

A Survey of Imitation Learning Methods, Environments and Metrics

NATHAN GAVENSKI, King's College London, United Kingdom

FELIPE MENEGUZZI, University of Aberdeen, United Kingdom

MICHAEL LUCK, University of Sussex, United Kingdom

ODINALDO RODRIGUES, King's College London, United Kingdom

Imitation learning is an approach in which an agent learns how to execute a task by trying to mimic how one or more *teachers* perform it. This learning approach offers a compromise between the time it takes to learn a new task alone by trial-and-error and the effort needed to collect the large number of samples needed for supervised learning. The idea is to focus on the samples provided by the teacher, who is assumed to be proficient at executing the task, deviating from the samples only when faced with unseen situations. This is especially useful when we want to imbue an agent with a desired behaviour that is difficult to specify through a particular reward function. As a result, the field of imitation learning has received considerable attention from researchers, resulting in many new techniques and applications. This sudden increase in publications poses some challenges when it comes to comparing work and evaluating results in a uniform manner. In this survey, we tackle these challenges by systematically reviewing the imitation learning literature and presenting our findings by (i) introducing novel taxonomies which focus on critical aspects of the imitation learning methods, environments and metrics; (ii) reflecting and discussing important issues observed across publications; and (iii) highlighting open challenges and suggesting directions for future work. This survey provides a solid foundation for understanding the increasingly diverse and dynamic imitation learning field. It allows the development of new approaches and a more systematic and uniform comparison of all techniques.

CCS Concepts: • Computing methodologies → Learning paradigms; Machine learning approaches.

Additional Key Words and Phrases: Imitation Learning, Inverse Reinforcement Learning, Behavioural Cloning

ACM Reference Format:

Nathan Gavenski, Felipe Meneguzzi, Michael Luck, and Odinaldo Rodrigues. 2024. A Survey of Imitation Learning Methods, Environments and Metrics. *Proc. ACM Meas. Anal. Comput. Syst.* 37, 4, Article 111 (August 2024), 35 pages. <https://doi.org/XXXXXX.XXXXXXX>

1 Introduction

Imitation learning is a socially-inspired machine learning approach that consists of learning a task from examples provided by a teacher. In other words, this approach refers to an agent's acquisition of skills or behaviours by observing a teacher performing a given task [47]. This approach benefits the agent since it learns from samples that theoretically indicate a successful behaviour instead of searching which behaviours solve the task (a process that trial-and-error learning techniques have to go through). Moreover, this idea from imitation learning is compelling since, when learning something new, the learner tries to learn from a proficient source and adopt more expert-like behaviour, which allows for more human-centric approaches.

Authors' Contact Information: **Nathan Gavenski**, nathan.schneider_gavenski@kcl.ac.uk, King's College London, London, United Kingdom; **Felipe Meneguzzi**, University of Aberdeen, Aberdeen, United Kingdom; **Michael Luck**, University of Sussex, Sussex, United Kingdom; **Odinaldo Rodrigues**, King's College London, London, United Kingdom.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 2476-1249/2024/8-ART111

<https://doi.org/XXXXXX.XXXXXXX>

In recent years, imitation learning evolved significantly from Pomerleau's original work on behavioural cloning [80]. Some work expanded the application of imitation learning with more complex learner structures, such as ensemble methods [51]. Other work proposed novel learning schematics, such as adversarial learning [44] and self-supervised learning [102]. These approaches rely on machine learning techniques to optimise an agent's behaviour, while other researchers focused on agent techniques by proposing the usage of exploration [38] and remapping of latent actions [29]. Given these improvements, imitation learning has seen many applications, including robotics [35, 59, 64, 73, 76, 77, 82, 97, 117, 120], game-playing [5, 29, 38, 42, 55, 69, 74, 77, 79, 87, 93, 94, 119, 122], and natural language processing [27, 56, 66, 81, 111].

With this increase in popularity, imitation learning has seen a rise in the number of experimental environments and performance metrics, as we show in this survey. These new metrics and environments, along with other surveys [47, 105, 121] focusing more on learning techniques, create a lack of standardisation in the evaluation process, making meaningful comparisons between different approaches challenging. Hence, this survey examines the field from a different perspective from the traditional model-free and model-based views employed in the past. It presents the field based on the most predominant learning techniques, metrics and environments, including the first taxonomies for environments and metrics and a new methodological one. We believe that our taxonomy for methods does not nullify existing methodological taxonomies but rather complements them by classifying novel trends in imitation learning methods in more detail. This paper thus acknowledges prior contributions from previous surveys and incorporates them when appropriate or necessary.

We first describe the selection criteria for including the various research efforts in the systematic review and highlight the main differences with previous reviews in Section 2. Section 3 lays out the formal underpinning of imitation learning, defining the critical elements involved in imitation learning tasks, and allowing our discussion to remain mathematically rigorous. We then present the different approaches to imitation learning in Section 4, introducing a new taxonomy to highlight a new trend among learning approaches. In Section 5, we present the different environments used in imitation learning and propose the first taxonomy for imitation learning environments based on the *role* of the environment in the learning process. Section 6 explains the different metrics used to evaluate imitation learning approaches, groups them based on each of their measures, and presents the first taxonomy for imitation learning metrics. In Section 7, we discuss key insights derived from the literature, pose key challenges for the field, and possible future directions for imitation learning. Finally, Section 8 summarises our main contributions by emphasising the need for consistent use of evaluation processes and environment and how researchers can use these new taxonomies to achieve a more solid and systematic approach.

2 Review preliminaries

In this work, we use a snowballing approach [115] and two different surveys: Hussein et al. [47] from 2017 and Zheng et al. [121] from 2021 as a starting point.

We select Hussein et al.'s work since it cites significant work on imitation learning and defines various approaches used to learn from demonstrations. With more than 800 citations, the snowballing process is bound to find the most relevant work to our desired subject. Given that the first survey is older, we add Zheng et al. as a more recent survey. By adding a more novel survey, we are more assured of finding relevant work that we might miss from an older survey in a snowballing process. To select all relevant work, we follow two general guidelines when filtering all papers found during the snowballing process. The first criterion is that all work must be from the inverse reinforcement learning or imitation learning literature because, historically, imitation learning approaches appeared in the reinforcement learning literature. Furthermore, imitation learning approaches that use inverse reinforcement learning require teacher demonstrations to be considered imitation learning, we do not exclude any inverse reinforcement learning method without further inspection. The second inclusion criterion refers to the publishing venues. We only use published peer-reviewed work, except for pre-prints published since

2023, given the slow nature of the rigorous review process for machine learning publication. This exception provides just one addition (Pérez-Dattari and Kober [82]). Although these rules set a general framework for selecting relevant work for our literature review, they are just guidelines that should not be seen as strict rules.

We use abstracts to remove all work that uses imitation learning outside the context of agent-related research. Examples of this are papers on meta-learning, which use imitation learning techniques applied to other domains, such as neural networks training [58]. From the later survey [121], we present 30 new methods, which shows the fast-growing nature of the imitation learning field. For completeness, we include other surveys (which others [105] omit) since imitation learning takes inspiration from other fields, such as reinforcement learning [2], multi-agent learning [13, 26] and transfer learning [75, 123].

To help bring some uniformity to our review, we deviate slightly in approach from previous surveys and create a unifying taxonomy under which different categories of environments and metrics can be understood and compared. Hence, we consider the field from a different perspective to the traditional model-free and model-based view employed by Torabi et al. [105]. We attribute the lack of standardisation in the imitation learning field to the lack of a taxonomy for imitation learning environments and metrics, which is the main reason for us to deviate from past surveys. In Hussein et al. [47], the authors mainly focused on methodologies, given the need for classifying new approaches from the literature. They set a background for future work, from which we take inspiration, and introduce relevant research following their learning approach, such as apprenticeship learning, behavioural cloning, and inverse reinforcement learning. We follow a similar approach for our classification of imitation learning methods, in which we present a taxonomy based on the most predominant learning techniques during training, such as inverse dynamics models and adversarial learning. Given its simplistic nature for classifying imitation learning methods, we deviate from Zheng et al., who classify each approach as either a behavioural cloning, inverse reinforcement learning or adversarial imitation learning approach or as high-level (which requires higher-level cognitive functions) or low-level (concrete operations, such as moving an object) tasks. Moreover, we maintain that our classification system not only acknowledges their taxonomy but also enhances it by providing a more detailed categorisation of emerging trends within imitation learning methods.

Finally, we present a novel taxonomy for imitation learning environments based on the role of the environment in the learning process and a taxonomy for imitation learning metrics based on the meaning they convey. Although Hussein et al. and Zheng et al. discuss different imitation learning environments, they do not classify them and only group them by domain, which provides no insight into the environment's role in the learning process. Similarly to environments, Hussein et al. present different metrics for evaluating imitation learning methods. However, their work does not classify them nor present in-depth metrics analysis.

