g., robot images), they perform poorly with respect to classic behavioral cloning [16].

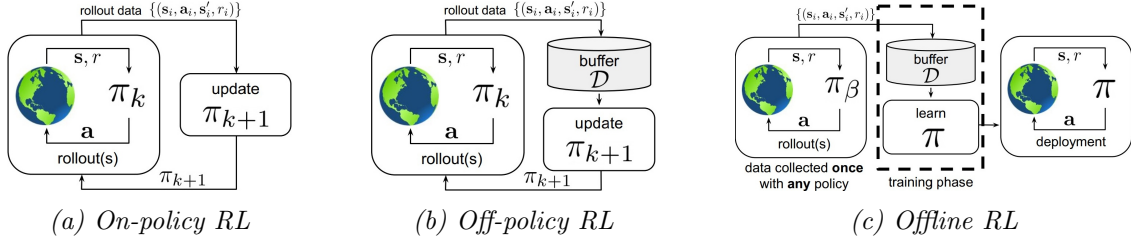


Figure 1.2: Graphical Representation of Reinforcement Learning Methods [14]

The need for both a task-specific reward function and expensive time-consuming training procedures are the main difficulties in applying such methods for real-world application [17].

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 performing a task). 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 Generative Adversarial Learning and Imitation Learning. In this setting the agent, which acts as generator, must produce a sequence of state transitions, such that a discriminator is not able to distinguish between transitions generated by the agent, and the ones generated by the expert and contained in the dataset.
- **Learning from Observation** (LfO), inspired by the fact that humans and animals are able to learn by just watching a demonstrator, without knowing the underlying performed actions (e.g., the joint position), here, the aim of the artificial agent is to reproduce the observed behavior, starting from state-only demonstrations [18].

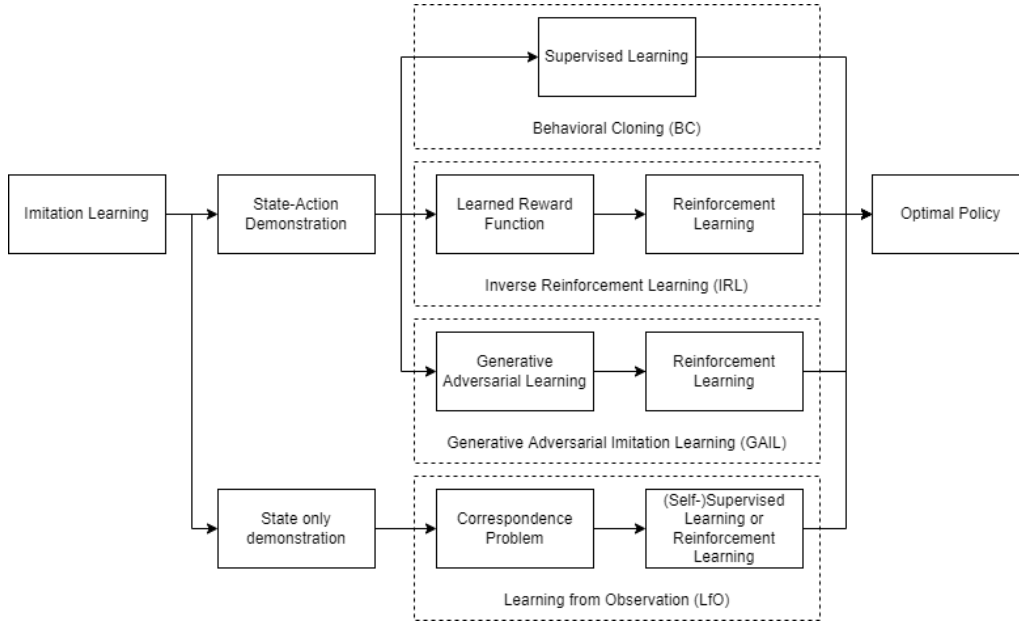


Figure 1.3: Imitation Learning: Taxonomy and main components

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 approaches, such as IRL, GAIL, 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 pure RL setting, since the demonstrations are used to “guide” the exploration; **(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, methods, based on this approach, 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 GAIL based methods improved both the performance and the efficiency 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 (e.g., video of humans performing a task), that are easy 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 demonstrations given in a space different from the robot one into the corresponding action in the robot space, must be solved.

Despite all the presented open-challenges, IL was successfully applied for solving complex tasks, such as driving a toy car [19], pick-and-place [20], and collaborative toolbox assembly [21], 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

The chapter reviews the State-of-the-Art related to the *Learning from Demonstration* problem. It will start with a general problem formulation (Section 1.2.1). Then it will briefly present 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, and π^E defines the expert one. In the broadest possible definition, π^L is obtained starting from a dataset $\mathcal{D} = \{(\tau_i, \mathbf{c}_i)\}_i^N$, where:

- τ_i is the i^{th} demonstrated trajectory, which can be described as:
 - A state-action sequence, i.e., $\tau_i = [s_0, a_0, \dots, s_T, a_T]$, when it is assumed to have access to the ground-truth action performed by the expert.
 - A state sequence, i.e., $\tau_i = [s_0, \dots, s_T]$, when it is assumed to not have access to the ground-truth action.
- \mathbf{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 object.

The policy π^L can be defined with respect to different abstraction levels [22, 10]: **(a) Symbolic Characterization**, the policy maps states, and context to a sequence of options, i.e., $\pi : s_t, \mathbf{c} \rightarrow [o_1, \dots, o_T]$, 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**, the policy maps context to trajectory, i.e., $\pi : \mathbf{c} \rightarrow \tau$. Because it allows the initial state to be mapped 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**, the policy maps states(-context) to actions, i.e., $\pi : s_t, \mathbf{c} \rightarrow a_t$. This representation makes it possible to map the current state directly to the corresponding action. However, it is easy for errors to accumulate in long-term processes.

The agent, by means of its policy π , acts on a system, which is modeled as a *Markov Decision Process* (MDP) [23]. A MDP is defined as a tuple (S, A, R, T, γ) , where:

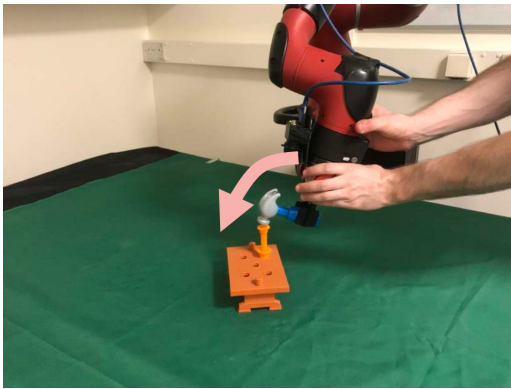
- $S \subseteq \mathbb{R}^n$, is a set of states (e.g. joints position and/or image).
- $A \subseteq \mathbb{R}^n$, is a set of actions, (e.g. desired end-effector pose, desired joints torque).
- $R(s, a, s')$, is a *reward function*, which expresses the immediate reward for executing action a in state 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*, can be given a priori or learned (Model-Based methods), or do not taken into account (Model-Free methods).

- $\gamma \in [0, 1]$, is a discount factor expressing the agent’s preference for immediate over future rewards.

With respect to the given MDP definition, the reward function plays different roles based on the used approach. Indeed, in BC methods, the reward function is not implicitly used, but a surrogate loss-function is used instead. In IRL the reward function is learned, assuming that the expert acts (near-)optimal w.r.t. some unknown reward function. 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

Regarding the way how the expert demonstrations can be obtained, according to [22], two categories can be identified: **(a)** direct demonstration; **(b)** indirect demonstration.



(a) Example of kinesthetic teaching [24]



(b) Example of teleoperation [20]

Figure 1.4: Examples of Direct Demonstration

Direct Demonstration

It allows samples to be directly obtained from the robot, either with kinesthetic teaching (Figure 1.4a) or teleoperation teaching (Figure 1.4b). 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, 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* [25]. It is a cloud-based teleoperation framework that enables the collection of high-scale demonstration dataset [26, 16], for both simulated and real-world robots, using a mobile-phone as controller (Figure 1.5). While this framework is interesting since it allows to collect data from a wide range of demonstrator with different demonstrations quality, it lacks of tactile information. When data are collected employing teleoperation, a way to collect contact information between the robot and the environment is represented by haptic interfaces [27, 28], which would allow the human operator to receive a force-feedback during teleoperation. However, the current literature has not yet focused on exploiting

haptic information in the context of Learning from Demonstration, mainly due to the presence of open-challenges related to exploiting and best representing robot motion-related information, as will be explained in the following sections.

