Università degli Studi di Salerno

Dipartimento di ingegneria dell'Informazione ed
Elettrica e Matematica applicata

Ph.D. Program in Information Engineering

Second year report

Author:
dott. Francesco Rosa

Supervisor:
Ch.mo. Mario Vento

XXXVII cycle - Academic year 2022/2023

Contents

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 4 mandatory chapters, Background 1, Research Plan 2, Preliminary Results 3, and Other Activities 4.

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 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 containes the Research Topic (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 contains the Dataset Description (Section 3.1) and the Experiment Results (Section 3.2) sections. The former describes the dataset used during the experiments and the type of data that it contains. While the latter reports the results obtained by running the designed experiments.

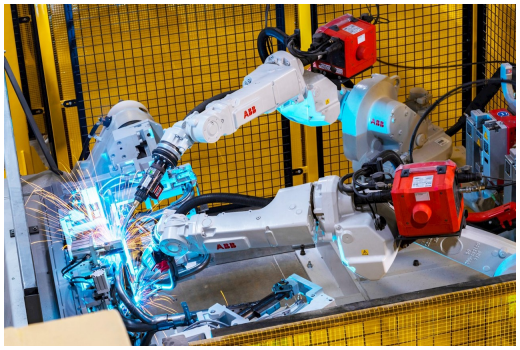
The fourth chapter shows all the activities carried out during the year, including courses attended at the university, and other research activities.

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.

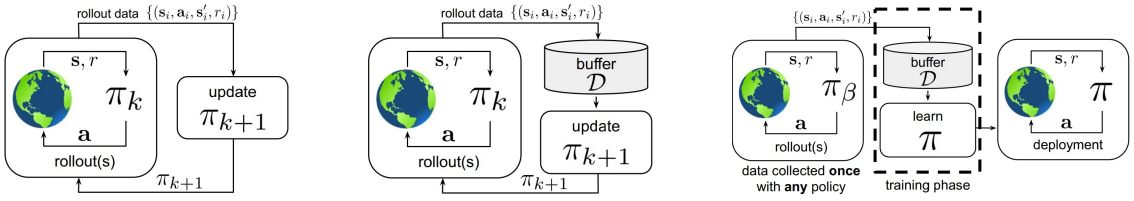
In this scenario, the desired characteristics of such robotic systems are: **(a) Adaptability to new conditions**, i.e., the system must be able to easly adapt to dynamc change in system and environmental conditions (e.g, motor failure or a different ground's coefficient of friction for a quadruped robot [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 easly 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].



(a) On-policy RL *(b) Off-policy RL* *(c) Offline RL*

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 underlaying 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].

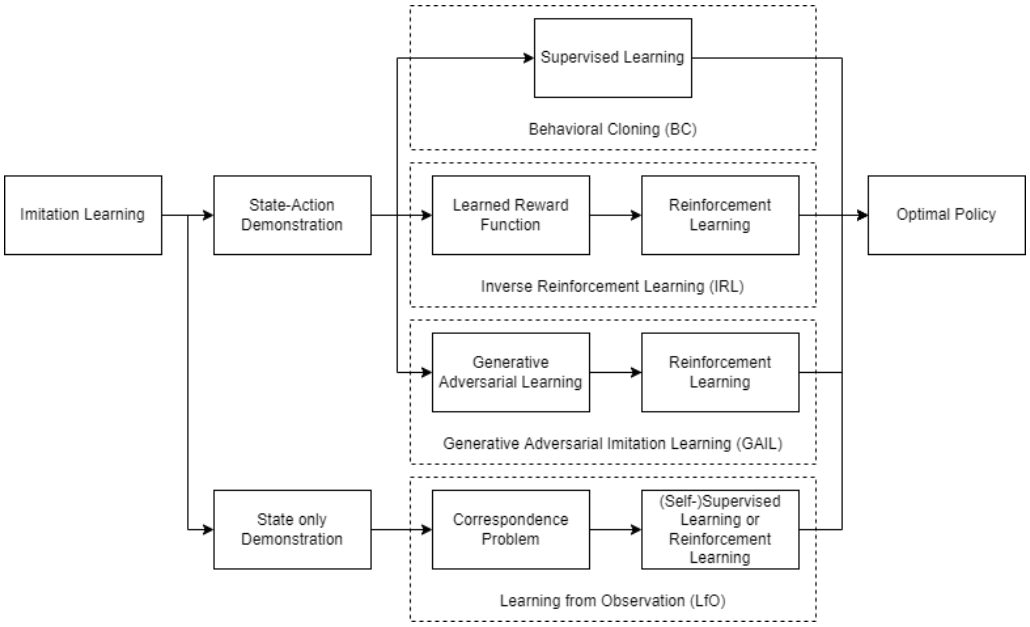


Figure 1.3: Imitation Learning: Taxonomy and main components

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 $\mathcal{D} = \{(\boldsymbol{\tau}_i, \boldsymbol{c}_i)\}_i^N$, where:

- $\boldsymbol{\tau}_i$ is the i^{th} demonstrated trajectory, which can be described as:
  - A state-action sequence, i.e., $\boldsymbol{\tau}_i = [s_0, a_0, \dots, s_T, a_T]$, when it is assumed to have access to the ground-truth action performed by the expert.
  - A state sequence, i.e., $\boldsymbol{\tau}_i = [s_0, \dots, s_T]$, when it is assumed to not have access to the ground-truth action.
- c_i is the *context-vector*, it contains task-related information, e.g. the initial state of the system s_0, the position of the target object, or a representation of the task to be executed (e.g., natural language description of the task or video demonstrations).

The state s_i can be represented by combination of images representing the robot's workspace and robot's proprioceptive information such as joints position, velocities and torques. While the ground truth action can be obtained by teleoperation or kinesthetic teaching (Section 1.2.2).
The policy π^L can be defined in two distinct ways. In one case, it is described as a deterministic function, meaning that it directly determines the action as follows: $a_t = \pi^L(s_t, c_i)$. In the other case, it takes on a probabilistic definition, where it represents a probability distribution from which to sample the desired action: $a_t \sim \pi^L(s_t, c_i)$.
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., $\pi : \mathbf{c} \rightarrow \boldsymbol{\tau}$. 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:

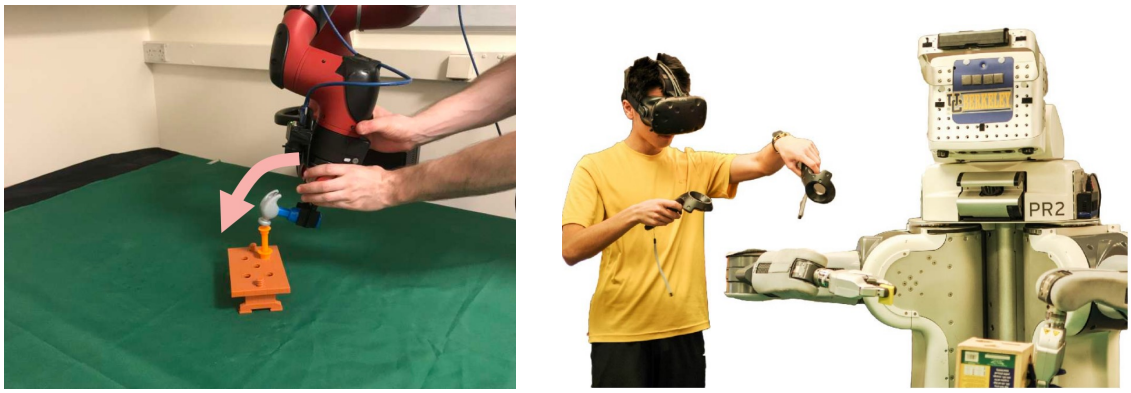
- $S \subseteq \mathbb{R}^n$, is a set of states (e.g. joints position and/or image).

- $A \subseteq \mathbb{R}^n$, 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.

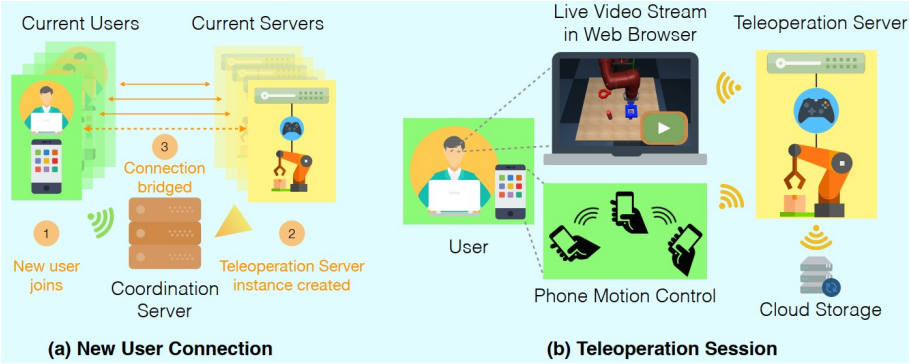


Figure 1.5: System diagram of Roboturk [25]

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, $\theta^* = \underset{\theta}{argmin}\, \mathbb{E}_{(\boldsymbol{\tau},\mathbf{c})\sim\mathcal{D}^E}\, [\mathcal{L}((\boldsymbol{\tau},\mathbf{c}),\, \pi_\theta^L)]$. 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:

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

$$\tau\dot{s} = y \tag{1.2}$$

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

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), $f(z(t)) = (g-s_0)\sum_{i=1}^{M}\psi_i(z(t))\omega_i z$. 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, $\{\omega_i, i = 1,\ldots,M\}$, 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 $\boldsymbol{\tau}$ is written as $\boldsymbol{\tau} = \prod_t \mathcal{N}(s(t)|\Psi(z(t))^T\omega, \Sigma_s)$, 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, $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$
    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

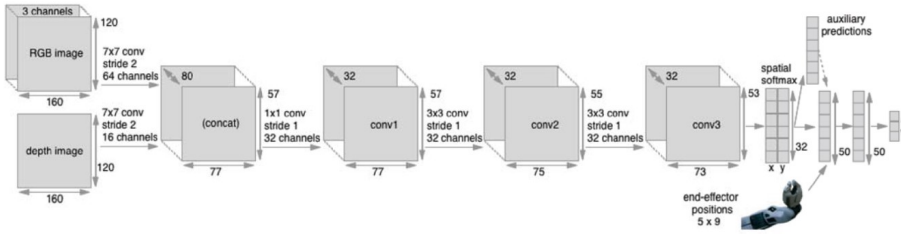


Figure 1.6: Architecture proposed in [20]

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

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 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.

Meta-Learning is based on the idea 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].

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

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 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 or 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.

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

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 $\mathcal{L}_\psi$, is the learned adaptive loss, which is used during the **Meta-Test step**, where the policy parameters are adapted with gradient descent given in input a video of human 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. Meta-Learning algorithms have

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.

demonstrated intriguing properties, notably their capacity for few-shot generalization to novel objects and object configurations. However, it has been observed that during the adaptation step, these methods tend to lose their effectiveness in performing other tasks. This limitation has underscored the need for the development of Multi-Task Imitation Learning methods, which aim to address these shortcomings and enable more versatile task execution in complex scenarios.

Multi-Task Imitation Learning aims to solve the problems related to Meta Learning. Indeed, the final goal of such methods is to obtain a general-purpose robot that is able to master a wide range of tasks and quickly learn a novel one by leveraging past experience.
Indeed, starting from an *expert dataset* containing n tasks, $\mathcal{D}^E = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$, where $\mathcal{D}_i = \left\{(\tau^j_{m_i},\ c_{m_i}), j = 1, \ldots, N,\ m_i \in \mathcal{M}_i\right\}$ is the *single-task dataset*, composed of:

- N expert demonstration for each m^{th} variation of the i^{th} task (Figure 1.10). M_i is the number of variation for the i^{th} task.
- Agent trajectories $\tau^j_{m_i} = (s_0, a_0, s_1, a_1, \ldots, a_{T-1}, s_T)$. where s_t is the state at time t and a_t is the corresponding action (Section 1.2.1).
- Task-conditioning c_{m_i} for the m^{th} variation of i^{th} task, which describes the desired task in terms of video demonstrations [51, 52, 53, 4], natural language description [54, 5, 55, 56, 57, 58, 59] or multi-modal prompt, that exploits both visual information (e.g., an image of the target object) and text information that contains the information related to the action to be performed [60].

