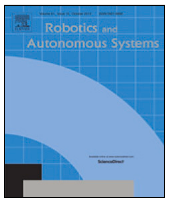
Robotics and Autonomous Systems 182 (2024) 104812

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



A survey of demonstration learning

André Correia *, Luís A. Alexandre

NOVA LINCS, Universidade da Beira Interior, R. Marquês de Ávila e Bolama, Covilhã, 6201-001, Castelo-Branco, Portugal

ARTICLE INFO

Keywords:
Machine learning
Demonstration learning
Imitation learning
Offline reinforcement learning
Behavior cloning
Learning from demonstrations

ABSTRACT

With the fast improvement of machine learning, reinforcement learning (RL) has been used to automate human tasks in different areas. However, training such agents is difficult and restricted to expert users. Moreover, it is mostly limited to simulation environments due to the high cost and safety concerns of interactions in the real-world. Demonstration Learning is a paradigm in which an agent learns to perform a task by imitating the behavior of an expert shown in demonstrations. Learning from demonstration accelerates the learning process by improving sample efficiency, while also reducing the effort of the programmer. Because the task is learned without interacting with the environment, demonstration learning allows the automation of a wide range of real-world applications such as robotics and healthcare. This paper provides a survey of demonstration learning, where we formally introduce the demonstration problem along with its main challenges and provide a comprehensive overview of the process of learning from demonstrations from the creation of the demonstration data set, to learning methods from demonstrations, and optimization by combining demonstration learning with different machine learning methods. We also review the existing benchmarks and identify their strengths and limitations. Additionally, we discuss the advantages and disadvantages of the paradigm as well as its main applications. Lastly, we discuss the open problems and future research directions of the field.

1. Introduction

The demand for intelligent systems that mimic human behavior increases. Future directions in artificial intelligence focus on replacing humans with machines that replicate the desired behavior more consistently. Notable examples include self-driving vehicles [1] and surgical robots [2]. This progress is driven by continuous advancements in the area of artificial intelligence, with increasingly complex problems being solved each year.

Emulating human behavior implies replicating a sequence of actions that a human would take in various situations. The human behavior can be recorded in the form of demonstration data sets, which are then used to train models to replicate the behavior by selecting the appropriate action based on the current state of the agent and the environment. Solving this problem entails learning the correct mapping between states and actions. This mapping is known in computer science as a policy, a function that selects an action based on the current state. Traditional programming approaches specify the action for every possible state. Moreover, such algorithms do not scale well to high-dimensional environments or continuous action spaces, because specifying the ideal action for all possible state conditions is tedious and computationally impractical. Additionally, these approaches require expertise in the specific area as well as programming knowledge, making them costly.

Because of this, machine learning (ML) techniques offer a solution to automatically learn policies from data, with Reinforcement Learning (RL) being a common method. In RL, the agent learns the policy through trial-and-error interactions with the environment. After each interaction, the agent receives feedback and adjusts its policy accordingly. The agent often learns super-human policies [3], often achieving the task's goals by performing actions that would be impossible or unlikely for a human. While this behavior can be advantageous for maximizing performance, it can be disadvantageous if the goal is for the agent to behave naturally. A clear drawback of reinforcement learning approaches is their need for a large number of interactions. Because the agent learns from failure and attempts random actions, the learning process can be unsafe in real-world environments, posing risks to the agent and its surroundings. This limitation hinders the application of RL in fields like robotics and healthcare. Additionally, reinforcement learning is highly data-inefficient, requiring many interactions to converge due to the need for exploration. Despite the significant progress and potential of deep RL, its applications remain largely confined to video games and simulations.

For these reasons, demonstration learning is an appealing alternative to reinforcement learning. In [4], the authors proposed an off-policy RL algorithm that can learn to play Atari games from image

* Corresponding author.
E-mail address: andrerspcorreia@gmail.com (A. Correia).

https://doi.org/10.1016/j.robot.2024.104812
Received 2 March 2023; Received in revised form 4 September 2024; Accepted 8 September 2024
Available online 13 September 2024

data. Two years later, AlphaGo [5] trained the first computer agent capable of beating a professional Go player. In such cases, the agent has access to a data set of interactions with the environment performed by an expert teacher. The agent learns the policy from the demonstrated state–action pairs present in the data set. The goal of demonstration learning is to learn complex policies, akin to RL, but from recorded data, similar to supervised learning. By providing examples of successful actions, the need for the agent to try incorrect actions during an exploration phase is avoided. Consequently, the agent remains safe during the learning process because it does not have to interact with the environment directly. Additionally, demonstrations ensure that the agent learns the desired behavior exhibited in the data set, allowing it to avoid local minimum and converge faster than with reinforcement learning.

The primary disadvantage of demonstration learning is that the agent is entirely dependent on the demonstrations for learning. The demonstrations must cover the state and action spaces for the agent to learn the task effectively. Depending on the task, this coverage can be difficult and expensive due to the size of both spaces. Demonstration learning methods attempt to generalize to unseen states. Although demonstration learning methods attempt to generalize to unseen states, the data is not i.i.d., making generalization challenging. If the agent encounters states outside the distribution of the data set, it is likely to fail, with potentially serious consequences in real-world applications. Therefore, the demonstration learning methods try to mitigate the distributional shift between the learned distribution and the data set's distribution, which is still an open problem. Additionally, recording demonstrations performed by a human requires capturing the state–action pairs, showcasing good behavior, and covering various scenarios. Demonstrations can suffer from sensor noise, inaccuracies, or inconsistencies in the demonstrator's actions. Consequently, demonstration learning approaches avoid directly copying the demonstrated behavior and instead attempt to generalize to non-demonstrated trajectories.

The use of demonstration data sets is a key differentiator among demonstration learning methods. Utilizing demonstrations reduces the programming overhead, making these methods accessible to non-experts. As a result, interest in this area has grown exponentially due to its significant potential. Once a policy is learned from demonstrations, it can be further refined through online interactions via RL, with the advantage that the initial policy is safer than a randomly initialized RL policy. In [3], an agent is pre-trained on demonstration data such that the convergence and its online interactions are safer. Later, [6] proposed an algorithm to learn solely from demonstrations, causing the field to gain traction. Demonstration learning is not limited to policy learning, the demonstrations can be used to learn a dynamics model, which allows the agent to collect new transitions and learn through online RL without having to interact with the environment. Alternatively, Inverse Reinforcement Learning (IRL) can utilize demonstrations to learn a reward function, which is often challenging to design in high-dimensional RL applications.

The paradigm of teaching robots through demonstrations emerged in the 1980s and has since been considered the future of robotics [7,8]. Over the years, this approach has been used to teach a wide range of tasks to various types of robots. Applications include aerial and ground navigation [9], video games [3], and controlling different kinds of robots, from manipulators [10] to humanoids [11]. The advent of deep learning has exponentially increased research interest in this area over the past two decades, leading to a significant rise in publications.

This rapid growth has resulted in numerous surveys on the topic. Early surveys reviewed the initial history of the paradigm and early attempts to teach robots from demonstrations. One such survey provided an overview of demonstration learning and defined the problem using four core questions for the field: how, what, when, and whom to imitate [12]. In a subsequent survey [13], the authors discussed various design choices and proposed a categorization for the field. Later, [14] focused on reviewing artificial intelligence methods used to estimate policies. Another survey [15] reviewed recent research and development, with a focus on demonstrating behaviors to assembly robots and extracting manipulation features. [16] provided a comprehensive review of inverse reinforcement learning. In [17], the authors gave an overview of various machine-learning methods, highlighting their advantages and disadvantages. Following this, [18,19] reviewed offline reinforcement learning methods. The timeline of these published surveys is represented in Fig. 1.

Despite the surveys mentioned above aiming to standardize terminology, the terms used in recent publications remain diverse. Terms such as demonstration learning, learning from demonstrations, imitation learning, behavioral cloning, and offline reinforcement learning are commonly used to describe the same paradigm. For consistency, this survey will use the term "demonstration learning" to refer to this paradigm. Given the rapid growth of the field, there is a need for a new comprehensive survey. This survey provides an overview of the steps required to learn from demonstrations and the various methods employed by researchers at each step. The reviewed literature encompasses a wide range of applications. The survey explains each step in a general manner, making it applicable to most tasks.

The following sections are organized as follows: Section 2 presents a formal definition of the demonstration learning problem. Section 3 discusses methods for collecting demonstration data and creating data sets. Section 4 explains the learning methods available from the demonstration data. In Section 6, we present the benchmarks available to evaluate demonstration learning methods. Section 7 lists the main applications of demonstration learning, followed by a discussion of the advantages and disadvantages of the paradigm in Section 8. Finally, Section 9 explores future research directions in demonstration learning, concluding with a summary in Section 10.

2. Problem definition

This section explains the demonstration learning problem and several relevant concepts. The overall sequence of steps in demonstration learning is illustrated in Fig. 2. The first step involves creating the demonstration data set, which requires selecting the demonstrator, the demonstration technique, and the data representation. The demonstrations can be generated by the agent itself using teleoperation, kinesthetic, or shadowing demonstration techniques. These techniques are categorized as direct demonstrations because the agent itself performs them. Alternatively, a teacher, either a human or a robot not acting as the agent, can perform the demonstrations. These are known as indirect demonstrations, as the learning agent does not perform them. In these techniques, demonstrations are recorded via observations or sensors on the teacher, with the captured information mapped to a format the learning agent can use.

Next, the agent learns through demonstration learning using the demonstration data set. The learning methods can be categorized by their objective, most commonly to learn a policy function. However, demonstration learning can also be used to learn a reward function through IRL or a dynamics model, both of which can then be applied in standard RL applications. Additionally, demonstration learning can be used to learn a generator, or a discriminator through Generative Adversarial Imitation Learning (GAIL), a sequence model, and seamlessly integrated with apprenticeship learning.

Subsequently, the learned model can be further enhanced using optimization techniques such as online interactions with RL and active learning. The evaluation of demonstration learning methods is usually quantitative, though some applications may require qualitative assessment. Each of these steps is discussed in more detail in their respective sections.

An agent is an entity that autonomously interacts within an environment towards achieving or optimizing a goal [20]. It receives information from the environment through its sensors and interacts with the environment using its actuators, based on its policy.

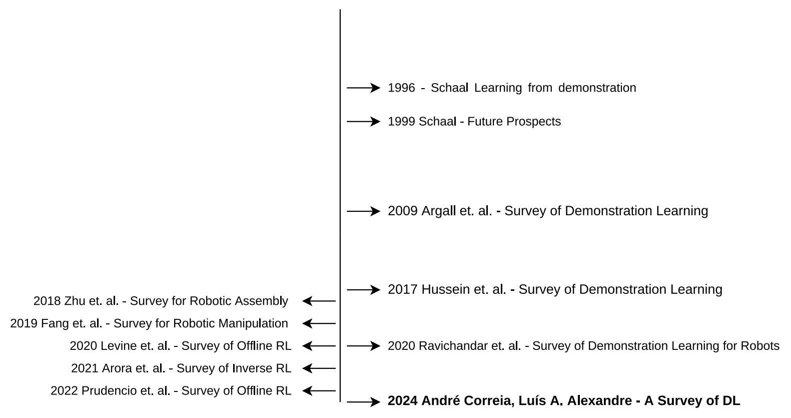

1996 - Schaal Learning from demonstration
1999 Schaal - Future Prospects
2009 Argall et. al. - Survey of Demonstration Learning
2017 Hussein et. al. - Survey of Demonstration Learning
2020 Ravichandar et. al. - Survey of Demonstration Learning for Robots
2024 André Correia, Luís A. Alexandre - A Survey of DL
2018 Zhu et. al. - Survey for Robotic Assembly
2019 Fang et. al. - Survey for Robotic Manipulation
2020 Levine et. al. - Survey of Offline RL
2021 Arora et. al. - Survey of Inverse RL
2022 Prudencio et. al. - Survey of Offline RL

Fig. 1. Timeline of surveys. General demonstration learning surveys are presented on the right side, while surveys specific to a sub-area or application are presented on the left side.

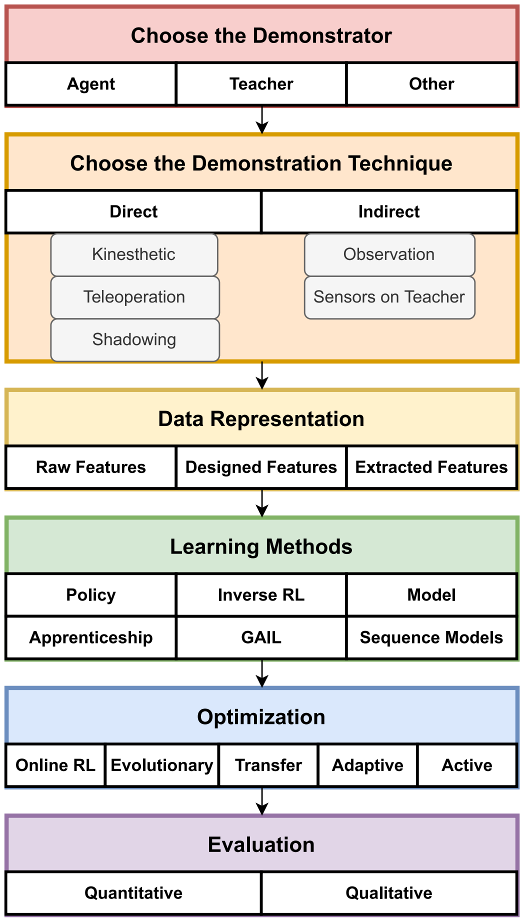


Fig. 2. Demonstration learning flowchart.

Demonstration learning is a mixture of supervised learning and reinforcement learning. In supervised learning, the agent receives the labeled training data and learns an approximation to the function that produced the data. In reinforcement learning, the agent must collect interaction data to learn from, by interacting with the environment through trial and error. In demonstration learning, the training data is a set of environment interactions collected beforehand by a teacher executing a task. The goal of demonstration learning is to learn a task from demonstrated examples performed by an expert demonstrator and recorded in a data set.

Demonstration learning expands from the RL paradigm commonly defined as a Markov Decision Process (MDP), formulated by the tuple $\langle S, A, P, \Delta_0, R, \lambda \rangle$ [21]. An MDP is a mathematical formulation that enables the creation of theoretical statements and proofs in RL, where S is the set of the possible environment states, $s \in S$, and Δ_0 denotes the initial state distribution. At each state, the agent can choose an action $a \in A$ from the set of possible actions. Acting changes the state of the environment. The mapping between states through the actions is defined by the state transition function $P(s_{t+1} \mid s_t, a_t) : S \times A \rightarrow S$. The Markov property determines that the transition function completely defines the dynamics of the environment. That is, the probability of state s_{t+1}, only depends on the current state s_t and selected action a_t, regardless of past transitions. The policy $\pi : S \rightarrow A$ is a function that selects an action given a state of the environment. A more common definition is to formulate the policy as a probability distribution $\pi(a_t \mid s_t)$, where the policy returns the probability of taking action a_t given the agent is at the current state s_t. The correct selection of the action for any given state is what allows the agent to perform the task. After interacting with the environment, the agent receives a reward $r_t = R(s_t, a_t, s_{t+1})$, which indicates the quality of the interaction. In RL, the policy is optimized to maximize the expected future rewards $\mathbb{E}[\sum_{t=0}^{\infty} \lambda R(s_t, a_t, s_{t+1})]$. A trajectory is a sequence of $H+1$ states and H actions and rewards $\tau = (s_0, a_0, r_0, \dots, s_H)$, where H is the episode's horizon, which may be infinite in the case of non-episodic environments. With these definitions, the probability density function for a given trajectory τ under the policy π is $\rho_\pi(\tau) = \Delta_0(s_0) \prod_{t=0}^{H-1} \pi(a_t \mid s_t) P(s_{t+1} \mid s_t, a_t)$. Lastly, λ is the discount factor.

In some settings, we do not have access to the full state information of the environment and have to work with observations $o_t \in O$. This formulation is named Partially Observable MDP (POMDP), defined by the tuple $\langle S, A, O, P, \Delta_0, E, R, \lambda \rangle$, where O is the set of observations and $E(o_t \mid s_t)$ is the function that maps states to observations. To overcome the limitations of learning from observations, methods combine consecutive past observations to supply the policy with time-varying information, such as velocity and direction.

One of the main reasons behind the success of machine learning methods is the usage of large data sets. However, in RL the data set is collected during training and can be expensive and unsafe to collect in the real-world.

In demonstration learning, the agent has access to a data set of N demonstrations $D_{demo} = \{\tau_i, i \in \{0, \dots, N-1\}\}$. Each demonstration is the sequence of visited states and the respective actions chosen by the expert demonstrator $\tau_i = (s_t, a_t, s_{t+1}, t \in \{0, \dots, L-1\})$, where L is the length of the sequence. The policy is estimated from the behaviors shown in the demonstration data set. The agent learns by increasingly better imitating the behavior of the teacher represented in the demonstrations. Therefore, the behavior of the agent should converge to a working and intended behavior.

Behavior cloning (BC) is the family of methods where the policy is trained to output the demonstrated action for a given state. Hence, the problem becomes a classification problem for discrete action spaces or a regression problem for continuous action spaces. However, the quality of the learned behavior is limited to the ones present in the demonstrations. Because of this, demonstrations can be used to formulate a reward function. This family of methods is called IRL. Here, the agent is rewarded for how similar the action is to the one in the data set for a given state. Alternatively, the demonstrations can include the environment rewards in addition to the states and actions: $\tau_i = (s_t, a_t, r_t, s_{t+1}, t \in \{0, \dots, L-1\})$. This family of methods is named offline RL because the agent has access to the interaction data in an offline manner. Here the ideal policy π^* is estimated by maximizing the expected accumulated reward for all trajectories τ: $\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\rho_\pi(.)}[R_\tau]$, where $R_\tau = \sum_{t=0}^{N-1} \gamma^t r_t$ is the discounted accumulated reward of trajectory τ with N transitions. Most methods estimate a state-value function to optimize the policy: $V^\pi(s_t) = \mathbb{E}_{\tau\sim\rho_\pi(.|s_t)}[R_\tau]$, which maps a state to the expected return when starting from that state. Similarly, an action-value function $Q^\pi(s_t, a_t)$, maps state–action pairs to the expected return starting from state s_t, and using action a_t.

Under offline RL, the goal is to use the data sets to generalize instead of naive imitation learning by finding the good parts of the demonstrated behavior. Even if the data set has bad behaviors, finding the good parts would result in an improvement over the demonstrations. Though distinguishing bad from good behaviors is difficult, offline RL accounts for long-term consequences of immediate actions through the value-function, unlike BC.

In the following section, we will delve into the process of creating a demonstration data set, exploring each step and the various options available. We will begin by discussing the selection of the demonstrator, followed by an examination of different demonstration techniques and data representation methods. Additionally, we will address the primary limitations that affect demonstration data sets, their potential consequences, and possible solutions.

3. Demonstration data set

The initial step in demonstration learning is creating the demonstration data set. This data set comprises a series of demonstrations, each represented as a sequence of state–action pairs. As stated previously, the data set can include extra information such as the environment's rewards for each interaction. The developer has a plethora of design options for the system. This section outlines the various choices for designing the demonstration data set, discussing their impacts on the final design, and comparing their advantages and disadvantages.

The first step is selecting the demonstrator. The next step involves choosing the demonstration technique, which depends on the type of demonstrator chosen. Afterward, defining how the data will be stored is crucial. The data must be usable by the learning agent, ideally mapping directly to state–action pairs usable by the learner. However, this direct mapping is not always feasible, and conversion mechanisms may be required. For example, learning from images requires the extraction of features, which may be manually-designed or learned. The demonstrated images are captured from the point of view of the teacher and may not be directly usable by the learning agent. Challenges that arise from the differences between the contexts of the demonstrator and the learner are named correspondence issues by [22]. All these steps are explored in the following sections.

3.1. Choosing the demonstrator

The demonstrator is the agent responsible for demonstrating the task. Two choices have to be made regarding the demonstrator: selecting who controls the demonstration and who executes the demonstration. The task can be demonstrated by either a human or a robot different from the learning agent, who controls the demonstration. However, through shadowing or teleoperation, a human can control the learning agent, allowing it to execute the task. The demonstration techniques will be explored in the following sections. In these cases, a human teacher controls the demonstration, while the learning agent executes the demonstration.

The choice of the demonstrator is critical to the success of the system, as it influences the algorithms that can be used. If the learning agent performs the demonstrations itself, for example, through teleoperation, the learner's state–action spaces will automatically align with those in the data set. However, if a different agent executes the demonstrations, the state and action spaces in the demonstrations will likely need to be mapped to the learner's corresponding spaces.

3.2. Demonstrator and learner matching

This subsection addresses the alignment of state–action pairs from the teacher, as represented in the demonstration data set, with the learner's state–action pairs, enabling the learner to perform the task. In [13], the authors defined two types of mappings: record mapping and embodiment mapping. While these terms are not widely used in the literature, we adopt them here to help classify the different demonstration techniques.

Record mapping corresponds to the mapping between the teacher's demonstrated state–action pairs and the recorded state–action pairs in the data set, denoted as $(s_{recorded}, a_{recorded}) = m_R(s_{teacher}, a_{teacher})$. Embodiment mapping corresponds to the mapping between the state–action pairs recorded in the data set and the state–action pairs performed by the learning agent, denoted as $(s_{agent}, a_{agent}) = m_E(s_{recorded}, a_{recorded})$. These mappings are represented in Fig. 3. Each mapping corresponds to applying a conversion function to the input to produce the output. Importantly, neither mapping alters the information in the demonstration data set. Instead, they change the data from the context of the teacher to the context of the agent.

Ideally, the data is recorded by the learning agent, eliminating the need for any mapping. In such cases, these mappings are equivalent to using the identity function $I(s, a)$. However, there are scenarios where the demonstrator's setting cannot be directly recorded or directly applied to the learner. In these instances, one or both conversion mappings must be created. For any given problem, each additional mapping introduces a potential source of errors. Consequently, the more mappings that are applied, the more challenging it becomes to accurately translate and reproduce the teacher's original behavior.

As an example, consider a human teacher demonstrating a task to a robot using their own body. A camera records these demonstrations as a series of images. An implicit record mapping is applied to convert the raw data captured by the camera into these images, as the images do not fully capture the entire environment. Additionally, the specific actions performed by the teacher between frames are unknown to the robot. The robot cannot infer what action was taken by the teacher to transition from one frame to the next. Therefore, an embodiment mapping is necessary to generate the state–action pairs that the robot can use to understand and replicate the demonstrated task.