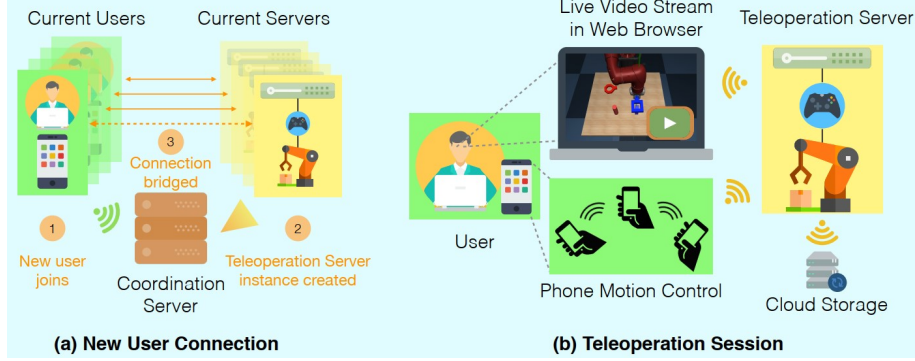


Figure 1.5: System diagram of Roboturk [25]

Indirect Demonstration

It allows samples to be acquired in a way that is completely disconnected from the robot platform. In this case the human motion is recorded by vision system [29, 30] or wearable device [31]. 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 [22, 18]. While Indirect Demonstration based on visual system allows to collect samples in the most intuitive and scalable way possible (potentially any video of performed task can be used), it lacks of tactile information, which can be obtained by using wearable devices such as gloves endowed with sensors able to measure the contact forces [32]. However, 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 ways this problem has been solved in the context of visual demonstration will be explained in detail.

1.2.3 Methods

In this section the different methods used to solve the Learning from Demonstration problem are presented. The paragraphs are organized according to the temporal order in which the different approaches were proposed for the context of interest. So the first paragraph will be devoted to methods related to Behavioral Cloning. A brief presentation will be made on methods related to Inverse Reinforcement Learning. In conclusion, recent methods related to Generative Adversarial Imitation Learning and Learning from Observation will be described and analyzed.

Behavioral Cloning (BC).

Behavioral Cloning is one of the first approach used to learn a task from a set of demonstrations. Generally, it is framed as a Supervised Learning problem. Algorithm 1 defines the classic procedure used to solve the learning task, where, given the dataset \mathcal{D}^E , a parameterized learner policy π_θ^L , and a loss-function \mathcal{L} , the goal is to find the policy's parameter that minimizes the loss-function, or in other terms, $\theta^* = \underset{\theta}{\operatorname{argmin}} \mathbb{E}_{(\tau, \mathbf{c}) \sim \mathcal{D}^E} [\mathcal{L}((\tau, \mathbf{c}), \pi_\theta^L)]$.

In this setting there are at least two questions to answer: **(1)** choose whether to optimize in trajectory space or action space; **(2)** what kind of representation to use for the policy. According to [10], BC methods can be categorized as Model-Free and Model-Based. Moreover, a further classification can be 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 π_θ^L

Ensure: The optimal set of policy parameter θ^*

Optimize \mathcal{L} w.r.t. policy parameter θ using \mathcal{D}^E

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

Model-Free methods that derive policy in the trajectory space were very popular in the context of Model-Free BC for trajectory planning, given their ability to explicitly model constraints (e.g. a smooth convergence to the goal state). In this setting well studied methods are the *Dynamic Movement Primitives* (DMPs) [33, 34], which can represent non-linear movements without losing the stability of the behavior. Authors in [34] formulated the Dynamic Movement Primitive for a single DoF point-to-point trajectory using the following set of non-linear differential equations:

$$\tau \dot{y} = \beta_s(\alpha_s(g - s) - y) + f(z) \quad (1.1)$$

$$\tau \dot{s} = y \quad (1.2)$$

$$\tau \dot{z} = -\alpha_z z, \quad z(t) = z_0 \exp\left(-\frac{\alpha_z}{\tau} t\right) \quad (1.3)$$

Where $\beta_s, \alpha_s, \alpha_z$ are constants, s is the system state, z is the phase-variable function of time t , and f is the forcing-term which describes the trajectory non-linear behavior. Generally, f is a linear combination of basis-function $\psi_i(z)$ (e.g., Gaussian basis function), $f(z(t)) = (g - s_0) \sum_{i=1}^M \psi_i(z(t)) \omega_i$. Basically, a DMP describes a point-attractor system, where the current system state s must converge to the goal state g , starting from s_0 . In this setting, the aim is to learn the set of weights, $\{\omega_i, i = 1, \dots, M\}$, which can be obtained by solving a supervised learning problem with loss function $\mathcal{L}_{DMP} = \sum_{t=0}^T (f_{target}(t) - f(z(t)))^2$, where $f_{target}(t) = \tau^2 \ddot{s}^E - \beta_s(\alpha_s(g - s^E) - \tau \dot{s}^E)$. DMPs formulation has been used in the context of robotic manipulation. For example in [35, 36, 37] the problem of task decomposition was faced, and DMPs have been used to model the sub-tasks. However, DMPs formulation has a series of problems related to: **(a)** how to handle stochasticity in demonstrations; **(b)** how to explicitly define basis function, given the desired behavior; **(c)** how to handle arbitrary desired trajectories, with intermediate via-points; **(d)** how to handle complex high-dimensional inputs. For all the above mentioned problems some solutions have been proposed. For example, the stochasticity problem was addressed in [38], where the Probabilistic Movement Primitives (ProMPs) method was proposed. In ProMPs the probability of observing a trajectory τ is written as $\tau = \prod_t \mathcal{N}(s(t) | \Psi(z(t))^T \omega, \Sigma_s)$, where Ψ is a time-dependent basis matrix. As in DPMs, the goal is to find the weights ω , by solving a supervised learning problem. As for the second problem, other trajectory representations have been proposed based on Hidden

Markov Model, Gaussian Mixture Models, Kernelized Movement Primitives. However, all these alternatives scale bad when the goal is to learn a policy starting from visual observations. To handle such complex high-dimensional input Deep Architecture have been proposed, which will be explained in detail in the next sections.

Regarding the methods that derive the policy in the action-space. One of the primal work was [39], which proposed *ALVINN*, an autonomous vehicle driving system based on a Neural Network, that infers the steering angle, given a synthetic camera image as input. The network was trained on pairs (image, steering-angle) and the training procedure was defined as a supervised classification problem, since the steering-angle was discretized over 45 units. 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} , which represents the next sample, violating the i.i.d assumption of Supervised Learning and generating a test-data distribution, that may be different from 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 $O(\epsilon T)$ [40, 41]. To attenuate this problem, interactive supervised learning algorithms have been proposed, such as the well-known *DAGger* [41]. Algorithm 2 describes the DAGger procedure. It is an aggregation strategy, based on the idea to train the policy π^L 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 [42].

Algorithm 2 DAGger Algorithm [41]

Require: Initial Dataset $\mathcal{D} \leftarrow \emptyset$, Initial policy π_1^L

Ensure: The best policy π_i^L

for $i = 1, \dots N$ **do**

 Sample $T - step$ trajectories using π_i^L

 Let $\mathcal{D}_i = (s_t, \pi^E(s_t))$, state s_t visited by policy π_i^L , and actions given by the expert

 Aggregate Dataset, $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$

 Train policy π_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) [43] 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 demonstrations and expert adjustments. In the context of Interactive Learning for Robot Manipulation, other works of interest include [44, 45]. In [45], 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 if the trajectory is not satisfactory, α if the trajectory is adjusted by

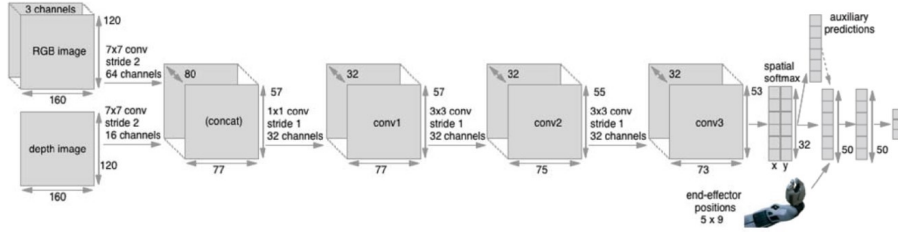


Figure 1.6: Architecture proposed in [20]

Table 1.1: Statistics of Training set, and Test Success rate [20]

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

the expert, where α is the ratio 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 was possible to have an agent capable of performing tasks such as picking up a cube or pulling a plug.

Despite the covariate-shift problem, [20] showed that very interesting performance can be obtained in the context of Robot Manipulation, by means of Behavioral Cloning and high quality demonstrations given by teleportation system. In this work, a Convolutional Neural Network was trained to predict the desired linear-velocity, angular-velocity of the end-effector, and 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 (Figure 1.6). The system was tested on 10 tasks, and the performance are reported in Table 1.1. The proposed system achieved a high success rate while evaluating all the tasks. The tests were carried out from different initial conditions but still quite similar to those present in the training set (e.g., the initial object positions have been uniformly distributed within the training regime, with random local variations around these positions). The analysis of failure cases showed that the leading cause of errors was the inability to detect critical points in the task execution, such as closing/opening the gripper to pick/place the object or detect the position of the object of interest in order to avoid collision with it. Another 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 task, using only a small number of training samples [46]. Algorithm 3 describes the steps followed by the *Model-Agnostic Meta-Learning* (MAML) algorithm [46], that is the base for different methods which apply One-shot Imitation Learning in the context of Behavioral Cloning [47, 48, 49].