3 Background

Imitation learning refers to an *agent*'s acquisition of skills or behaviours by observing a source performing a given task [47]. In the imitation learning context, we follow Russell and Norvig's definition [89, Chapter 3.] of an agent where an *agent* is an entity that autonomously interacts within an environment to achieve a goal. We refer to these sources of behaviour as *teachers*. Unlike other agent-based learning methods, where an agent learns through interactions with the environment, imitation learning uses this teacher's behaviours to guide the learning process. Most commonly, imitation learning refers to the teacher as an 'expert', but we deviate from this nomenclature, which we further discuss the reasoning behind in Section 7. In this setting, a teacher can be a human or a computational agent. We generally define a teacher as an agent who provides information on how to act in an environment. This information might carry expert knowledge, but it is not a requirement.

More formally, imitation learning problems use *Markov Decision Processes* to model the environment. This formulation represents a state-action network, where the transition of states is mapped from its state-actions and, therefore, is suitable for imitation learning. A more formal definition of the Markov decision process is:

Definition 1. *Markov Decision Processes* are represented by a five-tuple $M = \langle S, A, T, r, \gamma \rangle$, in which: S is the state space, A is the action space, T is the transition dynamics $T : S \times A \rightarrow S$, r is the immediate reward function $r : S \times A \rightarrow \mathbb{R}$, and γ is the discount factor such that $\gamma \in [0, 1]$.

The state space S is represented by vectors that contain information regarding the environment's current state. For the action space A , we can also encounter two forms to represent them: vector-based and single values. Vector-based actions represent either a sequence of decisions that the agent should enact in order, such as a sequence of buttons to press, or a list of actions to act simultaneously, different joints from a robot, for example. The transition dynamics of a Markov Decision Process dictate how a transition in an environment will occur, while the discount factor relates to the reward regarding the time domain. A discount factor of 1 indicates that future rewards are considered equally important as immediate rewards, while a discount factor close to 0 implies that only immediate rewards are prioritised. Moreover, a critical property of these networks is that the transition dynamics usually only depend on the previous state and action, regardless of earlier states. Nevertheless, in the context of imitation learning, although the network might carry information regarding reward and discount factors, we consider that this information is inaccessible to the agent, and the learning process does not depend on it. Fig. 1 displays a common implementation of the Markovian problem where the environment provides a state s to the agent, which predicts the most likely action a . The environment applies the state transition function T on the state-action pair to generate the new state s' , and computes the immediate reward using the discount factor γ and immediate reward function r .

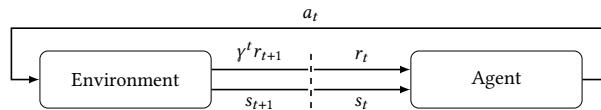


Fig. 1. The Markovian Decision Process.

Following a Markov Decision Process definition, we can generally state that the goal of imitation learning is for an agent to use S and A from a teacher to learn a mapping function called *policy*. A teacher also has a known or unknown policy that the agent is trying to approximate, and we define this function as:

Definition 2. A *policy* is a function π that maps states to actions $\pi := S \rightarrow A$. The policy has inner parameters θ , which represent the internal variables or weights within the policy that are adjusted during the learning process. The set of all possible internal variables is defined as $\Pi = \{\pi_1, \pi_2, \dots, \pi_z\}$.

Strictly defined, a policy maps states to probabilities of selecting each possible action $P(a | s)$ conditioned by the available teacher's data. Having the MDP definition allows us to define agents and teachers more formally.

Definition 3. An *agent* π_θ is one possible instantiation of Π . It selects an action $a \in A$ provided an environment state $s \in S$ according to its *fixed learned* parameters θ .

Definition 4. A *teacher* π_ψ is a special case of an agent, with respect to its parameters ψ , whose behaviour imitation learning wants to approximate.

Applying an action a to the environment results in a new state representation s' , which an agent can use to select a new action, forming a sequence of states and actions. All possible sequences from a given *teacher* refer to a smaller of all possible environment states S and actions A , which we denote as S_{π_ψ} and A_{π_ψ} , respectively. These states and actions are subsets because solutions from π_ψ may not require it to visit a specific state or predict an action. Thus, $S_{\pi_\psi} \subseteq S$ and $A_{\pi_\psi} \subseteq A$. For simplicity, in this work, we omit the policy information when referring to π_ψ , and explicitly inform the policy otherwise.

We still lack a definition for the sequential data collected from teachers to learn the policy function. These sequences of state and action interactions from the teacher in the environment are called *trajectories* and are an essential part of the imitation learning processes. The data within each trajectory is the only information an imitation learning agent can access when learning how to perform a task in an environment. Therefore, if too few samples are collected, the agent might encounter states with no prior information on how to act, or if the data is corrupted, it may act differently than its teacher. In this report, we formally define trajectories as:

Definition 5. A *trajectory* is an ordered list of state-action pairs $\tau = [(s_1, a_1), (s_2, a_2), \dots, (s_n, a_n)]$, where $n > 1$ and $s_{i+1} = T(s_i, a_i)$, for all $i > 1$.

A set of trajectories is composed of multiple trajectories, such that $\mathcal{T} = \langle \tau_1, \tau_2, \dots, \tau_k \rangle$, where $k \geq 1$. Given a set of trajectories \mathcal{T} , an agent π_θ and a teacher π_ψ , we can mathematically express the learning process for imitation learning as minimising the error for a given loss function ℓ between π_ψ and π_θ , as follows:

$$\arg \min_{\theta} \sum_{\tau \in \mathcal{T}} \sum_{s \in \tau} \ell(\pi_\psi(s), \pi_\theta(s)), \quad (1)$$

where s is a state sampled from the teacher's trajectory τ in the set of trajectories \mathcal{T} . It is important to note that early iterations of imitation learning used supervised learning losses to learn a policy. However, novel work [38, 51] have shown that imitation learning can be formulated as self-supervised and adversarial learning approaches (and further discussed in Section 4).

Equation 1 assumes we have access to π_ψ . However, it is sensible to assume that imitation learning could not depend on direct access to a teacher's policy. This level of access requires knowledge of the teacher's internal state, which cannot be done when the teacher is human. Therefore, imitation learning may use *demonstrations*, also known as *imitation learning from demonstrations*, to learn a policy, which we define below.

Definition 6. A *demonstration* is a state-action pair taken from a teacher's trajectory, such that $(s, a) \sim \tau \in \mathcal{T}$.

A list of all sampled demonstrations from one or more teachers is denoted as $D = [(s_1, a_1), \dots, (s_n, a_n)]$. We denote the subset of all states for a demonstration is $S = \{s \mid (s, a) \in D \text{ for some } a\}$, while the subset of all actions is $A = \{a \mid (s, a) \in D \text{ for some } s\}$. The probability of taking an action a given s in a list of demonstration D is:

$$P_D(s, a) = \begin{cases} P(s, a) & \text{if } (s, a) \in D \\ 0 & \text{otherwise} \end{cases}, \text{ where} \quad (2) \quad P(s, a) = \frac{|\{(s_i, a_i) \in D \mid s_i = s, a_i = a\}|}{|\{(s_i, a) \in D \mid s_i = s, \text{ for some } a\}|}. \quad (3)$$

Unlike trajectories, demonstration lists are not necessarily ordered lists; hence, s_{i+1} may not be the direct outcome from $T(s_i, a_i)$. As trajectories, demonstrations are vital for imitation learning and highly correlate with the agent's ability to generalise in unseen scenarios. For example, when solving different puzzles, the agent must adapt its strategy to solve one puzzle where some pieces do not share the same shapes from previous demonstrations. Consequently, when creating imitation learning from demonstrations, researchers must consider whether the data is representative enough of the environment, which we further discuss in Section 7.

Although imitation learning shies away from any direct signal from an environment by using demonstration, *experiences* such as an agent's trajectories can also be used. The difference between the two instances of data is that demonstrations provide the teacher's action to a given state, allowing a supervised learning approach, and experiences show the performed action, which may not be close to the source behaviour but also provides the reward of performing that action given the current state. However, using the reward function to optimise its policy further goes against the imitation learning paradigm. Hence, these work avoid using the returned reward from the environment and use experiences as state-action pairs. More formally, we define experiences as follows.

Definition 7. An *experience* is a special case of a demonstration, in the sense that its tuples are from the learned agent’s trajectories: $(s, a) \sim \tau \in \mathcal{T}_{\pi_\theta}$ instead of the teacher’s.

All definitions so far help us understand early approaches to imitation learning. Conversely, these definitions do not align with how humans acquire knowledge through observation. Thus, *imitation learning from observation* further restricts imitation learning by reducing demonstration information only to the teacher’s states. By learning without knowing which actions were performed, imitation learning from observation tries to employ the same learning approach humans do by figuring out the action performed from the observed changes between two successive states of the environment. In this context, imitation learning from observation represents its examples with less information.

Definition 8. An *observation* is a state pair (s_i, s_{i+1}) from a teacher.