The goal of Multi-Task Imitation Learning is to learn a *conditioned policy* $\pi^L_\theta(a_t|s_t, c_{m_i})$, that is able to map the current state and command into the corresponding action. Depending on how the policy is defined, various loss functions come into play. In the case of deterministic policies, the learning process focuses on minimizing the Mean-Squared Error (refer to Formula 1.5). However, for probabilistic policies, the learning process centers around minimizing the Negative Log-likelihood (refer to Formula 1.6). This approach aims to enhance the probability of correctly executing the action.

$$\mathcal{L}(\tau^j_{m_i}, c_{m_i}, \pi^L_\theta) = \frac{1}{T}\sum_{t=0}^{T}(a_t - \pi^L_\theta(s_t, c_{m_i})) \tag{1.5}$$

$$\mathcal{L}(\tau^j_{m_i}, c_{m_i}, \pi^L_\theta) = -\frac{1}{T}\sum_{t=0}^{T} log(\pi^L_\theta(a_t|s_t, c_{m_i})) \tag{1.6}$$

Regarding the methods that use a selection of frames as conditioning signal, there are approaches such as [51, 52] that use the first and last frame of the task demonstration. The fundamental concept underlying these methods involves the utilization of a Deep Convolutional Neural Network to encode this task representation. Subsequently, the resultant embedding serves as input for the control network, as illustrated in Figure 1.8. It is worth noting that these approaches tend to face a significant challenge, namely, that the conditioning signal fails to adequately encapsulate pertinent information regarding the optimal approach for solving the task. This limitation arises from the fact that the conditioning signal lacks comprehensive encoding of task-solving strategies.

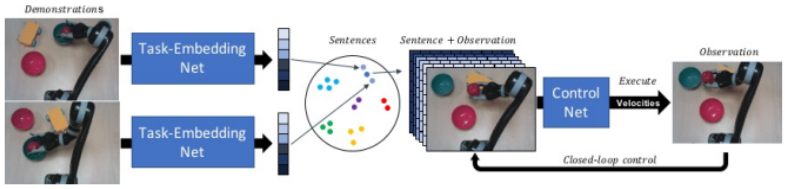


Figure 1.8: Architecture proposed in [51]

The methodologies introduced in [52] and [4] aim to advance the concept of an ideal multi-task agent, capable of replicating a new task based on a single demonstration, often performed by another agent (e.g., a robot or a human). The task execution takes place in a setting that differs from the one in which the demonstrator operated, potentially involving distinct object configurations, as illustrated in Figure 1.9. One particularly noteworthy contribution is presented in [4], where the authors introduce the Multi-task One-Shot imitation with self-Attention and Contrastive Learning (MOSAIC) architecture. This architecture is designed to model a demonstration-conditioned policy, denoted as $\pi_\theta^L(a_t|s_t, c)$, with c representing the current task demonstration, such as a video depicting a robot performing the task. MOSAIC addresses the challenge of multi-task imitation learning by incorporating two key components: **(1)** A time-contrastive loss, serving as additional supervision for representation learning, with the goal of obtaining similar embeddings for temporally proximate frames; **(2)** A self-attention model architecture, employed to extract contextual information from demonstrations, which is then utilized by the neural network to infer actions. To facilitate experimentation and evaluation, a dataset containing **seven distinct tasks** and **61 variations** (Figure 1.10) was generated by executing hand-written policies in a simulation environment. This dataset was employed to train and compare the proposed MOSAIC architecture against other one-shot (meta-)imitation learning methods, such as [48] and [53]. The results, presented in Table 1.2, demonstrate that MOSAIC surpasses previous methods in the Single-Task One-shot Imitation Learning Setting, specifically when addressing individual tasks with multiple variations and testing on previously unseen scenarios (i.e., novel objects configuration). Furthermore, when subjected to evaluation in a Multi-Task setting, wherein the system is trained on all the tasks and subsequently tested on each task separately, MOSAIC exhibits the capability to partially replicate the demonstrated tasks. It is important to note that transitioning from a single-task to a multi-task context presents inherent challenges. Despite being familiar with the tasks used during training, the success rate tends to decrease for almost all tasks. This phenomenon underscores the necessity for both training procedures and architectures capable of generating embeddings that accurately represent both the task itself and its various sub-tasks. This capacity is essential for reusing these embeddings when executing

new instances or entirely novel tasks.

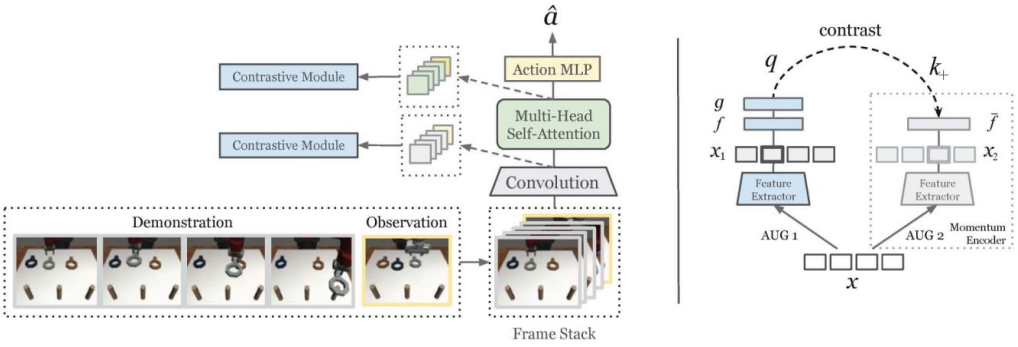


Figure 1.9: MOSAIC architecture [4]

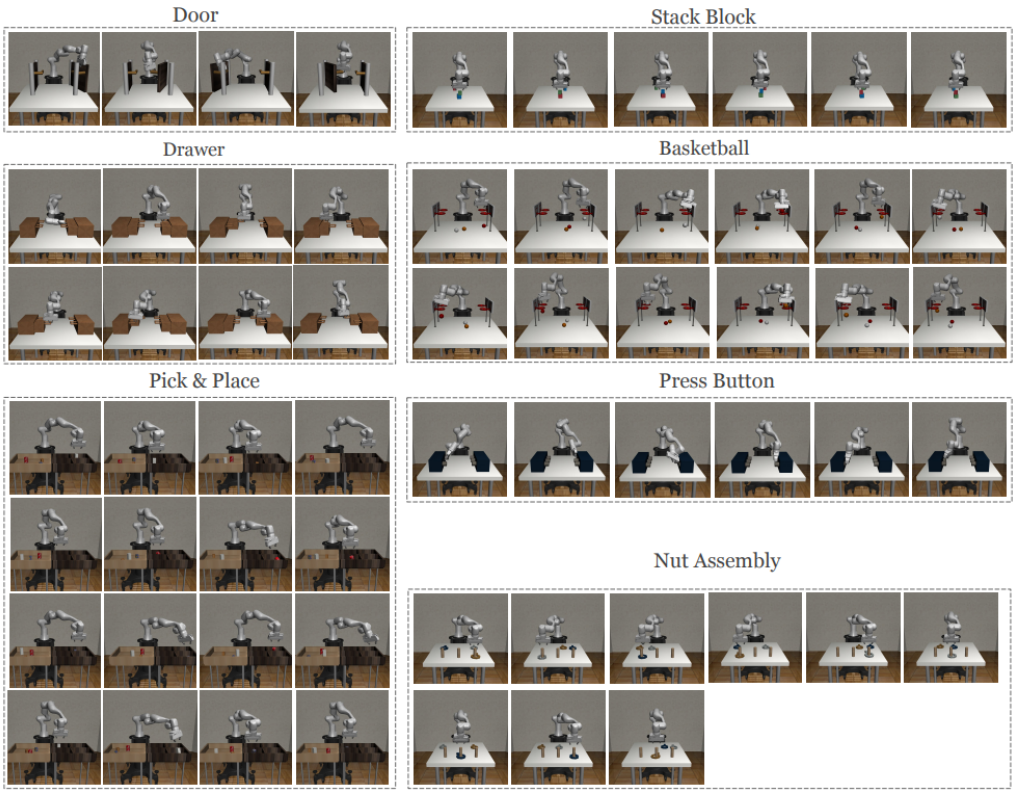


Figure 1.10: MOSAIC [4] proposed tasks

In addition to the previously discussed visual-conditioned techniques, the scientific literature has also explored an alternative approach to define a conditioning signal. This alternative method involves employing a ***natural language description*** of the task that is intended for execution. This approach enables a human operator to communicate with the robot, specifying the task it should perform. The inception of these methods was prompted by the realization that a natural language command, such as *"Pick the blue cube and place it in the red bowl"*, not only conveys information about the desired task (i.e., picking and placing) but also contains semantic information about the two target objects (i.e., the blue cube and the red bowl). Consequently, through training a neural network

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

Task	Setup	DAML [48]	T-OSIL [53]	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**

with a diverse set of tasks, the system should exhibit the ability to generalize its understanding to both previously unseen objects within familiar tasks and entirely novel tasks composed of the fundamental actions practiced during training. This approach showcases the potential for robust and adaptable human-robot interaction in real-world scenarios. Foundational research, exemplified by works such as [54] and [5], has sought to investigate the previously stated hypothesis. In particular, the authors of [54] introduced an innovative architectural framework, illustrated in Figure 1.11, marking the first instance where language, vision, and control tasks were seamlessly integrated. This model consists of two essential components: a Semantic Model, responsible for generating a task embedding denoted as e, using the initial scene image and the accompanying text command, and a Control Model, tasked with generating the control signal, utilizing the current robot state r_t and the task embedding e. Training of this model was conducted on two fundamental tasks, namely "Picking" and "Pouring", within scenarios featuring multiple objects of the same category, which served as distractors (see Figure 1.12). The subsequent testing experiments demonstrated the system's capability to successfully complete the picking task 98 out of 100 times and the pouring task 85 out of 100 times within novel scenarios. These results serve as compelling evidence of the efficacy and potential of language-conditioned methodologies in the field.

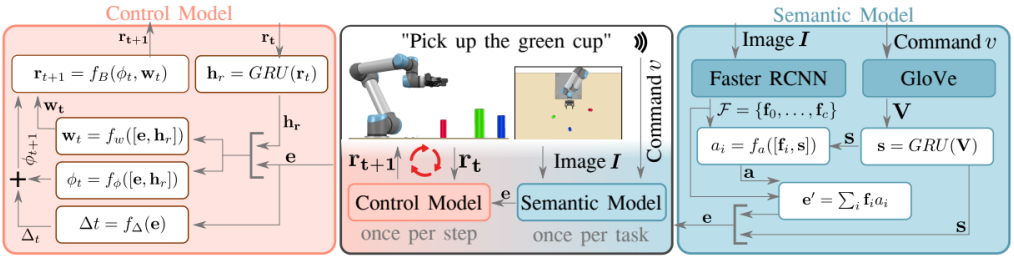


Figure 1.11: Architecture proposed in [54]

Authors in [5] make a step towards a more general agent, by proposing a large-scale dataset containing **100** diverse manipulation tasks. 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,

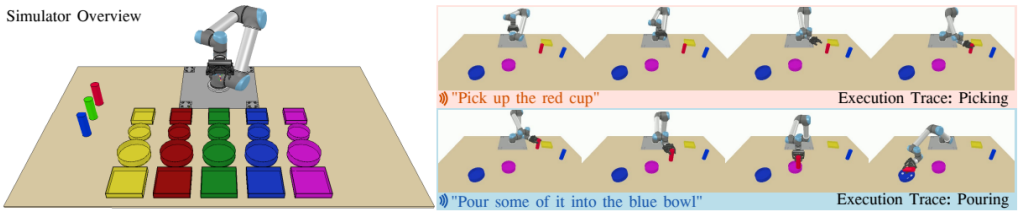


Figure 1.12: Set of object used in [54] (left), sample of task execution (right)

and push skills. The dataset was used to train the network in Figure 1.13. 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***. 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.