In [47], MAML algorithm was used to prove the effectiveness of Meta-Learning in the context of real robot manipulation, with visual observations, as opposite to [50]. A Convolutional Neural Network was trained by following the Algorithm 3, using as loss-function the Mean Squared Error, computed between the predicted action and the ground truth

Algorithm 3 Model-Agnostic Meta-Learning (MAML) [46]

Require: Distribution over tasks $p(\mathcal{T})$
 Randomly initialize θ
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

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 [50], and contextual network (i.e., a Convolutional Neural Network that takes in input the current observation and the image representing the target state). In [48], 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 $\mathcal{D}_{\mathcal{T}}^h$ and a set of robot demo $\mathcal{D}_{\mathcal{T}}^r$ (Figure 1.7), the *initial policy parameters* θ and the *adaptive loss parameters* ψ are learned, solving the problem in Formula 1.4.

$$\min_{\theta, \psi} \sum_{\mathcal{T} \sim p(\mathcal{T})} \sum_{\mathbf{d}^h \sim \mathcal{D}_{\mathcal{T}}^h} \sum_{\mathbf{d}^r \sim \mathcal{D}_{\mathcal{T}}^r} \mathcal{L}_{BC}(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\psi}(\theta, \mathbf{d}^h), \mathbf{d}^r) \quad (1.4)$$

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}_{ψ} , 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 demonstrating a new task \mathcal{T} , i.e., $\phi_{\mathcal{T}} = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\psi}(\theta, \mathbf{d}^h)$. 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.

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 [47] or different, but related tasks [48]. Very recent methods try to generalize BC to a huge variety of tasks [5, 4]. In [5], a large-scale dataset containing **100** diverse manipulation tasks was collected. The demonstrations were collected through both expert teleoperation and shared autonomy process (HG-Dagger [43]). 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 1.8. As it can be noted the samples were composed by the current robot observation, and a conditioning represented by either a natural language description or a human video demonstration. The idea was that training a conditioned policy over the current ob-



Figure 1.7: Tasks performed in [48]. (Top row) Human demonstration, (Bottom row) robot demonstration. (Left) Placing task, (Middle) pushing task, (Right) pick-and-place task.

servation o_t , and a task representation c , $\pi^L(a_t|o_t, c)$, it would allow the policy to generalize over new tasks in a zero-shot manner (i.e., without any fine-tuning). Experimental results shown that, over 28 held-out tasks, containing both completely new objects, 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 natural language instruction. The success rate dropped to **4%** in the hardest setting with 4 distractors and video conditioning. In their analysis, the authors pointed out that the system did not fail to identify the object, but in what they called “**last-centimeter errors**”, i.e., failing to close the gripper, failing to let go of objects, or a near miss of the target object when letting go of an object in the gripper. Authors in [4] did a further step towards Multi-task Imitation Learning. They proposed Multi-task One-Shot imitation with self-Attention and Contrastive Learning (MOSAIC), an architecture designed to model a demonstration-conditioned policy, $\pi_\theta^L(a_t|o_t, d)$ where d is the current task demonstration (e.g., a video of a robot performing the task), and solve the multi-task imitation learning problem by incorporating two key components: **(1)** a time-contrastive loss as additional supervision for representation learning, with the aim to obtain similar embeddings for temporally close-by frames; **(2)** a self-attention model architecture with the aim to extract contextual information from demonstrations, used by the Neural Network to infer the action. A dataset with 7 different tasks and 61 variations was collected by running hand-written policies in a simulation environment. It was used to train and compare the proposed architecture against other one-shot imitation learning methods [48, 51]. The results in Table 1.2 show that MOSAIC outperformed previous methods in the Single-Task One-shot Imitation Learning Setting, i.e., one-task with multiple variations and tested on unseen variations. Moreover, when tested in a Multi-Task setting, i.e., training the system on all the tasks and testing it on each task separately, the system partially replicates the demonstrated task. In general, transitioning from a single-task to a multi-task setting is not trivial. Indeed, the success rate decreases for almost all

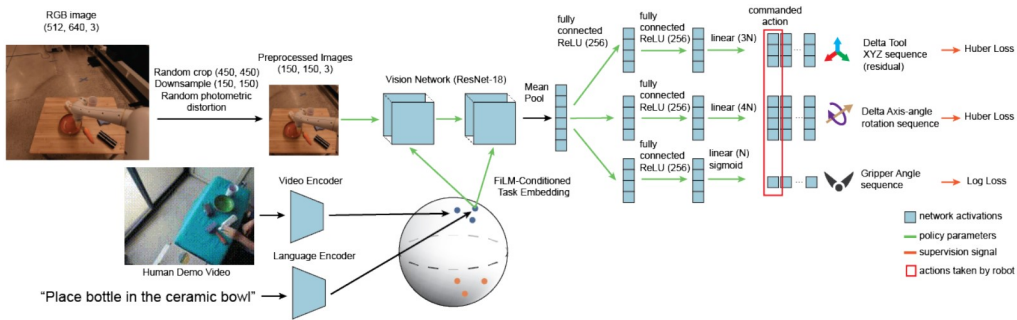


Figure 1.8: Architecture proposed in [5]

Table 1.2: Results obtained in single-task and multi-task one-shot imitation learning [4].

Task	Setup	DAML [48]	T-OSIL [51]	MOSAIC [4]
Open door	single	23.3 ± 5.2	57.9 ± 7.1	67.1 ± 5.5
	multi	10.8 ± 5.4	49.2 ± 6.0	68.3 ± 6.3
Open drawer	single	15.4 ± 5.5	57.5 ± 3.9	65.4 ± 3.4
	multi	3.3 ± 1.4	53.3 ± 4.0	55.8 ± 3.6
Press button	single	62.8 ± 3.9	56.4 ± 2.4	71.7 ± 3.9
	multi	1.7 ± 0.7	63.3 ± 3.5	69.4 ± 3.4
Pick-and-Place	single	0.0 ± 0.0	74.4 ± 2.1	88.5 ± 1.1
	multi	0.0 ± 0.0	19.5 ± 0.4	42.1 ± 2.3
Stack block	single	10.0 ± 1.8	13.3 ± 2.6	79.3 ± 1.8
	multi	0.0 ± 0.0	34.4 ± 3.4	70.6 ± 2.4
Basketball	single	0.4 ± 0.3	12.5 ± 1.6	67.5 ± 2.7
	multi	0.0 ± 0.0	6.9 ± 1.3	49.7 ± 2.2
Nut assembly	single	2.2 ± 1.4	6.3 ± 1.9	55.2 ± 2.8
	multi	0.0 ± 0.0	6.3 ± 1.3	30.7 ± 2.5

the tasks, even though the tasks are known (i.e., used during the training). It indicates the need to have both training procedures and architectures capable of extrapolating embeddings that are representative of both the task itself and the different sub-tasks, in order to reuse them when executing a new instance or a completely new task. The results reported by the different methods pointed out how there is a wide room for improvement in the context of both Single-Task and Multi-Task Few-shot Imitation Learning methods, which are particularly promising when it comes to task-level generalization, because the intent of the new task is correctly captured, especially in the case of conditioned policies. However, the success rate in task execution is relatively low for several reasons that can range from difficulty in identifying key points in task execution (e.g., the robot has approached the object and, therefore, the gripper must be closed) to compounding error, which leads to new observations different from those in the training set (e.g., the robot did not grasp the object correctly). As a result, the robot cannot complete the task because it performs actions inconsistent with the state of the task itself. Possible improvements can range from using a multi-modal system based on visual and tactile information in order to make explicit the transition between consecutive task phases, to adding bad demonstrations combined with recovery behavior during data collection.

The methods presented up to now partially solve the problems related to compounding error and task-generalization. Another relevant problem related to Behavioral Cloning, and more in general to Learning from Demonstration, is the execution of **multi-stage long-horizon tasks**. Indeed, it is pretty intuitive to think that a manipulation task is characterized by a modular structure which can be exploited to improve performance, following the idea that each sub-task can be learned more efficiently because each skill is shorter-horizon. In this setting, the main question to answer is: *How can a task be decomposed into the corresponding component skills?* In its classical formulation, this problem can be framed as a classification problem. In a preliminary work [35], the authors used a set of predefined primitive motions modeled as DMPs. The goal was to recognize these primitives within the demonstrated trajectory using an Expectation-Maximization algorithm. Also, recent methods [36, 37] exploit the DMPs formulation. However, they are not part of the process of task segmentation. For example, in [36], the action segmentation is performed based on either object proximity or explicit human command. When the robot manager identifies a new action, the corresponding DMP is learned, and the sequence of

tasks is organized in a hierarchical structure. The problem of task-segmentation was also solved starting from an unconstrained video demonstrations labeled with commands [52], activity [53], or completely unlabeled [49, 54]. In [49], the authors proposed an extension of DAML for multi-stage tasks. The main idea of this work was based on modeling the distribution of tasks, $p(\mathcal{T})$ as a set of primitives (e.g., reach primitive). During the meta-training phase, two systems were obtained: **(1)** a human/robot-phase predictor, whose responsibility was to indicate the completion rate of the current primitive given a human video demonstration and a robot video demonstration, respectively; **(2)** a Convolutional Neural Network trained according to DAML, which acted as a motion-primitive meta-model policy, parameterized by meta-parameters θ . During the meta-test phase, given as input a video of human performing a compound task, three steps were defined: **(1)** extrapolate the set of primitives by using the human-phase predictors; **(2)** for each primitive, starting from the meta-parameters θ , find the parameters ϕ_i by performing gradient descent on the learned loss function \mathcal{L}_ψ ; **(3)** execute the policy parameterized by ϕ_i , until the current sub-task is completed (completion detected by the robot-phase predictor). The work proposed in [54] addressed the multi-stage long-horizon tasks with a different approach. The authors did not focus on an explicit concept of task segmentation. Indeed, they started with the idea that similar tasks intersect at common regions of the state-space, and proposed a two-stage IL architecture able to exploit this intersecting structure to generalize to unseen start-goal state combinations.