3.3. Choosing the demonstration technique

In this section, we explore various techniques for acquiring demonstrations. According to [13], demonstration techniques are categorized into two groups: demonstration and imitation, based on their need for embodiment mapping. However, we adopt a simpler categorization proposed in [23], which divides demonstration techniques into indirect and direct demonstrations. In direct demonstration techniques, the learning agent itself performs the demonstration. Conversely, in indirect demonstration techniques, an external agent demonstrates the task.

In the direct category, no embodiment mapping is needed because the demonstration is performed directly by the target agent. Conversely, indirect techniques require embodiment mapping since the

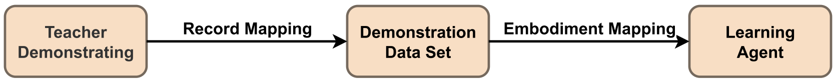


Fig. 3. Record and Embodiment Mapping as per [13].

Table 1
Categorization of the demonstration techniques.

	Direct			Indirect	
	Teleoperation	Kinesthetic	Shadowing	Observation	Sensors on Teacher
Mapping	No	No	Record	Embodiment and Record	Embodiment
Demonstrator	Learner	Learner	Learner	Not the learner	Not the learner
Data recorded	Learner's internal state	Learner's internal state	Learner's internal state	Visual	Sensor

demonstration is not performed on the learning agent. Furthermore, within each category, approaches are grouped based on whether they require record mapping. This categorization is presented in Table 1.

The demonstration techniques involve choosing how the demonstration data is provided to the learner. The most common approach is to have the data available beforehand, allowing the learner to derive a policy from a pre-existing data set. Alternatively, the data can become available incrementally during training, leading to ongoing policy updates. These incremental approaches are typically used to refine the policy based on its performance during training. For instance, in [24], the agent displays its confidence score for the selected action in a given state. The teacher can then choose to intervene and demonstrate the correct action or accept the agent's action.

[25] discusses techniques that enable a robot to refine its existing model, focusing primarily on interactive and active learning approaches. Interactive task learning, proposed in [26], involves the agent actively learning a task through natural interactions with a human instructor. The concept of modeling verbal and non-verbal cues used by the teacher during the teaching process to enhance learning is explored in [27]. Additionally, the idea of asking for help during the learning process is examined in [28], where a data-driven method was developed to estimate the human's beliefs after receiving a request and to create better requests that guide people towards providing useful help. We provide a categorization of the available demonstration techniques in Table 1.

3.4. Direct demonstration

Direct demonstration consists of techniques where the embodiment mapping is unnecessary because the learning agent performs the demonstrations. Hence, there is no need to convert the state–action pairs from the demonstrator's space to the learner's space, as they are inherently the same. However, record mapping may still be required if the state–action pairs performed by the demonstrator cannot be directly recorded in the data set.

3.4.1. Teleoperation

Teleoperation is the most direct method for transferring the teacher's behavior to the learner. In this setting, the human teacher operates the learning agent or an agent structurally identical to the learner. This agent can be a physical or simulated robot or a simulated agent, such as characters in video games. The state–action pairs of the demonstration are recorded directly from the learning agent's sensors. Because the agent performing the task is structurally identical to the learner, and the state–action pairs are extracted directly from the agent's sensors, neither embodiment mapping nor record mapping is required.

The main advantage of teleoperation is that it can be easily used in simulation environments and video games, unlike the other approaches. Additionally, teleoperation simplifies data collection, facilitating the development of new methods and benchmarks. However, a significant downside is that it requires the agent to be manually controllable, which limits its applicability to certain problems. This control can be executed through various interfaces, such as joysticks, graphical user interfaces, or virtual-reality interfaces. Another disadvantage is that not all users possess the necessary skills to teleoperate the agent effectively without extensive training. This need for technical proficiency undermines one of the key advantages of demonstration learning, which is its accessibility to a broader range of users.

Demonstrations using teleoperation have been applied to a wide variety of applications. For example, in [9,29], data from a robot helicopter flight, controlled with a joystick, was recorded and used to train an autonomous agent through RL. For manipulation tasks, [30] demonstrated how a robotic arm could be trained to change rolls on a paper roll holder from demonstrations. In [9], a humanoid robot was teleoperated using Virtual Reality technology, which translated the operator's arm and hand motions into those of the robot to generate demonstration data and develop a manipulation policy. Teleoperation has also been used in simulated environments. For instance, in [31], used teleoperation to transfer human skills in Robosoccer to a robot through demonstration learning. Additionally, [32] involved teleoperating a PR2 robot to touch a red cube on a table to create demonstration data, which was then used to train the robot to associate movements with labels and perform a sequence of trajectories based on this labeled data.

3.4.2. Kinesthetic

Kinesthetic teaching serves as an alternative to teleoperation when an external mechanism for controlling the agent is not available. In this approach, the teacher physically manipulates the agent by moving its joints through the correct positions that enable the agent to perform the task. Alternatively, the teacher can provide instructions through speech, where the robot is told specifically what to do. Similar to teleoperation, demonstration data in kinesthetic teaching is captured directly from the agent's sensors, so there is no need for any mapping. However, the quality of the demonstrations heavily relies on the capabilities of the human teacher. Even with expert demonstrators, the data obtained through kinesthetic teaching often requires post-processing techniques to refine the demonstrations.

The applications of kinesthetic demonstrations are similar to those of teleoperation. However, kinesthetic demonstrations are generally restricted to physical agents and are primarily used for manipulation tasks. For example, a learning method for collaborative and assistive robots based on imitation learning through kinesthetic demonstrations was applied to a robotic arm in various assistive scenarios [33]. To extract features from each state for kinesthetic teaching, a system that captures desired behaviors in the joint space was developed in [34]. Additionally, [35] explores how kinesthetic teaching can be used to capture demonstrations for modeling reward functions in manipulation tasks.

3.4.3. Shadowing

Demonstrations performed through shadowing are performed by the learning agent. Since the demonstration data is captured through the agent's sensors, there is no need for embodiment mapping. However, the agent performs the task by copying the movements of the teacher through some form of tracking. There is a record mapping that converts the actions performed by the teacher to the actions of the learning agent.

Shadowing has been effectively applied to both humanoid robots and navigation tasks. In [36], a humanoid robot learns to imitate a human demonstrator's arm gestures and is tested on a turn-taking gesture game. For navigation, a mobile robot learns routes demonstrated by a teacher in [37].

3.5. Indirect demonstration

Indirect demonstration encompasses techniques where an embodiment mapping is necessary. The demonstration data is not captured directly from the learning agent's sensors because a different agent demonstrates the task. Hence, the agent cannot directly apply the state–action pairs in the demonstration data set.

3.5.1. Observation

In indirect demonstration by observation, a teacher performs the task while sensors external to the learner capture the demonstrations. These sensors are often cameras or a camera system, and they may be complemented by additional sensors on the teacher. During this process, the learning agent remains a passive observer. The main advantage of this technique is its straightforward data collection process. However, it requires both record mapping and embodiment mapping to transform the captured data into usable state–action pairs for the agent. Since only image representations of the teacher's actions are recorded rather than the actual state–action pairs, record mapping is needed to convert these images into state–action pairs. Additionally, because the learning agent does not perform the task, embodiment mapping is necessary to translate the demonstration data into a format that the agent can use for learning. This is often accomplished through machine learning techniques, such as feature extraction or automated conversion from the teacher's context to the learner's context.

While indirect demonstration by observation is the simplest method for task demonstration, it necessitates both record mapping and embodiment mapping, making it less favorable compared to other techniques when they are viable options. This approach is particularly effective in settings with high degrees of freedom, where other demonstration techniques might be difficult to perform. However, it comes with challenges such as camera-related issues such as occlusions, blurriness, and noise-that can affect data quality and require extra post-processing steps.

Demonstration learning through observation is the most versatile and widely applicable demonstration technique. It has been employed in various applications, such as teaching a robot to perform assembly tasks from demonstrations in [38], learning house chores in both real-world and simulated environments as explored in [39], and manipulating a piece of cloth to create different shapes in [10]. Early works teach a robotic arm to balance a pole using demonstrations [40]. In [41], a robot demonstrates a task and transfers its skill through demonstration learning to a different robot. Moreover, demonstration learning from observation is frequently combined with other information sources beyond just camera data. For instance, [42] demonstrates how a robot learns to grasp objects by integrating visual observations with data from a force-sensing glove.

3.5.2. Sensors on teacher

Sensors on a teacher is a demonstration technique where the demonstrations are recorded from sensors on the teacher such as wearable devices [43]. In this approach, record mapping is not required because the data is captured directly from the teacher's sensors. However, because the learning agent did not perform the task, the recorded data cannot be directly used by the agent. This means that an embodiment mapping is required to convert the teacher's state–action pairs to the context of the learning agent. Because of this requirement, these approaches are used when the learning agent cannot be controlled to perform the task itself. Otherwise, teleoperation, kinesthetic, or shadowing techniques should be preferred. Human teachers are commonly used to demonstrate a task using their own bodies to train a humanoid robot. The advantage of using sensors on a teacher over passive observation lies in the precision of the data collection. Sensors provide detailed and accurate data about the teacher's actions, whereas passive observation requires indirect methods to infer and map the teacher's actions for the learner.

In [11], the data from sensors placed on a human is used to derive joint angles for a humanoid robot, enabling it to learn and perform reaching and drawing movements with one arm as well as tennis swings. This approach has also been applied in [44] to teach a biped robot to replicate human-like walking patterns. Additionally, [45] describes the use of a custom glove to capture hand position and tactile information for recording demonstration data performed directly by a human. The data was then used to obtain object model representations and optimize the policy to perform the task.

3.6. Data representation

The demonstration data needs to be recorded in a structured way. The structure is dependent on what technique was used for recording the demonstration and will determine which algorithms can be used to train the policy. The state representation is called a feature vector, which may encompass various types of information, including the agent's state, the environment, and individual objects. Due to the high dimensionality of the environment, it is often impractical to represent it in its entirety, and it may contain redundant or irrelevant information for the learning task. Hence, the selection of features needs to be adequate and efficient to convey enough relevant information to estimate a quality policy. The actions performed by the teacher are also normally included in the demonstrations. However, some approaches overcome the unavailability of actions in the data set and learn to infer the action that caused a state transition [46]. Additionally, in scenarios where the goal is to maximize a reward function, rewards for each state–action transition can also be part of the data set. Lastly, supplementary information such as episode termination indicators and safety constraint violations may be included [47,48].

3.6.1. Raw features

Raw features are the direct outputs from sensors collected during the demonstration and stored in the data set without any additional processing. Hence, these features are used in cases where there is no record mapping. However, the features may inherit noise from the sensors and may need to be pre-processed. If these features capture all the necessary information for the task, they can be used directly for training without further processing.

Such features can be easily obtained in virtual environments such as simulators or video games. For example, in [3], the agent learns to play a series of Atari games using state–action observations that are direct screenshots from the game. However, in the real-world, these features are often not readily available and typically require additional sensors to be collected.

3.6.2. Manually designed features

Manually designed features are extracted using custom-designed functions tailored to the specific problem at hand. These functions transform raw sensor data from the demonstration into a more efficient structure for training the agent. These functions filter out irrelevant and redundant information, often reducing dimensionality and focusing on the most useful features for the learning task.

In [49], a robot is trained to imitate human movement from observations. The authors define key points on the human demonstrator's body, and the agent learns to detect motions by identifying changes in the key points' locations. In [50], object positions are tracked and used as features of the demonstrations. For video games, [51] obtains screenshots of the Mario Bros game and divides them into binary cells, each signifying if the respective cell contains objects.

3.6.3. Extracted features

Extracted features are obtained from a learned function designed to process raw sensor data from demonstrations and identify relevant information for learning. Unlike manually crafted features created by experts through careful problem analysis, these functions are generated automatically through a model specifically trained for feature extraction. Typically implemented as neural networks, these models function as black boxes, extracting features in ways that are not explicitly understood by the programmer. Automatic feature extraction models are particularly useful when task-specific features cannot be identified through expert evaluation. Therefore, they offer the advantage of minimizing the need for task-specific knowledge and have broader applicability, as they can be trained to extract features for a variety of tasks. Consequently, they allow for the creation of a more streamlined demonstration learning pipeline.

In [52], deep learning techniques are employed to automatically extract features for training an agent to play a set of Atari games. Similarly, [53] demonstrates the use of features automatically extracted from observations to train a robot for various manipulation tasks. In [54], an encoder is utilized to extract features from state observations. Here, the encoder is trained simultaneously with the policy in an off-policy manner, improving sample efficiency.

3.6.4. Time as a feature

In this approach, demonstrations consist of time-action pairs. Such features can be applied to tasks where it can be assumed that the environment's state depends solely on time. Hence the agent can learn the ideal action for a specific time interval, and its behavior will always be optimal. It is assumed that there is a time loop synchronized with a fixed sequence of environmental states. The overhead of designing such features and algorithms is reduced tremendously. Furthermore, the efficiency of such policies will be high while the aforementioned time synchronization requirements remain the same. The main restriction of such features is their applicability, since rarely does the state of the environment depend solely on time.

Other limitations of time-based policies include issues with robustness and task dependency. Because these policies rely heavily on a fixed time structure, they are highly sensitive to changes in the environment that can disrupt time synchronization. Additionally, time-action pairs are typically specific to a particular task, making it challenging to adapt or scale these policies for similar tasks. To address these limitations, [55,56] explore methods for synchronizing different demonstration sequences. In their approach, frames from various demonstration sequences with the same time index are expected to exhibit similar behavior, representing the same point in the task. The model is then trained to extract features from these frames, aiming to generate consistent features for frames with the same timestamp while differentiating between features from frames at different timestamps.

3.7. Data set limitations

The performance of the policy is directly dependent on the quality of the information provided by the demonstration data set. In this section, we explore how various limitations of the data set can affect the agent's performance and outline important properties to consider when designing or utilizing a demonstration data set.

3.7.1. Incomplete data

The demonstration data set represents only a subset of the full MDP's distribution. The larger the distribution sample, the easier it is to generalize and tackle the curse of dimensionality problem. It also reduces the likelihood of encountering out-of-distribution states and dealing with the problem of distributional shift. However, collecting real-world demonstrations that adequately cover the entire MDP is often challenging and sometimes impossible, particularly in continuous state and action spaces. If certain states are missing from the demonstrations, the learner cannot estimate the optimal actions for those states during policy estimation. This section presents the ways demonstration learning approaches tackle missing data points in the data set. Human-generated demonstrations often produce narrow distributions, which can exacerbate issues related to out-of-distribution states. To mitigate these challenges, it is crucial to manage distributional shifts by ensuring that the agent does not venture beyond the data set's distribution. Additionally, another common issue is the non-inclusion or sparsity of reward signals in the data set. Creating a comprehensive reward function is often difficult, especially in complex state and action spaces. Consequently, it is sometimes easier to work with sparse or even absent rewards, though this approach can make the learning problem significantly more difficult.

The simplest idea to deal with limited data is to obtain new demonstrations. In such approaches, as the learner interacts with the system, it may encounter novel states and request a demonstration from the teacher for that given state. For instance, [57] proposes a confident execution approach, which focuses on learning relevant parts of the task where the agent identifies the need to request demonstrations. In this approach, the agent must decide between requesting a demonstration and executing actions autonomously. As the agent learns the task, it increases its autonomy, reducing both the teacher's training time and workload. In an alternative approach, [24] addresses this problem by having the agent provide the teacher with a confidence score for its chosen action. The teacher can then decide whether to intervene and provide a demonstration or to accept the agent's action [48,58].

The previous approach requires additional overhead to identify missing states in the data set and demands extra commitment from the teacher during the learning process. The alternative approach corresponds to generalizing using the available data. One way to generalize is to create new data from the existing set. Data augmentation is often used in machine learning to enlarge the data set and improve generalization to unseen data. Such techniques can be applied to state representations to generate unseen data points. In [59,60], different data augmentation schemes are compared and applied to off-the-shelf RL algorithms. However, naively applying data augmentation to RL can cause new problems. The authors of [61] identify pitfalls for naively applying transformations to RL algorithms and then teach how to properly use them. Additionally, [62] addresses the instability problem in RL by estimating the Q-values from an ensemble of agents. Another approach is to perform stitching, which involves combining portions of different unsuccessful trajectories to solve a task.

Another approach is to use transfer learning methods and learn from data of other tasks. In [63], it is shown that in certain conditions, the challenge of learning from few demonstrations for a given task can be mitigated by using demonstrations of other, related tasks. Even when rewards for the host task are either unusable or unavailable, they can be set to zero to ensure that the learning process remains effective. Furthermore, performance can be enhanced through the application of re-weighting methods to adjust the significance of transitions from these other tasks.

Alternatively, to generate new data without additional effort from the teacher, the learner can directly interact with the environment. In such approaches, the learner is pre-trained on the available demonstration data and is then fine-tuned with RL to collect further data. Since the agent starts with a solid foundation from pre-training, it is more competent and safer during exploration compared to an agent trained from scratch through RL alone. However, this method requires careful balancing between exploring new data and exploiting existing data for effective policy estimation. Additionally, it requires creating an exploration policy and a reward function that gives feedback based on the agent's actions in various states.

In [3], the agent is pre-trained on demonstration data before interacting with the environment. Then, the agent's policy is updated using both the demonstration data and the exploration data. In [64], a different approach is proposed where two policies are learned simultaneously. The primary policy executes the task, while a secondary policy enforces constraints to prevent the primary policy from taking actions that could lead to harmful outcomes. Another approach is created in [65], where a second agent is trained to make the learning of the main agent as difficult as possible. This adversarial setup creates more difficult conditions for the main agent during training, resulting in a policy that is more robust and resilient to various challenges.

Some methods explore the state space by maximizing the entropy of the visited state distribution [66,67]. In [66], a policy is trained to explore the state space while estimating the representations. The state space is clustered, and the policy is rewarded based on the distance between the visited states and the nearest cluster, thereby encouraging exploration of less-visited regions. In [67] a world model is estimated in conjunction with an exploration policy. Here, the policy is rewarded for maximizing the variance of the predictions of an ensemble of networks, which promotes exploration by favoring actions that lead to unpredictable outcomes. In [68,69], address the challenge of safe exploration by constraining the policy to ensure that it adheres to pre-defined safety restrictions.

3.7.2. *Inadequate data*

Most demonstration learning approaches assume that the quality of the data in the demonstration data set is optimal. However, this often is not the case. The demonstrated behavior can be sub-optimal, which in some cases can be intended if the goal is for the policy's behavior to appear human. Additionally, demonstration data can suffer from various issues such as sensor noise, blurriness, and occlusions. Furthermore, the data may be redundant or unevenly distributed, which can affect the learning process. To address these challenges, [70] introduces two algorithms designed to handle demonstration data corrupted by noise. These algorithms extract the idea of the expert demonstrator using Instrumental Variable Regression techniques from econometrics.

Another significant issue is data ambiguity, which occurs when there is inconsistency in the teacher's choices, resulting in different actions being selected for the same state across various demonstrations. This inconsistency means that a single state is mapped to multiple different actions in the data set.

Additionally, the data may contain unsuccessful demonstrations. When these demonstrations are appropriately labeled as unsuccessful or provide reward information that the policy aims to maximize, they can enhance the policy's robustness by guiding the agent to avoid certain state–action pairs. For example, if the policy is trained to learn from failed attempts, it can better understand what actions to avoid in similar situations. However, if unsuccessful demonstrations are mistakenly treated as successful, as often happens in BC approaches, they can degrade the quality of the learned policy. In short, the quality of the learned policy is directly affected by the quality of the data.

Some approaches leverage sub-optimal demonstrations to enhance generalization and achieve smoother behaviors in the learned policies. For example, [71], demonstrates how repeated demonstrations are used to encourage such behavior and smooth the policy. In another approach, [72] explores how data from multiple teachers can be used strategically to challenge the learning agent, resulting in a more robust policy. Some approaches identify inadequate demonstrations and choose to remove them from the data set before training the policy. For instance, [73] presents techniques for diagnosing and addressing sources of inadequacy in the demonstration data. Alternatively, [74] adopts a more nuanced approach by incorporating both successful and failed demonstrations. They separate the two types of demonstrations into clusters using an adapted version of Gaussian Mixture Models (GMM). They then perform regression using the Gaussian components from the cluster of successful demonstrations to generate improved trajectories for the learning agent.

Other solutions address inadequate data by seeking additional demonstrations from the teacher, as discussed in Section 3.7.1. Another effective strategy involves using RL to manage poor-quality data by collecting new interaction data. In this approach, the learner is first pre-trained on the available demonstration data and then fine-tuned through exploration-based methods. During this fine-tuning phase, the learner interacts with the environment and adjusts its policy based on feedback obtained either through a standard reward function or direct guidance from the teacher. For example, in [75], the authors propose a method where sub-optimal demonstrations are employed to constrain an RL algorithm. In contrast, [76] highlights the value of failed demonstrations, proposing a method that leverages these failures to train a policy. Their approach focuses on teaching the agent to avoid repeating unsuccessful behaviors observed in the failed demonstrations. In [77], an algorithm is proposed for learning policies from partially observable state environments. Alternatively, [78] choose to estimate the quality of demonstrations by estimating the competence of the demonstrator and filtering the transitions based on the competence level.

4. Learning from demonstrations

In this section, we explain the different methods available in the literature for using the demonstration data set. In general, the demonstration learning methods learn a policy or a world model. However, the demonstration data sets have also been used to learn other types of models which we will discuss.

4.1. *Learning problems*

Demonstration learning relies on the data set to learn the models. As discussed in Section 3.7, collecting the perfect data set is unfeasible for most applications, resulting in limitations that affect the learning method. To counter the limitations of the demonstration data sets, the methods should aim to generalize to regions outside the demonstrated regions in the data set. However, if the data set does not contain transitions that correspond to high-value decisions, such as those with high associated rewards, discovering these regions may be impossible. [18] argues that this challenge is insurmountable and that methods should assume that the data set contains sufficient information for developing a suitable model. To address imperfect demonstrations, methods should filter out poor demonstrations. Moreover, the method should learn to extract the beneficial parts of the demonstration, avoid the detrimental parts, and potentially combine parts from multiple demonstrations. Naive imitation in a self-supervised manner through BC risks copying bad behaviors. Therefore, methods that filter out poor demonstrations can achieve a better policy than the one represented by the data set.