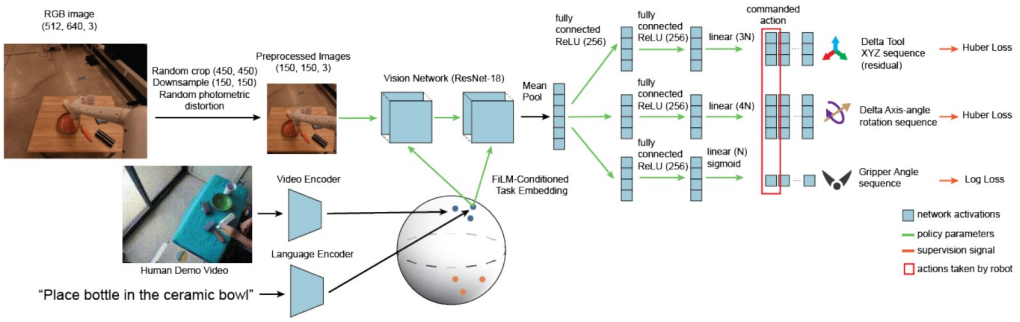
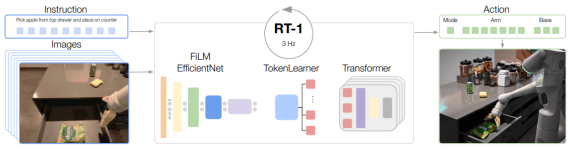


Figure 1.13: Architecture proposed in [5]

Foundational research in the field of Language-Conditioned Multi-Task Imitation Learning has demonstrated promising results in zero-shot generalization. However, the robustness of their performance remains a challenge. Subsequent research, as highlighted in [58, 55, 57], has focused on enhancing performance. In particular, the authors of [58] sought to investigate whether the transfer of knowledge from extensive, diverse, and task-agnostic datasets, which has enabled modern machine learning models to excel in zero or few-shot learning scenarios for new and specific tasks, is applicable within the realm of robotics. This inquiry arises due to the presence of high-capacity architectures capable of assimilating knowledge from such large datasets. To explore this prospect, the authors in [58] introduced a comprehensive dataset comprising over 130,000 demonstrations collected across more than 700 household tasks (as depicted in Figure 1.14a). Additionally, they proposed a Language-Conditioned Transformer-based architecture (as illustrated in Figure 1.14b).

It is worth noting the intriguing findings presented in Table 1.4. The model exhibits robustness in replicating tasks it has encountered before, and it even performs reasonably well on tasks it hasn't seen previously. However, a notable decline in performance becomes apparent when the model is exposed to novel backgrounds and scenarios featuring distracting objects, especially when the available data for these situations is limited. This observed trend holds significance because, unlike domains such as Computer Vision and Natural Language Processing where gathering large-scale datasets is relatively straight-

(a) Examples of household scenarios in RT-1 large-scale dataset



(b) RT-1 Language-Conditioned Transformer based architecture proposed in [58]

Figure 1.14: Household scenarios proposed in [58] (Figure 1.14a), architecture proposed in [58] (Figure 1.14b)

Table 1.3: Distribution of tasks in large-scale dataset proposed in [58]

Skill	Count	Description	Example Instruction
Pick Object	130	Lift the object off the surface	pick iced tea can
Move Object Near Object	337	Move the first object near the second	move pepsi can near rxbar blueberry
Place Object Upright	8	Place an elongated object upright	place water bottle upright
Knock Object Over	8	Knock an elongated object over	knock redbull can over
Open Drawer	3	Open any of the cabinet drawers	open the top drawer
Close Drawer	3	Close any of the cabinet drawers	close the middle drawer
Place Object into Receptacle	84	Place an object into a receptacle	place brown chip bag into white bowl
Pick Object from Receptacle and Place on the Counter	162	Pick an object up from a location and then place it on the counter	pick green jalapeno chip bag from paper bowl and place on counter
	9	Skills trained for realistic, long instructions	open the large glass jar of pistachios pull napkin out of dispenser grab scooper
Total	744		

forward, the collection of real-world robotic datasets is a laborious and time-consuming endeavor. Furthermore, these real-world robotic datasets often have limited applicability to other research due to **disparities in action space, robot morphology, and scene representation**, as pointed out by [58].Therefore, the aspiration is to develop a system capable of replicating tasks with minimal demonstrations specifically gathered from the particular robot in use.
Furthermore, the decline in success performance when faced with distractor objects emphasizes that addressing the policy-learning problem in an end-to-end manner, which involves mapping high-dimensional and high-level inputs like images and text directly to low-dimensional, low-level outputs such as the actions to be executed, may not be the most effective approach. This is because such models might lack the necessary perceptual components that enable them to initially recognize the target object within the scene, subsequently navigate towards it, and commence the manipulation process. This process aligns with how humans approach manipulation tasks [61].

Table 1.5: Results reported in [58] by training the same model RT-1 with different dataset size

				Generalization			
Models	% Tasks	% Data	Seen Tasks	All	Unseen Tasks	Distractors	Backgrounds
RT-1	100	**100**	97	73	76	**83**	59
RT-1	100	**51**	71	50	52	**39**	59
RT-1	100	**37**	55	46	57	**35**	47
RT-1	100	**22**	59	29	14	**31**	41

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 [62], activity [63], or completely unlabeled [49, 64]. 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 $\mathcal{L}_\psi$; (**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 [64] 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 [65] or Gaussian Process [66, 67]. 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 [68]. 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-

Algorithm 4 Classic feature matching IRL algorithm

Require: Dataset of expert trajectories $\mathcal{D} = \left\{\boldsymbol{\tau}_i^E\right\}_{i=1}^N$
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

function, i.e., a **linear** function of the features [69, 70] or a **non-linear** function of the features [71, 72, 73]. Generally, the IRL algorithms are characterized by the following problems: **(1)** the learning process could be time-consuming and impractical for high-dimensional problems, because of the double-nested optimization procedure; **(2)** the IRL problem is *ill-posed*, since there are different reward-functions that could lead to the same action.

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 [69, 71]; **(b)** *Maximum Entropy* (Max-Ent) approach [70, 72, 73]. 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,

$R(\boldsymbol{\tau}^E)$, is greater than the reward of all the alternative trajectories, $R_\omega(\boldsymbol{\tau})$, by a certain margin m, i.e., $\max_{\omega,m} m \;\; s.t. \; R_\omega(\boldsymbol{\tau}^E) \geq \max(R_\omega(\boldsymbol{\tau})) + m$. The main problem with this formulation is that it does not handle the case in which the expert behavior is sub-optimal, leading to an ambiguous notion of margin. To handle this problem the Max-Ent approach has been proposed in [70]. 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 $\max_{\psi} \mathcal{H}(\pi^{r_\psi}) \; s.t. \;\; \mathbb{E}_{\pi^{r_\psi}}[\mathbf{f}] = \mathbb{E}_{\pi^*}[\mathbf{f}]$. 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 [72], a deep-network was used to model the reward-function. In this work, the Nerual 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 [73]. In particular, [73] faced the problem of learning a cost function in a high-dimensional continuous state space, with **unknown dynamics**. Indeed, starting from the exponential trajectory distribution $p(\tau) = \frac{1}{Z} \; exp(-c_\theta(\tau))$, the main difficulty is the estimation of the partition function Z, needed to compute the negative log-likelihood loss function, $\mathcal{L}_\theta = \frac{1}{N} \sum_{\tau_i^E \in D_{demo}} c_\theta(\tau_i^E) + \log(Z)$. Since, the dynamic is unknown the idea was to estimate the partition function through the trajectories obtained from the current policy rollouts, with the hyphothesis 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 pro-

Algorithm 5 Guided-Cost-Learning Algorithm [73]

Require: Initial controller $q_k(\tau)$
  for i = 1, . . . , N **do**
    Generate D_{traj} from $q_k(\tau)$
    $\mathcal{D}_{samp} \leftarrow \mathcal{D}_{samp} \bigcup \mathcal{D}_{traj}$
    Update the 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 [74] and using $\mathcal{D}_{traj}$
  end for

posed. The algorithm [74] returns a *Linear-Quadratic Gaussian* controller q_k, such that $q_k = \underset{q}{argmin} \; \mathbb{E}[c_\theta(\tau)] - \mathcal{H}(\tau) \; s.t. \; p(s_{t+1}|s_t, a_t) = \mathcal{N}(s_{t+1}; f_{x_t} x_t + f_{u_t} u_t, F_t)$. 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 [75] tried to move IRL towards more complex learning scenario, such as learning a reward function from video-demonstrations. In [75] the architecture in Figure 1.15 was proposed. The goal was to obtain the parameters Ψ, of a cost function $C_\Psi(\hat{\boldsymbol{\tau}}, z_{goal})$, such that it was possible to derive a sequence of actions by minimizing the cost-function itself, i.e., $\mathbf{a}_{new} = \mathbf{a} - \eta \nabla_a C_\Psi(\hat{\boldsymbol{\tau}}, z_{goal})$, Different cost functions have been proposed, but all

with the same objective to reduce the distance between the current predicted keypoints and the goal keypoints configuration. The trajectory $\hat{\boldsymbol{\tau}}$ is a a sequence of predicted states $[\hat{s}_1, \ldots, \hat{s}_T]$, result of a learned dynamic model, $\hat{s}_{t+1} = f_{dyn}(\hat{s}_t, a_t)$. Experimental results have shown that it is possible to learn a cost-function starting from human/robot video demonstrations. However, the proposed setting was tested on a very simple reaching task, proving how a lot of work has to be made in order to prove the effectiveness or ineffectiveness of IRL in complex real-robot manipulation tasks, starting from video demonstrations.

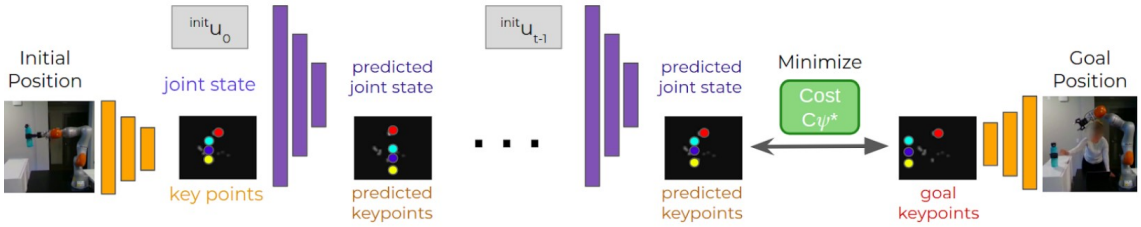


Figure 1.15: Architecture proposed in [75]

Generative Adversarial Imitation Learning (GAIL)

Generative Adversarial Imitation Learning was proposed for the first time in [76], with the idea to improve the IRL setting, which is expensive to run, because of the double-nested optimization procedure. The authors in [76], starting from a general Max-Ent formulation (Formula 1.7), obtained a characterization of the learned policy (Formula 1.8), 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 1.8 is that the $\psi - regularized$ IRL finds a policy whose occupancy measure is similar to the expert's one, measured by ψ^*. The next-step was to choose an appropriate regularization function. In particular, by choosing the regularizer in Formula 1.9, the conjugate in Formula 1.10 can be obtained, which is the classic Adversarial-Learning Loss, where the current policy π^L plays the role of GAN generator, and D is the GAN discriminator, which has to distinguish between state-action pairs generated either by the expert-policy or by the current policy.

$$IRL_\psi(\pi^E) = \underset{c \in R^{S \times A}}{arg\ max} - \psi(c) + (\min_{\pi^L \in \Pi} - \mathcal{H}(\pi^L) + \mathbb{E}_{\pi^L}[c(s,a)]) - \mathbb{E}_{\pi^E}[c(s,a)] \tag{1.7}$$

$$RL \circ IRL_\psi(\pi^E) = \underset{\pi^L \in \Pi}{arg\ min} - \mathcal{H}(\pi^L) + \psi^*(\rho_{\pi^L} - \rho_{\pi^E}) \tag{1.8}$$

$$\psi_{GA}(c) = \begin{cases} \mathbb{E}_{\pi^E}[g(c(s,a))] & if\ c < 0 \\ +\infty & otherwise \end{cases}, \quad g(x) = \begin{cases} -x - log(1 - e^x) & if\ c < 0 \\ +\infty & otherwise \end{cases} \tag{1.9}$$

$$\psi^*_{GA}(\rho_{\pi^L} - \rho_{\pi^E}) = \max_{D \in (0,1)^{S \times A}} \mathbb{E}_{\pi^L}[\log(D(s,a))] + \mathbb{E}_{\pi^E}[\log(1 - D(s,a))] \tag{1.10}$$