These methods are interesting because they allow the execution of a compound task from simple unlabeled videos, thus with as little effort as possible in dataset generation. However, since they do not have semantic information about which part of the task is being executed, they may err on the side of interpretability of actions, which is not the case with methods based on labeled videos or hierarchical structures.

Model-Based methods are characterized by the fact that during the learning process the system dynamic model is learned. There are several reasons why it may be necessary to use the dynamic system model in the context of Learning from Demonstration, for example: **(a)** a difference in the embodiment between the demonstrator and the learner; **(b)** a difference in the task execution conditions and parameters. Among these methods, the most important aspect is related to how the dynamic model is retrieved, e.g., Gaussian Mixture Model [55] or Gaussian Process [56, 57]. In recent years, a significant amount of attention has been placed on systems that can replicate a task from a video of human performing a task without any access to ground-truth action. While in the survey mentioned before, these methods were still considered Model-Based Behavioral Cloning. In the current literature, given the increasing amount of works, they can now belong to a new category called Learning from Observation. Methods belonging to this category will be explained in the specific paragraph.

Inverse Reinforcement Learning (IRL)

Inverse Reinforcement Learning or Inverse Optimal Control is a Learning from Demonstration approach that was introduced in [58]. 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 parametrize the reward-function, i.e., a **linear** function of the features [59, 60] or a **non-linear** function of the features [61, 62, 63]. Generally, the IRL algorithms are characterized by the following problems: **(1)** the learning process could be time-consuming and impractical for high-di-

Algorithm 4 Classic feature matching IRL algorithm

Require: Dataset of expert trajectories $\mathcal{D} = \{\tau_i^E\}_{i=1}^N$

Require: Reward function R_ω , policy π_θ^L

while policy improves **do**

 Evaluate the state-action visitation frequency μ of the current policy π_θ^L

 Evaluate the objective function \mathcal{L} , w.r.t. μ and the dataset \mathcal{D}

 Update the reward-function parameters ω based on \mathcal{L}

 Update the policy π_θ^L through an RL algorithm, using the updated R_ω

end while

mensional 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.

Some constraints to the optimization procedure have to be added to solve the problem of multiple solutions. Based on the constraint type, the two main IRL procedures can be identified, which are: **(a)** The *Maximum Margin Prediction* (MMP) approach [59, 61]; **(b)** *Maximum Entropy* (Max-Ent) approach [60, 62, 63]. The MMP methods assume that the demonstrated trajectories are optimal and work in a deterministic setting. The aim is to find the cost-function such that the reward of the demonstrated trajectories, $R(\tau^E)$, is greater than the reward of all the alternative trajectories, $R_\omega(\tau)$, by a certain margin m , i.e., $\max_{\omega, m} m \text{ s.t. } R_\omega(\tau^E) \geq \max(R_\omega(\tau)) + m$. 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 [60]. 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 $\max_{\psi} \mathcal{H}(\pi^{r_\psi}) \text{ s.t. } \mathbb{E}_{\pi^{r_\psi}}[\mathbf{f}] = \mathbb{E}_{\pi^*}[\mathbf{f}]$. The setting based on the

Maximum Entropy is the most popular 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., $r_\psi = \psi^T \mathbf{f}$. However, this reward formulation is not suited for high-dimensional feature space, which can require the capability to model non-linear reward structures. In [62], a deep-network was used to model the reward-function. In this work, the Neural Network maps the feature vector \mathbf{f} into the reward value, and trained according to the Maximum-Entropy setting. Experimental results have shown that the ability to approximate highly non-linear reward functions is essentially to successfully solve tasks in high-dimensional discrete state space. A generalization to continuous state space was proposed in [63]. In particular, [63] 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 dy-

namic is unknown the idea was to estimate the partition function through the trajectories obtained from the current policy rollouts, with the hypothesis that, during the learning of the cost function, the current policy drives the distribution towards regions where samples are more useful. Starting from these considerations, the Algorithm 5 was proposed. The algorithm [64] returns a *Linear-Quadratic Gaussian* controller q_k , such that

Algorithm 5 Guided-Cost-Learning Algorithm [63]

Require: Initial controller $q_k(\tau)$
for $i = 1, \dots, N$ **do**
 Generate D_{traj} from $q_k(\tau)$
 $\mathcal{D}_{samp} \leftarrow \mathcal{D}_{samp} \cup \mathcal{D}_{traj}$
 Update the cost-function parameters, using \mathcal{D}_{samp} and $\nabla_{\theta} \mathcal{L}(\theta)$
 Update the controller $q_{k+1}(\tau) \leftarrow q_k(\tau)$ according to [64] and using \mathcal{D}_{traj}
end for

$q_k = \underset{q}{\operatorname{argmin}} \mathbb{E}[c_{\theta}(\tau)] - \mathcal{H}(\tau) \text{ s.t. } p(s_{t+1}|s_t, a_t) = \mathcal{N}(s_{t+1}; f_{x_t}x_t + f_{u_t}u_t, F_t)$. 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 and the need to have a non-linear representation of the cost function for complex problems, outperforming the classic Max-Ent method. Recent method [65] tried to move IRL towards more complex learning scenario, such as learning a reward function from video-demonstrations. In [65] the architecture in Figure 1.9 was proposed. The goal was to obtain the parameters Ψ , of a cost function $C_{\Psi}(\hat{\tau}, z_{goal})$, such that it was possible to derive a sequence of actions by minimizing the cost-function itself, i.e., $\mathbf{a}_{new} = \mathbf{a} - \eta \nabla_a C_{\Psi}(\hat{\tau}, z_{goal})$. 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 $\hat{\tau}$ is a sequence of predicted states $[\hat{s}_1, \dots, \hat{s}_T]$, 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.

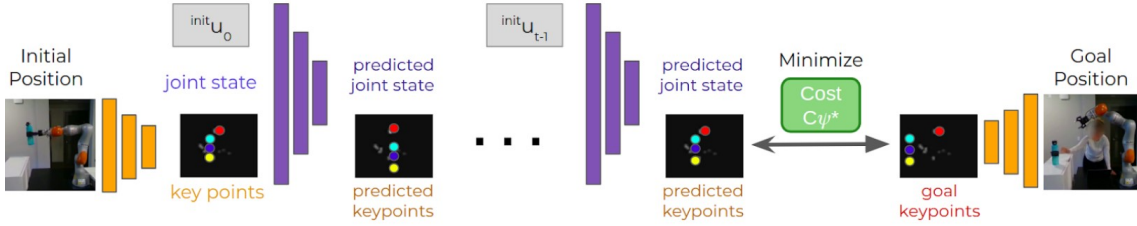


Figure 1.9: Architecture proposed in [65]

Generative Adversarial Imitation Learning (GAIL)

Generative Adversarial Imitation Learning was proposed for the first time in [66], with the idea to improve the IRL setting, which is expensive to run, because of the double-nested optimization procedure. The authors in [66], starting from a general Max-Ent formulation (Formula 1.5), obtained a characterization of the learned policy (Formula 1.6), 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 1.6 is that the ψ -regularized IRL finds a policy whose occupancy measure is similar to the expert's one, measured by

Algorithm 6 Generative Adversarial Imitation Learning Algorithm

Require: Expert Trajectories $\tau^E \sim \pi^E$, initial policy π_θ^L , discriminator D_ω
for $i = 1, \dots, N$ **do**
 Sample trajectories, $\tau_i^L \sim \pi_\theta^L$
 Update Discriminator, $\hat{\mathbb{E}}_{\tau_i^L} [\nabla_\omega \log(D_\omega(s, a))] + \hat{\mathbb{E}}_{\tau^E} [\nabla_\omega \log(1 - D_\omega(s, a))]$
 Update Policy π_θ , with TRPO [70], and cost-function $C(s, a) = \log(D_\omega(s, a))$
end for

ψ^* . The next-step was to choose an appropriate regularization function. In particular, by choosing the regularizer in Formula 1.7, the conjugate in Formula 1.8 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.

$$IRL_\psi(\pi^E) = \arg \max_{c \in R^{S \times A}} -\psi(c) + (\min_{\pi^L \in \Pi} -\mathcal{H}(\pi^L) + \mathbb{E}_{\pi^L} [c(s, a)]) - \mathbb{E}_{\pi^E} [c(s, a)] \quad (1.5)$$

$$RL \circ IRL_\psi(\pi^E) = \arg \min_{\pi^L \in \Pi} -\mathcal{H}(\pi^L) + \psi^*(\rho_{\pi^L} - \rho_{\pi^E}) \quad (1.6)$$

$$\psi_{GA}(c) = \begin{cases} \mathbb{E}_{\pi^E} [g(c(s, a))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases}, \quad g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases} \quad (1.7)$$

$$\psi_{GA}^*(\rho_{\pi^L} - \rho_{\pi^E}) = \max_{D \in (0,1)^{S \times A}} \mathbb{E}_{\pi^L} [\log(D(s, a))] + \mathbb{E}_{\pi^E} [\log(1 - D(s, a))] \quad (1.8)$$