If we have access to the teacher’s trajectory τ and the transition function T , we can retrieve any observation with $(s_i, T(s_i, a_i))$, such that $(s_i, a_i) \in \tau$ and $i \leq n$. On the other hand, if we lack these pieces of information, we can only retrieve observations (s_i, s_{i+1}) for all sequences of states where $i < n$. A list of observations is a not necessarily ordered list of all sampled observations from $\pi_\psi: O = [(s_1, s_2), \dots, (s_{n-1}, s_n)]$, where $n > 1$. Like demonstrations, the subset of states for observations is as follows: $\{s \mid (s_i, s_{i+1}) \in O\}$. The probability for observation pairs follows Equations 2 and 3, but uses the observation’s information rather than the demonstration’s. Assuming access only to the state information helps because datasets that explicitly give the actions performed between state changes (also known as *labelled datasets*) are uncommon in the real-world¹ and are costly to create. By lifting this restriction, imitation learning agents do not need to create a dataset that involves recording humans playing or training another agent to act as a teacher. However, by removing the action information, two problems may arise when dealing with more complex tasks: (i) Markovian problems often are non-injective, which means that the transition function T may map two different state-action pairs to the same state, and smaller datasets will not cover the range of transitions necessary for the policy to learn properly; or (ii) computing whether the collected data covers a significant part of the Markovian network might be impossible. Therefore, focusing on imitation learning from observation is vital to create more efficient imitation learning agents and removing the requirement for vast datasets. Fig. 2 shows a general approach to imitation learning, where the agent learns from demonstrations or observations provided by a teacher and uses the environment to collect new experiences.

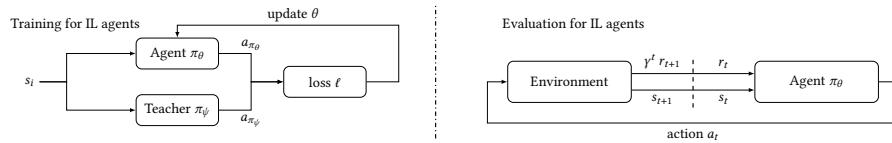


Fig. 2. A general approach to imitation learning.

So far, we have described all possible inputs for imitation learning agents as time-sensitive, but our agent formulation for now remains time-independent. Thus, we have to consider time as a possible input for all types of data, despite its ability to specify an instance of input and output. A policy that maps a sequence of states to actions, such as $P(A \mid s_0, s_1, \dots, s_t)$, is called *non-stationary*. These policies help map lasting consequences between states and sequential actions, such as the agent closing a path unintentionally and the different movements required to hit a ball with a racquet, respectively. Therefore, non-stationary policies are more naturally suited to learning motor trajectories [87]. Conversely, *stationary* policies ignore time and predict actions solely on the present information $P(A \mid s)$. The advantage of stationary policies is the ability to learn tasks when their length might

¹This is commonly referred to as *in the wild*

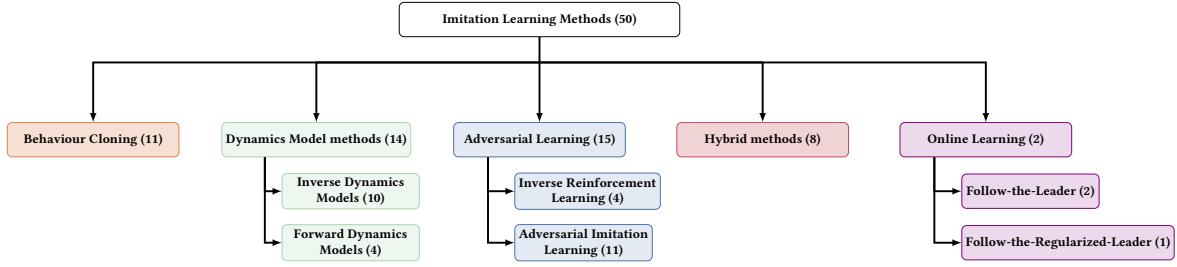


Fig. 3. A new taxonomy for imitation learning methods.

be too long or unknown [87], such as autonomous driving, where the agent can drive for an undetermined time. Moreover, non-stationary policies are difficult to adapt to unseen scenarios and changes in the parameters of a task [91]. Given the temporal characteristics of these policies, at one point, the trajectory can result in compounded errors as the agent continues to perform the remainder of the actions, such as an agent moving in a loop through a maze. The imitation learning literature focuses heavily on stationary agents.

Lastly, Zheng et al. [121] propose to further classify actions into two groups: (i) *low-level* actions, which refer to atomic decisions that an agent can perform in a given domain, typically involving simple commands such as movement or interaction; or (ii) *high-level* actions that are decisions made by an agent that determines the overall plan or strategy to be executed in a given task or domain, which are often complex and involve multiple lower-level actions or sequences of decisions. Most imitation learning work focuses on low-level actions of single-value outputs, given the stationary nature of most agents.

4 Imitation Learning Methods

Common forms of classification for imitation learning partition the methods into model-based and model-free methods. Model-based methods rely on learning to model the environment, such as predicting the consequences of

their actions and, afterwards, using this model in the learning process of the policy. Conversely, model-free methods do not use a model of the environment. Instead, they learn by trial and error, similar to reinforcement learning approaches. Model-based methods offer a trade-off in time and sample efficiency by leveraging the number of samples a teacher provides, the less time it takes to train a policy. However, model-free methods are usually more robust to unseen states since they learn by trial and error, which gives them more access to diverse states (outside the teacher's dataset), resulting in more generalisation from their policies. Lastly, imitation learning methods can also be classified as online. Online imitation learning methods assume that we have access to the teacher's policy. Since teacher policies are seldom available, these learning methods are less common.

Nevertheless, this classification can be quite coarse and does not truly reflect the nature of some methods. Instead, we propose the taxonomy with the main five categories given in Fig. 3, where the numbers in parentheses indicate the total of papers surveyed in each category. The key difference is that our classification distinguishes model-free methods that are based on behavioural cloning (Section 4.1) — the

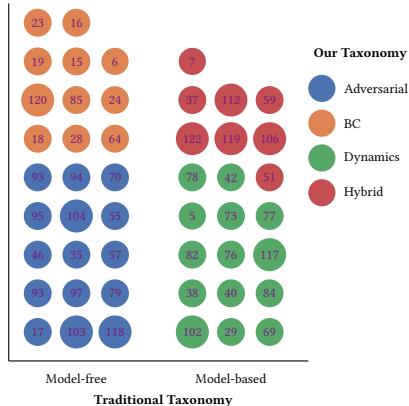


Fig. 4. Comparison between taxonomies. We remove online methods since both taxonomies are equal in that matter.

simplest form of imitation learning — from adversarial methods (Section 4.3); and partitions model-based methods into dynamics methods (Section 4.2) and hybrid methods (Section 4.4). For an illustration of how the IL papers discussed fit within our proposed taxonomy and within the traditional classification see Fig. 4.

In our review, we found 50 publications proposing new methods, which accounts for a big part of the reviewed work. We expected a high percentage of work to be methodological papers since imitation learning borrows environments and metrics from the reinforcement learning literature. On the one hand, this trend is sensible since there were fewer imitation learning methods in the past. On the other hand, this focus on imitation learning methodology shows a lack of research on other subjects pertinent to imitation learning, such as resiliency [39] and evaluation processes [67, 110]. The most common approach was adversarial learning (following the taxonomy illustrated in Fig. 3) with 15 published work, followed by 14 dynamic, 11 behavioural cloning, 8 hybrid, and 2 online methods. Although we expected to find many methods that use dynamic models, an interesting trend is the use of hybrid approaches to create imitation learning agents. We hypothesise that hybrid approaches and dynamic models are also becoming more popular with more researchers from backgrounds outside the reinforcement learning field. Indeed, hybrid methods offer a better trade-off between efficiency and effectiveness, which is a reason for their increase in popularity in recent years. Fig. 4 shows the difference between our taxonomy (colours) and other surveys taxonomies (location) for each of the papers present in this work.

4.1 Behavioural Cloning

The earliest approach to imitation learning was Behavioural Cloning [6, 80], which reduces the problem of learning to imitate a teacher into a supervised problem. In it, the agent learns to predict the most likely action given a state $\arg \max P(a | s_t)$ based on a teacher's dataset. To be precise, the agent uses a state-action pair of demonstrations (s_t, a_t) at time t from a proficient source to learn to act as the source based on previously seen data. Behavioural cloning tries to approximate the agent's *trajectory* to that of its teacher. A trajectory is a coherent sequence of demonstrations or experiences from one cycle of the teacher or agent's interaction with the environment. Such an approach becomes costly for more complex scenarios, requiring a large number of samples and information about the action's effects on the environment. For example, the number of samples used to solve tasks involving a higher number of possible actions (e.g., continuous actions) or intricate dynamics is approximately a hundred times higher than for classic control tasks [102]. This problem occurs for behavioural cloning policies since they fall short when approximating unseen states to known trajectories (generalisation), a common problem from supervised learning approaches. Additionally, requiring demonstrations becomes costly due to the need for labelled pairs. Usually, recording these trajectories involves training an agent via reinforcement learning, but some researchers record themselves playing, and as a result, they may not be demonstrating an optimal solution to the problem being solved. This scenario is not problematic as long as the measure of success depends on how good the agent is at imitating the teacher, rather than how good it is at actually solving the problem. Section 6 details this point further.