Another problem with demonstration learning is its inherent paradox. Demonstration learning combines supervised learning with the transition data of RL. To improve upon the policy of the data set, the goal is to answer what are the sequences of actions that generate the

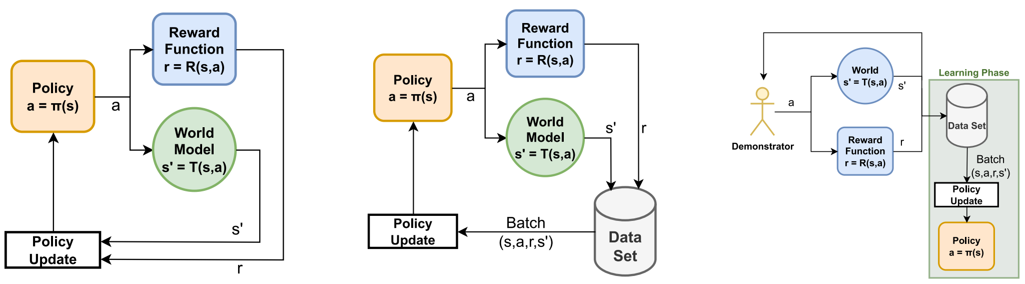


Fig. 4. Differences between on-policy reinforcement learning, off-policy reinforcement learning, and demonstration learning.

maximum reward. However, supervised learning methods assume that the data is i.i.d. The model should obtain good performance as long as the data it encounters comes from the same distribution as the one it was trained on. However, in demonstration learning, the goal is often to mimic or improve upon the behavior observed in the data set.

All these problems could be alleviated by interacting with the environment and testing uncertainties that the method may have. This is why demonstration learning is often followed by RL to refine the policy with online interactions. Pure demonstration learning is difficult because the agent cannot collect additional transitions and explore new regions. Technically, any off-policy method for online RL could be used to learn a model from the demonstration data set. However, these methods were created with the assumption that the agent could interact with the environment to correct existing errors. Demonstration learning estimates a model to perform a task defined by the state–action distribution Δ_{task}. Demonstration learning methods estimate the model using the demonstration data set which contains a set of transitions. The data set also has an associated distribution $\Delta_{demo} \subset \Delta_{task}$, which is a subset of the task's distribution. However, during deployment, the agent will likely encounter regions outside the distribution of the data set, $(s, a) \notin \Delta_{demo}$. The prediction of the model for such regions will result in larger mistakes than for in-distribution regions. Furthermore, these mistakes will accumulate and the agent will continue to diverge from the learned distribution. This snowball effect is known as the 'distributional shift'. Most policy learning methods in offline RL address the distributional shift in various ways. Some use BC to restrict the distribution to that of the dataset, which heavily limits generalization. Others propose to punish the distributional shift in the training loss by an estimation of uncertainty. Others constrain the agent to specific regions by making conservative estimates of future rewards for the Bellman update, by learning a lower bound estimation of the true value function. In [51], the authors proved that even with optimal action labels, the compound errors from distributional shift accumulate to a quadratic error in the best-case scenario. However, this error would scale linearly, if the agent was allowed to collect additional transitions. Demonstration learning methods struggle to balance generalization and avoiding the distributional shift.

4.2. Policy learning

Policy learning from demonstrations involves learning the correct mapping from states to actions from the demonstration data set. The teacher demonstrated a policy $\pi_{teacher}$ and the demonstrated state–action pairs in the data set are examples of the correct mapping. The closer the estimated mapping function is to the original mapping in the data set, the better the agent will reproduce the teacher's behavior. The agent can collect additional transitions by interacting with the environment with RL, receiving a reward, and adjusting the policy accordingly. By maintaining a history of past interactions, the agent can continuously update its policy. This approach is known as off-policy RL because the policy is updated with data collected by a previous policy. Conversely, on-policy RL updates the policy using the transitions collected by the current policy, and does not maintain a history of transitions. In demonstration learning, the agent learns from recorded data from the start, through the demonstration data set. The differences between the three settings are summarized in Fig. 4.

4.2.1. Behavior cloning

Behavior cloning (BC) is the simplest method for deriving a policy from a demonstration data set. In this approach, the policy is trained to directly imitate the teacher's actions for all states in the data set. The problem corresponds to either a classification or regression problem, for discrete or continuous action spaces, respectively. Formally, the policy is trained to minimize the error between its predicted action and the ground truth action for all state–action pairs in the data set:

$\pi^*_\theta = \arg\min_\theta(\pi_\theta(s) - a), \forall(s, a) \in D_{demo}$.

Another approach is to maximize the likelihood of actions in the demonstration:

$\max \mathbb{E}_{(s,a)\sim\mathbb{D}} log \pi(a|s)$.

However, because BC naively copies the data set, it is more reliant on the quality and size of the demonstration data set than other alternatives. The data set corresponds to a sub-set distribution $\Delta_{demo}(s|a)$ of the real distribution of states over actions for a given task, $\Delta(s|a)$. BC guarantees the agent's performance, as long as it only encounters states present in the demonstration data set. However, no such guarantees exist if the agent encounters an unseen state. In [78], the authors address the susceptibility of BC to the quality of demonstrations by estimating the competence of the demonstrator and filtering the transitions based on the competence level.

4.2.2. Offline reinforcement learning

Sometimes, direct imitation through BC is not adequate to reproduce the desired behavior and solve the task due to errors in the demonstration or poor generalization. The term offline RL is often used interchangeably with demonstration learning to describe various methods. In this context, we use offline RL to refer specifically to methods that apply RL techniques to a data set of demonstrations.

In offline RL, the agent has access to the rewards attributed by the environment to each transition. The policy is trained to maximize the expected accumulated reward $J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma R(s_t, a_t)]$, where γ is the discount factor.

In general, all RL algorithms follow the same basic train loop. The agent observes the current environment state $s \in S$, then interacts with the environment by selecting an action from its policy $a_t \sim \pi(s_t)$. This interaction changes the state of the environment to s_{t+1}, and the agent receives a reward $r_t = R(s_t, a_t)$. This process repeats for multiple interactions. The agent stores the transitions (s_t, a_t, s_{t+1}, r_t) in its memory and uses them to update the policy. In offline RL, the memory is provided by the demonstration data set D_{demo}.

Due to the limitations of BC, some approaches pre-train the agent on demonstration data and then optimize it to learn the remaining state–action space using online RL [3]. However, online RL is dangerous as some actions can lead the agent to catastrophic states which are unrecoverable in real-world scenarios. Because of this, some approaches choose to employ pure offline RL and apply regulators to reduce the impacts of distributional shift [79] or prevent the agent from going out of distribution [80].

One way to optimize the policy, parameterized by weights θ, for the Bellman objective is to estimate the gradient: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{(s_t,a_t,s_{t+1},r_t)\in D_{demo}}[\sum_{t=0}^{H} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^\pi(s_t,a_t)]$, where $Q^\pi(s_t,a_t)$ is the state–action value function.

Alternatively, we can use dynamic programming methods by first estimating the state or state–action value functions, and then using them to optimize the policy. The state value function $V^\pi(s_t)$ returns the estimated expected accumulated reward that can be obtained by starting at state s_t:

$V^\pi(s_t) = \mathbb{E}_{(s_t,a_t,s_{t+1},r_t)\in D_{demo}}[\sum_{t'=t}^{H} \gamma^{t'-t} R(s_t,a_t)]$.

The state–action value function $Q^p i(s_t,a_t)$ is similar and returns an estimation of the expected accumulated reward that can be obtained by starting at state s_t and performing action a_t:

$Q^\pi(s_t,a_t) = \mathbb{E}_{(s_t,a_t,s_{t+1},r_t)\in D_{demo}}[\sum_{t'=t}^{H} \gamma^{t'-t} R(s_t,a_t)]$.

From these definitions, we can reformulate them into a recursive form:

$Q^\pi(s_t,a_t) = R(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}\sim P(s_{t+1}|s_t,a_t), a_{t+1}\sim\pi(a_{t+1}|s_{t+1})}[Q^\pi(s_{t+1},a_{t+1})]$.

The algorithms that estimate the policy based on dynamic programming are mainly split into two families: Q-learning and Actor–Critic methods.

In Q-learning, the policy is obtained directly by estimating the state–action value function, and selecting the action that maximizes the expected accumulated reward: $\pi(a_t \mid s_t) = \arg\max_{a_t} Q(s_t,a_t)$. The Q-learning objective is defined by $Q_\theta(a_t,s_t) = R(s_t,a_t) + \gamma \mathbb{E}_{(s_{t+1}|s_t,a_t)}[\max_{a_{t+1}} Q_\theta(s_{t+1}, at+1)]$.

Actor–Critic algorithms are a mixture of policy gradients and dynamic programming because they use a policy, the actor, like policy gradients, but also use a value function, the critic, like dynamic programming. Actor–Critic algorithms learn the state–action value for the current policy $\pi_\theta(s_t)$: $Q^\pi(s_t,a_t) = R(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}\sim P(s_{t+1}|s_t,a_t), a_{t+1}|\pi(s_{t+1})}[Q^\pi(s_{t+1},a_{t+1})]$.

In early research, a set of algorithms were explored to fasten and improve RL. The authors in [81], compared eight RL frameworks, including pre-training with demonstration data, for performing a task of playing a 2D game. The authors concluded that pre-training the policy on demonstration data prevents the learner from falling in a local minimum and increases its scores. Furthermore, the improvements are more noticeable with the increase in the difficulty of the task. One notable example of the success of pre-training is the development of a RL agent capable of playing the game “Go” at a level that rivals the best human players [5]. In this application, the initial policy's weights are obtained from training on demonstration data, which are then refined through exploration using RL. In [3], the authors tackled the issue of RL being impractical to real-world issues, due to the risks associated with trial-and-error learning and the potential for severe consequences. Their approach involved pre-training the policy on demonstration data, which resulted in higher rewards during the initial learning iterations compared to standard RL methods. This demonstrates that pre-training on demonstration data leads to safer and more effective exploration strategies.

Additionally, a policy trained from online interactions can then demonstrate correct interactions. These demonstrations can then be used to transfer the knowledge to another agent. This approach bypasses the need for a human teacher by learning the policy entirely through trial and error, and then recording the successful interactions as demonstrations. It is especially valuable in situations where real-time policy updates are not possible [52].

4.2.3. Classification

In this context, classification refers to the process of assigning a specific action class to a given input state. This approach is used when the action domain is discrete, consisting of a finite set of predefined individual actions. For example, in a 2D platformer game where the agent can only walk left or right or jump, the inputs are categorized into these three actions. The policy's performance is evaluated by how often it attributes the correct action for any given input state.

Formally, the policy is a classifier $\pi(s)$, used to predict the action class a of an observation s. Where $a \in A, A = \{a_1,\ldots,a_n\}$ is a finite set of actions.

Classification methods can be applied to different levels of complexity ranging from low-level actions to complex behaviors. For instance, Bayesian networks were used in [82] for navigating an environment and avoiding obstacles. In [83], the authors created mapped representations of the environment and used a k-nearest neighbors algorithm to select the robot's actions. In [57], a GMM was used for classifying actions in navigational problems. Additionally, [84] evaluated four different classifiers for cooperative tasks in robot soccer.

More recently, neural networks have become the preferred classifiers due to their role as universal approximators, capable of modeling complex functions. For example, [85] employs Recurrent Neural Networks (RNNs) to train a robotic arm on manipulation tasks using demonstration data collected in a simulation environment. The RNN learns to predict trajectories in real-time, considering both the current position of the end-effector and the objects in the environment. Neural network classifiers are particularly effective for video games, where the number of possible actions is finite. In [3], the policy is represented by a neural network classifier with 18 neurons in the final layer, each corresponding to one of the 18 possible actions.

Some works have proposed ways of discretizing continuous action space [86]. This allows any discrete RL algorithm to be applied to the continuous state problem.

4.2.4. Regression

In this context, regression involves selecting a set of scalar values that define an action based on a given input state. These techniques are used when the action domain is continuous. For example, in the control of a robotic arm, each action can be defined by the robot's joint angles. The policy's performance is evaluated by comparing the estimated action values with the ground-truth action values in the data set for the given state.

Formally, the policy is a regressor $\pi(s)$ which maps a state $s \in S$ to actions $a \in A$, where each action is defined by a finite set of continuous values $a = \{a_1,\ldots,a_n | a_k \in \mathbb{R}\}$. Typically, regression approaches are applied to low-level motions and not high-level behaviors because high-level behaviors are a combination of low-level motions and are more likely to be discretized.

A traditional regression technique is Locally Weighted Regression (LWR), which is well-suited for learning trajectories composed of sequences of continuous values. In [87], a robotic arm is trained to execute a trajectory enabling it to perform manipulation tasks. Locally Weighted Projection Regression (LWPR) extends the previous approach to cope and scale with the input data's dimensionality and redundancy. [24] uses LWPR to teach a robot to perform basic soccer actions.

Similar to classification tasks, recent works commonly use neural networks for regression due to their ability to represent any function. In [55,56], a robotic arm is trained from demonstrations to perform manipulation tasks. The action is determined by a neural network, where the number and values of the last layer's neurons correspond to the number of joints of the robot.

Other approaches specify the type of task the agent can perform within the network. In [88], a robotic arm is trained to pick and place blocks. The network that outputs the actions has four neurons: the first two specify the position and rotation of the end-effector for picking up a block, while the other two are for placing the block. Therefore, the two groups of neurons are used separately.

In [89], an algorithm is proposed to convert discrete actions in the demonstration data set into continuous ones. An encoder is trained for this purpose to promote behavioral and data-distributional relations in the features. Then, an off-the-shelf algorithm can be used to train a

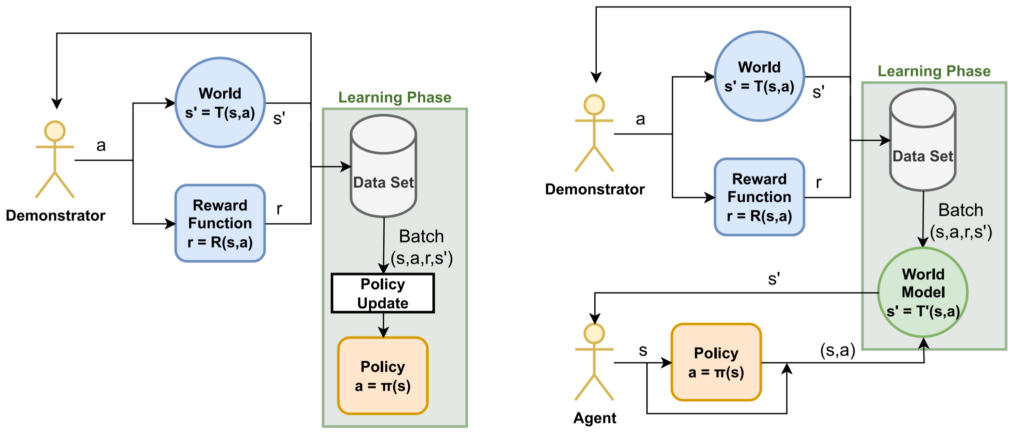


Fig. 5. Differences between policy learning and model learning from demonstrations.

policy using the new data set. However, because the policy outputs feature embeddings, the actions cannot be directly applied to the task environment. Therefore, the output of the policy is discretized by finding the action whose embedding is closer to the output of the policy.

4.3. Model learning

Model-based methods learn the dynamics of the environment by estimating the transition function $\psi(s_t, a_t) \sim P(s_{t+1} \mid s_t, a_t)$. This estimated transition function serves as a proxy for the real environment, allowing the agent to collect new transitions without directly interacting with the environment, remaining safe during the learning process. In standard RL, the agent must interact with the environment to collect transition data that represent the dynamics. In demonstration learning, the transition function can be estimated from the demonstration data set. These differences are represented in Fig. 5. The functions are typically estimated through standard supervised regression, using the states and actions as inputs and the next states as the desired output: $\mathbb{L}_\psi(s_t, a_t, s_{t+1}) = \|s_{t+1} - \psi(s_t, a_t)\|$. Model-based learning methods from standard RL can be used to learn from demonstrations [67, 90]. Standard online learning algorithms can be applied with minimal modification to train a model from demonstrated data.

However, because the policy learns from transitions simulated by the model, its performance is dependent on the quality of the estimated model, which in turn relies on the quality and coverage of the data distribution in the data set. In standard RL, the models can correct mistakes in the estimations by collecting new transitions. Similarly to policy learning, if the model is estimated solely from demonstrations, it can suffer from the distributional shift problem. In fact, the model can suffer from distribution shift regarding the true state distribution, and the true action distribution.

The distributional shift can cause the model to be exploited by the policy. The policy is optimized to maximize the expected accumulated rewards and may use the model to produce out-of-distribution states. Because these states are out-of-distribution, the model's predicted values are likely incorrect and may have an associated higher reward than the true state in the real MDP. Consequently, the policy learns to maximize erroneous transitions, leading to a worse performance once deployed to the real MDP.

A theoretical analysis presented in [90] formulates the bounds on the error between the learned policy and the policy in the data set, attributing these errors to distributional shifts in both the policy and the model. The methods for reducing the distributional shift in policy learning can also be applied to model learning. The main way to reduce the distributional shift problem in model learning is by learning an auxiliary model $\mathbb{U}(s, a) : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$, that punishes the reward function such that the agent avoids states outside the distribution: $R'(s, a) = R(s, a) + \mathbb{U}(s, a)$. Model-learning methods usually measure the uncertainty using an ensemble of models. In [91], the method only punishes the reward function if the disagreement of the ensemble is above a threshold. Alternatively, [92] adopts a pessimistic approach by selecting the maximum prediction uncertainty of the ensemble. In both approaches, the policy is penalized for visiting states where the model is likely to be incorrect. However, measuring uncertainty is challenging and often unreliable. To address this, [93] proposes learning the policy by generating a new data set from transitions produced by each of the models of an ensemble to counter uncertainty. Another approach to reduce the distributional shift without quantifying uncertainty is to use a regularizing term. For example, in [94] a model-based version of the Conservative Q-Learning (CQL) [79] algorithm is proposed.

Instead of learning the policy within the model, the learned model can be used to evaluate the policy without interacting with the environment. For instance, [95–97] utilize the model to estimate the expected return of the trajectories generated by the policy. In [98], a model is trained from demonstrations to estimate a task from images, which is particularly challenging due to their high dimensionality.

4.4. Inverse reinforcement learning

Reward functions map a state transition to a reward value based on the quality of the interaction: $R(s, a) : S \times A \rightarrow \mathbb{R}$. They define the objectives for the agent by guiding the learning process to maximize the expected accumulated reward. Traditionally, the reward functions are handcrafted by the programmer, which involves designing a function that assigns a reward value to each state–action pair. However, this task becomes increasingly challenging in high-dimensional domains, where covering the entire state space can be difficult and often results in sparse rewards. Transitions where the agent receives no feedback, through a reward of zero, hinder the convergence of the policy to an optimal one and sometimes may prevent convergence altogether. The requirement to create a reward function that covers the entire task, limits the applicability of learning algorithms to problems where a reward function can be easily specified.

An alternative is Inverse Reinforcement Learning (IRL) [99], also known as reward shaping. In IRL, the demonstration data set is used to infer a reward function, which is then employed to train a policy using standard online RL methods to maximize the expected accumulated reward defined by this function. IRL thus broadens the applicability of task learning models and reduces the manual effort required by programmers when demonstrations of the task are available. The

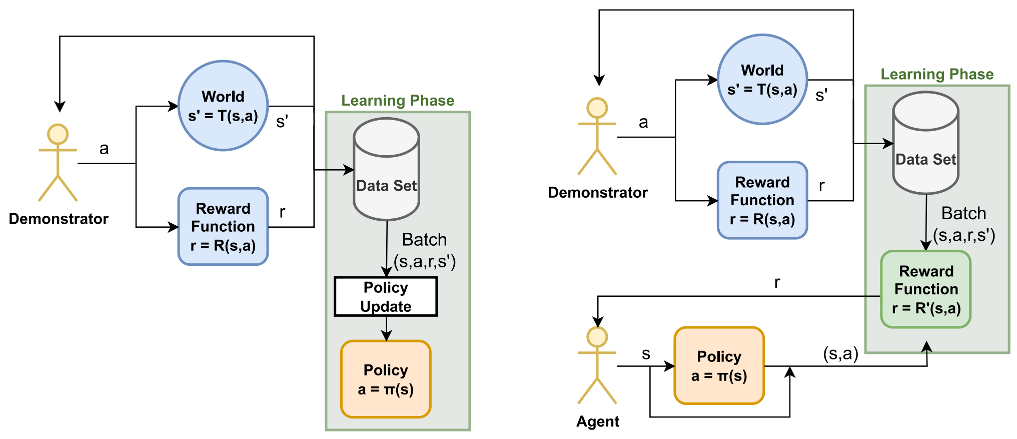


Fig. 6. Differences between policy learning and reward learning from demonstrations.

differences between policy learning and reward learning from demonstrations are illustrated in Fig. 6. [20] points out that the reward function is more transferable than a policy. While minor changes to the task can make a policy ineffective, such changes have a far less significant impact on the reward function. Typically, the learned reward function only needs to be extended to accommodate new states, rather than completely redesigned.

Demonstration learning assumes that the teacher follows a policy $\pi_{teacher}$ which is maximizing a reward function $R_{teacher}(s,a)$ when demonstrating a skill. The idea of IRL is to estimate the underlying reward function from the demonstrations. Formally, we have an MDP without the reward function, $MDP\backslash_R$, and a demonstration data set with N demonstrated trajectories $D_{demo} = \{\tau_i\}^N$, where each trajectory is a sequence of L state–action pairs $\tau_i = \{(s_j, a_j)\}^L$. The goal is to create an estimate $\hat{R}$ of the reward function that best describes the demonstrated behavior. In essence, IRL inverts the RL problem: rather than learning an optimal policy from demonstrations, potentially using the logged reward (s,a,r), IRL seeks to explain the demonstrated behavior by estimating the corresponding reward function.

IRL should estimate a reward function that generalizes from the demonstrated behavior. Hence, like other demonstration learning methods, IRL seeks to address the question: What happens if the agent were to perform a trajectory different from those in the data set? This is important because if we want the learning agent to improve upon the behavior seen in the data set, the agent must execute a trajectory that is different than the ones in the data set. However, most ML algorithms assume that the data is independent and identically distributed (i.i.d.). Consequently, addressing this question is challenging due to the problem of distributional shift.

Additionally, there can be many solutions to the reward function that describe the same behavior resulting in ambiguity. Some of these solutions, such as one that always returns the same reward, might accurately describe the observed behavior but be practically unusable. Due to this ambiguity, it is important to determine how to measure the performance of the estimated reward function. If the true reward function is available, we can directly measure the error between the predicted rewards for each state–action pair and the ground truth rewards. Alternatively, we can estimate a value function from the learned reward function and compare it to the real value function. However, the true reward function is often not accessible, which is precisely why IRL is employed in the first place. A more general way of measuring performance is to estimate a policy from the learned reward function, and assess its performance using the demonstration data set. The limitation of this method is the problem of how to evaluate the policy. Interacting with the environment is not possible because the true reward function is not available. Hence the only policy evaluation metric is to compare the policy predictions with the actions of the demonstrator for every state in the data set. However, this comparison is limited because even if the policy is only wrong in a single state, it can still result in compound errors at deployment. As a result, no single metric in IRL fully captures the performance of the learned reward function without access to the ground-truth. Furthermore, the design of the reward function's structure and the choice of its parameters are non-trivial. Using too many parameters can lead to overfitting and hinder generalization, while too few parameters may prevent the policy from effectively converging.