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 [60], 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 [67], or by modifying the reward function in input to the RL algorithm [68, 69]. All the cited methods have reported promising results on control tasks in a simulation environment [71], 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 [72, 73, 74, 75], it shown very poor results. In particular, with respect to the Adversarial Imitation Learning setting, works of interest are [74, 75]. In [74], 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 [74] 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 . These 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 (e.g., a different gripper color); **(2)** an early-stopping policy called 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

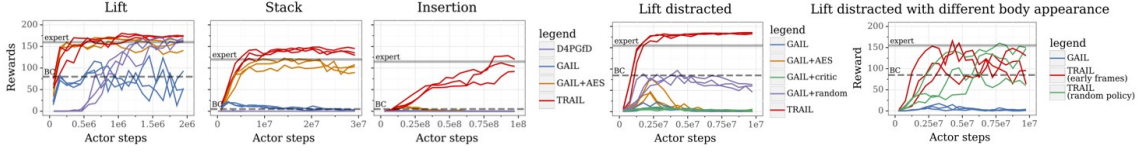


Figure 1.10: Experimental results on tasks without and with spurious features [74]

defined as $accuracy(\mathcal{I}_E, \mathcal{I}_A) = \frac{1}{2} \mathbb{E}_{s \in \mathcal{I}_E} [\mathbf{1}_{D_\omega \geq \frac{1}{2}}] + \frac{1}{2} \mathbb{E}_{s \in \mathcal{I}_A} [\mathbf{1}_{D_\omega < \frac{1}{2}}]$. In the end, the discriminator's parameters were the result of $\max_{\omega} G_\omega(s_E, s_A) - \mathbf{1}_{accuracy(\hat{s}_E, \hat{s}_A) \geq \frac{1}{2}} G_\omega(\hat{s}_E, \hat{s}_A)$, where G_ω is the discriminator binary cross-entropy, $\hat{s}_E \in \mathcal{I}_E$, and $\hat{s}_A \in \mathcal{I}_A$. An agent was trained on each single task, according to the Distributed Distributional Deterministic Policy Gradients (D4PG) [76] RL algorithm, with reward-function $R(s_t) = -\log(1 - D_\omega(s_t))$. Experimental results have shown how the proposed system overcomes the GAIL [66] baseline, both in setting with spurious features and without spurious features (Figure 1.10).

The authors of [75] 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 smaller dimension. Then subsequently go on to learn a dynamic model in that space. In this context the learning procedure is based on three main steps: **(1)** learn the *Latent Dynamic Model*, $(\hat{\mathcal{U}}_\beta, \hat{\mathcal{T}}_\beta, q_{\beta\epsilon})$, by maximizing the Evidence Lower Bound (Formula 1.9), where $\hat{\mathcal{U}}_\beta$ is the decoder, $q_{\beta\epsilon}$ 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 1.10); **(3)** train a *policy* π_θ^L , by maximizing the Value function (Formula 1.11).

$$\max_{\beta} \mathbb{E}_{q_{\beta}} \left[\sum_t \log(\hat{\mathcal{U}}_\beta(s_t|z_t)) + \mathbb{D}_{KL}(q_{\beta}(z_t|s_t, z_{t-1}, a_{t-1}) || \hat{\mathcal{T}}_\beta(z_t|z_{t-1}, a_{t-1})) \right] \quad (1.9)$$

$$\min_{\theta} \mathbb{E}_{(z,a) \sim \rho^E(z,a)} [-\log(D_\theta(z,a))] + \mathbb{E}_{(z,a) \sim \rho_{\hat{\mathcal{T}}}^L} [-\log(1 - D_\theta(z,a))] \quad (1.10)$$

$$\max_{\pi_\theta^L} V_{\theta,\beta}^K(z_t) = \max_{\pi_\theta^L} \mathbb{E}_{\pi_\theta^L, \hat{\mathcal{T}}_\beta} \left[\sum_{\tau=t}^{t+K-1} \gamma^{\tau-t} \log(D_\theta(z_\tau^{\pi_\theta^L}, a_\tau^{\pi_\theta^L})) + \gamma^K V_\beta(z_{t+K}^{\pi_\theta^L}) \right] \quad (1.11)$$

With this learning setting the proposed system outperforms previous works such as [73, 67] 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.

Learning from Observation (LfO)

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



Figure 1.11: Control tasks solved in [75]. From left to right: cheetah run, walker walk, car racing, claw rotate, baoding balls

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 a 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 can the embodiment mismatch be solved?; **(2)** in the case the demonstrator viewpoint differs from the imitator one, how can the correspondence problem between the two different viewpoints be handled?; **(3)** once the problems related to the perception subsystem are solved, how is the policy π^L obtained?.

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 demonstration shows a human performing a task, but the goal is to train a robot (Figure 1.12). To solve this problem, in [29, 77, 78], some sort of image-to-image translation was performed, while in [79] the *Temporal-Cycle Consistency* (TCC) [80] was used.

In particular, in [29, 78], the Cycle-GAN [81] 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 $G : X \rightarrow Y$, the second from the target to the source domain $F : Y \rightarrow X$. 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 [77] a similar approach was proposed, but the MUNIT [82] architecture was adopted for the image-to-image translation. With respect to the mentioned works, in [79] 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



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.

a correspondence between frames of different videos representing the same overall action (Figure 1.13).

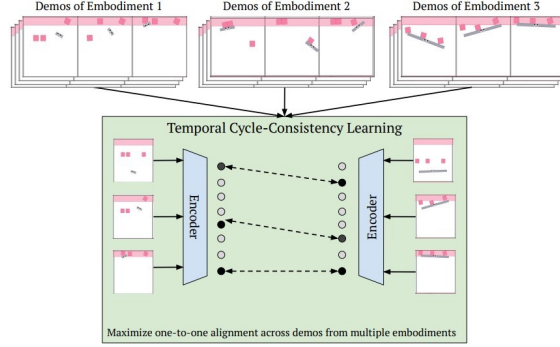


Figure 1.13: Temporal-Cycle Consistency representation, used to learn an embodiment-agnostic encoder in [79]

The correspondence problem between different viewpoints was faced by [30, 72]. In [30] a Convolutional Neural Network was trained by means of a *Triplet-Loss* [83]. 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-step, (x_i^a, x_i^p) , should have a similar embedding, while embedding produced by samples coming from the same viewpoint, but in different time-step, (x_i^a, x_i^n) , should have a different embedding (Figure 1.14a). In [72], a different approach was used, indeed, a *context translation problem* was solved by means of an Encoder-Decoder architecture (Figure 1.14b). The proposed architecture was trained on pairs of demonstrations, $\mathcal{D}_i = [o_0^i, o_1^i, \dots, o_T^i]$ and $\mathcal{D}_j = [o_0^j, o_1^j, \dots, o_T^j]$ composed of visual observations. Samples in \mathcal{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 \mathcal{D}_j conditioned on both \mathcal{D}_i and the first observation o_0^j from 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.

Concerning the way how the policy can be obtained, it is necessary to distinguish between Model-Based and Model-Free methods.

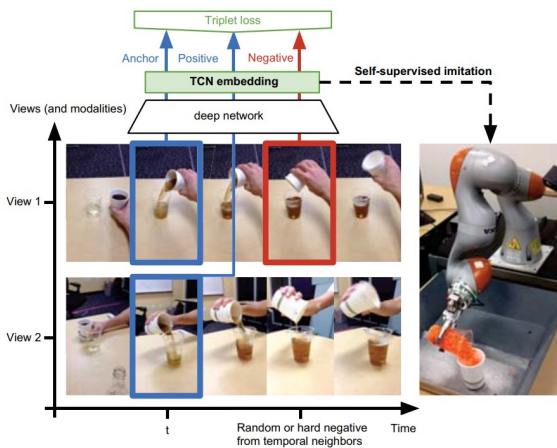
Among **Model-Free** methods, a further classification must be made between methods based on *Reward Engineering* and *Adversarial Learning*.

Methods based on Reward Engineering are [72, 30, 77, 79], 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 [72] 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 [30], 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

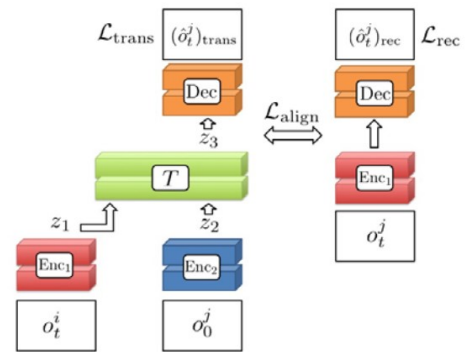
is the TCN embedding produced by the robot observation (Figure 1.14a). In [77], a keypoint-representation is obtained for both current robot observations z_t , and for each frames of the translated demonstration video $\{z_p^E\}_{p=1}^T$. Then, the reward is computed as $R(z_t, z_{t+1}, z^E) = -\lambda_1 \min_p \|z_t - z_p^E\| - \lambda_2 \min_p \|(z_{t+1} - z_t) - (z_{p+1}^E - z_p^E)\|$. In [79], the reward function was defined as $R(s_t) = -\frac{1}{k} \|\phi(s_t) - g\|_2^2$, where g is the goal embedding, defined as the mean embedding of the last frame of all the demonstration videos in the dataset, while $\phi(s_t)$ is the embedding of the current observation. Experimental results, on simulation data proved that the proposed method can be used to learn tasks from cross-embodiment demonstrations, outperforming baseline [30] in terms of both sample efficiency and performance.