Newer approaches usually employ behavioural cloning as a *bootstrapping* mechanism. Bootstrapping a policy means using a machine learning approach, such as supervised learning, to acquire knowledge from the environment or desired behaviour before applying another learning technique to fine-tune the agent's performance. An example is the work of Lynch et al. [64], where the authors use behavioural cloning to condition a policy to a set of goals and, afterwards, use planning to create an optimal trajectory. A second possible bootstrapping application is the work of Daftary et al. [23], in which the researchers record themselves flying a drone, apply behavioural cloning to learn their flying behaviour, and, as the final step, use reinforcement learning so the agent learns to adapt to different seasons (when the images look different). These approaches display some of the benefits of using behavioural cloning. The agent learns more efficiently *offline* (without requiring direct environment access) by applying this technique and, afterwards, using the acquired knowledge to reduce the number of steps required

for less efficient learning approaches, such as reinforcement learning. Additionally, behavioural cloning coupled with bootstrapping can condition the policy to a more human-like behaviour (when humans provide the data).

Conversely, behaviour cloning remains the sole learning approach when the work mainly focuses on aspects outside the learning process. One common approach among work such as [16, 18, 28, 120] is coupling behavioural cloning with other learning mechanisms, such as attention [28] or domain-shift [18, 120]. We use these as mere examples of mechanisms applied to imitation learning. Moreover, in these cases, the behavioural cloning approach could be swapped for another imitation learning paradigm with correct adaptations since the cost of acquiring labelled demonstrations remains significant.

4.2 Dynamics Model methods

A possible approach for solving the task of mimicking a teacher without any direct action information is by employing the use of dynamics models. These approaches are classified as model-based imitation learning methods since they learn a model of the environmental dynamics. Generating dynamics models involves some form of *online play*, which is when the agent interacts with the environment and focuses on a specific task. Dynamic models can appear in two forms: *inverse* and *forward*.

4.2.1 Inverse Dynamics Models. Inverse Dynamics Models avoid the need for labelled pairs in behavioural cloning by encoding environmental physics and retrieving the likelihood of each action given a state transition $P(a | s_t, s_{t+1})$. Nair et al. [73] use this form of learning to teach a robot to manipulate a rope in different ways. In their work, the dynamics model receives the current state of the environment and a state from a sequence of human demonstrators as the goal state from that sequence. With both states, the model predicts the action responsible for the desired transition conditioned to a desired goal. This process is applied sequentially, and the authors perform all experiments using only matrices (image) states.

Torabi et al. [102] later implement an inverse dynamics model, which they use in vector states for control and robotic tasks and coined as *Behavioural Cloning from Observation*. It uses its randomly initialised policy to learn a mapping function without access to the teacher’s action by creating a dataset containing labelled actions. Hence, Torabi et al.’s method does not rely on a desired state (or goal) from its teacher. Their method uses demonstrations (s_t, a_t, s_{t+1}) from π_θ to learn the dynamics model. This dynamics model learns a uniform transition function (given the uniformly distributed dataset due to the random initialisation from policy π with weights θ) and creates self-supervised labels for the teacher’s observations. Provided with these labelled samples, Torabi et al.’s work trains its policy in a supervised manner using behavioural cloning. Following Torabi et al.’s work, other methods augmented their approach to improve its performance [70], stability [38], sample efficiency [40, 78], and to other domains [84, 117].

Most notably, our work [38] improves Torabi et al.’s work by applying an exploration mechanism and fine-tuning its sampling mechanism, the first imitation learning method to use an exploration mechanism. The method assumes that the dynamics and policy models are not always sure about the correct action and samples from each model’s output using a softmax distribution. Therefore, if a model has an equal distribution between two actions, it will select each action 50% of the time. Conversely, if a model’s output mainly favours an action, such as [2.5, 0.1], the model will choose the first action approximately 90% of the time.

As behavioural cloning methods, inverse dynamics methods also combine their approach with reinforced learning methods [40, 78]. Pavse et al.’s work involves two phases. In the first phase, an inverse dynamics model is randomly initialised and learns state transitions from a random policy, just as Torabi et al.’s work would. In its second phase, the method alternates between generating agent experiences (with environment rewards) and optimising the policy with teacher demonstrations. This two-phase procedure aims to find the optimal policy in terms of total task reward (which may outperform the teacher) by using the expert demonstration as a guide. Our work [40] similarly alternates reinforcement learning with imitation learning. The difference between these

approaches is that instead of using all experiences, the second applies the goal-aware sampling mechanism from Gavenski et al.'s work and limits the size of the replay buffer from its reinforced learning counterpart to have less drastic updates when learning with experiences.

4.2.2 Forward Dynamics Models. Like inverse dynamics models, forward dynamics models model the environment's dynamics using state transitions. These models predict the next state, given some conditioning. Most researchers condition these models' predictions with the environment's actions to generate the next state [29, 77]. However, others rely on temporal information, such as using all states until the current moment [76, 82].

Most notably, Pathak et al. use the same idea from Nair et al., where it conditions its policy with a goal. However, instead of using current and goal states to predict the actions, the authors use current and goal state features coupled with the last action to predict the most likely next action and use the current and the predicted action to generate the next state. Pathak et al.'s policy uses a common technique to predict its information: using all previous states until timestep t , which helps to create a consistent action considering all previous states.

Nevertheless, the action information might be unavailable or can be insufficient. For example, suppose an agent walks towards an impassable wall and hits it. Even though the agents performed an action, such as 'move forward', the final transitions would show no movement. Thus, the agent should consider that the action responsible for such a transition is 'no action', which is sometimes absent in the environment dynamics. Edwards et al.'s work applies this premise to forward dynamics models by using these *latent actions* to condition its next state. By doing so, Edwards et al.'s method creates more faithful transitions according to the environment dynamics. As a final step, the method requires remapping the learned actions into environmental actions.

Forward dynamics models are less popular than inverse dynamics models, given their nature of predicting the next state from the Markov Decision Process, which is more challenging than predicting the action responsible for a state transition [34]. The agent can lower its generalisation capabilities by adding this sequential information into a behavioural cloning technique when the trajectory deviates from its teacher's dataset.

4.3 Adversarial learning methods

Adversarial learning and inverse reinforcement learning methods share the same task in imitation learning. Instead of trying to reproduce the teacher's behaviour by applying some form of supervised learning, these methods create an artificial reward function, which conditions the agent by rewarding similar behaviours. By creating this artificial reward function, imitation learning approaches can use reinforcement learning optimisation techniques to learn policies with similar behaviours from demonstrations. Most work use adversarial learning methods to create these reward functions [17, 46, 57, 79, 95–97, 103, 104, 118]; however, some earlier work rely on other inverse reinforcement learning techniques [35, 55, 94]. This method is model-free since the policy freely acts in the environment and learns by trial and error without modelling the environment dynamics. Nevertheless, not all inverse reinforcement learning methods use online play to learn its policy.

Before applying generative models, inverse reinforcement learning in an imitation learning setting used demonstrations to guide the agent's learning by applying a distance-based optimisation function to the teacher's and agent's trajectories. Finn et al. [35]'s work uses inverse entropy to retrieve an artificial reward function based on a random controller and the teacher's demonstrations. With the artificial reward, their method optimises a policy function, creating new demonstrations and allowing for further refinement of the retrieved reward function.

Ho and Ermon uses *adversarial learning* [41] and maximum entropy inverse reinforcement learning [124]. Adversarial learning is a type of training where a model trains to generate data according to a dataset, and a discriminator model has to learn how to discriminate samples from the dataset and those generated by the first model. The method uses a generative model, the same one that forward dynamics models use, to generate trajectories and a discriminator model to discriminate between teachers' and students' trajectories. Therefore, the policy role in this setting is to 'fool' the discriminator model by acting accordingly to the teacher demonstrations.

However, the method presents two problems: (i) it assumes that the teacher's actions are available during training, which inherits the cost of behavioural cloning approaches; and (ii) it is susceptible to local minima during its optimisation process, which requires prolonged environmental interactions. Nevertheless, Ho and Ermon is more sample-efficient than behavioural cloning, which decreases the severity of the first point. Ho and Ermon's work [44] is the most known work and perhaps one of the most influential, with several methods following the same setting [57, 70, 79, 103]. Most notably, Li et al.'s work [57] assumes that a dataset consists of various behaviours from one or more teachers. So, instead of creating a policy that tries to predict the most likely action solely on the state alone, it also conditions the agent with a teacher's parameter. Additionally, Peng et al.'s work changes how the discriminator discriminates between teacher and student. In it, the authors constrain the amount of information given to the discriminator by applying the output of the generative model into an encoder. Therefore, the discriminator uses an encoded value instead of the state to discriminate. Peng et al. show that applying this bottleneck helps the policy to generalise better and achieve better results than other adversarial methods that rely on state information.