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

Algorithm 6 Generative Adversarial Imitation Learning Algorithm

Require: Expert Trajectories $\tau^E \sim \pi^E$, initial policy π_θ^L, discriminator D_ω
  for $i = 1, \ldots, N$ **do**
    Sample trajectories, $\tau_i^L \sim \pi_\theta^L$
    Update Discriminator, $\hat{\mathbb{E}}_{\tau_i^L}\left[\nabla_\omega \log(D_\omega(s,a))\right] + \hat{\mathbb{E}}_{\tau^E}\left[\nabla_\omega \log(1 - D_\omega(s,a))\right]$
    Update Policy π_θ, with TRPO [80], and cost-function $C(s,a) = \log(D_\omega(s,a))$
  end for

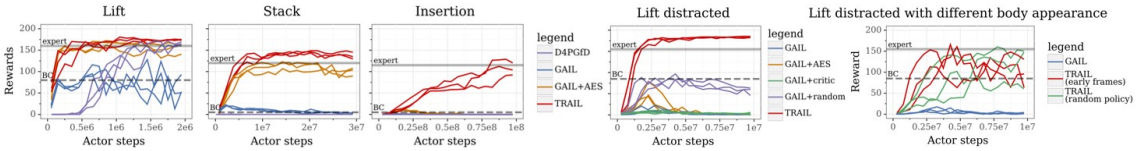


Figure 1.16: Experimental results on tasks without and with spurious features [84]

and the second related to the policy's parameter update. Since GAIL has proven to be more effective than classic IRL algorithm [70], 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 [77], or by modifying the reward function in input to the RL algorithm [78, 79]. All the cited methods have reported promising results on control tasks in a simulation environment [81], 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 [82, 83, 84, 85], it shown very poor results. In particular, with respect to the Adversarial Imitation Learning setting, works of interest are [84, 85]. In [84], 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 [84] 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 defined as $accuracy(\mathcal{I}_E, \mathcal{I}_A) = \frac{1}{2}\ \mathbb{E}_{s\in\mathcal{I}_E}\left[\mathbf{1}_{D_\omega \geq \frac{1}{2}}\right] + \frac{1}{2}\ \mathbb{E}_{s\in\mathcal{I}_A}\left[\mathbf{1}_{D_\omega < \frac{1}{2}}\right]$. In the end, the discriminator'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) [86] 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 [76] baseline, both in setting with spurious features and without spurious features (Figure 1.16).

The authors of [85] 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*, $(\hat{\mathcal{U}}_\beta, \hat{\mathcal{T}}_\beta, q_{beta})$, by maximizing the Evidence Lower Bound (Formula 1.11), where $\hat{\mathcal{U}}_\beta$ is the decoder, q_{beta} is the encoder, and $\hat{\mathcal{T}}_\beta$ is the transition model; **(2)** train a *discriminator*, D_θ, by minimizing the Adversarial Loss function (Formula 1.12); **(3)** train a *policy* π_θ^L, by maximizing the Value function (Formula 1.13).

$$\max_{\beta} \mathbb{E}_{q_\beta}\left[\sum_t \log(\hat{\mathcal{U}}_\beta(s_t|z_t)) + \mathbb{D}_{KL}(q_\beta(z_t|s_t, z_{t-1}, a_{t-1})||\hat{\mathcal{T}}_\beta(z_t|z_{t-1}, a_{t-1}))\right] \tag{1.11}$$

$$\min_{\theta} \mathbb{E}_{(z,a)\sim\rho^E(z,a)}\left[-log(D_\theta(z,a))\right] + \mathbb{E}_{(z,a)\sim\rho^{\pi^L}_{\hat{\mathcal{T}}}}\left[-\log(1 - D_\theta(z,a))\right] \tag{1.12}$$

$$\max_{\pi_\theta^L} V^K_{\theta,\beta}(z_t) = \max_{\pi_\theta^L} \mathbb{E}_{\pi_\theta^L, \hat{\mathcal{T}}_\beta}\left[\sum_{\tau=t}^{t+K-1} \gamma^{\tau-t} \log(D_\theta(z_\tau^{\pi_\theta^L}, a_\tau^{\pi_\theta^L})) + \gamma^K V_\beta(z_{t+K}^{\pi_\theta^L})\right] \tag{1.13}$$

With this learning setting the proposed system outperforms previous works such as [83, 77] both in terms of data-efficiency and overall performance, on a set of control tasks (Figure 1.17). Generally speaking, the Generative Adversarial Imitation Learning has shown very promising performance in simulated control tasks and simulated robot manipulation tasks, even in complex high-dimensional state-space. However, it is not so clear, how these methods could perform in real-world robotic manipulation tasks, in terms of data-efficiency, generalization capability, and safety during real-world interactions.

Figure 1.17: Control tasks solved in [85]. From left to right: cheetah run, walker walk, car racing, claw rotate, baoding balls

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 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 1.18). To solve this problem, in [29, 87, 88], some sort of image-to-image translation was performed, while in [89] the

Temporal-Cycle Consistency (TCC) [90] was used.
In particular, in [29, 88], the Cycle-GAN [91] 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 [87] a similar approach was proposed, but the MUNIT [92] architecture was adopted for the image-to-image translation. With respect to the mentioned works, in [89] a completely different approach has been proposed. Indeed, starting from a set of video demonstrations characterized by both different length and demonstrator embodiment, the TCC approach was used to learn an encoder able to map demonstration frame into the corresponding embodiment-independent embedding. The idea behind TCC was to learn a correspondence between frames of different videos representing the same overall action (Figure 1.19).
The correspondence problem between different viewpoints was faced by [30, 82]. In [30] a Convolutional Neural Network was trained by means of a *Triplet-Loss* [93]. 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

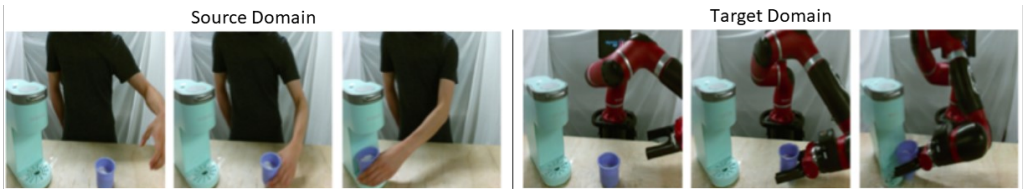


Figure 1.18: Representation of embodiment mismatch problem. (Left) The source domain represented by a video of human performing a task. (Right) The target domain, represented by the robot that executes the observed task.

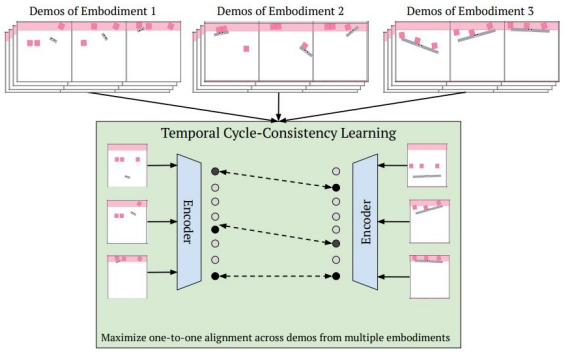


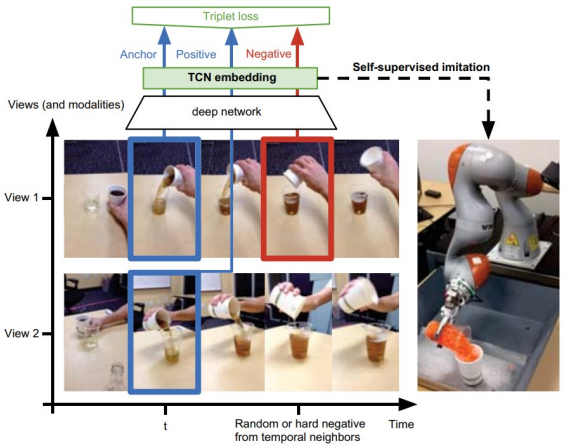
Figure 1.19: Temporal-Cycle Consistency representation, used to learn an embodiment-agnostic encoder in [89]

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 1.20a). In [82], a different approach was used, indeed, a *context translation problem* was solved by means of an Encoder-Decoder architecture (Figure 1.20b). 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 D_i comes from the source context ω_i, while samples in $\mathcal{D}_j$ comes from the target context ω_j. The model must output the observations in D_j conditioned on 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.

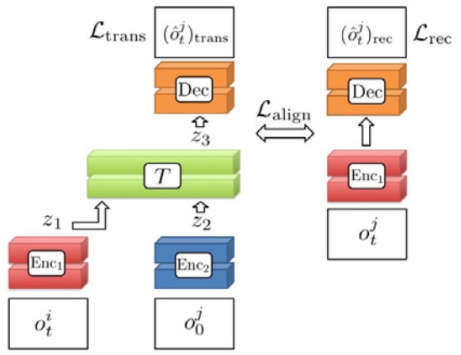
Concerning the way how the policy can be obtained, it is necessary to distinguish between Model-Based and Model-Free methods.

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 [82, 30, 87, 89], 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 [82] 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 1.20a). In [87], 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 [89], the



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



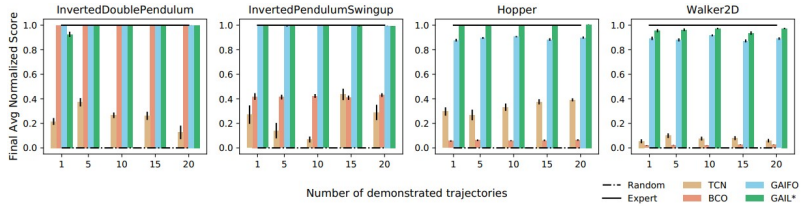
(b) Context-Translation network, proposed in [82]

Figure 1.20: Examples of how the mismatch between demonstrator viewpoint and learner viewpoint can be handled.

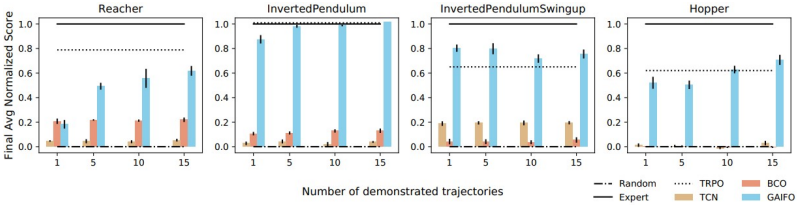
reward function was defined as $R(s_t) = -\frac{1}{k}\ ||\phi(s_t) - g||_2^2$, where g is the goal embedding, defined as the mean embedding of the last frame of all the demonstration videos in the dataset, while $\phi(s_t)$ is the embedding of the current observation. Experimental results, on simulation 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 [94, 95]. Obviously, these methods are strictly related to the Generative Adversarial Imitation Learning setting. However, with respect to the methods presented in GAIL paragraph (pag. 23), these methods do not assume access to the demonstrator action. Indeed, the goal of the preliminary work [94] 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 [95], that formalized the *GAIfO* algorithm, (Algorithm 7), that is an extension to state-only demonstration of GAIL [76]. The proposed algorithm was used to train a network to solve tasks in simulation environment [81], with both low-dimensional state representation, and visual-state representation. Results with respect to the number of demonstrated trajectories are reported in Figure 1.21. As it can be noted from the results, GAIfO outperforms previous observation based methods [30, 96], in a setting with a low number of expert trajectories. The main drawback of GAIfO is the high number of environmental interactions needed to learn a policy, since the model-free TRPO [80] algorithm was used to train the policy. This problem was solved by DEALIO [97], which replaced the model-free algorithm with PILQR [98] the model-based RL algorithm reported next.
Among the **Model-Based** methods a further classification has to be made between methods that obtain the *Inverse Dynamic Model* and methods that obtain the *Forward Dynamic Model*. The former, given in input a transition (s_t, s_{t+1}), obtain a function M, able to map state transition to action i.e., $a_t = M(s_t, s_{t+1})$. While the latter, given in input a state-action pair (s_t, a_t), aim to learn a function F able to generate the next state s_{t+1}, i.e., $s_{t+1} = T(s_t, a_t)$.
Methods that obtain the Inverse Dynamic Model are [99, 96, 100, 101]. In [99] 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

Algorithm 7 GAIfO algorithm [95]

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



(a) Experimental results in low-dimensional state space



(b) Experimental results in high-dimensional state space

Figure 1.21: Experimental results reported in [95].