Regarding the methods based on Adversarial Learning, there are [84, 85]. Obviously, these methods are strictly related to the Generative Adversarial Imitation Learning setting. However, with respect to the methods presented in GAIL paragraph (pag. 18), these methods do not assume access to the demonstrator action. Indeed, the goal of the preliminary work [84] 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 for a walking task, 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 state only demonstrations. The results obtained did not show a substantial difference between the two settings, supporting the hypothesis that in task learning the essential information is contained in the state.

The next remarkable work was proposed by the authors of [85], that formalized the *GAIfO* algorithm, (Algorithm 7), that is an extension to state-only demonstration of GAIL [66]. The proposed algorithm was used to train a network to solve tasks in simulation environment [71], 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 [30, 86], 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,



(a) Time-Contrastive network, proposed in [30].



(b) Context-Translation network, proposed in [72]

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

since the model-free TRPO [70] algorithm was used to train the policy. This problem was solved by DEALIO [87], which replaced the model-free algorithm with PILQR [88] the model-based RL algorithm reported next.

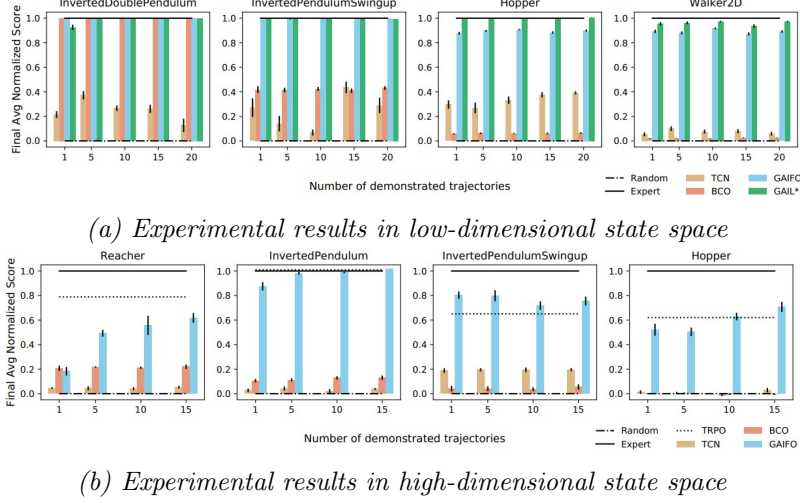


Figure 1.15: Experimental results reported in [85].

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., $s_{t+1} = T(s_t, a_t)$.

Methods that obtain the Inverse Dynamic Model are [89, 86, 90, 91]. In [89] 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 Convolutional Neural Network, that, taken in input a pair of images (I_t, I_{t+1}) representing two successive rope states, was able to obtain the action to perform in order to reach the state I_{t+1} from I_t . To train the proposed network, a dataset composed of 30K tuples (I_t, a_t, I_{t+1}) was collected by means of an exploratory policy. Authors in [86] proposed a general approach, depicted in Figure 1.16, and composed by 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_θ^i is updated by maximizing the probability $p_\theta(a_t|s_t, s_{t+1})$, 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

Algorithm 7 GAiFO algorithm [85]

Require: Initial policy π_ϕ^L , Initial Discriminator D_θ

Require: State-only expert demonstration trajectories $\tau^E = \{(s, s')\}$

while Policy Improves **do**

 Execute π_ϕ^L and collect state transitions $\tau^L = \{(s, s')\}$

 Update D_θ , 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 π_ϕ^L , with reward $r_{\pi_\phi^L} = - (\mathbb{E}_{\tau^L} [\log(D_\theta(s, s'))])$

end while

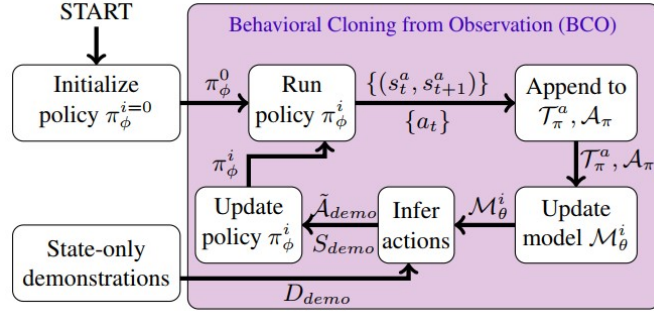


Figure 1.16: Representation of the learning procedure proposed by [86]

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 [90],

the same approach as in [86], but the agent’s policy was trained according to a linear combination of Behavioral Cloning and Advantage Actor Critic (A2C) objective function [92], i.e., $\mathcal{L}_\theta^{hyb} = \mathbb{E}_{s,a}[A(s) \log(a|s;\theta) + \alpha \mathcal{H}(\pi^L(\cdot|s))] + \mathbb{E}_{(\hat{s}_t, \hat{s}_{t+1}) \sim D}[\log(\pi^L(M(\hat{s}_t, \hat{s}_{t+1})|\theta))]$. Here, the main problem is the assumption to have access to reward function, which can reduce its applicability in real-robot manipulation tasks. In [91], the work in [93] 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 neural 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) [93], adapted for state-only demonstration, i.e., $g_{SOIL} = g + \lambda_0 \lambda_1^k \sum_{(s_t, \tilde{a}_t \in D')} \nabla_\theta \log \pi_\theta^L(\tilde{a}_t, s_t)$, 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 [29, 87]. In [29], 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 [94], which allows to obtain a controller optimized according to Linear-Quadratic Regulator (LQR) procedure. The idea was to optimize the policy in a low-dimensional high-regularized *latent space*, generated according to Variational Inference [95]. 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 [64]. Real world robotic experiments, shown that with **2 hours** of robot interaction it was possible to outperform previous works such as [30, 86] 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 [87] the sample-inefficiency problem of GAIfo [85] 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 [88]. The core of PILQR is the LQR optimization procedure. Generally speaking, it returns a *linear-gaussian controller* (Formula 1.12), that optimizes

a *quadratic-cost function* (Formula 1.13), under the assumption of *linear-gaussian dynamic* (Formula 1.14).

$$\pi(a_t|s_t) = \mathcal{N}(K_t s_t + k_t, S_t) \quad (1.12)$$

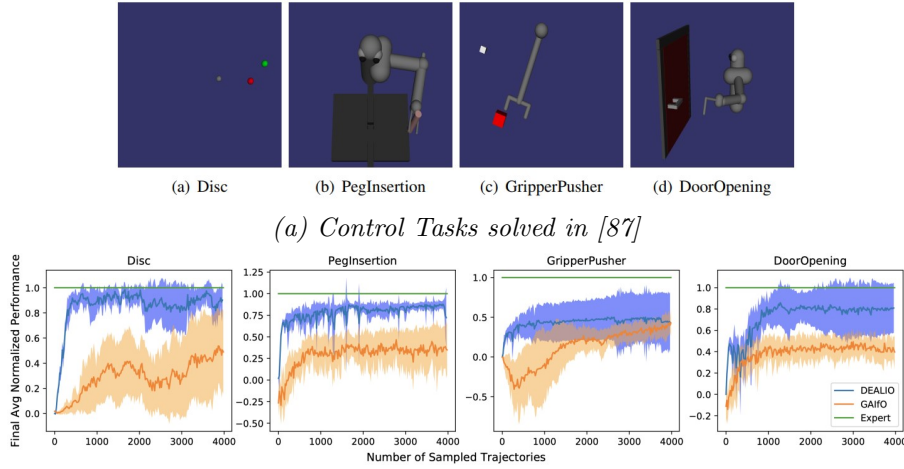
$$c(s_t, a_t) = \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T C_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T c_t + cc_t \quad (1.13)$$

$$s_{t+1} \sim P(s_{t+1}|s_t, a_t) = \mathcal{N}(F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t, \Sigma_t) \quad (1.14)$$

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 1.15) as function of the pair (s_t, a_t) .

$$D_\theta(s_t, s_{t+1}) = \frac{1}{2} \begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix}^T C^{ss}(s_t, s_{t+1}) \begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix} + \begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix}^T c^{ss}(s_t, s_{t+1}) \quad (1.15)$$

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.



(b) Performance of DEALIO [87] compared against GAIfo [85], with respect to the number of trajectories sampled during the learning process.

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

Generally, LfO methods have demonstrated interesting features, such as generating a policy from state-based information alone, supporting the hypothesis that the primary source of information for task learning is the sequence of state transitions. Extrapolating the valuable information to perform actions that induce the desired behavior may not be trivial, mainly if the state space is represented by images of a human operator, leading to the design of architectures composed of different stages, which increase not only the complexity of the system itself but also the amount and diversity of data required for their training. In addition, many methods of interest have been tested in simulated or otherwise

relatively simple scenarios, still leaving open the question of whether these methods can be used in real-world complex robotic manipulation tasks.