To obtain a unique reward function, IRL methods define additional optimization goals, the most common of which are maximum margin and maximum entropy. In the maximum margin setting, the reward function is the one that maximizes the difference between the best policy and all other policies. For example, [100] employs a maximum-margin-based IRL method to develop a policy for navigating rough terrain. In contrast, the maximum-entropy approach aims to find a distribution of policies that maximizes the entropy subject to certain constraints, such as feature matching, to ensure that the task's goals are reached. For instance, [101] uses the maximum entropy framework to learn a reward function for a driving task where there are multiple routes for the same destination in the demonstrations. The approach is later expanded to use deep learning in [102] for a table tennis task.

For discrete action spaces, IRL can be formulated as a classification problem, where for each state–action pair, the action is seen as the label for the state. The direct way to obtain a reward function is by estimating the action-value function which we explain in Section 2. This approach was used by [103] and later by [104]. However, this method assumes that the demonstrated state–action pairs are optimal.

Another approach to estimate the reward function involves assigning higher rewards to states encountered in the demonstrations, or similar states, than to states not found in the data set. For instance, in [35], the authors use demonstrations to estimate a Hidden Markov Model (HMM) which determines the associated reward for each state. Similarly, [40] applies this approach to the task of balancing a pole using a robotic arm. In [105], the authors explore three different methods for parameterizing reward functions from demonstrations and applied them to reaching, picking, and placing tasks. In [55,56], the reward is proportional to how close the images captured by the learning agent at a certain timestamp are to the respective frame of the demonstration video.

Some approaches employ an actor–critic algorithm, where a third-party critic defines the reward function and provides feedback on the actor's actions. For example, [64] describes a method where the critic's policy is trained simultaneously with the actor's policy. Initially, the critic imposes constraints to guide the actor's behavior, but as the learning process progresses and the actor's competency improves, the decision-making authority gradually shifts from the critic to the actor.

Reward functions obtained with IRL, can encourage the achievement of sub-goals or milestones during the task execution, that are represented in the demonstrations. In [106,107], the authors explore the intersection between RL and demonstration learning. Their results in two simulated domains show that reward-shaping methods can be more sample-efficient and robust against sub-optimal and inconsistent demonstrations than transfer learning algorithms.

4.5. Other learning methods

In this section, we discuss methods that complement or refine the previous learning methods to achieve greater accuracy, generalization, or robustness. Learning from demonstrations alone may not be sufficient to learn the task for all scenarios due to the limitations of the data set discussed in Section 3.7. Interacting with the environment allows the agent to collect extra data that it may use to refine the model.

The data set is unlikely to have a demonstration for all the possible environment states, particularly in high-dimensional spaces such as those encountered in real-world tasks. Hence, during the learning process, the agent must aim to generalize beyond the provided demonstrations. However, the agent may still not be able to generalize due to either limitation of the data set or the learning method. Generalizing in demonstration learning is especially difficult because the demonstrations are sequences of interactions where each action depends on the history of previous interactions, violating the i.i.d. assumption of supervised learning [58]. During training, the agent learns a sub-set distribution of the real task distribution. As a result, the agent learns from a subset of the real task's distribution, and during inference, it may encounter out-of-distribution states where it has no prior knowledge, potentially leading to unsafe actions. As discussed previously, some methods try to reduce this issue by explicitly reducing the distributional shift. However, constraining the agent to only operate within the demonstrated state distribution can limit the agent's performance. Consequently, methods that refine the agent's model through additional interactions with the environment, allow the agent to learn missing information. Such an agent is safer than a random agent with no task knowledge performing random interactions.

4.5.1. Reinforcement learning

RL models the problem as an MDP, as does demonstration learning. Instead of learning from a pre-existing data set of environment interactions, the RL agent interacts with the environment using its current policy and receives rewards based on these interactions. RL starts with a random policy and tunes its parameters towards maximizing the expected accumulated rewards. This approach can also refine parameters of a policy learned from demonstrations. An agent initialized with a demonstration-based policy is safer and converges faster, avoiding the risk of local minimum. This advantage was shown in [3] to learn Atari games. By exploring and learning from new state space regions, the agent enhances its generalization capabilities and robustness. However, when encountering unknown states, the agent may make mistakes, which can have catastrophic real-world consequences. Hence, such algorithms should be equipped with safety mechanisms. RL can also be used to train a policy from scratch in an online manner, which can then be used to generate the demonstration data set to train and evaluate the demonstration learning methods. This approach is limited to environments where online RL is possible and primarily serves to automatically generate demonstrations to evaluate the performance of demonstration learning methods. If online RL is viable, there is no need to train a secondary policy with demonstration learning. However, training a second policy can be beneficial if the RL agent does not act in real-time [52].

One of the most well-known application occurred when [5] trained an agent to play 'Go' to the extent of beating human experts. In this case, the agent is initially trained using demonstrations and then further refined through RL. In [108], RNNs are used to deal with POMDPs by incorporating past information to guide decision-making. Here, the agent is trained with RL, and demonstrations are used to determine which memories to retain.

4.5.2. Evolutionary algorithms

Like RL, optimization algorithms can be employed to learn or refine a policy to replicate a behavior. Evolutionary Algorithms (EA), inspired by natural animal behaviors, are popular optimization methods used to find solutions to various problems.

EAs can be used to generate trajectories, with the most common being Particle Swarm Optimization [109] and Ant Colony Optimization [110]. These algorithms, inspired by the behaviors of birds and ants respectively, aim to find optimal solutions within a search space. They have been extended with demonstrations to improve the learning process. In [31], EAs are used to optimize agents in a soccer simulation. The possible solutions, represented as chromosomes consisting of if-then rules, were derived from demonstration data. These solutions were then evaluated using a performance-measuring function, with the best-performing ones progressing to the next generation. In [111], PSO was utilized to find optimal behaviors, where demonstrations defined the initial behavior. Each particle modified its behavior by observing better-performing particles, with performance assessed by a fitness function. Additionally, in [112], Preference based Policy Learning was employed to teach a robot to navigate.

4.5.3. Transfer learning

Transfer Learning (TL) is a paradigm that leverages knowledge acquired from training on one task to facilitate learning a second task. Instead of training the second task from scratch, the knowledge from the first task can serve as a starting point, an optimization step, or, in rare cases, to perform the task completely. Formally, given a task T_s learned in the MDP domain D_s, the idea is to improve the learning of the goal task T_g in the MDP domain D_g using the knowledge of the previous task. TL is beneficial because it reduces the need to gather new samples; they can either be directly used by the new task, or the knowledge gained from training a policy on the original data can then be used to train the new task.

In demonstration learning, the demonstrations recorded for one task can be used to learn a second task. For instance, [113] employ transfer learning to enhance reward shaping (IRL). Reward shaping depends on prior knowledge, and transfer learning can leverage the knowledge of a policy learned for one task to shape rewards for a similar task. Additionally, [114] discovered that even if the agent overfits on the previous task, it can still adjust its weights sufficiently to recover and converge to the optimal weights of the second task. Their approach involves using graphs to identify previously encountered games and applying the relevant knowledge to the current game.

Alternatively, policies learned for a task can advise a learner on another similarity task. This knowledge can be transferred in the form of useful feature representations and specific parameter values. Moreover, an existing policy can serve as a foundation for developing a new policy for a different task. In [115], TL was used to learn a new soccer skill after learning a different one in a simulation. The experiments demonstrated that TL reduces the convergence time and achieves better performance.

4.5.4. Adaptive learning

Demonstration learning algorithms must consider that demonstration data sets are often incomplete, missing regions of the state space. A common approach to address this issue is to use pessimistic or conservative methods, which keep the policy close to the regions covered by the data set and avoid significantly different behaviors. However, these approaches can lead to sub-optimal estimations due to their strong restrictions. Additionally, agents can become stuck in certain states and repeat the same action over and over. To address this, policies should be adaptable, correcting poor choices. In [116], the authors train uncertainty-adaptive policies that incorporate a belief parameter, estimated from the interaction history using an ensemble of networks. After a failed interaction, the history changes, altering the belief value. Consequently, this new belief value prompts the agent to select a different action, preventing it from getting stuck in states.

4.5.5. *Active learning*

Active learning is a paradigm where the learning agent can query an expert for guidance, which is particularly useful in demonstration learning when the demonstration dataset is limited. When the agent encounters a state not represented in the demonstration data set, it might fail to choose the correct action, potentially leading to unsafe outcomes. Active learning addresses this issue by enabling the agent to request additional demonstrations from the teacher.

The approach to update the policy using both demonstration data and the teacher responses requires selecting which option to choose from at any given state. This can be achieved through a confidence score, as demonstrated in [24]. If the confidence for a given state–action pair is low, the learning agent queries the teacher for guidance. The learning agent progressively increases its confidence scores while obtaining a generalized policy, reducing the need for teacher queries over time. However, the main drawback of this approach is the additional investment required from the teacher, which may be unfeasible in some cases. In [117], the authors use active learning in human–robot cooperative tasks. For successful cooperation, the robot must be able to adapt its behavior to complement the human counterpart. Active learning is used after each round of interactions, with expert feedback provided via a graphical interface. The expert's feedback is provided by a graphical interface, recorded, and added to a database. This feedback is recorded, added to a database, and subsequently used to update the robot's policy. Results indicate that the robot's policy converged more smoothly using this method, particularly in tasks such as standing-up and assisted walking.

4.5.6. *Generative adversarial imitation learning (GAIL)*

The authors of [46] introduced a model-free demonstration learning method called GAIL, which adapts the Generative Adversarial Network (GAN) [118] framework to the demonstration learning paradigm. In GAILs, the reward function is learned from the demonstration data and then used in RL for learning the policy. GANs consist of two neural networks: a generator and a discriminator. The generator creates synthetic data points, while the discriminator has to distinguish between the synthetic data and real data from the data set. The discriminator is trained to correctly identify whether each data point is real or generated, and the accuracy of these classifications adjusts the weights of both networks. The generator seeks to deceive the discriminator into misclassifying generated data as real, and it is rewarded when it succeeds. Conversely, the discriminator aims to avoid being fooled and is rewarded for accurately classifying data.

In GAILs, the learned policy π serves as the generator in the adversarial setup, while the discriminator D_ϕ is responsible for determining whether a given state–action pair originates from the demonstration data set or is produced by π. The goal of π is to improve its behavior to more closely replicate the demonstration data, thereby generating trajectories that deceive D_ϕ. To achieve this, D_ϕ is trained as a binary classifier to differentiate between real state–action pairs from the demonstration data and fake pairs generated by π. The generator π is rewarded for successfully confusing D_ϕ, and treating this reward as if it were an external analytically-unknown reward from the environment through RL.

In [119], the authors present two algorithms: one designed for offline GAIL and another for online GAIL. These algorithms improve upon existing state-of-the-art methods by enhancing the efficiency and effectiveness of the GAIL framework. In [120], a discriminator is trained to distinguish between two data sets with significant differences in quality. The discriminator is then used as a filter for the policy to avoid learning from sub-optimal data. In [121], a policy is trained to perform multiple small skills, where each skill is represented by a discriminator, a replay buffer, and a demonstration buffer. Each discriminator learns to differentiate between descriptors built from a pair of consecutive states sampled from either the replay or demonstration buffer. The reward is higher when the policy fools the discriminator into thinking the consecutive states were demonstrated.

Similar to GANs, GAILs suffer from severe sample inefficiency, which hinders the agent's ability to learn effectively from a limited number of interactions with the environment. This challenge has been addressed in subsequent research, such as in [122]. In [123,124], a discriminator is used to distinguish between generated and demonstration state–action pairs to learn multiple similar tasks at once. This method facilitates the generalization of the learned policy to additional, contextually similar tasks.

Similar to GAILs, [125] introduces a zero-sum game framework where a second player acts as an antagonist to perturb the transition probabilities of the protagonist. In this setup, the antagonist operates with a perturbation budget designed to optimize the protagonist's policy against the worst-case alpha percentile of transitions, thus providing safety guarantees. More recently, [126] proposes to use adversaries in place of the critic in actor–critic algorithms to improve sample efficiency.

4.5.7. *Embedding space*

Learning from visual states requires applying a function $f(s)$ that extracts a set of N values, known as features, from the observations: $f(s) : S \to \mathbb{R}^N$, where N is the dimension of the embedding space. With deep learning, these features, and the corresponding embedding space, are typically estimated by applying a set of convolutions and subsampling operations to the input images. In demonstration learning, the states from the demonstration data can be used to explicitly learn an embedding space to extract features with specific characteristics.

Contrastive learning is a self-supervised learning paradigm that compares different images sharing a common signal to learn robust representations. It has been applied to multiple ML fields, with a notable example being image classification [127]. In such applications, contrastive learning creates an embedding space where images from the same class are pulled closer together, while images from different classes are pushed apart. A linear classifier [128] is then trained on top of the embedding space for a few epochs to classify images based on these learned features. In [129], this approach is adapted to demonstration learning, where demonstrations captured from multiple camera view points are used to learn a viewpoint-invariant embedding space. This learned embedding space facilitates the development of a viewpoint-invariant policy, thereby enhancing the policy's robustness to changes in camera position and different perspectives.

In [130], the representation learning part is decoupled from policy estimation, where the embedding space is estimated by contrasting images that appear close to each other in a sequence of frames. In [131], the different views are obtained through transformations applied to the original image. Alternatively, in [132] the representations are obtained by contrasting the similarity between the sequence of actions required to reach each contrasting state. In [55,56], an embedding space for view-invariant features is estimated from a multi-view data set through triplet learning. Similarly, in [133] the authors train an encoder to estimate similar features for concurrent frames of multi-view synchronized videos. However, the criterion here is cycle consistency, where for two views, a data point is cycle-consistent if the nearest neighbor of its nearest neighbor is the point itself.

Siamese networks [128,134] have been paired with contrastive learning, where each network is responsible for extracting features from different views of the same data. In [134], the different views are obtained through data augmentation and applied to motion simulation tasks. Other forms of obtaining different views from a single image are by using different image channels. For example, [135] converts images to the Lab color space, where the L and ab components are treated as two views of the image. Then contrastive loss is applied to learn an embedding space. Their findings indicate that increasing the number of views, such as by using additional channels or data augmentations, improves the quality of the learned features.

In [66,136], exploration is performed to estimate an embedding space without task-specific returns. In this approach, the embedding space is initially learned through exploration, and later refined for specific tasks using the reward functions of those tasks. The core idea is to encourage the embeddings to capture meaningful skills by maximizing the mutual information between state transitions and the associated embedding. This is achieved through a contrastive loss that approximates the lower bound of the mutual information, ensuring that the learned embeddings are informative about state transitions. For exploration, the agent is trained to maximize rewards that are proportional to the entropy of state transitions, which fosters exploration of diverse state regions. This two-phase process allows the agent to develop a robust embedding space through exploration and then fine-tune it for particular tasks to optimize performance.

In [137], the authors use bisimulation to generate an embedding space where functionally identical states from different tasks are mapped to the same embedding. Using this embedding space, the learning agent is trained to adapt to different tasks that are functionally identical to previously learned tasks.

4.5.8. *Sequence models*

Sequence models learn from a series of transitions instead of a single transition, leveraging the history of past interactions to make more informed decisions By considering the sequential nature of data, these models utilize past experiences to improve their predictions and reduce the likelihood of deviating from the intended distribution of actions. In sequence models, the objective is to optimize the model over entire trajectories $\pi(\tau)$, by finding the best distribution of actions over these trajectories.

Sequence modeling with deep networks has evolved from Long-Short Term Memory (LSTM) architectures to Transformer architectures with self-attention [138]. The latter have revolutionized many Natural Language Processing (NLP) tasks. Recently, they have been applied to RL by re-formulating it as a sequence modeling problem [139,140]. These treat RL as a supervised learning paradigm that predicts action sequences from trajectories and task specification (e.g., target goal or returns), instead of traditionally learning Q-functions or policy gradients. In the Decision Transformer (DT) [139], the agent is conditioned on past trajectories and the accumulated reward to be collected in the future, the returns-to-go (RTG). While the DT has demonstrated success across various tasks, its reliance on a fixed RTG sequence can limit its effectiveness in stochastic environments where the reward structure is not predetermined and must be specified by the user [141]. This requirement for manual reward specification can be challenging and may impact performance. In contrast, the Trajectory Transformer [140] employs the Transformer model both as a policy and as a model of the environment.

Subsequent research has proposed several methods to address issues with the DT. In [142], online learning is used to train the Transformer. Alternatively, [143] pre-trains the Transformer on large corpus of text which in turn increases performance on seemingly unrelated tasks. To address the DT's dependency on the RTG, several alternative methods have been proposed. One notable approach is presented in [144], where a value function is trained using the demonstration data set to replace the RTG sequence with state value predictions. While this method mitigates issues related to RTG sequences and environmental stochasticity, it introduces a dependency on the quality of the value function, which is constrained by the available demonstration data. Other works target the stochasticity problem of the DT. The method in [145], aims to estimate environmental stochasticity using a Transformer model to aid policy learning of the main Transformer.

While Transformers have achieved remarkable success due to their self-attention mechanism, their scalability is constrained by quadratic scaling relative to the size of the context window. In contrast, Structured State Space Models (SSSM) [146] have gained attention for their linear scalability with respect to the sequence length. Notably, the Mamba architecture [147] merges the context-dependent reasoning of Transformers with the linear scalability of SSSMs through its selection mechanism. Mamba has demonstrated superiority over Transformers in numerous sequence processing tasks [148]. Like the Transformers before it, Mamba has potential to impact demonstration learning applications.

4.6. *Multi agent*

Cooperation between agents is useful for robotic tasks and has been explored in RL. In RL, multi-agent systems often explore how agents can collaborate to achieve shared goals. Despite its importance, it has not received the same attention in demonstration learning. The shift from single-agent to multi-agent settings introduces significant complexity, as it requires not only learning individual policies but also managing interactions among multiple agents.

The state space must be expanded to include the status of all agents, as each agent's decisions are interdependent on the states of the others. Consequently, the reward function in these systems must also reflect the collective or individual goals of the agents. In a cooperative setting, the reward function is designed to encourage agents to work together to maximize a shared cumulative reward. Conversely, in a competitive setting, the reward function is structured so that one agent aims to maximize its own reward while strategically minimizing the rewards of other agents.

In [31], the team of robots works together to prevent the opposing team from scoring in a soccer game. In this approach, all robots share a common policy, which is updated collectively based on the actions of any individual agent. This method, while effective, essentially mirrors single-agent learning techniques. Alternatively, in [84] each of the agents learns different roles separately that in the end complement each other. Despite these advances, true multi-agent cooperative learning remains an open problem.

4.7. *Learning modifications*

In this section, we will discuss modifications that have been employed by demonstration learning algorithms to tackle the problems that plague them.

Constraints serve as loss terms designed to impose specific characteristics on the learned model. These terms are either distribution constraints or action constraints. Most commonly, these constraints are employed to ensure that the model remains within the data distribution of the demonstration data set, thus mitigating distributional shift and its negative impacts. Constraint methods can be divided into two categories: direct and indirect.

Direct methods estimate the policy of the data set through BC π_{demo} and use it to constrain the learned policy π_θ, such that the divergence between the distributions of the two policies is below a threshold ϵ: $|\Delta_{\pi_\theta} - \Delta_{\pi_{demo}}| < \epsilon$. However, a major drawback of direct methods is their dependence on the quality of the behavior policy. Estimating the behavior policy is difficult due to its reliance on the quality of the demonstration data set. Then, an incorrect behavior policy can cause methods that use it to constrain the learning process to fail. For example, a behavior policy that was learned from sub-optimal or incorrect demonstrations will constrain the policy learning method on such states, causing the policy to be too pessimistic which is undesirable. Furthermore, if more demonstrations become available, direct constraints require re-estimating the behavior policy. In [6], the algorithm estimates the behavior policy using a parametric generative model and constrains the learning policy to make sure it only chooses actions that the behavior policy would choose. Later in [149], the authors argue that since constraining the distribution does not take into account the quality of the actions, action constraining is superior. In [150], the authors applied a value penalty in the state-value function to improve performance.

Table 2
Distinction between evaluation methods.

	Quantitative	Qualitative
Goal	Performance	Believability
Judgement	Objective	Subjective
Metric	Distance to goal	Subjective Analysis

Contrarily, indirect methods avoid the need to estimate a behavior policy and instead modify the learning objective and use samples from the data set. The most common approach is to minimize the Kullback–Leibler divergence between the distribution of the learned model and the distribution of the demonstration data set. In [151,152], the algorithms estimate advantage functions to constrain the policy to reduce variance and increase sample efficiency. In [153], the authors add a regularizer based on BC by penalizing the difference between actions from the learned policy and the data set.

Alternatively, instead of imposing constraints, other methods can incentivize the model to have specific behaviors independent of the demonstrated data set. If the regularization term is $\mathbb{T}$, the learning objective is adjusted to incorporate the regularization term: $J'(\pi) = J(\pi) + \mathbb{T}$. Examples of regularization terms include penalizing the weights of the networks [3], and state entropy [66], to avoid overfitting. In [154], an entropy regularization term is proposed to control the stochasticity of the policy and promote exploration, preventing premature convergence, improving robustness and stability. In [79], the method learns a lower bound of the true Q-function by adding a regularization term in its estimation.

Next, we can relax the constraints and regularization based on how much we trust the model. For instance, if we estimate the uncertainty of the model, we can reduce safety constraints in low-uncertainty state regions: $J'(\pi) = J(\pi) + \sigma\mathbb{T}$, where σ is a function which weights how much to emphasize the regularizing term. Entropy estimation methods can be applied as regularization terms such as clustering the state space in [66] or ensembles of models [155].

We provide a categorization of demonstration learning methods in the Table 4. The methods are categorized by demonstration technique, type of the input data, the learning objective of the method (such as learning a policy or a world model), the inference type (classification, regression or both of them), the evaluation metrics and the applications.

5. Evaluation

Like other ML paradigms, demonstration learning methods require evaluation using specific metrics. These metrics are categorized into quantitative and qualitative, as outlined in Table 2. Demonstration learning inherits metrics from RL and supervised learning, with common performance metrics including success rate, accumulated reward, and classification or regression error on a demonstration test set. However, some applications prioritize human-like behavior. Additionally, demonstration learning faces challenges in specifying hyper-parameter values.