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 [96] proposed a general approach, depicted in Figure 1.22, 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 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 [100], the same approach as in [96], but the agent's policy was trained according to a linear combination of Behavioral Cloning and Advantage Actor Critic (A2C) objective function [102], i.e., $\mathcal{L}_\theta^{hyb} = \mathbb{E}_{s,a}[A(s)\log(a|s;\theta) + \alpha\, \mathcal{H}(\pi^L(.|s))] + \mathbb{E}_{(\hat{s}_t,\hat{s}_{t+1})\sim D}[\log(\pi^L(M(\hat{s}_t, \hat{s}_{t+1})|\theta)]$. Here, the main problem is the assumption to have access to reward function, which can reduce its applicability in real-robot manipulation tasks. In [101], the work in [103] 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) [103], 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, 97]. 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 [104], which allows

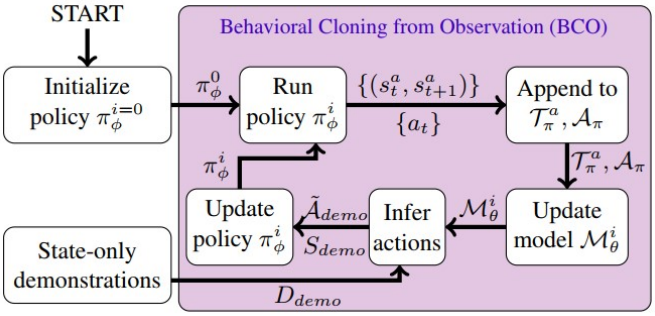


Figure 1.22: Representation of the learning procedure proposed by [96]

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 [105]. 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 [74]. Real world robotic experiments, shown that with **2 hours** of robot interaction it was possible to outperform previous works such as [30, 96] and classic BC algorithm, on tasks such as "coffee making" (Figure 1.18) and cup-retrieving (i.e., the robot has to take a cup from a closed drawer). In [97] the sample-inefficiency problem of GAIfO [95] was addressed. The idea was to exploit the adversarial learning setting with state-only demonstration, which has shown promising results (Figure 1.21), and combining it with a more data-efficient RL algorithm, such as PILQR [98]. The core of PILQR is the LQR optimization procedure. Generally speaking, it returns a *linear-gaussian controller* (Formula 1.14), that optimizes a *quadratic-cost function* (Formula 1.15), under the assumption of *linear-gaussian dynamic* (Formula 1.16).

$$\pi(a_t|s_t) = \mathcal{N}(K_t s_t + k_t, S_t) \tag{1.14}$$

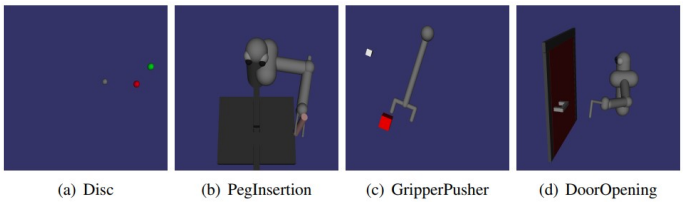
$$c(s_t, a_t) = \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T C_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T c_t + cc_t \tag{1.15}$$

$$s_{t+1} \sim P(s_{t+1}|s_t, a_t) = \mathcal{N}(F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t, \Sigma_t) \tag{1.16}$$

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

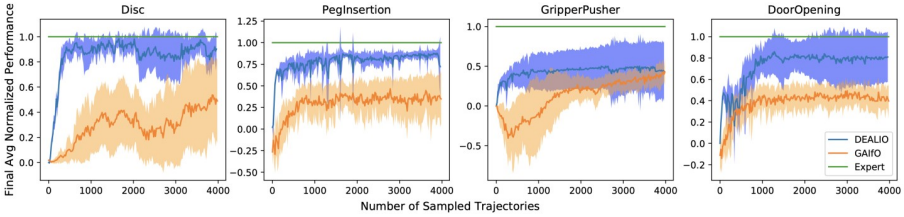
$$D_\theta(s_t, s_{t+1}) = \frac{1}{2} \begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix}^T C^{ss}(s_t, s_{t+1}) \begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix} + \begin{bmatrix} s_t \\ s_{t+1} \end{bmatrix}^T c^{ss}(s_t, s_{t+1}) \tag{1.17}$$

Experiments performed in simulation with low-dimensional state space have shown promising results (Figure 1.23b), in terms of sample-efficiency with respect to the GAIfO baseline. However, improvements would be made in order to: **(1)** reduce the variance, in order to make the learning process more reliable, **(2)** increase the overall performance, **(3)** adapt the algorithm to work with real-world robot manipulation tasks.



(a) Disc (b) PegInsertion (c) GripperPusher (d) DoorOpening

(a) Control Tasks solved in [97]



(b) Performance of DEALIO [97] compared against GAIfO [95], with respect to the number of trajectories sampled during the learning process.

Figure 1.23: DEAILO: (1.23a) Control Tasks, (1.23b) 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 environ-

ment, 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 is dedicated to the presentation of the research project. In Section 2.1, the *Research Topic* will be described. Specifically, starting from the state of the art presented in Section 1.2, which contains a comprehensive overview with respect to the Learning from Demonstration problem, the context of interest will first be identified, going on to describe the application scenario in which the proposed system will be placed. Next, the reference approach currently used to solve the Learning from Demonstration problem given the scenario of interest will be identified. Finally, based on the considerations made earlier, some research gaps will be identified, highlighting how current methods partially solve these gaps and proposing some hints on how advances can be made based on the current state of the literature. In Section 2.2 the *Research Plan* will be described. Specifically, starting from the gaps reported in the previous section, the proposed hints will be formalized, defining hypotheses of methods and procedures that can lead to their implementation, and pointing out why the proposal can improve the current literature. Next, the experimental techniques that will be followed to evaluate and compare the proposed methods will be described.

2.1 Research Topic

One of the primary objectives in robotics is to develop autonomous robots capable of performing a wide range of manipulation tasks, such as pick-and-place and assembly operations, in response to specific commands. These tasks are often characterized by some degrees of variability, which may be related to factors like object categories and positions. In the context of traditional industrial operations, manually coded control rules can effectively manage situations in which robots follow fixed and repetitive paths. This is achieved through prior knowledge of object categories and positions, within an environment where the robot's working area is known and fixed over time(e.g., the workcell in Figure 1.1). In contrast to this scenario, the contemporary landscape of social robotics and modern industrial applications demands a higher level of adaptability and flexibility. Robots are expected to work in environments alongside human operators, receiving commands and engaging in interactions with them in a collaborative or cooperative manner. For example, a robot may be tasked with picking up a tool and delivering it to the operator. This entails the robots' ability to recognize various object categories and estimate their positions but also correlate the outcomes of environmental analysis with the requested commands and

adapt their actions accordling, showing "intellingent" behaviors.

As extensively discussed in Chapter 1, the scientific community has directed its attention towards addressing these challenges by evaluating the utilization of *data-driven* approaches. These methods encompass Imitation Learning algorithms, which utilize data derived from examples of desired behaviors, often referred to as demonstrations, to equip a robot with the capability to replicate the demonstrated tasks.
Multi-Task Imitation Learning methods, as highlighted in [5, 53, 4, 58], are notably promising due to their capacity to fulfill the requirements of both high *adaptability* and *flexibility*. As extensively explained in Section 1.2.3, these methods use a **multi-task dataset** $\mathcal{D}^E = \{\mathcal{D}_1, \dots \mathcal{D}_n\}$, which contains demonstrations for n different tasks (e.g., $\mathcal{D}_1$ contains demonstrations of pick-place task, $\mathcal{D}_2$ contains demonstrations of nut-assembly task and so forth), to fit a *single control function* denoted as $\pi_\theta^L(a_t|s_t, c_{m_i})$, that is able to map the current observed state s_t and the command c_{m_i}, into the corresponding action a_t that will be executed by a phisical robot. The command c_{m_i} referes to the m^{th} variation of the i^{th} task (e.g., the pick-place task can have different variations based on the target object and the target placing spot as depicted in Figure 1.10). Tables 1.2 and 1.4 illustrate the potential of the proposed approaches, yet challenges remain in both single-task and multi-task problems. As reported in [5, 48], existing systems face difficulties in identifying critical task points, such as determining when to close the gripper near an object or open it in proximity to the placement area, rather than understanding the broader intent of the task at hand. Furthermore, performance drops occur when scenes involve distractor objects, i.e., objects that do not contribute to the task execution. This highlights the need for improved capabilities in correctly correlating commands with the results of scene analysis to identify target objects.

In the context presented so far, this research activity aims to address the problem of *Learning from Demonstration* in multi-task scenarios. Our aim is to develop a system that can make a step towards the realization of a *versatile collaborative robot* suitable for industrial applications. This robot should possess the capability to execute tasks directed by a human operator and acquire new skills based on a limited number of demonstrations, building upon its existing knowledge. To illustrate potential applications, consider a collaborative workspace where a human operator could instruct the robot to provide a tool or engage in assembly tasks.
With respect to the general formulation extensively explained in Section 1.2.3, we are interested in exploring methods that rely on *visual inputs*, i.e., the current state s_t is an image that depicts the current state of robot workspace, and the command c_{m_i} is given in terms of video demonstrations of an agent (e.g., robot or human) that executes the desired task.
In addition to the overall assessment we conducted following the literature review, it is essential to emphasize that significant modifications have been made in relation to the Research Proposal presented at the conclusion of the first year. These alterations were prompted by interesting insights obtained from preliminary experiments, which subsequently enabled us to identify a previously unexplored research gap and formulate the ensuing research question:

Reason-then-act: How the separation between perception and action can affect the performance of an Imitation-Learning System
In both *"Language-Conditioned Imitation Learning"* and *"Video-Conditioned Imitation Learning"*, the primary objective is to obtain a function π_θ^L, that can effectively map

the current state and command pair to the corresponding action. These systems need to address two key challenges:

1. **Command analysis and understanding**: The first challenge involves the analysis of the given command. Specifically, from an **high-level task command** the system should be able to understand the **intent** (e.g., picking and placing rather than assembling), identify the relevant objects (e.g., selecting the blue box rather than the red one), and recognize the mentioned actions (e.g., reach, follow, pick). In language-conditioned systems, neural language models [54, 5, 58], are commonly employed to extract and represent the semantics of the command. In image-conditioned systems, image processing and computer vision techniques (e.g., deep learning architectures [53, 4]) are used to extract relevant features from the video demonstration.

2. **Action Generation**: The second challenge is to correlate the features generated from the analysis of the observed state and the information obtained from the command, with the aim to create an intermediate representation that captures the relevant aspects from both inputs (e.g., there is a focus on the image portion that contains the target object). Techniques such as feature concatenation [51, 54, 52], attention mechanisms [53, 4], or feature-wise linear modulation [58] are utilized to combine the visual information from the state and the information from the command. The fused representation should capture the important cues for inferring the correct action based on the current input and command.