1.2.4 Summary

This section summarizes all the approaches seen so far, trying to understand their pros and cons. Answering which approaches leads to better performance than the others is not trivial for several reasons ranging from the fact that, as mentioned above, methods are tested in different environments (e.g., some in simulation, others in the real world), on different tasks and using different robotic platforms. It can be said that, by analyzing the state of the art, the most studied approach is Behavioral Cloning. This approach, from a pure implementation point of view, is the simplest, as, in principle, it allows the execution of a purely offline training procedure. Unfortunately, some problems have to be handled, ranging from compounding-error to dataset creation. About the former, some solutions have been proposed, based on interactive learning algorithms, which reduce the covariate-shift phenomena, but introduce the need for active human supervision during the learning. During the past few years, there has been increasing interest in Meta-Learning algorithms, with the ultimate goal of achieving a system that can generalize across different tasks. However, as it turns out, the generalization problem is far from solved. Regarding approaches such as IRL, GAIL, and LfO. Compared to BC, methods such as IRL, GAIL, and LfO are more efficient regarding required demonstrations. However, they introduce RL algorithms into the optimization loop, which requires interaction with the environment, which can be risky and time-consuming. Moreover, despite their promising results, relatively few methods of these three approaches have actually been used in real-world vision-based manipulation tasks, effectively leaving unanswered the question of whether these methods can be used.

CHAPTER 2

RESEARCH PLAN

This chapter will present the research project. Specifically, the discussion will begin with a description of the various research gaps that can be extrapolated from the current literature and the importance of addressing these gaps with respect to the ultimate goals of Learning from Demonstration. Next, the research proposal for the next two years will be presented, contextualizing it within the previous study and highlighting its strengths and importance. Along with the presentation of the project, the techniques and methodologies used to implement the project and evaluate the results obtained will also be described.

2.1 Research Gaps

Regardless of the application area, whether industrial or social, one of the main goals of robotics is to have a system capable of performing various tasks in dynamic environments characterized by a high degree of uncertainty. This system must be characterized by several properties, among which are **high adaptability** to new scenarios and operating conditions, **high reliability** in execution, and it must be able to run in **real-time** on **real-world robot systems**. As seen from the study of the literature, learning-based methods appear to be of interest, given their ability to infer a task execution policy from a series of demonstrations, mainly when the application context is characterized by high uncertainty, such as the location or type of the target object or the task itself that has to be executed. Although the problem of Learning from Demonstration has been studied for many decades, there are still substantial gaps at different levels of abstraction. Gaps can be identified that are both methodological and purely applicative.

Regarding the methodological aspects, there are different questions that, despite being studied, have not yet been fully answered, these gaps will be briefly discussed.

How can the compounding-error be mitigated? As reported in Section 1.2, the compounding-error is the most important limitation related to Imitation Learning. Solutions proposed in the literature are mainly based on algorithms that require active interaction between the learner and the expert. This may lead to both safety problems and time-consuming learning procedures, especially for complex long-horizons tasks. While in the context of autonomous vehicle driving is prohibitive and very risky to generate bad examples, in the context of robotic manipulation, it may have sense to produce a set of low-quality demonstrations (e.g., demonstrations where the robot fails to pick the target object), label these demonstrations with a quality index (e.g., based on execution time,

smoothness of the performed action) or event classes (e.g., “approaching” class for those frames where the robots approaches the target object, “closing gripper” for those frames where the robot closes the gripper, etc. . .), then show recovery behaviors and label them. In this way, the system can learn the correct task execution and actions to be performed in case of failure. In this case the main challenge is to design an architecture able to identify a divergence situation, and performing recovery actions, based on mainly exteroceptive sensors.

How can a system generalize both to different instances of the same task and to different tasks? When it comes to systems based on Artificial Intelligence techniques, the leading property of this should be the generalization capability. In applying such systems in the context of task control and execution, there are at least two levels of generalization. The first is a single-task generalization capability, and the second is a multi-task generalization capability. The former is related to the desire to obtain a system that can correctly execute different instances of the same task (e.g., different objects and/or different initial conditions). The latter, on the other hand, is related to the desire to have a system able to solve different tasks without complete retraining, which can be time-consuming and data inefficient. To achieve this goal, recent methods proposed to use Meta-Learning algorithms, showing promising results but with relatively low overall performance. Improvement can be achieved by better exploitation of temporal information and by the improvement of learning procedures and architectures that must be guided by the goal of extrapolating (sub-)task related embeddings, independent of context (e.g., demonstrator and workspace visual appearances) so that they can be reused and combined for the execution of new instances or new tasks. Moreover, given the relatively low amount of data in the produced dataset, the exploitation of temporal information should be combined with efficient and meaningful task representation, given by latent-variable in Model-Based methods.

How can a dataset be reused across different works? When AI-based methods are used, the dataset plays a key role. The ability to train and test methods on a common dataset is crucial for a fair comparison, for pointing out methods’ criticality, and consequently, for findings hints for possible improvements. Generally speaking, dataset collection is a tedious and time-consuming process, and this is particularly true when collecting task demonstrations. Having the ability to reuse a dataset can save time, allowing a greater focus on the learning procedure and overall architecture, increasing potentially the rate of advancements. Unfortunately different aspects may severely limit the reuse of a dataset, for example, the used robotic platform or the different visual appearance. The former plays a critical role, especially when robot-specific information, such as the joints’ position/velocity/torque, are used. In order to avoid the naive approach based on collecting demonstrations by using different robots, a starting point could be the collection of a dataset that contains platform-independent information, such as the trajectory of the manipulated object or video of a human performing the desired task. This solution leads to the other important challenge, that is the difference in appearance between a public dataset and the actual testing conditions (e.g., different objects but of the same category, different backgrounds, viewpoints etc.). To solve these problems, one might think of using simulated environments, which allows to train and test different methods on the same environment. However, in the end, the system must be deployed in a real-world context, and the problem of Sim-to-Real Transfer must be handled. Instead of having a shared environment to use for training and testing the proposed methods, an alternative solution can be found by formalizing the problem differently. In fact, from datasets showing differ-

ent demonstrators (both human and robot) in different environments and with different objects but performing the same task, the goal can be to develop a training procedure and architecture capable of extrapolating task-dependent embeddings but context agnostic. This setting is very similar to Learning from Observation, where early examples of applications of these concepts can be found [30, 79].

How can the perception errors be mitigated? As stated in the paragraph dedicated to Behavioral Cloning (pag. 20), one of the main reasons for task failure is related to errors in perception and identification of key points during task execution (e.g., failure to close the gripper on object approach, hitting the object of interest because the system does not recognize its presence). A possible way to solve these problems is to use a multi-modal perception system, which could include tactile and/or proximity information in addition to RGB(-D) visual information, in order to explicitly model, in the demonstrations, the fact that the robot touches an object then it closes the gripper. Moreover, the idea behind Hierarchical Task Decomposition methods may also play an important role in solving such problems since they model sub-policy activation/deactivation when a given event is detected, however the event detection should not be based on hand-written conditions (e.g., distance between objects computed through the identification of specific markers), but the system should be able to learn, and then produce automatically the sub-tasks relationships. With respect to current solutions, the main challenge is to guarantee a certain level of explainability and safety of the performed actions, in order to increase the reliability of such systems, and move towards systems that can actively collaborate with human to perform a task.

How can a human describe the task to be executed? Being able to express the intent as efficiently and naturally as possible can lead to several benefits, both in terms of pure human-robot interaction (e.g., during the system operation, the human can describe the task to execute using a voice command), and during the training procedure itself. Regarding this last point, the human intent can be described by means of video demonstrations and natural language task descriptions. These kinds of intent representation can be exploited: **(1)** in the multi-task setting, where they can be used as policy conditioning, in this way, the policy can infer the action with respect to both the current observation and the required task; **(2)** in the LfO approach, where the whole dataset can be composed of only unlabeled video of human performing a task. It is reasonable to think that having either a system able to infer task-relevant features as effectively as possible may increase the training efficiency and the overall system performance. The former is a very important aspect to consider because in the context of robotics it is not so trivial to obtain the amount of samples that are usually needed to train a system based on deep architectures.

As for the questions more related to application aspects. Based on the proposed state of the art, there is one that tower above all others, mainly related to the approaches of IRL, GAIL, and LfO, that will be discussed below.

Can these approaches be effectively used for solving complex manipulation tasks on real robot platforms? In particular, the theoretical properties behind training frameworks related to IRL and GAIL are well known. In addition, proposed experiments both in simulation and real environments on more or less complex tasks have shown promising results, especially for methods based on GAIL and LfO. However, on the basis of current knowledge, there is a lack of experimental analyses in real environments that can point out flaws of such approaches when applied in a real context and thus guide the development of new methods to solve them. For example, most LfO methods applied in robot manipulation are based on some image-to-image translation, where the video

of a human demonstration is translated into the corresponding robot video by means of complex generative architecture, but, instead of using the video demonstration as it is, *can a more efficient and effective task representation be obtained?* Such representation could be an object centric trajectory extrapolated from the human demonstration, or a set of keypoints tracked from the human arm movements. In this way, the problem becomes either to directly find the robot action that generates the desired object movement, or find the mapping between human and robot motions, in order to solve the task.

All the gaps presented up to now are stricly related each others, and focusing on a specific aspect can lead to hints for solving other gaps.

2.2 Proposed Research Activity

In this section the proposed research activity will be presented, and linked with the gaps discussed in Section 2.1.