Experiments are typically conducted on specific robots or simulators tailored to the method being tested. Due to the limited number of benchmarks, evaluations can be challenging. Demonstration learning can use RL benchmark simulation environments, some of which include demonstration data sets. Even if demonstration data sets are not provided, a policy can be trained through RL to learn the simulation task and generate the necessary data sets. However, real-world benchmarking remains complex due to required hardware, varied backgrounds, and safety concerns. As a result, custom task environments and demonstration data sets are often created for each individual method, though some real-world demonstration data sets do exist. This section elaborates on the evaluation processes in demonstration learning.

5.1. Quantitative

Quantitative evaluation metrics are specific to tasks where the performance of a policy can be directly measured. These approaches are divided into two categories: online and off-policy evaluation, with online evaluation being more common. After training the policy, a set of N online rollouts $\tau_1, \dots, \tau_N$ are performed on the environment, and a metric is applied to these rollouts. A rollout is the sequence of transitions generated by selecting actions from the current policy and obtaining the next state and reward from the transition and reward functions, respectively: $\tau_i = (s_0, a_0, r_0), \dots, (s_H, a_H, r_H)$, where H is the length of the trajectory. The most common measurement of online performance is the average success rate of the policy at performing the task $J(\pi) = \frac{\sum_{i=0}^{N} \mathbb{G}(\tau_i)}{N}$, where $\mathbb{G}$ is 1 if the trajectory completed the task and 0 otherwise. For example, in [87], the success is determined by whether the ball falls inside the cup or outside.

If a reward function is available, the performance of the model can be measured by the accumulated rewards of a rollout [3]: $J(\pi) = \sum_{t=0}^{H} R(s_t, a_t, s_{t+1})$. Similarly, the average or maximum accumulated rewards over multiple rollouts can also be used as performance metrics. In video games, a similar approach involves obtaining the score directly from the environment and using it as an evaluation metric. For example, in [51], performance can be evaluated by the distance traveled.

Alternatively, if the goal is to closely imitate the teacher, the performance measurement can be the distance between the agent's actions and the teacher's actions [156]. This can be quantified through classification or regression error for discrete or continuous action spaces, respectively: $J(\pi) = \sum_{(s_i,a_i)\in D_{demo}} \|\pi(s_i) - a_i\|$.

Time can also be used as an evaluation metric. This can include the time taken to execute a task or the time to converge during learning, such as the number of training steps. For instance, in [115], RoboCup soccer agents were simulated with the goal of keeping the ball away from the opposing team. Thus, the duration for which the ball is kept away from the enemy team can serve as a performance metric.

Another way to evaluate a policy's performance online is by measuring the safety of the method. Safety metrics can include the length of the episode (the number of transitions) or defining a set of safety constraints and counting the number of times the agent violated the constraints. This can be done by associating certain states with violation occurrences or by defining violations separately from the state space and checking for any violations after the agent executes an action. In [64], an advisor agent aims to prevent the main agent from violating constraints that could cause damage during learning. The challenge with online evaluation is that it relies on interactions with the environment, which can be dangerous. This makes it problematic to evaluate a policy while learning, because it might be too dangerous to deploy.

Demonstration learning inherits challenges from ML, particularly in determining the optimal hyper-parameter values. Identifying these ideal values before or early in the training process saves time and computing resources by reducing the need for repeated experiments with different value sets. Furthermore, selecting appropriate hyper-parameters can help prevent dangerous interactions after deployment.

Off-policy evaluation (OPE) involves evaluating a policy using past experience, which can come from demonstration data sets, memories of online interactions, or a combination of both. In [157], various OPE methods are reviewed for selecting hyper-parameter values. Most demonstration learning methods do not use OPE and evaluate performance on a set of pre-defined hyper-parameters. The choice of these parameters is often influenced by state-of-the-art practices, small ablation studies, past methods, or random selection. Alternatively, some approaches train the model using multiple sets of hyper-parameter values to find better configurations.

OPE methods rely on a dataset of past interactions, D_{demo}, and an optimization objective, $J(\pi)$. Some OPE methods use the transition function $P(s_{t+1} \mid s_t, a_t)$ and the reward function $R(s_t, a_t, s_{t+1})$ to

Table 3
Summary of the available benchmarks for demonstration learning methods. RLU stands for RL Unplugged, and Sim. for simulation.

Benchmark	Data set	Action space	State space	Dynamics	Observability	Type
AI Habitat	Included	Discrete	Continuous	Deterministic	Full	Sim.
Adroit	D4RL	Continuous	Continuous	Deterministic	Full	Sim.
ALE	No	Discrete	Visual/Cont.	Stochastic	Full	Sim.
Atari	RLU	Discrete	Visual/Cont.	Stochastic	Full	Sim.
BSuite	No	Discrete	Continuous	Deterministic	Full	Sim.
DM Control	RLU	Continuous	Continuous	Both	Full	Sim.
DM Lab	No	Continuous	Visual/Cont.	Deterministic	Partial	Sim.
DM Locomotion	RLU	Continuous	Visual	Deterministic	Both	Sim.
Google Research Football	No	Discrete	Continuous	Deterministic	Full	Sim.
Gym-MuJoCo	D4RL	Continuous	Continuous	Deterministic	Full	Sim.
Gym-Retro	No	Discrete	Visual/Cont.	Stochastic	Full	Sim.
Meta-World	No	Continuous	Continuous	Stochastic	Full	Sim.
MineRL	Included	Both	Visual	Deterministic	Partial	Sim.
RWRL	No	Continuous	Continuous	Deterministic	Full	Sim.
RoboTurk	Included	Continuous	Continuous	Stochastic	Both	Real/Sim.

evaluate the policy. If these are not available for the current problem, model-based methods estimate a dynamics model $\psi(s_t, a_t) \sim P(s_{t+1} \mid s_t, a_t)$ and a reward function through IRL $\hat{R}(s_t, a_t, s_{t+1})$. Using the transition function, reward function, and the actions selected by the current policy π, we can calculate the expected return: $J(\pi) = \mathbb{E}_{a_t \sim \pi(s_t), s_{t+1} \sim P(s_t, a_t)}[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1})]$.

Alternatively, instead of relying on a transition and reward function, we can estimate a state–action value function $Q(s, a)$ by minimizing the Bellman error using the dataset D_{demo} and the current policy π. In this approach, the OPE objective is to evaluate the expected accumulated rewards as given by the state–action value function, expressed as: $J(\pi) = \mathbb{E}_{(s,a) \sim D_{demo}}[Q(s, a)]$.

An extension of this evaluation method involves weighting the importance of each reward using importance sampling [158]. In this approach, the weights are derived by first estimating a policy from the demonstration data set through BC, denoted as π_{BC}. A common weighting scheme involves calculating the ratio of the product of the probabilities of the actions under the current policy to the product of the probabilities under the behavior policy, given by: $w = \frac{\prod_{t=0}^{H} \pi(a_t|s_t)}{\prod_{t=0}^{H} \pi_{BC}(a_t|s_t)}$. The weights can be used to regulate the original objective as such: $J(\pi) = \mathbb{E}_{a_t \sim \pi(s_t), s_{t+1} \sim P(s_t, a_t)}[w \sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1})]$.

5.2. *Qualitative*

Qualitative metrics are used for tasks where the agent's behavior is more important than the performance it achieves. As previously mentioned, some applications require the agent to simulate human-like behavior, which can be challenging to quantify objectively. One approach to evaluating this believability is to involve multiple judges, each assessing and scoring the agent's behavior based on their own analysis. This method leverages diverse perspectives to gauge how convincingly the agent mimics human actions.

In [159], the authors explored various methods for generating controllers that best replicate human behavior. Although these methods were evaluated using the "Super Mario Bros" game, they are applicable to a range of other tasks. The evaluation was conducted qualitatively, where users were shown pairs of gameplay sequences-one performed by a trained controller and the other by a human. The users were asked to identify which gameplay was performed by a human for each pair.

6. Benchmarks

The demonstration data set is often gathered from rollouts of a policy that was trained using an RL algorithm. Alternatively, a human demonstrator can interact with the environment and generate demonstrations. Hence, online RL benchmarks can be employed for demonstration learning in these scenarios. We summarize the available benchmarks in Table 3.

AI Habitat is a simulation platform designed for the research and development of embodied agents in an efficient 3D environment, with the goal of transferring learned skills to real-world applications. Another benchmark is BSuite [160], which offers a diverse set of experiments to evaluate the capabilities of learning methods. This library automates the evaluation process across nine varied environments. DeepMind's (DM) Control Suite [161] is a benchmark that provides a collection of RL environments built on the MuJoCo simulator. It includes tasks for controlling a variety of agents, such as Acrobot, Ball-in-Cup, Cart-Pole, Cheetah, Finger, Fish, Hopper, Humanoid, Manipulator, Pendulum, Point-Mass, Reacher, Swimmer, and Walker, with state spaces that are non-visual. DM Lab [162] is a 3D learning environment built on the Quake III Arena game. It challenges agents with visual observations and complex 3D navigation and puzzle-solving tasks. One of the most widely used visual benchmarks is OpenAI's Gym, which includes environments for training agents to perform various tasks such as walking with a Humanoid agent, similar to DM Control Suite, using MuJoCo. Gym additionally offers environments for classic Atari games. Gym Retro extends the OpenAI Gym framework by providing environments for over 1000 classic games. Google Research Football [163] introduces a novel RL environment with a physics-based 3D football simulation. This benchmark allows agents to control either a single player or an entire team, making it suitable for multi-agent and multi-task learning scenarios.

Meta-World [164] is a benchmark designed for meta-RL and multi-task learning, featuring 50 distinct robotic manipulation tasks. Meta-learning algorithms can acquire new skills more quickly, by leveraging prior experience to learn how to learn. The Real-World Reinforcement Learning (RWRL) Suite [165] identifies nine challenges that prevent RL agents from being applied to the real-world. It also describes a framework and a set of environments to evaluate the method's potential applicability to the real-world.

In practical applications, such as the real-world, we do not have access to a policy. In such scenarios, the data might come from non-Markovian agents, such as humans, which differs significantly from the data generated by online RL policies Consequently, the data sets generated by the online RL policies are not representative of practical applications. The field is continuously developing a novel benchmarks with varying properties to address these challenges. The demonstration data sets and environments should take into account the properties explained in Section 3. Continuous action and state spaces are generally more challenging than discrete ones, since it is impractical to explore every possible state and action. Therefore, an agent that learns in such domains is obligated to generalize beyond the visited data. Furthermore, visual observations are considered more challenging than non-visual observations. Another common real-world issue is the occlusion of state representations, which results in partially observable states and turns the problem into a POMDP. This scenario is significantly harder and closely mirrors real-world situations where complete state

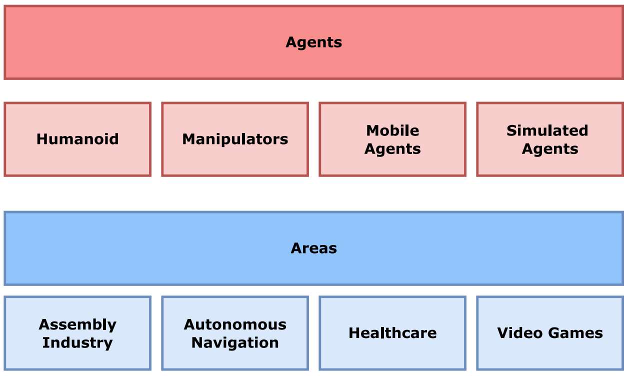


Fig. 7. Agents and areas demonstration learning methods can be applied to.

information is rarely available. Lastly, real-world environments are often stochastic rather than deterministic. Thus, environments that incorporate stochastic transition functions are generally more desirable for evaluating methods.

The D4RL benchmark [166] provides demonstration data sets for OpenAI Gym's MuJoCo tasks. These data sets come from human demonstrations and vary in quality levels such as random, medium, and expert. The RL Unplugged benchmark [167] provides demonstration data set for four different suites: Arcade Learning Environment (ALE) [168], DeepMind Control Suite [161], DeepMind Locomotion [169], and the Real-World Reinforcement Learning Suite [165]. The disadvantage is that the data sets of RL Unplugged are obtained from online RL policies. This means they lack the desirable non-Markovian properties which arise from human-generated data.

MineRL [170] is a benchmark built on top of the game Minecraft. The benchmark includes a wide array of tasks, including finding a cave, creating a pen, making a waterfall, building a house, and locating a diamond. These tasks are characterized by their sequential nature, which presents significant challenges for learning algorithms, such as sparse rewards and long horizons. The benchmark provides a large demonstration learning data set with over 60 million frames of recorded human gameplay.

For real robots, RoboTurk [171] provides the largest collection of demonstration data sets of a variety of real-world robots performing diverse manipulation tasks. The data set is increased through crowdsourcing. It also provides a framework for generating demonstrations for a personal robot controlled through teleoperation using a phone.

The Deep Off-Policy Evaluation (DOPE) benchmark [172] offers a standardized framework for the evaluation and comparison of various OPE algorithms. Because it is not a benchmark to evaluate demonstration learning methods, it is not included in Table 3. This benchmark is structured into two main suites: DOPE RL Unplugged and DOPE D4RL. DOPE allows the selection of different learning objectives, such as ensuring the estimated value of a policy is as close as possible to the true value, selecting the best possible policy from a set of policies, and hyper-parameter tuning with early stopping.

In Table 4, we categorize various demonstration learning methods. The table organizes the methods based on key attributes: the demonstration technique used to obtain the data set; the type of input data used to train the method; the learning objective (e.g., policy learning); the type of inference type (classification, regression, or both); the metrics used to evaluate the method; and the tasks the method was applied to.

7. Applications

Demonstration learning approaches have been successfully applied to a wide range of domains, as illustrated in Fig. 7. This section provides a detailed exploration of these applications, highlighting a selection of relevant examples. The primary applications of demonstration learning are robotics and simulation environments. RL can learn complex skills but at the cost of interaction with the environment which can be dangerous or expensive in real-world scenarios. Moreover, the sample inefficiency of RL requires millions of interactions even for simple tasks in the real-world. For this reason, demonstration learning can be critical for applying policy learning algorithms to real-world settings.

7.1. Assistive robots

Demonstration learning can be applied to assistive robots which aim to help humans in their daily tasks. This application presents unique challenges, because these robots must be capable of adapting to a wide range of scenarios that the untrained human may require. In [117], the authors adapt the training process to account for the dynamic and evolving behavior of humans. Robots may be used to interact with humans [48] or help with social or mental problems [207]. Assistive robots likely have to imitate how a human would assist, which is something that demonstration learning can provide over RL.

7.2. Autonomous navigation

Self-driving cars have been a central focus of ML research in recent years. The goal is to use the data captured from a wide range of sensors to control the vehicle safely in complex environments. Given the immense complexity and high dimensionality of the problem, RL represents a viable solution by learning from interactions. However, the unsafe characteristics of RL represent an obstacle, as the consequences of unsafe driving are catastrophic. Demonstration learning avoids this problem since the agent does not have to interact with the environment to learn.

Early research in [208] proposes a method for learning to fly an aircraft from demonstrations recorded via teleoperation using IRL. Demonstration learning has also been applied successfully to autonomous aerial navigation, such as drones for their potential for the delivery of goods, and helicopters. In [29] and in [9], data of a robot helicopter flight, using a joystick, was recorded and used to train an autonomous helicopter agent through RL and apprenticeship learning, respectively. With advances in sensors for capturing data, demonstration learning has been increasingly used to learn control policies. Beyond aerial applications, demonstration learning has also been employed for the locomotion of bipedal and quadrupedal robots. The demonstrations for training such robots are obtained either by teleoperation or observation. Some works have applied demonstration learning to driving tasks [58,101,173]. [209] collected a dataset for navigation tasks using a camera mounted on a robot. In [57], the authors evaluated active learning techniques based on demonstrations in both simulated and real-world environments. Similarly, [83] utilized a mobile robot to gather environmental data and select appropriate algorithms from a pre-defined database based on situational context. Full autonomous control of a vehicle in the real-world is such a complex problem that it likely cannot be solved using a single field of ML. Nonetheless, demonstration learning offers tremendous potential to learn skills from data, without having to interact with the dangerous environment of driving in the real-world.

RobotCar [210] and BDD100K [211] are large data sets containing real-life driving demonstrations in the form of videos. Demonstrations have been used towards autonomous driving in [1,177,191,212]. The safety of demonstration learning can be used to employ RL methods to autonomous driving. For example, in [47,68,195] the agent cannot violate safety constraints learned from demonstrations.

Table 4
Categorization of Demonstration Learning papers.

Name		Year	Technique	Data	Learned goal	Inference	Evaluation	Benchmark
Abeel et al. 2004	[173]	2004	Teleoperation	Raw Data	IRL	Regression	Acc. Reward	Grid World Car Driving Simulation
ABPS	[174]	2021	N/A	N/A	Policy Learning	Classification	Acc. Reward	Custom Grid World
Align-RUDDER	[175]	2022	N/A	Raw Data	Policy Learning, IRL	Regression	Succ. Rate	Minecraft
AOG	[43]	2017	Sensors on Teacher	Sensor data, Image	Classification	Classification	Succ. Rate	Water bottle opening
APE-V	[116]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, Succ. Rate	D4RL, Procgen Mazes
APID	[75]	2013	Teleoperation	Raw Data	Policy Learning	Regression	Time, Acc. Reward	Path Finding
AQuaDem	[86]	2021	Teleoperation	Raw Data	Policy Learning	Classification	Acc. Reward, Succ. Rate	D4RL
ARC	[132]	2018	N/A	Image	Policy Learning, IRL	Regression	Acc. Reward	Navigation, Robot Pushing
ATAC	[126]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	D4RL
ATC	[130]	2021	Observation	Image	Policy Learning	Regression	Acc. Reward, Train Time	DM control, Atari, DM Lab
AT-Net	[56]	2020	Teleoperation	Image	Policy Learning	Regression	Alignment Error, Accuracy, Succ. Rate	Manipulation
AWAC	[152]	2020	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, Succ. Rate	Simulated, Real Robot Manipulation
AWR	[151]	2019	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	OpenAI Gym, Simulated Robot
BCQ	[6]	2019	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	OpenAI Gym MuJoCo tasks
BEAR	[149]	2019	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	OpenAI Gym MuJoCo
BRAC	[150]	2019	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	OpenAI Gym MuJoCo
BREMEN	[93]	2020	Teleoperation	Raw Data	Model-Based Policy Learning	Regression	Acc. Reward, KL Divergence	OpenAI Gym MuJoCo
CDS	[176]	2021	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, KL Divergence	MuJoCo, Meta-world
ChauffeurNet	[177]	2018	Teleoperation	Sensor data	Policy Learning	Regression	Distance Error	CARLA driving simulator
CIC	[136]	2022	N/A	N/A	Policy Learning	Regression	Acc. Reward	Mujoco, Simulated Jaco Robot
CLfD	[129]	2022	Observation	Image	Policy Learning	Regression	Alignment Error, Success Rate, Acc. Reward	Simulated Panda Manipulation
Coarse-to-Fine IL	[178]	2021	Observation	Image	Policy Learning	Regression	Error, Succ. Rate	Target Reaching
Codevilla et al. 2018	[1]	2018	Teleoperation	Sensor data	Policy Learning	Regression	Succ. Rate, Missed turns, Interventions, Infractions	Real Truck Driving
Confidence-Based LfD	[57]	2007	Teleoperation	Raw Data	GMM	Classification	Accuracy, Collision Rate	Custom Simulation
Context-Aware Translation	[39]	2018	Observation	Image	Policy Learning,IRL	Regression	Error, Succ. Rate	MuJoCo Manipulation
COPO	[179]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, Cost, Violations	Walk-Around-Grid, Bipedal Walker
COMBO	[94]	2021	Teleoperation	Raw Data	Model-Based Policy Learning	Regression	Acc. Reward, Model Error	D4RL
CORRO	[180]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	Mujoco
CQL	[79]	2020	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, Value Error	D4RL
Cross-Context IL	[88]	2020	Observation	Image	Policy Learning	Regression	Distance, Succ. Rate	Manipulation
CSI	[104]	2013	Teleoperation	Raw Data	IRL	Both	Acc. Reward	Mountain Car, Driving Simulator
CVaR	[125]	2022	N/A	N/A	Policy Learning	Regression	Acc. Reward	Custom Grid World
CVPO	[47]	2022	N/A	N/A	Policy Learning	Regression	Cost, Acc. Reward	Custom Simulation Tasks
DAgger	[51]	2011	Teleoperation	Raw Data	Policy Learning	Classification	Falls, Distance Traveled	Super Tux Kart, Super Mario Bros.
Dalal et al. 2018	[68]	2018	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, Violations	Custom MuJoCo tasks
DIS	[106]	2016	Teleoperation	Raw Data	Policy Learning, IRL	Regression	Acc. Reward	Maze, Mario AI
DQfD	[3]	2018	Teleoperation	Image	Policy Learning	Classification	Acc. Reward	ALE
DQN	[4]	2015	N/A	N/A	Policy Learning	Classification	Acc. Reward	Atari
Dogged Learning	[24]	2007	Teleoperation	Raw Data, Image	Other Predictive Learning	Both	Student visual inspection, Error	Ball seeking, head mirroring tail
DoubIL/ResiduIL	[70]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Loss	OpenAI Gym
DT	[139]	2021	Teleoperation	Raw Data	Sequence Model	Regression	Acc. Reward	Atari, OpenAI Gym

(*continued on next page*)

Table 4 (*continued*).