Torabi et al.'s work [103] applies a similar strategy to Ho and Ermon's work to apply imitation learning from observation. It assumes that adversarial methods follow a convex conjugate optimisation problem, where the generative model and policy function learn to approximate state transitions alone. To approximate teacher and student, a discriminator has to discriminate over the agent and teacher trajectories observations. By learning to differentiate between origins, the discriminator retrieves an artificial reward function, which the method uses to optimise the policy. By applying these changes, Torabi et al.'s work improves adversarial methods performance in more dynamic environments and reduces the cost of learning new policies. Torabi et al.'s work has also seen further improvement by Sun et al.'s work [97]. Sun et al. use an adversarial learning approach to create a policy that minimises the policy with a no-regret function bounded to one. Therefore, if the optimisation function is zero, the policy acts as the teacher; however, if the artificial reward function returns one, the policy is as far as possible from its teachers. As with other work, acquiring new demonstrations to further policy improvement is done via online play, which becomes costly if many interactions are needed. Additionally, Torabi et al. further explore his methods capabilities by applying it to vision tasks [104] by using convolutional neural networks as the discriminator model. Since vectorised information is not typical for applications, optimising a policy using only visual information from a convolutional neural network while holding the vectorised representation as the input for its policy allows for more optimal solutions.

Although adversarial learning methods became the standard for model-free approaches, inverse reinforcement learning techniques have also demonstrated remarkable advancements recently. Chang et al.'s work [17] discards the usage of discriminator models to match the policy's trajectories to the teacher's using a distance metric. However, using this approach, the method can enter a seeking behaviour state, producing low probabilities for observations from the teacher data. Therefore, Chang et al. implement a noise-conditioned normalising flow, which adds noise to observations conditioned to the state during training and nothing during evaluations. Doing so reduces the chance of low probabilities and biased predictions from the policy. In multi-agent systems, inverse reinforcement learning methods have also achieved some success. In Song et al.'s work [95], the method is divided into centralised (or cooperative), decentralised, and zero-sum. In centralised and decentralised approaches, the policy operates with the environment to match the teacher reward by using adversarial techniques. On the other hand, in the last approach, the policy interacts directly with the discriminator to match the reward from the teacher. Conditioning also appears in multi-agent settings. For example, Shih et al. proposes to create a policy that adapts its strategy depending on other agents [94]. Therefore, the method has to condition itself by classifying which strategy from the teachers' dataset is being used.

Finally, these approaches also appear in different settings and tasks. For example, Shang and Ryoo assumes that the agent only has access to third-person views from the teacher (a more human-like experience of observing others) [93]. Thus, outside of learning the teacher's behaviour, the agent must also learn to adapt the data from

its teacher to suit its first-person view when enacting the same task. In [Yin et al.](#)'s work [118], the method has to predict a sequence of actions from a behavioural cloning and adversarial agent. After both policies predict all actions, a planning model forms a plan of which actions should be used throughout the trajectory. Finally, in the domain of self-driving vehicles, [Hu et al.](#)'s work [46] experiments with sharing weights with the policy and generative model, which the method uses to optimise the policy using a reinforcement learning technique.

4.4 Hybrid approaches

Hybrid methods leverage dynamics models with adversarial techniques to learn their policies. Recent work in imitation learning has focused on taking inspiration from adversarial and dynamics methods [7, 37, 51, 59, 70, 106, 112, 119, 122]. By combining dynamics models and adversarial learning, these methods create more robust policies since they have a temporal signal that helps minimise teacher and student differences and some intrinsic knowledge about the environment dynamics. Although they use discriminators to achieve their task, these methods have significantly improved imitation learning efficiency, with some requiring a single trajectory [122].

Towards sample efficiency and the issue of saddle points, [Wang et al.](#) [112] use a variational autoencoder to increase diversity with relatively fewer demonstrations and achieve one-shot learning to new trajectories. They apply a temporal neural network to create a non-stationary policy. It is worth noting that even though the generative model relies on a forward dynamics model, the agent's optimisation does not rely on it, increasing efficiency during optimisation.

In learning from demonstration hybrid methods, [Baram et al.](#) develop a method that optimises the forward dynamics model [7]. Instead of concatenating state and action into a neural network to predict the next state, the method uses a recurrent neural network (temporal) to encode the state and another non-temporal encoder and perform an element-wise multiplication to feed as the input of a third neural network. By doing so, the authors achieve teacher behaviours earlier during training and reduce the number of teacher samples. In the same context, hybrid methods have also seen use in partial demonstrations from teachers [119]. In their work, the demonstrator is a single teacher's life in Atari environments (less than a whole episode – three lives), and [Yu et al.](#) use a variational autoencoder to encode a forward dynamics model and a discriminator to learn an artificial reward function. Although learning from demonstration work is not as common as its observation counterpart, these methods tackle problems significant for imitation learning agents. Removing the complexity of learning policies without action information allows researchers to tackle other, more complex problems.

Hybrid methods for learning from observation are more prevalent in recent research. These methods focus on efficiency [51, 70, 106, 122] and eliminating the seeking behaviour [59, 122]. [Zhu et al.](#)'s work [122] solves the issue of imitation learning agents adopting a seeking mode by using a discriminator to swap between a covering and seeking mode when the agent heavily diverges from the teacher. It does so by using ten teacher's trajectories for each environment, but [Zhu et al.](#) assume that no more than a single state-action pair can result in the same next state, which is not valid in most environments where there is momentum for example. Most notably, [Kidambi et al.](#)'s work uses a forward dynamics model with adversarial learning and couples it with an exploration mechanism [51] with ten teacher trajectories. However, their method assumes that all episodes initiate in the same state, which causes it to fail to generalise as state sizes grow. Moreover, [Kidambi et al.](#)'s method exploration mechanism comes with the cost of using an ensemble method (more than one model during prediction), which later work by [Monteiro et al.](#)'s solves. In addition to solving efficiency problems, [Liu et al.](#)'s work [59], the authors experiment with encoders across different contexts, creating dynamics models to translate between teacher and student states and apply a discriminator model to approximate both behaviours. By doing so, their work eliminates the need to fine-tune a model every time an agent's context changes and only requires training a new encoder, allowing for translating teacher behaviours into new contexts.

More recently, [Gangwani et al.](#)'s work assumes that a policy has to learn from a teacher's trajectory that diverges from the agent. Different from [Liu et al.](#)'s work, teachers and agents are in the same context, but the transition function for both is different. In other words, the teacher's observations should guide the policy towards a goal, similar to other methods that condition their policies as a goal state. In their work, the policy learns a forward dynamics model to output the next state from the agent's transition function perspective that aligns with the teacher's trajectories. [Gangwani et al.](#)'s work [37] shows that it performs better in these settings than other methods. However, the results are far from the teacher, which shows that such a setting needs additional research.

We note that hybrid methods have become more popular. Although adversarial learning methods are less sample efficient, they help model-based methods generalise and learn more human-like behaviour. Methods such as [Kidambi et al.](#)'s and [Zhu et al.](#)'s work achieve teacher results with ten or fewer trajectories, but some recent studies show that there are still other mechanisms to research. For example, exploration mechanisms and cross-domain contexts have been uncommon in imitation learning approaches. By exploring, agents can acquire knowledge unattainable before due to the need for more variety in teacher samples. When applying cross-domain contexts, imitation learning agents can learn from a third-person view or different settings for the agent, such as fewer joints in a robot.

4.5 Online Imitation Learning

A different approach to learning from a teacher is online imitation learning. In this approach, the agent must also learn the teacher's behaviours. However, instead of using demonstrations from a pre-recorded dataset, online imitation learning involves access to the teacher's information in real-time. Online learning can be detrimental when designing a reward function that is challenging to define or a policy function that is difficult to model. Given the requirement of direct access to a teacher, such as another agent, this approach is less commonly researched than the other areas. Nevertheless, online imitation learning is more suitable when real-time interaction with a teacher is possible or when the agent needs to adapt to changing conditions.

[Ross et al.](#) implement an iterative online imitation learning algorithm that uses demonstrations from online play and teachers to learn a behaviour [87]. It uses a *Follow-The-Leader* technique to optimise the policy behaviour based on the demonstrations it retrieves during the first iterations and from the teacher. *Follow-The-Leader* tries to minimise a cumulative loss function or maximise the agent's reward sequentially. Since reward values are not available to the agent in imitation learning approaches, *Follow-The-Leader* tries to minimise the difference between teacher and student behaviour. [Lavington et al.](#) further improved the *Follow-The-Leader* strategy by applying a regularisation term, which they coined as *Follow-The-Regularised-Leader* [54]. In it, [Lavington et al.](#) consider that the teacher's demonstrations might have noise, and the *Follow-The-Leader* approach can lead to oscillations resulting in non-optimal behaviour. The researchers show that a policy achieves better results after applying regularisation to the optimisation function.

4.6 Learning Methods Discussion

Upon analysing the literature from this section, we notice a change in the development of new learning methods, which use hybrid mechanisms. Instead of relying solely on dynamics models or adversarial approaches to learn the teacher's behaviour, these methods mix different aspects from both sides to achieve better results. Furthermore, recent approaches focus more on imitation learning from observation and efficiency problems. In Section 7, we further expand on challenges researchers may focus on rather than efficiency and also reflect on different aspects learning methods may not account for.

This literature review shows a lack of imitation learning methods applied to online learning and multi-agent systems. We believe this dearth of work for online learning comes from the premise of having access to the teacher, which limits the number of applications and the complexity of learning from a policy in an environment where

more than one agent impacts the transition function. Like online learning, multi-agent systems use highly dynamic environments to learn their policies. In these environments, the overall combination of all possible states can be virtually infinite, which leads to imitation learning agents requiring a lot of data or drastic assumptions on how to model the environment (that might not hold true in diverse cases). Therefore, we argue that imitation learning methods tailored specifically to the challenges posed by these applications need to be researched and developed.