As seen, the problem of Learning from Demonstration is far from being solved. Indeed, there is a relevant amount of open questions that cover both well-studied Behavioral Cloning and Inverse Reinforcement Learning methods and the novel Generative Adversarial Imitation Learning and Learning from Observation methods. From the study of the state of the art and the reported research gaps, it could be seen that, in recent years, there has been a particular interest on the two aspects, the first is related to the question *“How can a system generalize both to different instances of the same task and to different tasks?”* and the second is related to the question *“How can a human describe the task to be executed?”*. Developing and proposing a solution that can answer these two questions would result in a **highly adaptable system**. In fact, solving the first problem would result in a system that can be applied to different scenarios as it is, minimizing the effort required to fine-tune for the specific application. While solving the second problem through human video demonstrations and natural language descriptions would result in a highly human-centered human-machine interaction, thus making available a robot programming technique that follows a very natural and intuitive process. By keeping in mind these aspects, the final goal of this research activity is to investigate the possibility to design and develop an innovative Artificial Intelligence based system able to replicate a set of tasks given in input expert demonstrations represented by robot/human video performing a task, and further natural language task description. In particular, starting from the methods that have been presented in the state of the art, the aim is to make an improvement in both methodological and technological terms, going on to investigate the following aspects:

Dataset and Task representation

As already discussed in Section 2.1 the dataset is one of the key element for an AI based system. With respect to the datasets collected by the presented methods, the goal will be to investigate the following: **(1)** Whether the use of additional information, such as a frame label of the performed (sub-)task, can improve the ability of the method to extrapolate embeddings related to the (sub-)task and infer the action based on the current observation of the robot (first-person or third-person visual information), the current state of the robot (joint position, speed), the label of the current (sub-)task. **(2)** Closely related to (1), the second investigation related to the dataset and task representation concerns how the intent of the task can be provided most naturally and intuitively as possible. Here, one aspect to be investigated is how a natural language description combined with

a video demonstration can be exploited to act as conditioning or (self-)supervision during the training in order to extract both high-level task information (e.g., pick-and-place) and task-specific information (e.g., pick the green ball and place in the red cup). Concerning these two points, the investigation will begin by collecting this information in a simulated environment, (e.g., robosuite [96] or openai-gym [71]). Since, it is easier and faster to automatically collect a massive dataset containing different scenarios and robotic platforms. Once extensive tests have been performed in simulation, and the pros and cons have been weighed, the next step will be designing and developing a real-world activity demonstration recording system for both human and robotic demonstration videos.

Methods and learning procedure

The investigation of methods and learning procedures is strictly related to the dataset and task representation. As already discussed, the goal is to obtain a vision-based multi-task imitation learning system. As seen in the State-of-the-Art section, learning-based approaches are essential for solving this problem. Building upon the currently proposed methods, the investigation will be devoted to designing and developing a system based on modern architectures (e.g., Recurrent Neural Network, Transformers, (conditional) Variational-Autoencoder) that have been proved to be effective in learning contextual information, and latent-space representation from temporal sequences. The challenge here is to apply such methods in multi-task learning from demonstration. In this setting, both high-level and task-specific embeddings must be extracted and combined, given in input video and natural language task descriptions. The experimental validation will be performed first in simulation, where pros and cons could be identified in a safe environment. Then based on the obtained results, the system will be tested on a real-world robot platform.

Computational efficiency and real-time constraint

Deep Learning architectures are known to be able to solve very complex problems. Still, they generally require an appropriate amount of computational resources and capability to produce an output in a reasonable time. This problem becomes critical when the system must control an agent interacting with the environment by performing actions. In proposing a novel method, a particular focus must be posed on the computational requirements of the method itself. The right trade-off between two properties must be found: (1) the inference time, which is in general as low as the complexity of the system decreases; (2) the system accuracy, which is as high as the complexity of the system increases. Generally speaking, two possible solutions can be found for real-world robot applications. The first is to deploy the system on an expensive workstation connected to the robot through some communication protocol. The second is related to cloud-based solutions, where the method is deployed on an external server, communicating through Internet connection. Generally speaking, since these proposed systems must work on light-load cobots or social robots, a workstation-based solution can be inadequate regarding the total cost. In contrast, a cloud-based solution can be inadequate because the internet connection may be missing or introduce unacceptable delays. A third, less explored solution is based on running the system on a low-cost embedded platform (e.g., Nvidia Jetson Xavier) connected directly to the robot. Based on all these considerations, the final goal is to design and deploy a system that can solve the problem at hand with a reasonable success-rate while running on an embedded platform.

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. The course presents the Logic Programming paradigm and the Functional Programming paradigm, related to the more general declarative paradigm. During the course, exercises were conducted through which it was possible to analyze and solve problems using the paradigms presented, thus understanding their importance and the main differences from the more well-known Imperative Paradigm.
- English Language C1.

3.1.2 Additional Courses

I decided to add the following two 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] 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.
- [4] 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.
- [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. Kumar, J. Hong, A. Singh, and S. Levine, “Should i run offline reinforcement learning or behavioral cloning?,” in *International Conference on Learning Representations*, 2021.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.

- [26] 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.
- [27] C. Systems, “Cyberforce.”
- [28] D. Systems, “Touch,” Jun 2020.
- [29] 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.
- [30] 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.
- [31] H. Liu, C. Zhang, Y. Zhu, C. Jiang, and S.-C. Zhu, “Mirroring without overimitation: Learning functionally equivalent manipulation actions,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8025–8033, Jul. 2019.
- [32] H. Liu, X. Xie, M. Millar, M. Edmonds, F. Gao, Y. Zhu, V. J. Santos, B. Rothrock, and S.-C. Zhu, “A glove-based system for studying hand-object manipulation via joint pose and force sensing,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6617–6624, IEEE, 2017.
- [33] A. Ijspeert, J. Nakanishi, and S. Schaal, “Learning attractor landscapes for learning motor primitives,” *Advances in neural information processing systems*, vol. 15, 2002.
- [34] 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.
- [35] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, “Movement segmentation using a primitive library,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3407–3412, IEEE, 2011.
- [36] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, “Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction,” *Autonomous Robots*, vol. 43, no. 6, pp. 1291–1307, 2019.
- [37] A. Agostini, M. Saveriano, D. Lee, and J. Piater, “Manipulation planning using object-centered predicates and hierarchical decomposition of contextual actions,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5629–5636, 2020.
- [38] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” *Advances in neural information processing systems*, vol. 26, 2013.
- [39] D. A. Pomerleau, “Alvin: An autonomous land vehicle in a neural network,” *Advances in neural information processing systems*, vol. 1, 1988.

- [40] 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.
- [41] 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.
- [42] 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.
- [43] 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.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] 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.
- [48] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine, “One-shot imitation from observing humans via domain-adaptive meta-learning,” in *Proceedings of Robotics: Science and Systems*, (Pittsburgh, Pennsylvania), June 2018.
- [49] T. Yu, P. Abbeel, S. Levine, and C. Finn, “One-shot hierarchical imitation learning of compound visuomotor tasks,” *arXiv preprint arXiv:1810.11043*, 2018.
- [50] 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.
- [51] S. Dasari and A. Gupta, “Transformers for one-shot visual imitation,” in *Conference on Robot Learning*, pp. 2071–2084, PMLR, 2021.
- [52] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, “Neural task programming: Learning to generalize across hierarchical tasks,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3795–3802, IEEE, 2018.

- [53] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos, “Robot learning manipulation action plans by” watching” unconstrained videos from the world wide web,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.
- [54] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, “Gti: Learning to generalize across long-horizon tasks from human demonstrations,” in *Proceedings of Robotics: Science and Systems*, (Corvalis, Oregon, USA), July 2020.
- [55] D. B. Grimes and R. P. Rao, “Learning actions through imitation and exploration: Towards humanoid robots that learn from humans,” in *Creating Brain-Like Intelligence*, pp. 103–138, Springer, 2009.
- [56] P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters, “Probabilistic model-based imitation learning,” *Adaptive Behavior*, vol. 21, no. 5, pp. 388–403, 2013.
- [57] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, “Multi-task policy search for robotics,” in *2014 IEEE international conference on robotics and automation (ICRA)*, pp. 3876–3881, IEEE, 2014.
- [58] 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.
- [59] 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.
- [60] 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.
- [61] 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.
- [62] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” *arXiv preprint arXiv:1507.04888*, 2015.
- [63] 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.
- [64] 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.
- [65] 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.
- [66] J. Ho and S. Ermon, “Generative adversarial imitation learning,” *Advances in neural information processing systems*, vol. 29, 2016.

- [67] 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.
- [68] J. Fu, K. Luo, and S. Levine, “Learning robust rewards with adversarial inverse reinforcement learning,” in *International Conference on Learning Representations*, 2018.
- [69] 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.
- [70] 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.
- [71] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [72] 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.
- [73] S. Reddy, A. D. Dragan, and S. Levine, “Sqil: Imitation learning via reinforcement learning with sparse rewards,” in *International Conference on Learning Representations*, 2019.
- [74] 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.
- [75] 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.
- [76] 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.
- [77] 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.
- [78] 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.
- [79] 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.
- [80] 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.

- [81] 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.
- [82] 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.
- [83] 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.
- [84] 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.
- [85] F. Torabi, G. Warnell, and P. Stone, “Generative adversarial imitation from observation,” *arXiv preprint arXiv:1807.06158*, 2018.
- [86] 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.
- [87] 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.
- [88] 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.
- [89] 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.
- [90] 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.
- [91] 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.
- [92] 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.
- [93] 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.

- [94] 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.
- [95] 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.
- [96] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, “robosuite: A modular simulation framework and benchmark for robot learning,” *arXiv preprint arXiv:2009.12293*, 2020.