DWBC	[120]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, Discriminator Accuracy	D4RL
EnsembleDAgger	[181]	2019	N/A	N/A	Policy Learning	Regression	Acc. Reward	OpenAI Gym
ExORL	[182]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	DeepMind control suite
GAIL	[46]	2016	Teleoperation	Raw Data	GAIL	Regression	Acc. Reward	MuJoCo
GCB	[137]	2022	Observation	Image	Policy Learning	Regression	Succ. Rate	Pybullet
Gradient-Based IRL	[105]	2021	Observation	Image	Model Learning, IRL	Regression	Train Time, Distance	Teaching
GTI	[183]	2021	Teleoperation	Raw Data	Policy Learning	Regression	Succ. Rate, Generalization	Manipulation
Guo et al. 2022	[77]	2022	Observation	Image	Policy Learning	Regression	N/A	N/A
HAMMER	[50]	2005	Observation	Image	Bayesian Belief	Regression	N/A	N/A
Hayes et al. 2014	[38]	2014	Observation	Image	Active learning	Regression	Execution Paths	Lego Montage
HDT	[141]	2022	Teleoperation	Raw Data	Sequence Model	Regression	Acc. Reward	UR3 Reaching, D4RL
IRIS	[184]	2020	Teleoperation	Raw Data	Policy Learning	Regression	Succ. Rate, Acc. Reward, Traj. Length	Graph Reach, RoboTurk, Robosuite
Ijspeert et al. 2002	[11]	2002	Sensors on Teacher	Sensor Data	LWR	Regression	Error	Tennis Swings
ILEED	[78]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	Simulated Minigrid tasks
LazyDAgger	[185]	2021	N/A	N/A	Policy Learning	Regression	Acc. Reward, Interventions	MuJoCo
Levine et al. 2016	[53]	2016	N/A	Image	Policy Learning	Regression	Distance, Succ. Rate, Error	Manipulation
Levine et al. 2018	[186]	2018	Observation	Image	Policy Learning	Regression	Failure Rate	Manipulation Task
LDM	[80]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, Succ. Rate	OpenAI Gym MuJoCo, SimGlucose
LTSD	[187]	2019	Teleoperation	Raw Data	Policy Learning, IRL	Regression	Acc. Reward	BiMGame, AntTarget, AntMaze
LfMD	[10]	2015	Teleoperation	Point Cloud	Other	N/A	Succ. Rate	Pick Place, Towel Folding
Maeda et al. 2017	[33]	2017	Kinesthetic	Sensor Data	Policy Learning	Regression	Distance Error	Manipulation Tasks
MAGIC	[96]	2016	N/A	N/A	Policy Learning (OPE)	Regression	Regression Error	ModelFail/Win, Maze, Mountain Car, Cart
Max. Ent. IRL	[101]	2008	Sensors	GPS Data	IRL	N/A	Matching	Path Following
MBPO	[90]	2019	N/A	N/A	Policy Learning (OPE)	Regression	Acc. Reward, Regression Error	MuJoCo
MERLION	[89]	2021	Teleoperation	Raw Data	Policy Learning	Classification	Acc. Reward	Maze, Dialogue, Recommendation
MIR	[188]	2021	Observation	Image	Policy Learning	Regression	Succ. Rate	MuJoCo, Manipulation
MOPO	[92]	2020	Teleoperation	Raw Data	Model-Based Policy Learning	Regression	Acc. Reward	D4RL
MOReL	[91]	2020	Teleoperation	Raw Data	Model-Based Policy Learning	Regression	Acc. Reward	OpenAI Gym MuJoCo
Motion2Vec	[189]	2021	Kinesthetic, Observation	Robot Data, Image	HMM	Regression	Loss, Segmentation Accuracy, Noise	Pick-and-Place, Suturing
MRDR	[95]	2018	N/A	N/A	Policy Learning (OPE)	Regression	Regression Error	ModelFail/Win, Maze, Mountain Car, Cart
Mülling et al. 2013	[102]	2013	Kinesthetic	Raw Data	LWR	Regression	Cost, Succ. Rate, Acc. Reward	Tennis Swings
Multi-AMP	[121]	2022	Observation	Image	Policy Learning	Regression	Stand Duration	4 Legged Robot Movements
MVP	[190]	2022	Observation	Image	Policy Learning	Regression	Succ. Rate	PixMC
ODT	[142]	2022	N/A	Raw Data	Policy Learning	Regression	Acc. Reward	D4RL
Pan et al. 2017	[191]	2017	Teleoperation	Sensor data, Image	Policy Learning	Regression	Speed, Succ. rate	AutoRally
PC-GMM	[74]	2020	Teleoperation	Raw Data	GMM, GMR, RL	Regression	Succ. Rate, Error	Peg-in-Hole
PEMIRL	[123]	2019	Teleoperation	Raw Data	IRL	Regression	Acc. Reward	MuJoCo
PI2-ES-Cov	[35]	2020	Kinesthetic	features from point cloud	Policy Learning	Regression	Succ. Rate	Simulated, Real Robot Manipulation
Pinto et al. 2016	[192]	2016	Observation	Image	Policy Learning	Regression	Accuracy	Manipulation Task
PTS	[113]	2015	N/A	N/A	IRL	Regression	Score	Mounting Car 3d, Cart Pole, Mario

(*continued on next page*)

Table 4 (*continued*).

Recovery RL	[193]	2021	Teleoperation	Raw Data	Policy Learning	Regression	Succ. Rate, Violations	Simulated task, Real Robot
REPaIR	[194]	2020	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward, AUC, Accuracy	Robot Reaching
Rhinehart et al. 2018	[195]	2018	Teleoperation	Raw Data	Model-Based Policy Learning	Regression	Succ. Rate, Classification Metrics	CARLA driving simulator
RLfD	[107]	2015	Teleoperation	Raw Data	Policy Learning, IRL	Regression	Acc. Reward	Mario AI, Cart Pole
RIS	[196]	2021	N/A	N/A	Policy Learning	Regression	Acc. Reward, Distance, Succ. Rate	Navigation, Manipulation
RPL	[197]	2019	Teleoperation	Raw Data	Policy Learning	Regression	Succ. Rate	MuJoCo
RUDDER	[198]	2019	N/A	Raw Data	Policy Learning, IRL	Regression	Acc. Reward	Atari
Ruppel et al. 2020	[45]	2020	Sensors on Teacher	Sensor Data	Sequence Model	Regression	Succ. Rate, Error	Manipulation
S4RL	[59]	2022	Teleoperation	Image	Policy Learning	Regression	Acc. Reward	D4RL, MetaWorld, RoboSuite
SafeDAgger	[199]	2017	Teleoperation	Raw Data	Policy Learning	Regression	Score, Safety, loss	TORCS Driving Car
SAILR	[48]	2021	N/A	N/A	Policy Learning	Regression	Acc. Reward, Constraints	Point Robot, OpenAI Gym MuJoCo
SAM	[122]	2019	Teleoperation	Raw Data	GAIL	Regression	Acc. Reward	MuJoCo
SCIRL	[103]	2012	Teleoperation	Raw Data	IRL	Classification	Acc. Reward	Highway
Sepsis Treatment	[200]	2017	N/A	N/A	Policy Learning	Classification	Mortality	MIMIC-3.
Silver et al. 2010	[100]	2010	Kinesthetic	Raw Data	LWR	Regression	Cost, Loss, Distance, Speed	Navigation
SMILe	[58]	2010	Teleoperation	Raw Data	Policy Learning	Regression	Falls, Distance	Mario Bros
SPiRL	[201]	2020	Teleoperation	Raw Data	Policy Learning	Regression	Succ. Rate	D4RL
SQUIRL	[124]	2020	Observation	Image	Policy Learning, IRL	Regression	Succ. Rate	Pick-Carry-Drop, Pick-Place
SRL-RNN	[202]	2018	N/A	N/A	Sequence Model	Both	Estimated Mortality	MIMIC-3.
SWIRL	[203]	2019	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	RCCar, Acrobot
SWITCH	[97]	2017	N/A	N/A	Policy Learning (OPE)	Regression	Regression Error	UCI data sets
TCC	[133]	2019	Observation	Image	Representation Learning	N/A	Classification Accuracy, Kendall's Tau	Pouring and Penn action data sets
TCN	[55]	2018	Observation	Image	Policy Learning	Regression	Alignment Error, Classification Error, Acc. Reward	Pouring
TD3+BC	[153]	2021	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	D4RL
TPIL	[204]	2021	Observation	Image	Policy Learning	Regression	Alignment Error, Loss	PyBullet, Minecraft
UCT	[52]	2014	Teleoperation	Image	Regressor/Classifier	Both	Score	ALE
UDS	[63]	2022	Teleoperation	Raw Data	Policy Learning	Regression	Acc. Reward	D4RL
visual MPC	[205]	2018	Observation	Image	Model-Based policy learning	Regression	Regression Error, Succ. Rate,	Manipulation Task
Weak Label LfD	[32]	2020	Teleoperation	Sensor Data, Image	Classifier	Classification	Effort	Manipulation
Yang et al. 2022	[206]	2022	Teleoperation	Raw Data	Policy Learning, GAN	Regression	Acc. Reward	D4RL

7.3. Manipulators

Demonstration learning has been applied to manipulators in manufacturing applications since the 1980s. Training manipulators through demonstrations, usually through kinesthetic teaching, is more efficient than hard-coding their behavior. One prominent application of this technology is in assistive and healthcare robotics, where robots are designed to help humans with various tasks. ML methods are used for such tasks because the robot must perform effectively in diverse environments. Training a policy to generalize to new scenarios is more practical than coding every possible situation manually. Additionally, since assistive robots operate in close proximity to humans, there is a greater emphasis on safety, which is enhanced through the convergence and stability guarantees of learned policies compared to hard-coded behaviors. Furthermore, effective collaboration requires the desired robotic movements to be similar to those of humans. Moreover, effective collaboration with humans necessitates that robotic movements closely mimic human actions, making demonstration learning a good approach for developing such behaviors in the policies.

Several studies have utilized demonstrations for learning robotic grasping [186,192]. Examples of demonstration learning methods applied to such robots include handing over objects to humans in [213], where the authors encode coordination from demonstration data to enable humans to transfer control of objects to other entities. Additionally, a model-based algorithm [205] learns a variety of manipulation skills.

Policies for learning manipulation tasks have the potential to replace humans in healthcare. However, this raises significant safety concerns, as exploration required by RL is impractical [214]. In [2], the authors train multiple robotic arms to perform surgical tasks using trajectories learned from demonstrations. Later, [215] applies a model-based method for the treatment of lung cancer. The MIMIC-III dataset [216], which contains 60,000 Intensive Care Unit (ICU) records, has been used for drug recommendations [200] and sepsis treatment [202] by leveraging demonstrated treatment processes from the data set. The automation of healthcare through demonstration learning has the potential to optimize procedures and improve outcomes.

7.4. Humanoid robots

Humanoid robots are perhaps the most obvious application of demonstration learning. These are robots with a structure similar to humans whose goal is to perform tasks typically carried out by

Table 5
Summary of the advantages and disadvantages of demonstration learning methods.

Advantages	Diverse and flexible: can be combined with other paradigms. Data efficiency: faster convergence from fewer data over reinforcement learning. Performance guarantees: the method will at least be as good as the demonstrations. Reduction of programming load: no need to program the model's decision for every case. Safety: the agent learns without interacting with the environment. Simplification: demonstrate the task and estimate a policy with a SOA method.
Disadvantages	Creation of the data set: difficult to create large numbers of high quality demonstrations. Distributional shift: generalize from the data set or remain inside its distribution. Quality of the data set: performance relies heavily on the demonstrated behavior.

people. Since the required tasks are already performed by humans, demonstration learning is a fitting approach. Demonstrations can be captured using a sensor suit and converted into joint values for the robot. The tasks vary from utilizing only part of the robot [217] to engaging the entire humanoid body [218,219].

In [11], a humanoid robot was trained from human demonstrations captured by sensors placed directly on the human body to perform reaching and drawing movements with one arm, as well as tennis swings. In [36], a humanoid robot learned to imitate human arm gestures and was tested in a turn-taking gesture game. A human tele-operated a humanoid robot in [9], where Virtual Reality technology was used to convert the operator's arm and hand motions into those of the robot to learn a manipulation policy. In [55], an embedding space was learned from demonstrations, allowing the humanoid robot to replicate human movements projected into this space. In human–robot interaction, beyond learning task control, the robot must also learn where to focus its attention.

7.5. Video games

ML methods, particularly RL, have been successfully applied to video games. Key applications include creating agents to learn and master games or generating controllers for non-playable characters. However, video games often present challenges such as sparse rewards and high-dimensional spaces, which can hinder policy convergence during exploration. Additionally, non-playable characters often need to exhibit human-like behavior [220]. Therefore, demonstration learning serves as a bridge to reach these goals that RL struggles to achieve.

Demonstration learning has been applied to racing games such as Mario Bros in [51,159]. The ALE [168] offers a platform for benchmarking algorithms on Atari games. Approaches such as [3] have used this environment to evaluate their method and compare them to other state-of-the-art methods for Atari games. While these examples involve relatively simple problems with small state and action spaces, they demonstrate potential that can be extended to more complex games in the future.

The variety of genres and the growing complexity of video game domains present challenges increasingly akin to real-world difficulties. Methods often rely on simulation to validate their correctness due to safety concerns associated with real-world deployment. However, creating effective simulators is challenging and often yields inconclusive results, as the real-world remains more complex. Video games offer a diverse array of problems for demonstration learning methods to tackle. Addressing progressively difficult problems in video games provides a pathway for validation that is closer to real-world scenarios.

8. Advantages and disadvantages of demonstration learning

Different demonstration learning algorithms bring different advantages and disadvantages, which are summarized in Table 5. This section will explore the overall advantages and disadvantages of demonstration learning methods.

8.1. Advantages

The most significant advantage of demonstration learning paradigm over others is its potential to eliminate the need for expert programming. Although the field has not yet reached this goal, the aim is to train an agent to perform tasks through demonstrations alone. This approach enhances adaptability by enabling agents to learn behaviors from demonstrations rather than being programmed on static instructions. Demonstrating a manipulation task is often much easier than detailing every step a robot must take to complete it. Additionally, the field allows agents to learn from various external sources, including humans or other agents with different types of hardware.

Additionally, the paradigm is highly data efficient, especially compared to RL which is severely data-inefficient. Several approaches use a small number of demonstrations. In contrast, RL rely on trial-and-error interactions to learn optimal policies, resulting in numerous failed attempts. Demonstration learning allows the agent to learn the task by providing the correct choices for the input states, thus solving high-dimensionality problems and effectively addressing the curse of dimensionality that plagues ML problems suffer from. This property allows the paradigm to solve high-dimensionality problems and effectively addresses the curse of dimensionality ML problems suffer from. Furthermore, RL agents' policies can converge to local minima with suboptimal performance. In demonstration learning, agents learn the behavior demonstrated to them, offering performance guarantees for the policies learned. Then, demonstration learning can be combined with other ML fields, such as RL, to further enhance the learned methods.

Lastly, one of the main limitations of RL is that it is difficult to apply it to real-world problems. The RL agents learn through trial-and-error interactions, which is ideal for simulation environments. However, in the real-world, errors can have serious consequences, such as collisions that may severely damage a robot, hindering or preventing learning altogether. Demonstration learning is extremely valuable in applications where interaction is impractical, expensive, or dangerous. It leverages demonstration data sets to provide interaction data, allowing the agent to learn without directly interacting with the environment. This approach ensures much safer learning, making it applicable to real-world problems. Even when it is safe to interact with the environment, using demonstrations can speed up convergence and improve generalization in complex domains. After learning a policy from demonstrations, the policy can still be improved with online interactions. Since the policy already possesses task knowledge, it interacts with the environment more safely compared to a random policy in RL.

8.2. Disadvantages

While demonstration learning is appealing and addresses many issues associated with RL, it presents challenges when working with limited data. Technically, any off-policy RL algorithm can be used to learn from a demonstration data set. However, these algorithms were originally designed for online RL, where the agent interacts with the environment and can correct its mistakes. As a result, these methods often fall short when applied to limited data.

The disadvantages of demonstration learning are primarily related to the data set, as discussed in Section 3.7. The quality of the policy is directly dependent on the quality of the demonstrations within the data set. Sub-optimal or incorrect demonstrations can hinder or even prevent the policy from converging to adequate behavior. To address this issue, existing solutions aim to identify and filter out these sub-optimal demonstrations.

Additionally, collecting demonstrations is expensive, and the data set may be insufficient, often lacking coverage of all possible state–action pairs. Creating the environment with the sensors, demonstrating, collecting the data, and creating adequate embodiment and record mappings for the data set, are all non-trivial steps. Then, the distributional shift discussed in previous sections can severely hinder the performance of the model. The demonstration data set represents only a subset of the real MDP. Unless the data set is optimal the agent cannot generalize to cover the entire problem. Hence, when queried in out-of-distribution states, it will likely fail. Restricting the agent to in-distribution states to avoid queries for actions it cannot generalize to from limited data is challenging, and most methods must address this issue explicitly.

9. Future directions

In this section we will discuss the open research problems in the field of demonstration learning.

9.1. Benchmarking

The goal of demonstration learning is to teach a real-world agent a task from demonstrations. However, most methods are evaluated in simulation environments. There are few standardized benchmarking environments and data sets, particularly for real-world environments. Standardization in any research area is beneficial as it facilitates the comparison of different methods for the same problem, and demonstration learning is no exception. The lack of a real-world demonstration learning benchmark is an open issue in the field, limiting its progression and applicability to real-world tasks.

Current demonstration learning methods are often evaluated using a limited range of RL environments, frequently creating their own demonstration data set. A robust method should be able to generalize across various environments and perform tasks with high repeatability. Additionally, the method should be computationally efficient and, in some cases, capable of producing behavior that appears human-like.

Although some environments provide corresponding demonstration data sets, methods that generate their own data sets create uncertainty about the method's quality. The number of demonstrations in the data set significantly impacts the method's performance. In real-world scenarios, collecting demonstrations is challenging, resulting in smaller data sets. However, some methods utilize large numbers of demonstrations, offering greater coverage of the state–action space, which is a distinct advantage over methods with smaller data sets. Measuring and comparing the quality of demonstrations across different data sets is difficult. It is reasonable to assume that two different data sets will vary in quality. Consequently, the performance of a method is directly proportional to the quality of the demonstrations used.

Existing stochastic environments are very limited. Additionally, non-stationary environments, which are very common in the real-world, are also scarce, with the notable exception of the RWRL suite. Additionally, there is a lack of available multi-agent environments and data sets. To address these gaps, future research should focus on creating more standardized environments, demonstration data sets, and evaluation metrics, and encourage their usage.

9.2. Context problem

The context encompasses the set of characteristics that define the environment, such as background objects, illumination, and camera positioning. Most demonstration learning methods are designed to learn a policy for a single, fixed context. When any of these contextual elements change, they form a new context, and the policy trained for the original context is often unable to perform effectively in the new one. This issue is frequently overlooked because evaluation is typically conducted in simulation or controlled environments where the context remains consistent with the training conditions, allowing the policy to succeed. However, this limitation severely affects the scalability of the policy for real-world applications, where context changes can be drastic.

9.3. Goal specification

Demonstration learning methods focus on learning one specific task at a time. Pick-and-place tasks, for example, involve picking up an object from one location and placing it somewhere else. In the literature, picking a different type of object or placing it in a different location is often considered a distinct task from the original [124], despite both being variations of the same fundamental task. As a result, the effort required to train an agent to handle multiple goals scales linearly with the number of distinct goals, as a separate policy must be developed for each one. This approach overlooks the fact that these tasks share redundancy and common information. Future research should aim to develop policies that can scale and adapt to different goals specified by the input, rather than requiring a new policy for each individual goal.

9.4. General demonstration learning framework

The end goal of demonstration learning is to enable the training of an agent without the need for expert programming knowledge. Ideally, an agent should be able to learn the task solely through demonstrations. However, current approaches still require the design of algorithms for feature extraction, reward specification based on the type of agent and task, and policy derivation algorithm. A learning framework that could be universally applied to any task would allow a teacher to train an agent solely by demonstrating the task.

9.5. General feature extraction

All demonstration learning approaches rely on the quality of the data set. In high-dimensional environments, images are the most practical state representations. Learning from images requires the extraction of quality features, which currently vary depending on the specific task. Creating a general feature extractor framework would eliminate the need for engineered feature extraction frameworks for each task. This is one of the open problems preventing the creation of a general demonstration learning algorithm.

9.6. Generalization

The policies should be able to select the correct action for any possible state. However, since policies are estimated from a demonstration data set, they are inherently limited to the state–action distribution present in that data set. Generalization is the ability of a policy to make correct decisions in unseen scenarios based on previous experiences. To improve generalization, several techniques are employed: some approaches expand the data set through methods such as data augmentation [60,61], while others fine-tune the policy using online data [178]. Additionally, some methods restrict the policy to in-distribution states to prevent it from being queried for actions it has not been trained on [47].

9.7. Hyper-parameter selection

Hyper-parameters are a critical aspect of most ML approaches and must be specified in advance. Examples include determining the number of hidden layers in a neural network and the number of neurons in each layer. The ideal values for the hyper-parameters are often difficult to estimate and are usually determined through trial and error. As noted by [221], many studies present results based on carefully tuned hyper-parameters for a limited set of tasks, which are insufficient for evaluating the robustness of methods. Instead, methods should be trained multiple times with varying hyper-parameter configurations, and the resulting policies should be assessed using the proposed metric, 'Expected Validation Performance'.

Performing policy rollouts in real-world settings is generally challenging due to costs and safety concerns, particularly when evaluating policies that are still being trained. Evaluating a policy before completing its training to determine the optimal hyper-parameter values would allow for adjustments without incurring the full cost of training a complete policy, thereby saving time and resources. This is another factor limiting demonstration learning to simulators. Off-policy methods aim to evaluate a policy from past interactions, without requiring new interaction with the environment. For example, [222] conducted a study evaluating thirty-three different OPE methods by measuring the distance between the estimated policy value compared to the true policy value. The study concluded that evaluation with the state–action function outperforms the remaining methods. To help accelerate the development of OPE methods, [172] propose the DOPE benchmark. The available options are using inaccurate OPE methods or training the model for a fixed number of steps and adjusting the values based on the early results. While hyper-parameter selection is a common challenge across various machine learning paradigms, demonstration learning needs to overcome this problem to truly reduce the requirement of expert programming. We believe that developing general, accurate, and efficient OPE methods is crucial for the growth of the field and for expanding its applications to real-world scenarios.

9.8. Long-horizon tasks

Offline RL relies on a reward function to estimate policies. More frequent rewards make it easier to develop effective policies, but creating a reward function that provides feedback for every step is often challenging, especially for complex tasks. As a result, demonstration learning has been restricted to simple tasks that complete within a few steps and require minimal interactions. Long-horizon tasks are more common in real-world scenarios, which consist of either a series of small interactions or a single extended and complex interaction. These tasks require a greater number of steps, leading to a more extensive state space for the agent to explore and learn from. Although learning to handle long-horizon tasks remains an open challenge, some methods have achieved partial success in addressing these complex problems.

The method in [183] detects intersections among demonstrations in the data set at certain states to generalize new trajectories. [187,203] demonstrate that complex tasks can be decomposed into sub-tasks, which are learned from the demonstration data set. The agent receives a reward upon completing each sub-task. [196] estimates a state distribution to ensure that these sub-goals correspond to reachable states. A particularly promising approach involves using goal-conditioned policies across multiple layers of hierarchy in RL [223]. Additionally, some methods generate sequences of sub-goals through a divide-and-conquer approach [224].

Another approach is to learn an embedding space of skills from unstructured demonstrations [201] and then train the policy on the embedding space. In [201], skills are defined as useful sequences of actions, and an embedding space of these skills is learned from unstructured demonstrations. They then train a RL policy on the embedding space using a hierarchical model where the high-level component generates embeddings of skills which the decoder then converts into sequences of actions. A particularly promising approach was proposed, using goal-conditioned policies at multiple layers of hierarchy for RL [223]. Alternatively, other methods generate sequences of sub-goals with a divide-and-conquer approach [224].