Despite the progress made with Multi-Task Imitation Learning, the performance of these agents still lacks consistent robustness when faced with challenges such as distractor objects and varying backgrounds [58]. During our preliminary experiments (Chapter 3) we tried to emphasize this problem in order to **focus our attention to scene analysis and cognitive aspects** rather than pure control problems, by testing the models in scenarios where there are objects that differ in very high-level features such as the color. Indeed, our preliminary experiments revealed a noteworthy observation. While our system consistently generates what can be referred to as "valid trajectories", i.e., it effectively carries out the actions of picking up an object and correctly placing it in the intended position, it often struggles with the critical task of identifying the precise target object. In other words, there is a recurrent issue where the selected object for manipulation is not the correct one.
The observed results can be explained through a comprehensive analysis of the underlying learning procedure. Specifically, the reference methods [53, 4, 58] propose end-to-end architectures that starting from **high-level inputs** (e.g., images and text) directly generate the required action, assuming that there is an implicit solution to the initial challenges associated with the command analysis and comprehension. These architectures are trained based on the principles of Behavioral Cloning (BC). BC offers two main approaches: one involves minimizing the disparity between the predicted and correct actions (as shown in Formula 1.5), while the other centers on maximizing the likelihood of executing the correct action (as shown in Formula 1.6). The choice between these approaches hinges on the nature of the policy, whether deterministic or probabilistic. It's important to note that all of these loss functions primarily target the predicted action, directly addressing step 2, with an implicit assumption of resolving step 1. This approach has exhibited effectiveness in simple scenarios where the robot has to learn a single task without variations [20, 50] and in multi-task settings where scenes are comprised of easily distinguishable objects [53, 4, 58]. However, in complex scenarios composed of different distractor objects, the

cognitive task becomes considerably more intricate and may be beyond resolution solely through the information derived from the action-based loss, increasing the gap that exists between the training metric and the actual goal (i.e., understand the command, genarete the action based on the current state and the command comprehension, and solve the desired task). Starting from all the considerations above, the stated problem can be faced in two ways: (1) Enhancing dataset diversity by expanding the representation of objects and employing a large-scale model capable of assimilating all the knowledge inherent in the dataset. (2) Dividing the Decision-Making Problem into two primary blocks, namely a reasoning module and an action module. This segmentation enables the independent validation of each task within the decision-making process and the separate evaluation of the performance of each block. Subsequently, a method should be devised to integrate these two components, creating a modular system capable of solving complex problems. This system would consist of interconnected modules designed to address simpler problems. In our second year of research, we opted to concentrate on a procedure aligned with the second approach. This choice was motivated by its effectiveness in assessing the various sub-tasks inherent in a decision-making problem, such as the control of a robotic system. Comprehensive details regarding the exact procedure and its formalization will be presented in the forthcoming Section 2.2.

2.2 Proposed Research Activity

After the presentation of the context of interest and the aspects to be treated in the current research proposal, this section will formalize the hypotheses reported in Section 2.1, identifying the methods of interest and the procedures that will be used to validate and compare the proposed approach against the state-of-the-art methods. Building upon the discussion presented in Section 2.1, a pertinent question emerges. In scenarios where a robot is required to perform a task based on a provided command, *"How can we enhance the robot's cognitive capabilities to accurately interpret the given command?"* This question becomes particularly relevant in light of our observation that a significant challenge lies in the system's proficiency to precisely identify the target object as specified in the command. As a result, our research centers on the task of accurately identifying the target object within the agent's surroundings when provided with a desired command.
In literature, there are recent *Imitation Learning methods*, that follow an object-oriented paradigm [106, 107, 108, 60]. However, methods like [107] primarily focus on learning the affordance of individual objects, taking in input the current observation and the desired state of a specific object. On the other hand, techniques in the field of vision-based manipulation, such as [108, 60], leverage pre-trained object detectors like Mask-RCNN [109]. These detectors identify the top-k predicted bounding-boxes that **potentially contain objects** in the scene and use this information to infer the appropriate actions. In contrast to these methods, our approach draws inspiration from Computer Vision tasks like "Vision Question and Answering" also known as "Visual-Scene Reasoning" [110]. In this task, the system is responsible for generating an answer to a given query (e.g., a textual representation of a question) based on an input image. What distinguishes this approach is its capability to **selectively focus on specific segments of the image**, guided by the content of the question, as illustrated in Figure 2.1

In conclusion, our proposal aims to explicitly address the **cognitive aspect** of task understanding by introducing a dual-stage system. The first stage focuses on command analysis and understanding, while the second stage tackles action generation based on in-

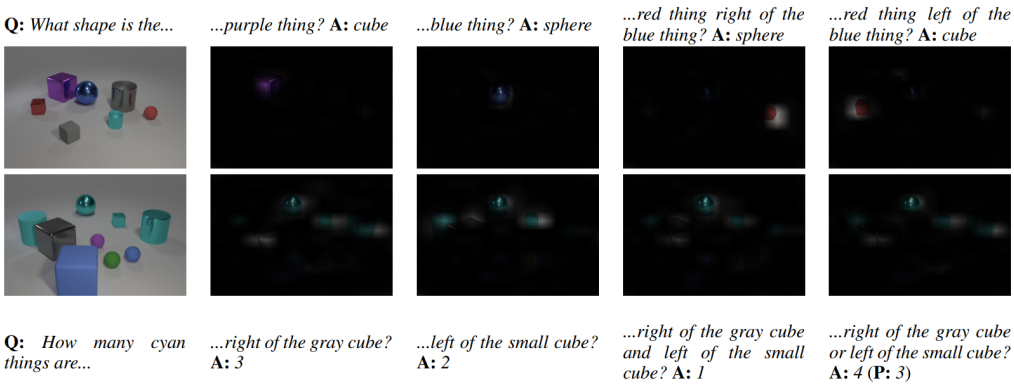


Figure 2.1: Vision Question and Answering task, starting from a an image and a text, the method is able to generate an answer by focusing on specific part of the image based on the question.

formation from the first module. Given the challenge of accurately identifying the target object, we introduce a *Conditioned-Target Object Detection task*, where the system identifies the target object location by predicting its bounding-box, $bb_t^{target} \in \mathbb{N}^4$, defined as vector of 4 integer elements $[x^{upper-left}, y^{upper-left}, x^{bottom-right}, y^{bottom-right}]$ representing the coordinates in pixel of the upper-left and bottom-right corners. The inputs are the current image observation $s_t \in \mathbb{R}^{H \times W \times 3}$ and the command $c_{m_i} \in \mathbb{R}^{T \times H \times W \times 3}$ expressed as demonstration video, i.e., $bb_t^{target} = \mathcal{F}_\psi(s_t, c_{m_i})$, where $\mathcal{F}_\psi$ is a parametrized function approximator such as Deep Neural Network. This approach should **simplify** the action-inference process, as the system benefits from **explicit low-level information** about the objects to be manipulated. The overall goal is to enhance the system's performance by breaking down the task into distinct stages, improving cognitive task understanding, and streamlining subsequent action generation. In order to validate the hypothesis that following a modular approach can affect positively the system performance, we will follow an incremental approach, divided into 4 main steps: (1) **Baseline definition**, this initial step is crucial to establish a baseline and point out critical aspects of the current state-of-the-art methods; (2) **Incorporate ground-truth information into the baseline's control module**, after assessing the baseline, the second phase focuses on empirically assessing whether adding target-object information to the control modules can enhance system performance. During this phase, to prevent *error propagation*, the control module will receive input based on ground-truth information; (3) **Development of a Conditioned-Target Object Detector System**, once the second step is completed, we can formulate the Conditioned-Target Object Detector problem and evaluate its efficacy in addressing the tasks at hand; (4) **Integration of the developed system with the baseline**: After achieving a stable system in the previous step, we will integrate the developed system with the baseline architecture to assess overall system performance

In the initial iteration, the methods will undergo training in a Single-Task Multi-Variation scenario, i.e., we consider single-task such as pick-place and nut-assembly, each task composed of different variations (Figure 1.10). Subsequently, leveraging the class-agnostic property, the system will be extended to a Multi-Task scenario. Preliminary experiment will be performed in simulation, then a real-world dataset will be collected and used to train models to be validated on a real-robot setup. Results and additional details concerning these problems will be presented in Chapter 3."

Chapter 3

Preliminary Results

This chapter is dedicated to presenting the outcomes achieved during the second year of my PhD endeavor. While the initial year was primarily focused on the comprehensive study of Machine Learning and Deep Learning fundamentals, coupled with an extensive review of the literature on Learning From Demonstration as detailed in Chapter 1, the second year marked a significant shift in focus. It was dedicated to the development of the actual research proposal, building upon the hypotheses introduced in Chapter 2.

More specifically, Section 3.1 offers an in-depth description of the dataset employed throughout the experimental phase, providing essential context for the ensuing research. Meanwhile, Section 3.2 provides a detailed exposition of the most pertinent findings and results obtained during this second year.

3.1 Dataset Description

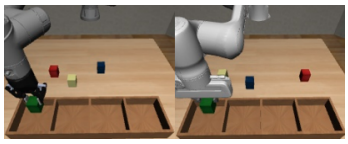
In Chapter 2, we have outlined our primary research focus, which revolves around the field of Multi-Task Imitation Learning. Specifically, we are directing our attention to methods based on video conditioning, with reference to seminal works such as [4] and [53], which have been chosen as our initial reference points due to their promising outcomes, as presented in Table 1.2. In this context, we aim to establish a robust baseline for tasks that encompass a blend of highly multimodal behaviors and fine-grained manipulations. To achieve this, we have commenced our investigations by narrowing our focus to two specific tasks: *"Pick-and-Place"* and *"Nut-Assembly"* both of which are proposed among the seven tasks in [4].

Simulated Dataset
Table 3.1 provides a summary of the dataset's cardinality, including the number of variations for each task and the number of trajectories collected for each variation. These statistics align closely with the parameters outlined in [4].

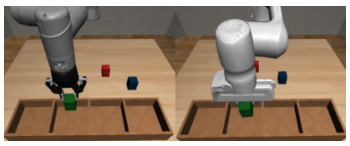
Table 3.1: Reference dataset's cardinality

	#Variations	**#Trajectories per variations**	**#Trajectories**
Pick-Place	16	100	1600
Nut-Assembly	9	100	900

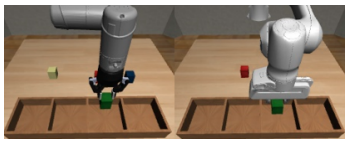
In the simulated environment, agent trajectories $\tau^j_{m_i} = \{s_0, a_0, s_1, \ldots, a_{T-1}, s_T\}$ are generated using predefined manual control strategies. The robot's actions are determined by programs that rely on accurate information provided by the environment, such as the target object and the desired placement location. During trajectory execution, various elements are recorded, including a frontal view image of the scene (see Figure 3.1a) to represent the state s_t and the robot's gripper pose to represent the action a_t. In addition to agent trajectories, we have also collected demonstrator examples under the same scenarios and cardinality as detailed in Table 3.1. However, in this case, we changed the robot from the Universal Robot UR5e to the Franka-Emika Panda. For the demonstrator, we are specifically interested in recording only frontal view images of the scene, as we use the execution video for the command c_{m_i}. Figure 3.1 and Figure 3.2 illustrate some examples of variations for each task. As it can be noted the designed scenarios show some difficulties: (•) In the dataset a given object (e.g., the red box) plays the role of both target object and distractor. This poses a significant challenge because the method cannot learn a pre-established pattern where a specific category of objects consistently assumes the role of the target while another category consistently acts as a distractor. Consequently, the demonstration video and its role in conditioning the policy become pivotal factors in addressing this challenge; (•) The objects in a given task share the same category and the same shape (e.g., in pick-place there are only box with the same dimensions). This characteristic can increase the complexity in target object localization task because methods need to acquire high-level features, such as color, to distinguish between different objects in the scene.
As discussed in Section 2.2, these factors play a pivotal role and can significantly impact the performance of the methods employed. Another critical consideration pertaining to the dataset concerns the distribution of actions. This aspect is exemplified in Figure 3.3, which illustrates the trajectory distribution along the y-axis, that is parallel to the table and most involved during task execution. It is evident that, for a given variation, the target object exhibits varying starting positions. This characteristic adds a layer of complexity to the learning task because the model must generalize to different starting position of the target object.