5 Imitation Learning Environments

Environments are a vital part of the imitation learning process. They simulate various tasks and domains and allow the agent to interact with the simulation as they would in a real-world application. Creating an environment most often involves creating the environment's rules, such as physics for each object, and implementing rules to guide the agent, for example, how agents should interact. Nevertheless, they allow different work to evaluate their results using the same dynamics, creating a more fair comparison since all methods receive the same demonstrations and solve the same problem. In the literature review, we notice a lack of consensus on which environments imitation learning literature should use.

In the literature on agents, there are different descriptions of what environments consist of. We follow [Russell et al.](#)'s definition, which describes environments in four different aspects [90, Chapter 2]. The first aspect entails the *accessibility* of the states from the environment. Most commonly, environments use fully observable states, meaning the agent can obtain complete, accurate, up-to-date information. However, most real-world environments are inaccessible in this sense. The second aspect is *certainty*. *Deterministic* environments are those in which any action has a single guaranteed effect. In these environments, there is no uncertainty about the state resulting from an action. We call environments that are not deterministic *non-deterministic*. The third aspect describes the environment dynamics. *Static* environments are those that are safe to assume will remain unchanged except by the performance of an action by an agent. In contrast, a *dynamic* environment has other processes operating, changing in ways beyond the agent's control, such as the physical world, which is a highly dynamic environment. Lastly, the fourth aspect entails how things are represented. *Discrete* environments are those with fixed, natural numbers and a finite set of actions, while *continuous* are those represented by rational numbers, hence an infinite set of actions with upper and lower limits.

Environments can also be domain-specific, such as autonomous driving, or more general, such as robotic simulations, which use 2D and 3D simulations to evaluate agents in various similar tasks. Most work use the application to differentiate each environment, such as assistive and humanoid robotics [47]. However, such a classification is shallow since it only focuses on a specific set of problems per environment, not what the environment tests the agent on. Therefore, we propose a threefold classification for these environments: (i) validation, which helps researchers to validate ideas without added complexity; (ii) precision, which helps to test agents that require less temporal modeling, but require precision; and (iii) sequential, which tests an agent's actions and their temporal consequences. Fig. 5 presents the proposed taxonomy for the environments.

It is important to note that other forms of classifying environments exist, such as how the states are represented (images or vectors), actions (discrete or continuous), and whether a task is a maintenance task (a set of states are unreachable to the agent) or an achievement task (the agent should reach a set of states). However, our classification focuses more on the agent's intent, such as 'to be as precise as possible', than on the task or domain itself. Additionally, this taxonomy also contains domain-specific and task-specific classifications after the broader term. For each environment, we present whether the environment presents domain-specific or task-specific problems, which should help researchers decide if an environment is desirable for their experiments.

5.1 Validation environments

Validation environments are more straightforward simulations in general. These environments help to validate whether an agent can learn to perform a task from a teacher with no extra complexity involved. Usually, the agent is only required to learn the task, and the environment does not diverge much when initialising the agent. These environments are more commonly split into discrete and continuous action environments.

The most common discrete environment in the imitation learning literature is CartPole [9] (Fig. 6a). This environment is a perfect example of a validation environment. It consists of a pole attached by an unactuated joint to a cart, which moves along a frictionless track. The pole is placed upright on the cart, and the goal is to balance the pole by applying forces to the left and right of the cart. Thus, CartPole is a maintenance task-specific environment. Each episode starts with uniformly random values between -0.05 and 0.05 for the cart position, cart velocity, pole angle and pole angular velocity. Therefore, each episode has a small variety, and since there is no differentiation between training and validation, such as different mechanics, the agent is not required to generalise. Achieving an optimal result within the CartPole environment does not demonstrate that the agent can learn and perform complex tasks. However, it is helpful as an initial validation. Alongside the CartPole, the most common environments for discrete actions are the MountainCar [71] environment and Atari games [12]. In the MountainCar environment, a car is randomly placed at the bottom of a valley shaped like a sine wave. The only available actions are accelerating the car in either direction or not accelerating. The MountainCar problem deals with momentum, making it harder for simple or random agents to solve since it carries some sequential impact. If an agent builds momentum and selects to use force opposite to its direction, the agent will lower its speed and fail to reach the goal. On the other hand, Atari games are environments that consist of visual state representations. They are commonly used in imitation learning tasks because they allow for visual qualitative assessment of the agent's behaviour. Moreover, Atari games were a long benchmark for comparing how agents perform in different tasks and allow for a direct comparison with humans using the video game score [65].

The most common continuous environments are simulations from Multi-Joint dynamics with Contact (MuJoCo) [101] (Fig. 6b). In these environments, the agent is a robotic simulation of different joints and body parts configurations, such as a cheetah or a human. The agent aims to move right by applying force to these joints and body parts. Similar to the CartPole environment, these environments are simplistic in the sense of not posing new challenges to the agent. The agent has no barrier to overcome or other dynamics it has to adapt to. However, differently from the CartPole environment, these environments still pose a challenge in the sense that the agent has to learn to control its body joints together to achieve the goal.

Most work use a combination of validation and another environment type to experiment with the limitations of imitation learning agents. However, some work [69, 97, 102, 103, 106, 118, 119] use only validation environments, which can be troublesome. By only using these environments, the agent can learn to overimitate its teacher without comprehending the underlying task. Additionally, considering the initialisation is not diverse, the agent can force a teacher's trajectory into the environment, causing the same problem as reward-based metrics. In other

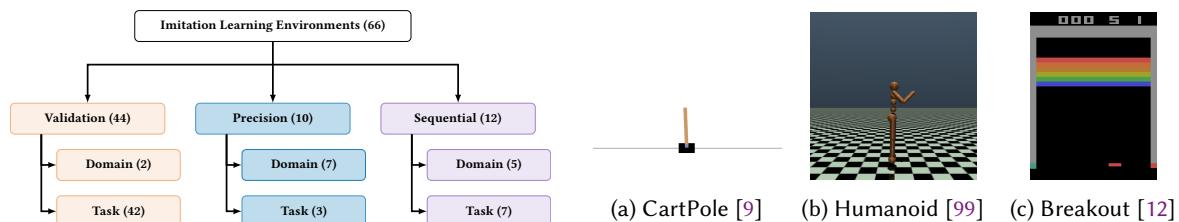


Fig. 5. Imitation learning environments taxonomy.

Fig. 6. Examples of validation environments.

words, during an agent’s first interactions with an environment, the agent can predict actions that will take it to the same states present in the teacher’s dataset, and afterwards, the agent will only have to perform state matching to achieve optimal results. Other frameworks for simulation, such as Google’s suite [108], promise a diverse set of initialisations, but further testing shows that these simulations are vulnerable to the same seeking behaviour. Table 2 shows all validation environments with Russell and Norvig’s environment attributes information.

5.2 Precision environments

Precision environments are generally independent of long action consequences and require precision, as the name suggests, from the agent. Russell et al. classify this autonomy from time as episodic environments [90, Chapter. 2]. It is important to note that an episode is a sequential set of states from the agent in imitation learning literature, while Russell et al. classify an episode as a single step from the agent (an experience). In episodic environments, the task divides the agent’s experience into atomic experiences. The agent receives a state in each experience and then performs a single action. Additionally, the following experience is independent of the actions taken in the previous one. However, this characteristic depends on how the environment computes these experiences. For example, an experience usually comprises four frames in Atari games, while it is a single frame in Mario games. Therefore, being episodic has some space for interpretation.

Precision environments usually involve robotics tasks where agents require precision, and time for each decision is irrelevant. The most common precision environment type is robotic arms. This simulation works for various tasks, such as opening doors [106], pushing pins [15], and arranging blocks [28]. The most common setting for these tasks is a robotic arm with an unmovable base and some range of movement on other joints (Fig. 7c). In these tasks, researchers aim for agents to overimitate a teacher in the same setting and adapt slightly in newer settings, such as posing blocks in different positions [28] or moving with different body configurations [79].

Robot walking and navigation is also common tasks for precision environments. For this task, the environment simulates a robot’s body, such as MuJoCo, but to a higher degree of detail. The idea is to test whether the agent can navigate different obstacles and spaces. Furthermore, the agent can have different body configurations, from vacuums to bipedal designs. For example, micro air vehicles, or MAVs, is a common domain for these precision environments (Fig. 7b), where the agent has to fly through obstacles without getting hit. Some flying environments consider different seasons to understand how agents adapt to variations in images [23].

Usually, work that use precision environments experiment with physical robots as the final evaluation step [23, 35, 59, 76, 77, 82, 117, 120]. However, this approach is expensive, given the requirement of a robotic arm or different robot settings. Therefore, some authors [78] rely on simulations to test different settings, such as blocks world [28], where the robotic arm has to arrange blocks in order, and the authors split some initial settings and goals only for evaluation, making sure the agent is not only overimitating and has learned the underlying task correctly. Table 3 shows all precision environments with some information regarding the environments’ attributes.