Hierarchical approaches learn a high-level planner and a low-level controller [184,203]. The high-level planner identifies a sequence of sub-goal states that guide the agent towards the main task goal, while the low-level controller is conditioned to achieve these sub-goals. This extra guidance helps the agent learn in sparse reward environments and long horizon tasks. Conditioning RL and demonstration learning approaches on goal observations improves sample efficiency. In [196] the high-level policy is encouraged to predict intermediate states between the current state and the goal state.

RUDDER [198] redistributes the reward by identifying key steps in the demonstrations and increasing the reward of the respective transition. However, RUDDER uses LSTMs to predict the associated reward and models require many demonstrations to generalize well. Because of this, Align-RUDDER replaces the LSTM model with a profile model adapted from bio-informatics. This model can be estimated with very few sequences. In Align-RUDDER, a sequence is aligned with the profile, providing an alignment metric that indicates how well the sequence matches the model. The reward for a transition is the difference between the alignment values of the sequence with and without the transition.

9.9. Multi-agent demonstration learning

Most demonstration learning research is focused on single-agent learning, while real-world applications often require agents to interact and cooperate with one another. Although there has been some exploration of multi-agent cooperation and competition, as discussed in the relevant section, the applications of the methods are limited. Hence, multi-agent demonstration learning remains an open problem.

9.10. Learning from sub-optimal data

Demonstration learning heavily relies on the quality of the demonstration data set. In many cases, the data is sub-optimal due to various factors. Sensor data often contains noise and errors, which can increase the distributional shift between the estimated models and the demonstrated behavior. Some works address inadequate data through identification and removal of poor samples [30] or by correcting the data [194]. Collecting demonstrations is generally expensive, resulting in small data sets. To mitigate this, some methods use demonstrations from other tasks, enhancing the policy's generalization and adaptability. For instance, [180] gathers data sets from multiple tasks sharing the same state and action spaces but differing in reward functions and dynamics. An encoder is trained to encode trajectories into an embedding space, and the policy is conditioned on a task embedding alongside the state. However, using data sets from other tasks can increase the distributional shift. To address this, [176] proposes a method to learn policies from multiple task data sets without causing a distributional shift, by relabeling transitions from the other tasks' data sets.

Additionally, the estimated distribution of the policy may differ significantly from the demonstrator's distribution. A larger distributional shift typically results in poorer generalization of the policy. In [77], the authors address this mismatch by deriving a tractable bound on the distributional shift between the offline data set and the learned policy. This bound is then used as an extra regularization parameter in the optimization step.

Offline RL requires reward specification which is expensive in the real-world scenarios. Beyond IRL, unsupervised techniques offer a promising direction by leveraging unlabeled data. For instance, [63] demonstrates that a policy can be effectively learned by combining large sets of unlabeled data with a smaller set of labeled data. Additionally, [182] introduces a downstream reward labeling method applied to unlabeled datasets.

9.11. Safety concerns

RL has gained significant attention in recent years due to its diverse applications. However, its trial-and-error nature can pose safety risks. In contrast, demonstration learning allows the agent to learn policies from a dataset without interacting with the environment, remaining safe while learning. In offline RL, the goal is to maximize the expected accumulated rewards, while in demonstration learning the goal is to align action choices with those of the demonstrator. However, these goals do not take into account safety concerns. Additionally, the limitations of the data set can result in sub-par policies and the agent can still reach an unsafe state if it exists the distribution of the data set. The demonstrator is unlikely to cover all potential safety–critical scenarios, and the reward function often does not prioritize safety. Safety is paramount in real-world applications, particularly in human–robot interaction settings [117,225].

One approach to safe RL is to maintain the agent within a safe distribution of states. For example, [226] proposes learning a manifold that captures natural variations in the environment and uses a secondary policy to guide the agent back into the distribution of visited states. The method in [227] introduces an advantage-based mechanism to determine when the recovery policy should intervene. Existing RL algorithms address safety by specifying constraints that the agent cannot violate during execution [47,68,69,193]. For instance, [228] proposes learning a barrier function that constrains the agent's policy to remain within a set of states that do not violate constraints. The method proposed in [179] initially finds a reward-optimal policy, which is then projected onto the feasible set of policies that satisfy the cost constraints. However, these constraints are often task-specific and agent-specific, making it impractical to specify all the constraints for every situation. Alternatively, [229] proposes a zero-sum game where a second player perturbs the agent's transition probabilities to optimize the worst-case transitions to produce a more robust policy. Methods in [48,174] require access to sets of safe and unsafe states to train the agent safely.

Active learning can address uncertainty. The DAgger framework trains a student policy by allowing it to query an expert policy [58], where it learns from data generated by both itself and the expert. Before each interaction with the environment, a decision rule decides whether to use the student's or the expert's policy. Since selecting the student's action early on can be unsafe, SafeDAgger [199] introduces a criterion that only allows the student to act if the difference between its action and the expert's action is below a threshold. EnsembleDAgger [181] further refines this approach by requiring the prediction of an ensemble of networks to be below a threshold for the student to act. However, these algorithms depend on the availability of an expert policy.

Some methods leverage expert demonstration data sets to combine the advantages of RL and demonstration learning. For instance, [80] proposes learning a Lyapunov function to ensure the agent's policy remains within the distribution of states from the data set. In [230], the authors use a pre-existing expert policy to filter the agent's action if it differs from the expert's.

10. Conclusion

We have presented a comprehensive survey on demonstration learning. Demonstration learning reduces the programming overhead by teaching the agent tasks through demonstrations. The paradigm comprises two main phases. The first phase involves collecting and building the demonstration data set, with key considerations including selecting the demonstrator, recording the demonstrations, representing the data, and addressing common data set limitations. The second phase focuses on extracting the behavior from the data set and training the agent accordingly.

The main challenge in demonstration learning is the generalization to unseen scenarios. Demonstrations only cover a subset of the distribution, and direct imitation through BC learns this limited distribution. When the agent encounters out-of-distribution cases, it may not know how to respond, potentially leading to catastrophic real-world consequences. To address this, offline RL methods aim to reduce the distributional shift. Various strategies and mitigations were discussed, including model learning methods that enable the agent to generate new transitions without interacting with the environment. Additionally, methods for refining the behavior were explored, such as trial-and-error interactions using RL, skill transfer through transfer learning, active teacher querying, and optimizing behavior with EAs.

Demonstration learning is predominantly used across various types of robots and video games, with potential applications extending to diverse domains such as healthcare and industry. The main advantages of demonstration learning include reduced reliance on expert programming, greater data efficiency than RL, and enhanced safety during the learning process. Despite these benefits, demonstration learning requires a good framework to create high-quality demonstration data sets and accurately estimate the policy.

Demonstration learning is a promising area within the broader field of ML. This survey has identified several key research challenges, highlighting that the development of robust benchmarks and high-quality data sets are crucial for the field's advancement. As noted in [18] much of the success of deep learning is due to large data sets. Even recent applications still utilize relatively old methods paired with improved models and large data sets. As explained in this chapter, many of the problems of current Demonstration learning methods stem from the limitations of the data sets. Creating more, better and larger data sets or pipelines for generating such data sets is the fundamental direction for the success of the field.

CRediT authorship contribution statement

André Correia: Writing – original draft, Investigation, Conceptualization. **Luís A. Alexandre:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported by NOVA LINCS ref. UIDB/04516/2020 (https://doi.org/10.54499/UIDB/04516/2020) and ref. UIDP/04516/2020 (https://doi.org/10.54499/UIDP/04516/2020) with the financial support of FCT.IP, and also through the research grant 2022.14197.BD.

References

[1] F. Codevilla, M. Müller, A. López, V. Koltun, A. Dosovitskiy, End-to-end driving via conditional imitation learning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 4693–4700.
[2] H. Wang, J. Chen, H. Lau, H. Ren, Motion planning based on learning from demonstration for multiple-segment flexible soft robots actuated by electroactive polymers, IEEE Robot. Autom. Lett. 1 (2016) 391–398.
[3] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al., Deep q-learning from demonstrations, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.
[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533.
[5] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[6] S. Fujimoto, D. Meger, D. Precup, Off-policy deep reinforcement learning without exploration, in: International Conference on Machine Learning, 2019, pp. 2052–2062.
[7] S. Schaal, Learning from demonstration, in: Advances in Neural Information Processing Systems, vol. 9, 1996.
[8] S. Schaal, Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences 3 (1999).
[9] P. Abbeel, A. Coates, A. Ng, Autonomous helicopter aerobatics through apprenticeship learning, Int. J. Robot. Res. 29 (2010) 1608–1639.
[10] A. Lee, A. Gupta, H. Lu, S. Levine, P. Abbeel, Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2015, pp. 5265–5272.
[11] A. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), vol. 2, 2002, pp. 1398–1403.
[12] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Survey: Robot Programming by Demonstration, Springer, 2008.
[13] B. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robot. Auton. Syst. 57 (2009) 469–483.
[14] A. Hussein, M. Gaber, E. Elyan, C. Jayne, Imitation learning: A survey of learning methods, ACM Comput. Surv. 50 (2017) 1–35.
[15] Z. Zhu, H. Hu, Robot learning from demonstration in robotic assembly: A survey, Robotics 7 (2018) 17.
[16] S. Arora, P. Doshi, A survey of inverse reinforcement learning: Challenges, methods and progress, Artif. Intell. 297 (2021) 103500.
[17] H. Ravichandar, A. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration, Annu. Rev. Control Robot. Autonom. Syst. 3 (2020) 297–330.
[18] S. Levine, A. Kumar, G. Tucker, J. Fu, Offline reinforcement learning: Tutorial, review, in: And Perspectives on Open Problems, vol. 5, 2020.
[19] R. Prudencio, M. Maximo, E. Colombini, A survey on offline reinforcement learning: Taxonomy, review, and open problems, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[20] S. Russell, Learning agents for uncertain environments, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, 1998, pp. 101–103.
[21] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[22] C. Nehaniv, K. Dautenhahn, et al., The correspondence problem, in: Imitation in Animals and Artifacts, vol. 41, 2002.
[23] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, F. Sun, Survey of imitation learning for robotic manipulation, Int. J. Intell. Robot. Appl. 3 (2019) 362–369.
[24] D. Grollman, O. Jenkins, Dogged learning for robots, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 2483–2488.
[25] S. Chernova, A. Thomaz, Robot Learning from Human Teachers, Morgan & Claypool Publishers, 2014.
[26] J. Laird, K. Gluck, J. Anderson, K. Forbus, O. Jenkins, C. Lebiere, D. Salvucci, M. Scheutz, A. Thomaz, G. Trafton, et al., Interactive task learning, IEEE Intell. Syst. 32 (2017) 6–21.
[27] A. Saran, E. Short, A. Thomaz, S. Niekum, Enhancing robot learning with human social cues, in: 2019 14th ACM/IEEE International Conference on Human-Robot Interaction, HRI, 2019, pp. 745–747.
[28] T. Kessler Faulkner, S. Niekum, A. Thomaz, Asking for help effectively via modeling of human beliefs, in: Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, 2018, pp. 149–150.
[29] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang, Autonomous inverted helicopter flight via reinforcement learning, in: Experimental Robotics IX: The 9th International Symposium on Experimental Robotics, 2006, pp. 363–372.
[30] J. Chen, A. Zelinsky, Programing by demonstration: Coping with suboptimal teaching actions, Int. J. Robot. Res. 22 (2003) 299–319.
[31] R. Aler, O. Garcia, J. Valls, Correcting and improving imitation models of humans for robosoccer agents, in: 2005 IEEE Congress on Evolutionary Computation, vol. 3, 2005, pp. 2402–2409.
[32] Y. Hristov, S. Ramamoorthy, Learning from demonstration with weakly supervised disentanglement, 2020, arXiv preprint arXiv:2006.09107.
[33] G. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, J. Peters, Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks, Auton. Robots 41 (2017) 593–612.
[34] Y. Shavit, N. Figueroa, S. Salehian, A. Billard, Learning augmented joint-space task-oriented dynamical systems: A linear parameter varying and synergetic control approach, IEEE Robot. Autom. Lett. 3 (2018) 2718–2725.
[35] C. Eteke, D. Kebüde, B. Akgün, Reward learning from very few demonstrations, IEEE Trans. Robot. 37 (2020) 893–904.
[36] M. Ogino, H. Toichi, M. Asada, Y. Yoshikawa, Imitation faculty based on a simple visuo-motor mapping towards interaction rule learning with a human partner. Proceedings, in: The 4th International Conference on Development and Learning, 2005, 2005, pp. 148–148.
[37] O. Akanyeti, U. Nehmzow, C. Weinrich, T. Kyriacou, S. Billings, Programming mobile robots by demonstration through system identification, in: European Conference on Mobile Robots, ECMR 2007, 2007.
[38] B. Hayes, B. Scassellati, Discovering task constraints through observation and active learning, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 4442–4449.
[39] Y. Liu, A. Gupta, P. Abbeel, 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, 2018, pp. 1118–1125.
[40] C. Atkeson, S. Schaal, Robot learning from demonstration, in: ICML, vol. 97, 1997, pp. 12–20.
[41] R. Dillmann, M. Kaiser, A. Ude, Acquisition of elementary robot skills from human demonstration, in: International Symposium on Intelligent Robotics Systems, 1995, pp. 185–192.
[42] M. Lopes, J. Santos-Victor, Visual learning by imitation with motor representations, IEEE Trans. Syst. Man Cybern. B 35 (2005) 438–449.
[43] M. Edmonds, F. Gao, X. Xie, H. Liu, S. Qi, Y. Zhu, B. Rothrock, S. Zhu, Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2017, pp. 3530–3537.
[44] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, M. Kawato, Learning from demonstration and adaptation of biped locomotion, Robot. Auton. Syst. 47 (2004) 79–91.
[45] P. Ruppel, J. Zhang, Learning object manipulation with dexterous hand-arm systems from human demonstration, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2020, pp. 5417–5424.
[46] J. Ho, S. Ermon, Generative adversarial imitation learning, in: Advances in Neural Information Processing Systems, vol. 29, 2016.
[47] Z. Liu, Z. Cen, V. Isenbaev, W. Liu, S. Wu, B. Li, D. Zhao, Constrained variational policy optimization for safe reinforcement learning, in: International Conference on Machine Learning, 2022, pp. 13644–13668.
[48] N. Wagener, B. Boots, C. Cheng, Safe reinforcement learning using advantage-based intervention, in: International Conference on Machine Learning, 2021, pp. 10630–10640.
[49] A. Billard, M. Matarić, Learning human arm movements by imitation:: Evaluation of a biologically inspired connectionist architecture, Robot. Auton. Syst. 37 (2001) 145–160.
[50] Y. Demiris, A. Dearden, From Motor Babbling to Hierarchical Learning by Imitation: A Robot Developmental Pathway, Lund University Cognitive Studies, 2005.
[51] S. Ross, G. Gordon, 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, 2011, pp. 627–635.
[52] X. Guo, S. Singh, H. Lee, R. Lewis, X. Wang, Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning, in: Advances in Neural Information Processing Systems, vol. 27, 2014.
[53] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (2016) 1334–1373.
[54] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, R. Fergus, Improving sample efficiency in model-free reinforcement learning from images, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 10674–10681.
[55] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, G. Brain, Time-contrastive networks: Self-supervised learning from video, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 1134–1141.
[56] K. Ramachandruni, M. Babu, A. Majumder, S. Dutta, S. Kumar, Attentive task-net: Self supervised task-attention network for imitation learning using video demonstration, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 4760–4766.

[57] S. Chernova, M. Veloso, Confidence-based policy learning from demonstration using Gaussian mixture models, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, 2007, pp. 1–8.

[58] S. Ross, D. Bagnell, Efficient reductions for imitation learning, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 661–668.

[59] S. Sinha, A. Mandlekar, A. Garg, S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics, in: Conference on Robot Learning, 2022, pp. 907–917.

[60] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, A. Srinivas, Reinforcement learning with augmented data, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 19884–19895.

[61] N. Hansen, H. Su, X. Wang, Stabilizing deep q-learning with convnets and vision transformers under data augmentation, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 3680–3693.

[62] K. Lee, M. Laskin, A. Srinivas, P. Abbeel, Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning, in: International Conference on Machine Learning, 2021, pp. 6131–6141.

[63] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, C. Finn, S. Levine, How to leverage unlabeled data in offline reinforcement learning, in: International Conference on Machine Learning, 2022, pp. 25611–25635.

[64] L. Zhu, Y. Cui, T. Matsubara, Dynamic actor-advisor programming for scalable safe reinforcement learning, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 10681–10687.

[65] L. Pinto, J. Davidson, R. Sukthankar, A. Gupta, Robust adversarial reinforcement learning, in: International Conference on Machine Learning, 2017, pp. 2817–2826.

[66] D. Yarats, R. Fergus, A. Lazaric, L. Pinto, Reinforcement learning with prototypical representations, in: International Conference on Machine Learning, 2021, pp. 11920–11931.

[67] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, D. Pathak, Planning to explore via self-supervised world models, in: International Conference on Machine Learning, 2020, pp. 8583–8592.

[68] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, Y. Tassa, Safe exploration in continuous action spaces, 2018, arXiv preprint arXiv:1801.08757.

[69] R. Cheng, G. Orosz, R. Murray, J. Burdick, End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3387–3395.

[70] G. Swamy, S. Choudhury, D. Bagnell, S. Wu, Causal imitation learning under temporally correlated noise, in: International Conference on Machine Learning, 2022, pp. 20877–20890.

[71] J. Aleotti, S. Caselli, Robust trajectory learning and approximation for robot programming by demonstration, Robot. Auton. Syst. 54 (2006) 409–413.

[72] P. Pook, D. Ballard, Recognizing teleoperated manipulations, in: [1993] Proceedings IEEE International Conference on Robotics and Automation, 1993, pp. 578–585.

[73] M. Kaiser, H. Friedrich, R. Dillmann, Obtaining good performance from a bad teacher, in: Programming by Demonstration Vs. Learning from Examples Workshop at ML, vol. 95, 1995.

[74] M. Hamaya, F. Drigalski, T. Matsubara, K. Tanaka, R. Lee, C. Nakashima, Y. Shibata, Y. Ijiri, Learning soft robotic assembly strategies from successful and failed demonstrations, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2020, pp. 8309–8315.

[75] B. Kim, A. Farahmand, J. Pineau, D. Precup, Learning from limited demonstrations, in: Advances in Neural Information Processing Systems, vol. 26, 2013.

[76] D. Grollman, A. Billard, Robot learning from failed demonstrations, Int. J. Soc. Robot. 4 (2012) 331–342.

[77] H. Guo, Q. Cai, Y. Zhang, Z. Yang, Z. Wang, Provably efficient offline reinforcement learning for partially observable Markov decision processes, in: International Conference on Machine Learning, 2022, pp. 8016–8038.

[78] M. Beliaev, A. Shih, S. Ermon, D. Sadigh, R. Pedarsani, Imitation learning by estimating expertise of demonstrators, in: International Conference on Machine Learning, 2022, pp. 1732–1748.

[79] A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative q-learning for offline reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1179–1191.

[80] K. Kang, P. Gradu, J. Choi, M. Janner, C. Tomlin, S. Levine, Lyapunov density models: Constraining distribution shift in learning-based control, in: International Conference on Machine Learning, 2022, pp. 10708–10733.

[81] L. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, Mach. Learn. 8 (1992) 293–321.

[82] T. Inamura, M. Inaba, H. Inoue, Integration model of learning mechanism and dialogue strategy based on stochastic experience representation using Bayesian network, in: Proceedings 9th IEEE International Workshop on Robot and Human Interactive Communication. IEEE RO-MAN 2000 (Cat. No. 00TH8499), 2000, pp. 247–252.

[83] J. Saunders, C. Nehaniv, K. Dautenhahn, Teaching robots by moulding behavior and scaffolding the environment, in: Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human–Robot Interaction, 2006, pp. 118–125.

[84] S. Raza, S. Haider, M. Williams, Teaching coordinated strategies to soccer robots via imitation, in: 2012 IEEE International Conference on Robotics and Biomimetics, ROBIO, 2012, pp. 1434–1439.

[85] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, Learning manipulation trajectories using recurrent neural networks, 2016, arXiv Preprint arXiv:1603.03833.

[86] R. Dadashi, L. Hussenot, D. Vincent, S. Girgin, A. Raichuk, M. Geist, O. Pietquin, Continuous control with action quantization from demonstrations, 2021, arXiv preprint arXiv:2110.10149.

[87] J. Kober, J. Peters, Learning motor primitives for robotics, in: 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 2112–2118.

[88] S. Yang, W. Zhang, W. Lu, H. Wang, Y. Li, Cross-context visual imitation learning from demonstrations, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 5467–5473.

[89] P. Gu, M. Zhao, C. Chen, D. Li, J. Hao, B. An, Learning pseudometric-based action representations for offline reinforcement learning, in: Proceedings of the 39th International Conference on Machine Learning, vol. 162, 2022, pp. 7902–7918.

[90] M. Janner, J. Fu, M. Zhang, S. Levine, When to trust your model: Model-based policy optimization, in: Advances in Neural Information Processing Systems, vol. 32, 2019.

[91] R. Kidambi, A. Rajeswaran, P. Netrapalli, T. Joachims, Morel: Model-based offline reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 21810–21823.

[92] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, T. Ma, Mopo: Model-based offline policy optimization, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 14129–14142.

[93] T. Matsushima, H. Furuta, Y. Matsuo, O. Nachum, S. Gu, Deployment-efficient reinforcement learning via model-based offline optimization, 2020, arXiv preprint arXiv:2006.03647.

[94] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, C. Finn, Combo: Conservative offline model-based policy optimization, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 28954–28967.

[95] M. Farajtabar, Y. Chow, M. Ghavamzadeh, More robust doubly robust off-policy evaluation, in: International Conference on Machine Learning, 2018, pp. 1447–1456.

[96] P. Thomas, E. Brunskill, Data-efficient off-policy policy evaluation for reinforcement learning, in: International Conference on Machine Learning, 2016, pp. 2139–2148.

[97] Y. Wang, A. Agarwal, M. Dudık, Optimal and adaptive off-policy evaluation in contextual bandits, in: International Conference on Machine Learning, 2017, pp. 3589–3597.

[98] R. Rafailov, T. Yu, A. Rajeswaran, C. Finn, Offline reinforcement learning from images with latent space models, Learn. Dynam. Control (2021) 1154–1168.

[99] P. Norvig, S. Intelligence, A modern approach. Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage, Knowledge-Based Syst. 90 (2015) 33–48, 2002.

[100] D. Silver, J. Bagnell, A. Stentz, Learning from demonstration for autonomous navigation in complex unstructured terrain, Int. J. Robot. Res. 29 (2010) 1565–1592.