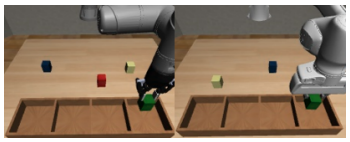
(a) First variation, green box into first bin



(b) Second variation, green box into second bin



(c) Third variation, green box into third bin

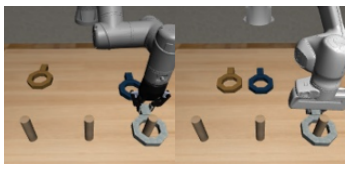


(d) Fourth variation, green box into fourth bin

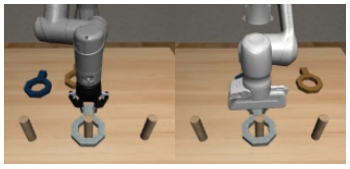
Figure 3.1: Example of variations for the Pick and Place task. The same set of variations is repeated for each block. Left the agent UR5e, Right the demonstrator Panda

Real-World Dataset and Setup

During this second year we also designed and implemented an experimental setup in a con-



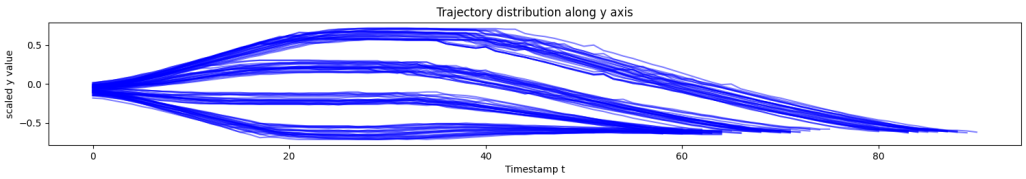
(a) First variation, assembly gray nut with right peg



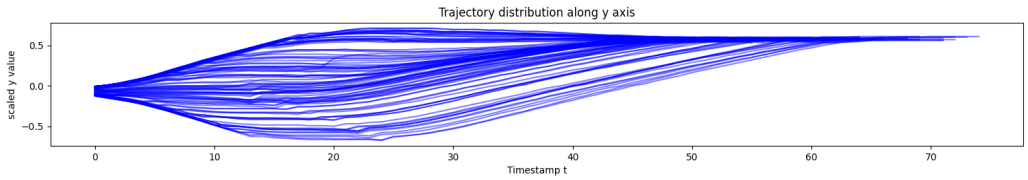
(b) Second variation, assembly gray nut with middle peg

(c) Third variation, assembly gray nut with left peg

Figure 3.2: Example of variations for the Nut-Assembly. The same set of variations is repeated for each nut. Left the agent UR5e, Right the demonstrator Panda



(a) Trajectory distribution along the y axis for the first Pick-Place variation



(b) Trajectory distribution along the y axis for the first Nut-Assembly variation

Figure 3.3: Trajectory Distribution along the y axis for the first variation of Pick-Place (3.3a) and Nut-Assembly (3.3b) task

trolled environment that allows the evaluation of the proposed methods on a real robotic platform. It is composed of: (•) The UR5e robot [111], endowed with the Robotiq2f-85 gripper [112], that will play the role of the agent; (•) 4 Zed-Mini stereo cameras [113], 1 camera is mounted on the gripper while the other three are mounted around the robot in order to have complete coverage of the workspace. Also in real-world setup we consider the pick-place and the nut-assembly problem. In this case agent trajectories are obtained by *teleoperating* the robot, and a dataset with the same cardinality as in Table 3.1 is under collection. Example of pick-and-place demonstration is reported in Figure 3.4.

(a) Frames of pick-place task from frontal camera

(b) Frames of pick-place task from lateral left camera

(c) Frames of pick-place task from lateral right camera

(d) Frames of pick-place task from gripper camera

Figure 3.4: Frames of pick-place task from different cameras

3.2 Experiments Results

In this section, we will present the most significant preliminary results achieved during the current year of our research activities. To begin, Section 3.2.1 is dedicated to establishing baseline performance. Following that, Section 3.2.2 introduces the Conditioned-Target Object Detector Problem, and Section 3.2.3 presents the outcomes of integrating the scene analysis component with the action module.

3.2.1 Baseline Definition

As first step we started by trainining the method presented in [4], using the dataset reported in Section 3.1. In order to valide the trained method we performed 10 rollouts for each variation of each task. For the pick-place task, we performed 160 rollouts and for the nut-assembly task 90 rollouts. We evaluated 3 statistics: (1) *Reaching rate*, it evaluates the ratio between the number of times the robot successfully reaches the target objects and the total number of rollouts. This metric allows us to assess the system's capacity to identify the target object and guide the robot to reach it; (2) *Picking rate*, it evaluates the ratio between the number of times the robot pick the target object and the total number of rollouts. This metric takes into account the system's proficiency in executing the picking task, as it necessitates precise control; (3) *Success rate*, it evaluates the ratio between the number of times the robot complete the given task and the total number of rollouts. This metric considers the system's proficiency in accurately identifying the task's target (e.g., the target bin in pick-place or the target peg in nut-assembly) and generating valid actions to accomplish the task.
The baseline performance are reported in Table 3.2, with respect to Table 1.2 we observe a relevant performance drop on both pick-place and nut-assembly tasks.

Table 3.2: Baseline performance

Task	**Reaching Rate** [%]	**Picking Rate** [%]	**Success Rate** [%]
Pick-Place	66.8	64.3	58.7
Nut-Assembly	40.0	38.8	36.6

For each task, we conducted both quantitative and qualitative analyses of the robot's behavior, focusing on identifying the most pertinent error cases, which are detailed in Table 3.3. It is evident from both cases that the most significant error involves the robot successfully completing the task but using the incorrect object, as illustrated in Figure 3.5. These findings underscore the system's proficiency in generating valid trajectories but highlight a deficiency in target recognition, as described in Section 2. Consequently, we initiated an assessment of the potential to separate the two tasks inherent in the decision-making process. Potential solutions are discussed in Section 3.2.2 and Section 3.2.3.

Table 3.3: Relevant error cases in pick-place task (3.3a) and nut-assembly task (3.3b)

(a) Relevant error cases in pick-place task

Error case	**#Occurrences**
Pick wrong object	**52**
Place wrong bin	6
Other	8

(b) Relevant error cases in nut-assembly task

Error case	**#Occurrences**
Pick wrong object	**45**
Fail to pick	9
Other	2

Figure 3.5: Example of error where the robot complete the task but with a wrong object

3.2.2 Conditioned-Target Object Detector

In this section, we will introduce the architectural design dedicated to the *command analysis task*. We explicitly aim to identify the target object position in the image space by predicting its bounding-box bb_t^{target}. This prediction is made through a parametrized function denoted as $\mathcal{F}_\psi$ that takes in input the current agent observation s_t and the command representation c_{m_i}, i.e., $bb_t^{target} = \mathcal{F}_\psi(s_t, c_{m_i})$. In the realm of Computer Vision, Object Detection is a well-established and extensively researched problem, with modern methods achieving high levels of performance in detection and classification tasks. However, we cannot directly apply the methods employed in previous work [60, 108]. This is because our objective is not to detect all objects in the scene, but our focus is on identifying the "target object", which can vary depending on the specific task variation at hand. Furthermore, we aim to design a method that is agnostic to object categories, meaning it doesn't need to answer the question, "What category does the object belong to?" Instead, its primary goal is to answer, "Where is the target object located?" This approach enables the method to be easily adapted to scenarios where multiple object categories are present. To address this task, we drew inspiration from the "Visual Question and Answering" problem 2.2. Our approach is rooted in the concept that, much like the architecture presented in [110] can direct attention to specific regions of an image in response to input queries, we aim for our model to exhibit a similar ability. Specifically, we want the model to focus on a particular portion of the image based on the task command c_{m_i}. The proposed architecture is reported in Figure 3.6 It is composed by the following modules: (•) *ResNet-18*

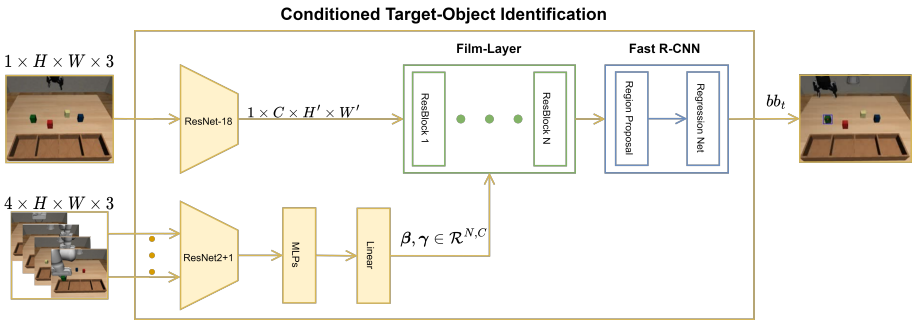


Figure 3.6: Proposed architecture for solving the Conditioned-Target Object Detector

[114], as feature-extractor for the agent observazione $s_t \in \mathbb{R}^{H \times W \times 3}$, rapresented as an RGB image; (•) *ResNet2+1* [115], as a feature extractor for the command description $c_{m_i} \in \mathbb{R}^{4 \times H \times W \times 3}$. This description is represented by a set of 4 frames sampled from the demonstrator's execution video. The chosen architecture facilitates the extraction of both spatial and temporal features. Subsequently, the generated features are flattened and processed through a Multilayer Perceptron (MLP) network; (•) The FiLM-Layer, as introduced in [110], serves as the module responsible for integrating the features derived

from both the agent and command processing stages. In detail, this layer carries out a modulation of the c^{th} activation through a feature-wise affine transformation, expressed as $FiLM(\mathbf{F}_c|\gamma c, \beta_c) = \gamma_c \mathbf{F}_c + \beta_c$. Here, γ and β represent parameters generated by the linear module, based on the desired input; (•) *Fast R-CNN* [116], is the dual-stage anchor-based object-detector, that has to infer the bounding box position based on the features maps generated by the previous module. The whole architecture has been trained according to the classic object-detector loss-function $\mathcal{L} = w_1\mathcal{L}_{reg} + w_2\mathcal{L}_{cls} + w_3\mathcal{L}_{class}$, where: (•) $\mathcal{L}_{reg}$ is the *L1-loss* function that compares the predicted bounding-box offset and the ground truth offset; (•) $\mathcal{L}_{cls}$ is a *binary-cross entropy loss* that allows the model to learn the difference between foreground and background bounding-box; (•) $\mathcal{L}_{class}$ is a cross-entropy loss designed to accommodate object classification. In our scenario, owing to the class-agnostic nature of our model, we trained it to distinguish between two object classes: *target* and *no-target*.

Table 3.4: Conditioned-Target Object Detector performance

Task	**Precision@0.5**	**Recall@0.5**
Pick-Place	0.87	0.99
Nut-Assembly	0.96	0.99

The model's performance is documented in Table 3.4. To validate the module we follow the test procedure explained in Section 3.2.1 but moving the robot with the manually defined control rules used to collect the simulated dataset. It is evident that we have achieved a highly stable detector in terms of both precision and recall. This stability is maintained throughout various phases, encompassing the initial frames when the robot approaches the target object and during the manipulation itself. For a quantitative analysis of prediction distribution in both pick-place and nut-assembly scenarios refer to Table 3.5. It's worth noting the remarkably low number of false-negative predictions, that indicate frames for which the model does not predict any bounding-box for the target object. About the false positive cases, for the pick-and-place task, the system predicts a bounding box for the target object that is completely non-overlapping with the ground truth only 192 times out of 625, and 52 times out of 186 for the nut-assembly task. In all other cases, we have an Intersection Over Union (IoU) less than 0.5, but it still indicates that the system can identify the region of the image where the target object is located, as in the second frame of Figure 3.7. These results emphasize the effectiveness of the proposed system in efficiently handling the command analysis task, which involves precise and consistent localization of the target object. Notably, even when the demonstrator and agent scenarios exhibit variations in object configuration, our system consistently demonstrates the ability to accurately detect the target object. This achievement is made possible by the establishment of a meaningfull correlation between the current agent's observations and the command, which has been a central focal point in the proposed architecture, achieved through the usage of the FiLM conditioning layer.