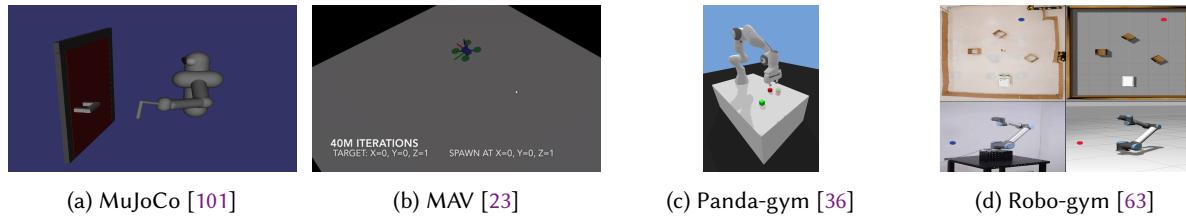


Fig. 7. Examples of precision environments.

5.3 Sequential environments

Sequential environments could have long-lasting consequences, given the agent's prior actions. They are the opposite of precision environments. However, it is essential to observe that not all actions must impact future experiences. These environments are harder to solve for imitation learning agents because they require the agent to plan for future actions, which might be learning an abstraction from the task, and usually rely on non-stationary policies. For example, in Lunar Lander, an environment where a rocket has to use its engines to land between two flags, the agent have to learn how to map a trajectory depending on where the agent spawns. If we suppose the agent overimitates teachers' trajectories instead of adapting actions to consider the difference between previous demonstrations and the current setting, the agent will not learn how to land the rocket where it is suppose to, but instead, it will learn how to land the rocket where the teacher landed it.

Sequential environments are most commonly used for autonomous driving domains, where the agent must learn how to drive and navigate different scenarios. In these environments, different aspects can be experimented with. For example, while CARLA [25] has a more realistic city traffic simulation, TORCS [116] proposes a scenario for open racing car simulation, where the agent competes against other cars (Fig. 8a and 8b, respectively). Simulation of Urban Mobility, or SUMO [60], allows for recreating various urban scenarios, but it does not have CARLA's realistic LIDAR and RADAR sensors. In contrast, SuperTuxKart offers a driving simulation with a more cartoonish style where the agent has different powers to use against other computer-controlled agents (Fig. 8c). Therefore, each environment can have drastic differences in information and settings, and researchers must ponder which scenario is more relevant to their research and correctly evaluates their agents.

On the other hand, if the domain of autonomous driving is not required, the usual sequential environment is video games. Video games, such as Mario [49] and Overcooked [16], allow for easy abstraction of tasks, such as navigation, or just for modelling temporal consequences. In the latter, these environments allow for the manipulation of maps to test specific behaviours from the agent. For example, VizDoom [50] (Fig. 8d) allows developing agents to play DOOM, a 1993 video game where you play a marine fighting through hordes of monsters in first person, using visual information. The environment has visual editors that allow for custom scenarios and to control variables for each level, allowing for a more consistent benchmark between researchers. Another advantage of these environments is that they provide diverse contexts for both learning and assessment, such as changing the number of enemies, weapons and level design between training and evaluation. CoinRun [21] (Fig. 8e), where the agent learns how to navigate through a level from a game where the goal is to pick one or more coins, poses a similar setting. During training, the level has no hindrances to the agent, but during the evaluation, the level will have monsters and lava, which can kill the agent. Consequently, sequential environments are great for evaluating not only long-lasting consequences but also the agent's ability to generalise to different settings.

Nevertheless, this diversity of settings for sequential environments can cause problems when comparing different work. For example, in Duan et al.'s work use a maze to evaluate whether their agent learned the underlying task [28]. However, in Gavenski et al.'s the authors use a different maze to evaluate their agent [38]. In the first, the authors use a more simplistic structure (following a 'C' and 'S' shape). The latter, uses a more



Fig. 8. Examples of sequential environments.

complex structure, with more turns and dead ends with different sizes (3×3 , 5×5 , and 10×10). Thus, making the comparison between these two work harder.

Combining sequential and validation environments is ideal when creating imitation learning agents. These long-lasting consequences usually are the downfall of these agents. More so, these environments allow for diverse customisation of different levels of requirements. Table 4 shows all sequential environments with some information regarding the environments' attributes.

5.4 Environments Discussion

In this section we analysed all environments used in the literature and classified them in three different categories: validation, precision, and sequential. In their usage, we observe a behaviour similar to Zipf's law, where from the 66 environments used to evaluate the various imitation learning work (presented in Tables 2, 3, and 4), six environments account for more than 80% of the total number of times these environments were used. Fig. 9 illustrates this behaviour by displaying the distribution of the environments used in the literature. Walker2D, MuJoCo robotic arm simulation, Hopper, CartPole, MountainCar and HalfCheetah are the most common environments. We classify five of these six as validation environments, except for the robotic arm. As pointed out in Section 5.1, this trend worries us since using only validation environments might incur the wrong evaluation of the imitation learning agent. Nevertheless, 42 out of the 66 environments appear only once in the literature. This behaviour is worrisome because it shows a lack of experimental protocols. We recognise that various tasks and domains may require particular settings. However, when analysing the work for autonomous vehicles, for example, we observe that the simulations were used only once on all domain-relevant work. Furthermore, Zheng et al. also point out this behaviour in their survey [121], when presenting the high-level and low-level tasks, where they argue that researchers have sometimes might bias their evaluation by selecting different environments. We attribute this behaviour to reviewers and researchers looking to evaluation metrics and not to evaluation protocols, which shows a need for a more formal definition of more thorough protocols.

6 Imitation Learning Metrics

Imitation learning evaluation metrics can drastically differ from each work. Some work rely on more domain-specific metrics, while others present metrics that measure how the agent performs compared to its teacher. The most common taxonomy for evaluation metrics is the quantitative and qualitative classification. However, this differentiation fails to consider the context of a metric. For example, a metric that evaluates the distance

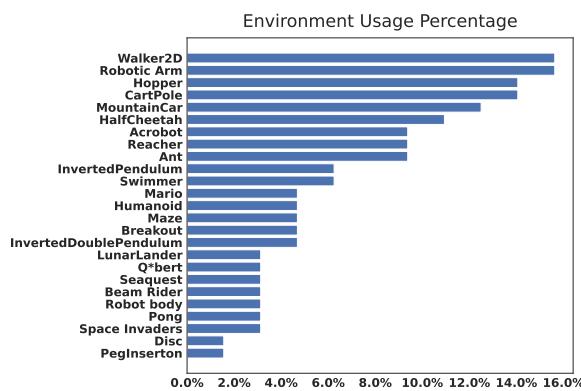


Fig. 9. 25 most used environments distribution.

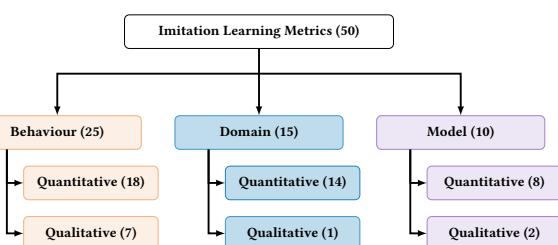


Fig. 10. Imitation learning metrics taxonomy.

between an agent's trajectory and its teacher in a task requiring precision will convey that the agent is performing well. Conversely, if the environment requires less precision and a goal state holds higher importance, using a metric that measures an entire trajectory might convey that an agent that does not reach its goal state but follows the teacher's trajectory until a point, such as stopping right before the goal, is better than one that takes a significantly different trajectory but reaches the goal. Therefore, we classify metrics as quantitative and qualitative but under behaviour, domain and model metrics. Behaviour metrics are those that measure the agent's behaviour, they convey how distant an agent is from its teacher. One might argue whether behaviour metrics measure teachers behaviour [39], but we shy away from this discussion for now. Domain metrics measure domain-specific properties; they bring a more contextual measurement for tasks with more specific requirements, such as how many traffic infractions an agent commits in a self-driving car environment. Finally, model metrics measure the learning procedure of agents, such as how accurate a model is when predicting an action.

6.1 Behaviour metrics

Behaviour metrics are the most common metrics among all imitation learning work. Usually, they depend on the environment's reward or the distance from one property of the teacher. The most used metrics are reward-based ones since comparing the reward from an agent to its teacher intuitively shows how the agents perform comparatively. Three variations from the reward-based metrics are present in the reviewed literature: (i) accumulated reward; (ii) average episodic reward; and (iii) performance.