[101] B. Ziebart, A. Maas, J. Bagnell, A. Dey, et al., Maximum entropy inverse reinforcement learning, Aaai 8 (2008) 1433–1438.

[102] K. Mülling, J. Kober, O. Kroemer, J. Peters, Learning to select and generalize striking movements in robot table tennis, Int. J. Robot. Res. 32 (2013) 263–279.

[103] E. Klein, M. Geist, B. Piot, O. Pietquin, Inverse reinforcement learning through structured classification, Adv. Neural Inf. Process. Syst. 25 (2012).

[104] E. Klein, B. Piot, M. Geist, O. Pietquin, A cascaded supervised learning approach to inverse reinforcement learning, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September (2013) 23-27, Proceedings, Part I 13, 2013, pp. 1–16.

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

[106] H. Suay, T. Brys, M. Taylor, S. Chernova, Learning from demonstration for shaping through inverse reinforcement learning, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, 2016, pp. 429–437.

[107] T. Brys, A. Harutyunyan, H. Suay, S. Chernova, M. Taylor, A. Nowé, Reinforcement learning from demonstration through shaping, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[108] M. Zhang, Z. McCarthy, C. Finn, S. Levine, P. Abbeel, Learning deep neural network policies with continuous memory states, in: 2016 IEEE International Conference on Robotics and Automation, ICRA, 2016, pp. 520–527.

[109] Y. Zhang, S. Wang, G. Ji, et al., A comprehensive survey on particle swarm optimization algorithm and its applications, Math. Probl. Eng. 2015 (2015).

[110] C. Zhang, Z. Zhen, D. Wang, M. Li, UAV path planning method based on ant colony optimization, in: 2010 Chinese Control and Decision Conference, 2010, pp. 3790–3792.

[111] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inform. Sci. 291 (2015) 43–60.

[112] J. Bongard, G. Hornby, Combining fitness-based search and user modeling in evolutionary robotics, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, 2013, pp. 159–166.
[113] T. Brys, A. Harutyunyan, M. Taylor, A. Nowé, Policy transfer using reward shaping, in: AAMAS, 2015, pp. 181–188.
[114] G. Kuhlmann, P. Stone, Graph-based domain mapping for transfer learning in general games, in: Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21 2007. Proceedings 18, 2007, pp. 188–200.
[115] L. Torrey, T. Walker, J. Shavlik, R. Maclin, Using advice to transfer knowledge acquired in one reinforcement learning task to another, in: Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7 2005. Proceedings 16, 2005, pp. 412–424.
[116] D. Ghosh, A. Ajay, P. Agrawal, S. Levine, Offline RL policies should be trained to be adaptive, in: International Conference on Machine Learning, 2022, pp. 7513–7530.
[117] S. Ikemoto, H. Amor, T. Minato, B. Jung, H. Ishiguro, Physical human–robot interaction: Mutual learning and adaptation, IEEE Robot. Autom. Mag. 19 (2012) 24–35.
[118] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A. Bharath, Generative adversarial networks: An overview, IEEE Signal Process. Mag. 35 (2018) 53–65.
[119] Z. Liu, Y. Zhang, Z. Fu, Z. Yang, Z. Wang, Learning from demonstration: Provably efficient adversarial policy imitation with linear function approximation, in: International Conference on Machine Learning, 2022, pp. 14094–14138.
[120] H. Xu, X. Zhan, H. Yin, H. Qin, Discriminator-weighted offline imitation learning from suboptimal demonstrations, in: International Conference on Machine Learning, 2022, pp. 24725–24742.
[121] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, M. Hutter, Advanced skills through multiple adversarial motion priors in reinforcement learning, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, 2023, pp. 5120–5126.
[122] L. Blondé, A. Kalousis, Sample-efficient imitation learning via generative adversarial nets, in: The 22nd International Conference on Artificial Intelligence and Statistics, 2019, pp. 3138–3148.
[123] L. Yu, T. Yu, C. Finn, S. Ermon, Meta-inverse reinforcement learning with probabilistic context variables, in: Advances in Neural Information Processing Systems, vol. 32, 2019.
[124] B. Wu, F. Xu, Z. He, A. Gupta, P. Allen, Squirl: Robust and efficient learning from video demonstration of long-horizon robotic manipulation tasks, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2020, pp. 9720–9727.
[125] M. Godbout, M. Heuillet, S. Raparthy, R. Bhati, A. Durand, A game-theoretic perspective on risk-sensitive reinforcement learning., in: SafeAI@ AAAI, 2022.
[126] C. Cheng, T. Xie, N. Jiang, A. Agarwal, Adversarially trained actor critic for offline reinforcement learning, in: International Conference on Machine Learning, 2022, pp. 3852–3878.
[127] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, 2020, pp. 1597–1607.
[128] I. Melekhov, J. Kannala, E. Rahtu, Siamese network features for image matching, in: 2016 23rd International Conference on Pattern Recognition, ICPR, 2016, pp. 378–383.
[129] A. Correia, L. Alexandre, Multi-view contrastive learning from demonstrations, in: 2022 Sixth IEEE International Conference on Robotic Computing, IRC, 2022, pp. 338–344.
[130] A. Stooke, K. Lee, P. Abbeel, M. Laskin, Decoupling representation learning from reinforcement learning, in: International Conference on Machine Learning, 2021, pp. 9870–9879.
[131] M. Laskin, A. Srinivas, P. Abbeel, Curl: Contrastive unsupervised representations for reinforcement learning, in: International Conference on Machine Learning, 2020, pp. 5639–5650.
[132] D. Ghosh, A. Gupta, S. Levine, Learning actionable representations with goal-conditioned policies, 2018, arXiv preprint arXiv:1811.07819.
[133] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, A. Zisserman, Temporal cycle-consistency learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1801–1810.
[134] G. Berseth, F. Golemo, C. Pal, Towards learning to imitate from a single video demonstration, 2019, arXiv preprint arXiv:1901.07186.
[135] Y. Tian, D. Krishnan, P. Isola, Contrastive multiview coding, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28 2020, Proceedings, Part XI 16, 2020, pp. 776–794.
[136] M. Laskin, H. Liu, X. Peng, D. Yarats, A. Rajeswaran, P. Abbeel, Cic: Contrastive intrinsic control for unsupervised skill discovery, 2022, arXiv Preprint arXiv:2202.00161.
[137] P. Hansen-Estruch, A. Zhang, A. Nair, P. Yin, S. Levine, Bisimulation makes analogies in goal-conditioned reinforcement learning, in: International Conference on Machine Learning, 2022, pp. 8407–8426.
[138] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, vol. 30, 2017.
[139] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, I. Mordatch, Decision transformer: Reinforcement learning via sequence modeling, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 15084–15097.
[140] M. Janner, Q. Li, S. Levine, Offline reinforcement learning as one big sequence modeling problem, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 1273–1286.
[141] A. Correia, L. Alexandre, Hierarchical decision transformer, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2023, pp. 1661–1666.
[142] Q. Zheng, A. Zhang, A. Grover, Online decision transformer, in: International Conference on Machine Learning, 2022, pp. 27042–27059.
[143] M. Reid, Y. Yamada, S. Gu, Can Wikipedia help offline reinforcement learning? 2022, arXiv Preprint arXiv:2201.12122.
[144] H. Hsu, A. Bozkurt, J. Dong, Q. Gao, V. Tarokh, M. Pajic, Steering decision transformers via temporal difference learning.
[145] A. Villaflor, Z. Huang, S. Pande, J. Dolan, J. Schneider, Addressing optimism bias in sequence modeling for reinforcement learning, in: International Conference on Machine Learning, 2022, pp. 22270–22283.
[146] A. Gu, K. Goel, C. Ré, Efficiently modeling long sequences with structured state spaces, 2021, arXiv Preprint arXiv:2111.00396.
[147] A. Gu, T. Dao, Mamba: Linear-time sequence modeling with selective state spaces, 2023, arXiv Preprint arXiv:2312.00752.
[148] R. Bhirangi, C. Wang, V. Pattabiraman, C. Majidi, A. Gupta, T. Hellebrekers, L. Pinto, Hierarchical state space models for continuous sequence-to-sequence modeling, 2024, arXiv preprint arXiv:2402.10211.
[149] A. Kumar, J. Fu, M. Soh, G. Tucker, S. Levine, Stabilizing off-policy q-learning via bootstrapping error reduction, in: Advances in Neural Information Processing Systems, vol. 32, 2019.
[150] Y. Wu, G. Tucker, O. Nachum, Behavior regularized offline reinforcement learning, 2019, arXiv preprint arXiv:1911.11361.
[151] X. Peng, A. Kumar, G. Zhang, S. Levine, Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019, arXiv preprint arXiv:1910.00177.
[152] A. Nair, A. Gupta, M. Dalal, S. Levine, Awac: Accelerating online reinforcement learning with offline datasets, 2020, arXiv preprint arXiv:2006.09359.
[153] S. Fujimoto, S. Gu, A minimalist approach to offline reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 20132–20145.
[154] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, 2018, pp. 1861–1870.
[155] R. Agarwal, D. Schuurmans, M. Norouzi, An optimistic perspective on offline reinforcement learning, in: International Conference on Machine Learning, 2020, pp. 104–114.
[156] S. Schaal, A. Ijspeert, A. Billard, Computational approaches to motor learning by imitation, Philos. Trans. R. Soc. Lond. Ser. B 358 (2003) 537–547.
[157] T. Paine, C. Paduraru, A. Michi, C. Gulcehre, K. Zolna, A. Novikov, Z. Wang, N. Freitas, Hyperparameter selection for offline reinforcement learning, 2020, arXiv preprint arXiv:2007.09055.
[158] R. Zhang, B. Dai, L. Li, D. Schuurmans, Gendice: Generalized offline estimation of stationary values, 2020, arXiv preprint arXiv:2002.09072.
[159] J. Ortega, N. Shaker, J. Togelius, G. Yannakakis, Imitating human playing styles in super Mario Bros, Entertain. Comput. 4 (2013) 93–104.
[160] I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepesvari, S. Singh, et al., Behaviour suite for reinforcement learning, 2019, arXiv preprint arXiv:1908.03568.
[161] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al., Deepmind control suite, 2018, arXiv preprint arXiv:1801.00690.
[162] C. Beattie, J. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al., Deepmind lab, 2016, arXiv preprint arXiv:1612.03801.
[163] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, et al., Google research football: A novel reinforcement learning environment, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 4501–4510.
[164] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, S. Levine, Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, in: Conference on Robot Learning, 2020, pp. 1094–1100.
[165] G. Dulac-Arnold, N. Levine, D. Mankowitz, J. Li, C. Paduraru, S. Gowal, T. Hester, An empirical investigation of the challenges of real-world reinforcement learning, 2020, arXiv preprint arXiv:2003.11881.
[166] J. Fu, A. Kumar, O. Nachum, G. Tucker, S. Levine, D4rl: Datasets for deep data-driven reinforcement learning, 2020, arXiv preprint arXiv:2004.07219.
[167] C. Gulcehre, Z. Wang, A. Novikov, T. Paine, S. Gómez, K. Zolna, R. Agarwal, J. Merel, D. Mankowitz, C. Paduraru, et al., Rl unplugged: A suite of benchmarks for offline reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 7248–7259.

[168] M. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: An evaluation platform for general agents, J. Artificial Intelligence Res. 47 (2013) 253–279.
[169] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, Y. Tassa, dm_control: Software and tasks for continuous control, Softw. Impacts 6 (2020) 100022.
[170] W. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, R. Salakhutdinov, Minerl: A large-scale dataset of minecraft demonstrations, 2019, arXiv preprint arXiv:1907.13440.
[171] 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, 2018, pp. 879–893.
[172] J. Fu, M. Norouzi, O. Nachum, G. Tucker, Z. Wang, A. Novikov, M. Yang, M. Zhang, Y. Chen, A. Kumar, et al., Benchmarks for deep off-policy evaluation, 2021, arXiv preprint arXiv:2103.16596.
[173] P. Abbeel, A. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the Twenty-First International Conference on Machine Learning, vol. 1, 2004.
[174] C. He, B. León, F. Belardinelli, Do androids dream of electric fences? Safety-aware reinforcement learning with latent shielding, 2021, arXiv Preprint arXiv:2112.11490.
[175] V. Patil, M. Hofmarcher, M. Dinu, M. Dorfer, P. Blies, J. Brandstetter, J. Arjona-Medina, S. Hochreiter, Align-rudder: Learning from few demonstrations by reward redistribution, 2020, arXiv preprint arXiv:2009.14108.
[176] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, S. Levine, C. Finn, Conservative data sharing for multi-task offline reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 11501–11516.
[177] M. Bansal, A. Krizhevsky, A. Ogale, Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst, 2018, arXiv preprint arXiv:1812.03079.
[178] E. Johns, Coarse-to-fine imitation learning: Robot manipulation from a single demonstration, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, 2021, pp. 4613–4619.
[179] N. Polosky, B. Da Silva, M. Fiterau, J. Jagannath, Constrained offline policy optimization, in: International Conference on Machine Learning, 2022, pp. 17801–17810.
[180] H. Yuan, Z. Lu, Robust task representations for offline meta-reinforcement learning via contrastive learning, in: International Conference on Machine Learning, 2022, pp. 25747–25759.
[181] K. Menda, K. Driggs-Campbell, M. Kochenderfer, Ensembledagger: A Bayesian approach to safe imitation learning, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2019, pp. 5041–5048.
[182] D. Yarats, D. Brandfonbrener, H. Liu, M. Laskin, P. Abbeel, A. Lazaric, L. Pinto, Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning, 2022, arXiv preprint arXiv:2201.13425.
[183] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, L. Fei-Fei, Learning to generalize across long-horizon tasks from human demonstrations, 2021.
[184] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, F. 0001, A. Garg, D. Fox, IRIS: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data, in: 2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31 2020, 2020, pp. 4414–4420.
[185] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. Brown, D. Seita, B. Thananjeyan, E. Novoseller, K. Goldberg, Lazydagger: Reducing context switching in interactive imitation learning, in: 2021 IEEE 17th International Conference on Automation Science and Engineering, CASE, 2021, pp. 502–509.
[186] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, Int. J. Robot. Res. 37 (2018) 421–436.
[187] S. Paul, J. Vanbaar, A. Roy-Chowdhury, Learning from trajectories via subgoal discovery, in: Advances in Neural Information Processing Systems, 32, 2019.
[188] Y. Zhou, Y. Aytar, K. Bousmalis, Manipulator-independent representations for visual imitation, 2021, arXiv Preprint arXiv:2103.09016.
[189] A. Tanwani, A. Yan, J. Lee, S. Calinon, K. Goldberg, Sequential robot imitation learning from observations, Int. J. Robot. Res. 40 (2021) 1306–1325.
[190] T. Xiao, I. Radosavovic, T. Darrell, J. Malik, Masked visual pre-training for motor control, 2022, arXiv Preprint arXiv:2203.06173.
[191] Y. Pan, C. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, B. Boots, Agile autonomous driving using end-to-end deep imitation learning, 2017, arXiv Preprint arXiv:1709.07174.
[192] L. Pinto, A. Gupta, Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours, in: 2016 IEEE International Conference on Robotics and Automation, ICRA, 2016, pp. 3406–3413.
[193] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. Gonzalez, J. Ibarz, C. Finn, K. Goldberg, Recovery RL: Safe reinforcement learning with learned recovery zones, IEEE Robot. Autom. Lett. 6 (2021).
[194] T. Faulkner, E. Short, A. Thomaz, Interactive reinforcement learning with inaccurate feedback, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 7498–7504.
[195] N. Rhinehart, R. McAllister, S. Levine, Deep imitative models for flexible inference, planning, and control, 2018, arXiv preprint arXiv:1810.06544.
[196] E. Chane-Sane, C. Schmid, I. Laptev, Goal-conditioned reinforcement learning with imagined subgoals, in: International Conference on Machine Learning, 2021, pp. 1430–1440.
[197] A. Gupta, V. Kumar, C. Lynch, S. Levine, K. Hausman, Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning, 2019, arXiv preprint arXiv:1910.11956.
[198] J. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, S. Hochreiter, Rudder: Return decomposition for delayed rewards, in: Advances in Neural Information Processing Systems, vol. 32, 2019.
[199] J. Zhang, K. Cho, Query-efficient imitation learning for end-to-end simulated driving, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, 2017.
[200] A. Raghu, M. Komorowski, I. Ahmed, L. Celi, P. Szolovits, M. Ghassemi, Deep reinforcement learning for sepsis treatment, 2017, arXiv Preprint arXiv:1711.09602.
[201] K. Pertsch, Y. Lee, J. Lim, Accelerating reinforcement learning with learned skill priors, in: Conference on Robot Learning, CoRL, 2021, pp. 188–204.
[202] L. Wang, W. Zhang, X. He, H. Zha, Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2447–2456.
[203] S. Krishnan, A. Garg, R. Liaw, B. Thananjeyan, L. Miller, F. Pokorny, K. Goldberg, SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards, Int. J. Robot. Res. 38 (2019) 126–145.
[204] J. Shang, M. Ryoo, Self-supervised disentangled representation learning for third-person imitation learning, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2021, pp. 214–221.
[205] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, S. Levine, Visual foresight: Model-based deep reinforcement learning for vision-based robotic control, 2018, arXiv preprint arXiv:1812.00568.
[206] S. Yang, Y. Feng, S. Zhang, M. Zhou, Regularizing a model-based policy stationary distribution to stabilize offline reinforcement learning, in: International Conference on Machine Learning, 2022, pp. 24980–25006.
[207] R. Bemelmans, G. Gelderblom, P. Jonker, L. De Witte, Socially assistive robots in elderly care: A systematic review into effects and effectiveness, J. Am. Med. Directors Assoc. 13 (2012) 114–120.
[208] C. Sammut, S. Hurst, D. Kedzier, D. Michie, Learning to fly, in: Machine Learning Proceedings 1992, 1992, pp. 385–393.
[209] K. Mo, H. Li, Z. Lin, J. Lee, The adobeindoornav dataset: Towards deep reinforcement learning based real-world indoor robot visual navigation, 2018, arXiv preprint arXiv:1802.08824.
[210] W. Maddern, G. Pascoe, C. Linegar, P. Newman, 1 Year, 1000 km: The Oxford robotcar dataset, Int. J. Robot. Res. 36 (2017) 3–15.
[211] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, T. Darrell, et al., Bdd100k: A diverse driving video database with scalable annotation tooling, 2018, p. 6, 2, arXiv preprint arXiv:1805.04687.
[212] M. Bojarski, D.Del. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, 2016, arXiv Preprint arXiv:1604.07316.
[213] K. Strabala, M. Lee, A. Dragan, J. Forlizzi, S. Srinivasa, M. Cakmak, V. Micelli, Toward seamless human–robot handovers, J. Hum.-Robot Interact. 2 (2013) 112–132.
[214] O. Gottesman, F. Johansson, M. Komorowski, A. Faisal, D. Sontag, F. Doshi-Velez, L. Celi, Guidelines for reinforcement learning in healthcare, Nat. Med. 25 (2019) 16–18.
[215] H. Tseng, Y. Luo, S. Cui, J. Chien, R.Ten. Haken, I. Naqa, Deep reinforcement learning for automated radiation adaptation in lung cancer, Med. Phys. 44 (2017) 6690–6705.
[216] A. Johnson, T. Pollard, L. Shen, L. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L.Anthony. Celi, R. Mark, MIMIC-III, a freely accessible critical care database, Sci. Data 3 (2016) 1–9.
[217] D. Vogt, H. Ben Amor, E. Berger, B. Jung, Learning two-person interaction models for responsive synthetic humanoids, J. Virtual Real. Broadcastings 11 (2014).
[218] S. Calinon, A. Billard, Incremental learning of gestures by imitation in a humanoid robot, in: Proceedings of the ACM/IEEE International Conference on Human–Robot Interaction, 2007, pp. 255–262.
[219] A. Ude, C. Atkeson, M. Riley, Programming full-body movements for humanoid robots by observation, Robot. Auton. Syst. 47 (2004) 93–108.
[220] P. Hingston, Believable Bots: Can Computers Play Like People? Springer, 2012.
[221] V. Kurenkov, S. Kolesnikov, Showing your offline reinforcement learning work: Online evaluation budget matters, in: International Conference on Machine Learning, 2022, pp. 11729–11752.
[222] C. Voloshin, H. Le, N. Jiang, Y. Yue, Empirical study of off-policy policy evaluation for reinforcement learning, 2019, arXiv preprint arXiv:1911.06854.
[223] O. Nachum, S. Gu, H. Lee, S. Levine, Data-efficient hierarchical reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 31, 2018.

[224] K. Pertsch, O. Rybkin, F. Ebert, S. Zhou, D. Jayaraman, C. Finn, S. Levine, Long-horizon visual planning with goal-conditioned hierarchical predictors, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 17321–17333.
[225] A. De Santis, B. Siciliano, A. De Luca, A. Bicchi, An Atlas of physical human–robot interaction, Mech. Mach. Theory 43 (2008) 253–270.
[226] A. Reichlin, G. Marchetti, H. Yin, A. Ghadirzadeh, D. Kragic, Back to the manifold: Recovering from out-of-distribution states, in: International Conference on Intelligent Robots and Systems, IROS, 2022.
[227] N. Wagener, B. Boots, C. Cheng, Safe reinforcement learning using advantage-based intervention, in: International Conference on Machine Learning, 2021.
[228] R. Cheng, G. Orosz, R. Murray, J. Burdick, End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019.
[229] M. Godbout, M. Heuillet, S. Raparthy, R. Bhati, A. Durand, A game-theoretic perspective on risk-sensitive reinforcement learning., in: SafeAI@ AAAI, 2022.
[230] S. Ross, G. Gordon, 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, 2011.

André Correia received his B.Sc. (2018), and M.Sc. (2020) from the University of Beira Interior (UBI). His research interests are reinforcement learning, demonstration learning, computer vision, and their applications. Is currently doing his Ph.D. in computer engineering at UBI under the research grant 'FCT - Fundação para a Ciência e Tecnologia 2022.14197.BD'. His Ph.D. research topic is focused on improving the robustness of demonstration learning algorithms.

Luís A. Alexandre received his B.Sc. (1994), M.Sc. (1997) and Ph.D. (2002) from the University of Porto, and an Habilitation on Computer Science and Engineering (2013) by UBI. His research interests are neural networks, computer vision and their applications, particularly to robotics. He has been PI of several research projects with total budgets over 2Me, involving both academia and industry, has co-chaired 2 international conferences, participated in more than 50 international conference program committees and has been an evaluator for the A3ES (Portuguese agency for the Quality Assurance in Higher Education). He was also a member of the governing board of both the International Association for Pattern Recognition and of the European Neural Network Society and the president of the Portuguese Association for Pattern Recognition.