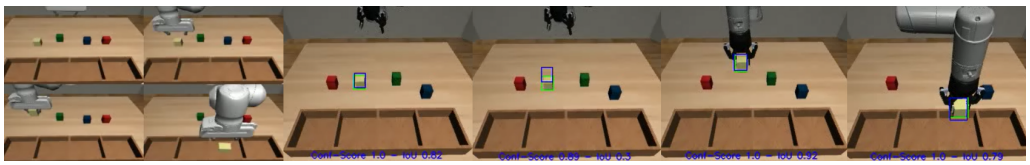


Figure 3.7: Example of predictions during trajectory execution

Table 3.5: Conditioned Target Object Detector prediction distribution

Task (#Frames)	**TP**	**FP pre-picking**	**FP post-picking**	**FN pre-picking**	**FN post-picking**
Pick-Place (11229)	9718	625	812	74	0
Nut-Assembly (5643)	5187	186	80	10	0

3.2.3 Object Oriented Multi-Task Learning from Demonstration

As discussed in Section 3.2.1, the system is capable of generating valid trajectories but struggles to select the correct object. To test the hypothesis that enhancing the control module with low-level information, such as the target object's position, would simplify the control task and lead to improved performance, we trained a model similar to the one proposed in [4], but giving in input to the control module also the information about the target-object position, rapresented by the bounding box coordinates. Consequently the novel system can be defined as a Conditioned-Policy $\pi_\theta^L(a_t|s_t, c_{m_i}, bb_t^{target})$ that generates the current action a_t based on 3 inputs: (1) The current agent observation $s_t \in \mathbb{R}^{H\times W\times 3}$; (2) The task command $c_{m_i} \in \mathbb{R}^{4\times H\times W\times 3}$; (3) The target-object bounding box $bb_t^{target} \in \mathbb{N}^4$.
Table 3.6 provides a clear comparison between the outcomes achieved using the baseline approach and those achieved with our proposed method. We specifically applied two different variations for the analysis: (1) The control module receives in input the Ground Truth (GT) bounding box; (2) The control module receives in input the bounding box predicted with the module designed in Section 3.2.2.
Table 3.6 reports the results obtained by adding the low-level bounding-box information to the control module. From these results we can draw the following preliminary con-

Table 3.6: Performance comparison on pick-place and nut-assembly tasks, adding the information about the target object position

Task	**Setup**	**Reaching Rate** [%]	**Picking Rate** [%]	**Success Rate** [%]
Pick-Place	Baseline	66.8	64.3	58.7
	Baseline with GT bb	100	96.5	76.8
	Baseline with predicted bb	**97.5**	**92.5**	**78.1**
Nut-Assembly	Baseline	40.0	38.8	36.6
	Baseline with GT bb	100.0	98.8	60.0
	Baseline with predicted bb	**38.8**	**38.8**	**26.6**

clusions: (1) Incorporating bounding-box information proves to be beneficial for system performance. When the control module is provided with ground-truth data as input, it results in a consistent ability of the system to reach the target object and effectively accomplish the task. This enhancement is reflected in improved success rates for both the pick-place and nut-assembly tasks; (2) While the introduction of the Conditioned-Target Object Detector module shows promising aspects, notably the substantial 19.4% increase in success rate for the pick-place task, it also brings to light limitations tied to what is termed "error propagation". In other words, errors originating from an upstream module can influence subsequent modules in the system. This limitation becomes evident in the nut-assembly task, where there is a notable decline in performance. In this scenario,

the prediction module encounters difficulties in accurately identifying the correct object, particularly during the initial frame. As a consequence, the robot ends up choosing the wrong object, as reported in Table 3.7b and illustrated in Figure 3.8. It's important to emphasize that we did not observe this phenomenon in Section 3.2.2. In that case, the module initially generated a false-positive prediction in the first frame. However, once the robot made its first movement toward the correct object, the subsequent predictions were accurate. This observation suggests the potential existence of a correlation between the position of the robot's arm and the predicted bounding box.

Table 3.7: Relevant error cases in pick-place task (3.7a) and nut-assembly task (3.7b)

(a) Relevant error cases in pick-place task

Error case	#Occurrences
Wrong bin	**17**
Fail to pick	9
Other	8

(b) Relevant error cases in nut-assembly task

Error case	#Occurrences
Wrong object correct peg	**30**
Wrong object wrong peg	22
Correct object wrong peg	11

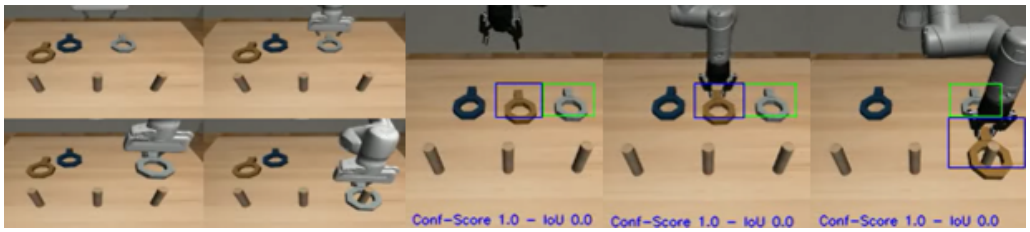


Figure 3.8: Example of error for nut-assembly task. The robot picks the wrong object due to a false positive generated by the bounding-box predictor

In conclusion, the proposed approach has initiated the validation of the hypothesis that dividing the Decision-Making Process into two stages, with explicit handling of a task associated with the Command-Analysis stage, could potentially yield a more robust system. However, there are several areas where improvements are needed, particularly concerning the Conditioned-Target Object Detector. Enhancements should focus on improving the initial frame predictions to minimize object confusion and the potential correlation with the position of the robot's arm. These improvements can be achieved through the application of appropriate data augmentation techniques, such as introducing bounding boxes that do not encompass any objects and categorizing them as "no-target", or by adding random patches within the robot's arm region. Furthermore, following this validation in a simulated environment, our next step is to extend the modules to a real-world controlled setting, leveraging the framework described and detailed in Section 3.1.

Chapter 4

Other Activities

4.1 Attended Courses

4.1.1 Compulsary Courses

The mandatory courses supported were:

- *Funding and Management of Research Project.* The course primarily emphasizes three aspects related to Research Projects: how to secure funding for a research project, how to craft a research proposal to obtain a grant, and how to effectively manage a project once the funding has been secured. First, the course introduced the European funding programs and their associated evaluation criteria. Particular attention was given to the Marie Skłodowska-Curie Actions (MSCA), specifically the Post-Doctoral Fellowship. Subsequently, the course covered project management methodologies, starting with an overview of project phases and the project life cycle, and concluding with concepts related to cost and risk management. Towards the conclusion of the course, we were tasked with composing a proposal for the MSCA Post-Doctoral Fellowship as the first part of the assignment, and a Project Management Plan and a First Year Report as the second part.

- *Exploitation of Research Results.* The course is dedicated to presenting two strategies for leveraging research outcomes: the establishment of spin-off ventures and the creation of patents. In the initial segment, the regulations and methodologies employed to initiate a university spin-off have been presented. The subsequent section delves into the processes involved in crafting a patent. As a final activity, we were asked to prepare a proposal for a potential spin-off venture.

A self-certification containing the CFUs earned during the first two years of doctoral studies can be found at the end of this file.

4.2 Other Activities

In conjunction with my primary research project centered on Robotic Learning from Demonstration, I was involved in a second research activity related to ***Graph Representational Learning*** (GRL). GRL, which is a subfield of Machine Learning and Deep

Learning, is dedicated to obtain valuable information from data structured in the form of graphs. This approach to data representation finds relevance in diverse domains, including but not limited to Recommendation Systems, Social Network Analysis, and Biological Networks, among others.

Under the supervision of Professor Pasquale Foggia and Professor Vincenzo Carletti, I directed my focus toward the topic of Anomaly Detection within the context of Internet-of-Things Device Networks. Our research aims to propose novel architectures and innovative learning paradigms to dynamically analyze such networks. The ultimate objective is to discern whether a subset of devices exhibits anomalous behaviors, an indicative sign that could potentially indicate the presence of cyber-attacks in progress. This pursuit not only enriches the field of Graph Representational Learning but also holds profound implications for the cybersecurity landscape.

4.3 Mandatory requirements

4.3.1 International Conference as presenting author and Paper as corresponding author

In the second year of our project, we did not produce any scientific publications for either conferences or journals. Nonetheless, because our initial findings presented intriguing insights, we are in the process of preparing a paper for the "European Robotic Forum 2024" (ERF 2024), scheduled to take place in Rimini, Italy from March 13 to 15, 2024. The submission deadline for scientific materials is set for November 1, 2024. In addition, we have intentions to expand and validate our current experiments, with the aim of creating another publication in reputable journals like "IEEE Robotics and Automation Letters" (RA-L).

4.3.2 Period Abroad

I will spent the period abroad next year. We are currently assessing various opportunities with tutors who can enhance my skills, focusing on areas that complement what I've learned so far. These tutors will also provide valuable feedback on my research in AI-Enabled Robotics. Specifically, we are considering the tutorship with the following professors:

1. *Alberto Sanfeliu*, director of the Artificial Vision and Intelligent System Group (VIS) at the "Universitat Polit'ecnica de Catalunya - Barcelona (Spain)" (UPC). His research interests include intelligent robotics, human–robot interactions, computer vision, and pattern recognition.

2. *Francesc Serratosa*, full professor of Computer Science at the "Universitat Rovira i Virgili - Tarragona (Spain)". He appears in the list of the 2% most influential researchers in the world, presented in "Updated science-wide author databases of standardized citation indicators". He is active in research in the areas of Computer Vision, Machine Learning and Artificial Intelligence.

3. *Xiaoyi Jiang*, full professor of Computer Science at the University of Munster - Munster (Germany). His current research interests include 3D image analysis, and structural pattern recognition.

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."

[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, "Alvinn: 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. James, M. Bloesch, and A. J. Davison, "Task-embedded control networks for few-shot imitation learning," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, vol. 87 of *Proceedings of Machine Learning Research*, pp. 783–795, PMLR, 2018.

[52] V. Bhutani, A. Majumder, M. Vankadari, S. Dutta, A. Asati, and S. Kumar, "Attentive one-shot meta-imitation learning from visual demonstration," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8584–8590, 2022.

[53] S. Dasari and A. Gupta, "Transformers for one-shot visual imitation," in *Conference on Robot Learning*, pp. 2071–2084, PMLR, 2021.

[54] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor, "Language-conditioned imitation learning for robot manipulation tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13139–13150, 2020.

[55] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 7327–7334, 2022.

[56] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu, "Do as I can, not as I say: Grounding language in robotic affordances," in *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand* (K. Liu, D. Kulic, and J. Ichnowski, eds.), vol. 205 of *Proceedings of Machine Learning Research*, pp. 287–318, PMLR, 2022.

[57] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning over unstructured data," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 4, pp. 11205–11212, 2022.

[58] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. T. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "RT-1: robotics transformer for real-world control at scale," in *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023* (K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, eds.), 2023.

[59] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*, pp. 785–799, PMLR, 2023.

[60] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: General robot manipulation with multimodal prompts," in *Fortieth International Conference on Machine Learning*, 2023.

[61] K. Grill-Spector, "The neural basis of object perception," *Current opinion in neurobiology*, vol. 13, no. 2, pp. 159–166, 2003.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

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

[73] 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.

[74] 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.

[75] 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.

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

[77] 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.

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

[79] 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.

[80] 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.

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

[82] 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.

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

[84] 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.

[85] 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.

[86] 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.

[87] 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.

[88] 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.

[89] 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.

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] 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.

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

[96] 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.

[97] 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.

[98] 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.

[99] 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.

[100] 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.

[101] 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.

[102] 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.

[103] 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.

[104] 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.

[105] 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.

[106] J. Park, Y. Seo, C. Liu, L. Zhao, T. Qin, J. Shin, and T.-Y. Liu, "Object-aware regularization for addressing causal confusion in imitation learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3029–3042, 2021.

[107] S. Belkhale and D. Sadigh, "Plato: Predicting latent affordances through object-centric play," in *Conference on Robot Learning*, pp. 1424–1434, PMLR, 2023.

[108] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Object-centric imitation learning for vision-based robot manipulation," in *Conference on Robot Learning*, pp. 1199–1210, PMLR, 2023.

[109] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[110] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[111] U. Robots, "Ur5e cobot per automazione industriale."

[112] Robotiq, "Pinze modello 2f-85."

[113] Robotiq, "Zed mini - mixed reality."

[114] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[115] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

[116] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

List of Figures

List of Tables