Accumulated reward is how much reward the agent accumulates in an episode. We reiterate here that an episode in this work differentiates from Russell and Norvig's definition [89] and refers to a set of experiences from an agent in an environment. The final value is given by summing the reward retrieved from the Markov decision process in an episode with n steps (Equation 4). This metric conveys how the agent performs the environment task and allows researchers to compare student and teacher rewards. However, using a single episode might be misleading. For example, suppose the *seed* (a random numerical value used to initialise the environment) used in the evaluation is present in the teacher dataset. In that case, the agent must only match the given states to those in the training set to achieve the same reward. Moreover, even when the seed is absent in the teacher dataset, some initial states might be closer to the teacher's initial state than others. Thus, some states require less generalisation from the agent than others, making some initial states 'easier' than others. Therefore, the accumulated reward might not be the best metric to measure how well the agent generalises in some unfavourable instances.

$$AR(\pi) = \sum_{i=1}^n \gamma^n r(s_i, \pi(s_i)) \quad (4)$$

A more statistical approach would be to use the average reward for a set number of episodes and study the deviation between each experience. Therefore, the chance of finding diverse seeds increases, and the standard deviation among all episodes can show whether the agent is consistent among different experiences. The average of all accumulated rewards is called the *average episodic reward*. In it, the agent records its accumulated reward for k number of episodes, and the average is calculated (Equation 5). The number of episodes is usually 100 or 10, and the standard deviation is shown alongside it. The average episodic reward has the same benefits as the accumulated reward and solves the problem from untested generalisation, as long as there are enough episodes. However, it can be the case that an agent's average episodic reward is closer to that of a random initialised agent than to that of a teacher in a specific task. To understand how an agent compares between all agents, all three reward values would have to be given.

$$AER(\pi) = \frac{1}{k} \sum_{j=1}^k AR(\pi) \quad (5)$$

Performance solves the comparison issue by applying normalisation in the accumulated reward. It uses a random policy's π_ξ accumulated reward as the minimum and the teacher's as the maximum (Equation 6). In this metric, if an agent accumulates as much reward as a random one, the performance will be 0. However, if the same agent accumulates the same reward as the teacher, its performance will be 1. An agent can achieve a performance higher than 1 when it accumulates more than its teacher and lower than 0 when its rewards are inferior to a random agent. The performance metric is usually retrieved from k trajectories to get the same benefits the average episodic reward has. However, some work directly uses the average episodic reward instead of the accumulated reward when computing the metric (Equation 7).

$$\mathcal{P}_\tau(\pi) = \frac{AR(\pi) - AR(\pi_\xi)}{AR(\pi_\psi) - AR(\pi_\xi)} \quad (6)$$

$$\mathcal{P}_{\mathcal{T}}(\pi) = \frac{AER(\pi) - AER(\pi_\xi)}{AER(\pi_\psi) - AER(\pi_\xi)} \quad (7)$$

Even though the performance metric has all the benefits of the other reward-based metrics, it inherits the same issues from linear normalisation procedures when dealing with skewed data [39]. For example, suppose we assume that the minimum reward (random) is 0, the maximum (teacher) is 100. The mean score from agents is 50, and the median score is 20. In this case, the majority of the performance for the agents will be clustered towards the minimum, and there will be very little differentiation among them. Thus, performance becomes a less useful metric when the agent's performance is close.

Although the metric does not optimise the agent, it can misrepresent the agent's behaviour in a given environment. For example, imagine two scenarios for an environment where the threshold to be considered optimal is a 100 reward: (i) the teacher reward is 75, and the random reward is 0; and (ii) the teacher reward is 100, and the random reward is again 0. In the first scenario, when an agent achieves teacher behaviour, it will score a performance of 1, even though the agent was 25 points short from an optimal result. By achieving the same reward from the first scenario (75) in the second scenario, the agent will only achieve a performance of 0.75. This example shows the problem of assuming as a premise that the teacher is an optimal policy. If the selected teacher is not optimal, the performance metric will not accurately represent how good the agent is. Moreover, in the first scenario, although the agent is 25 points from being considered optimal for the task, the metric conveys optimality. Hence, there is a need to know beforehand how optimal the teacher is, which can be a complex problem in several application domains. On the other hand, if an agent achieves a reward of 150 in the same example, the performance will be 2 and 1.5 for the first and second scenarios, respectively. Although the agent achieves performance higher than the teacher, this result shows that: (i) the samples might not be ideal; and (ii) the policy might not have learned the exact behaviour from the teacher.

Distance metrics are the most common metrics after reward-based ones. They usually involve measuring how far the agent is from one or more properties of the teacher. For example, some work measures the distance between the teacher's and agent's trajectories or between the artificial reward function and the MDP's r . Although the distance metric can vary, the most common distances are the *Manhattan distance* (Equation 8, where $p = 1$) and *Euclidean distance* (Equation 8, where $p = 2$), where an input I may be single states, trajectories or state-action pairs, and the *Kullback-Leibler divergence* (Equation 9), where P and Q are two probability functions. A common issue with distance-based metrics is the requirement for teachers to be accessible to measure the chosen property (when observations do not provide the desired property). These metrics require some higher degree of control for measuring the same property for the agent. An example would be measuring the precision of unseen trajectories. One might reserve part of the trajectory data to measure how close agents and teachers are. However, this setup would require a more significant number of data from researchers. A second issue with distance-based metrics is that they might misrepresent the behaviour of an agent due to compounding errors from different actions during the early stages of an episode. Suppose an agent acts differently from its teacher earlier on. In that case, the action might cause the trajectory to be further apart even though the agent's behaviour might be close to

the teacher after these earlier steps. This approach to measuring would require researchers to have a tree of all possible teacher's actions to compare the agent's actions given a diverging path from its initial data.

$$d(I_{\pi_\theta}, I_{\pi_\psi}, p) = \left(\sum_{i=1}^n \left| I_{\pi_\theta}^i - I_{\pi_\psi}^i \right|^p \right)^{\frac{1}{p}} \quad (8)$$

$$d(P \parallel Q) = \sum_{s \in S} P(s) \ln \left(\frac{P(s)}{Q(s)} \right) \quad (9)$$

The *success rate* (*SR*) also appears as a viable quantitative behavioural metric. In it, the authors usually select a goal g for the environment (it could be the environment's goal or a user-defined one) and collect a set of trajectories \mathcal{T} , one for each episode, and check whether each trajectory τ has g in it (Equation 10).

$$SR(g, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} G(g, \tau), \text{ where} \quad (10)$$

$$G(g, \tau) = \begin{cases} 1 & \forall n s_n \in \tau \mid \exists_t s_n = g \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Coupling a reward-based and distance-based metric with a success rate can help researchers understand how the agent behaves. Both initial metrics convey information about the agent's trajectory while the latter conveys if the agent achieves its intended goal.

Nevertheless, these metrics do not have information regarding how human-like the agent behaviour is (considering the agent trained with human demonstrations). Therefore, some work employ qualitative metrics like questionnaires and Turing tests. For questionnaires, questions are usually regarding how similar to humans the agents look [55] or how easy it is to play with the agent [61]. In the Turing Test, participants classify if a video from a agent interacting with the environment is human or agent. Thus, the agent aims to deceive the human participants (similar to adversarial learning). Moreover, other researchers [15, 23, 35] also observe the agent's trajectory and do some qualitative analysis. More related to robotics, this analysis usually displays a graph with various trajectories for the robot's movement, and the authors explain why the divergence in trajectory is good or acceptable in a precise task. Unfortunately, the qualitative metrics that require human participants are costly in comparison to the quantitative ones. Thus, they are less common in more recent work. Although these metrics are essential for evaluating imitation learning agents, we understand that to use qualitative metrics requires significant effort. Creating a questionnaire requires question design, which is uncommon, and the Turing test requires participants to watch numerous videos of humans and agents playing to vote.

6.2 Domain metrics

It is not always the case that behaviour metrics convey the information required to evaluate whether an agent performs adequately in an environment. Some domains allow the usage of specific metrics to measure the optimality of the agent. These metrics are specific to these domains and do not make sense to use anywhere else. The most common domain-specific metrics are those used in autonomous driving and robotics.

For driving tasks, different behaviours from the agent can be measured, such as distance travelled [23, 87], traffic infractions [18], time to collision and distance headway [46], allowing for more accurate measurement of how safely the agent navigates through the environment. Even though the reward function could contain this information, by separating them, the researcher can better comprehend the agent's behaviour and try to optimise different things without altering the reward function to represent the desired behaviour better.

In robotics, most domain-specific metrics measure time [23, 24, 87] or failure rate [73, 87]. In the case of time, these metrics convey how efficient the robot arm is in performing a task. For example, how long it takes to order a set of blocks, or how precise the robot can perform a variety of movements, such as the pixel difference between teacher and agent in the same settings. For failure rate, most metrics display the percentage of times the robot failed to reach its goal, such as running in a simulation. Conversely, we did not find any work that gave intuition

behind the failure, which is crucial to understanding whether the final result is good or bad. We understand that doing this analysis is more costly and adds a qualitative tone to it, which the domain-specific metrics lack; however, only providing a number representing failure, such as how many times the robot fell during a run, makes it hard to compare. Compared to humans, competitors almost never fall during a running competition, and to other robots, the results might be significant.

Considering that imitation learning agents do not use reward functions to learn, it would not be very sensible to require a reward function to measure how optimal an agent is. Therefore, some work use domain-specific scoring to show how an agent performs. For example, in some environments, researchers use how many coins [29, 55] the agent collects during an episode or how many goals a team scores [24, 86]. Nevertheless, these metrics lack the finesse of other metrics and should be used with behavioural- or model-specific metrics.

6.3 Model metrics

Model metrics are the usual metrics we see in learning research. They convey how accurate or inaccurate a model is. Most work use *accuracy* since imitation learning approaches use some form of behavioural cloning, which is a supervised-learning approach, where accuracy is defined as the ratio between the number of correct predictions and the total number of predictions. Other work use the *error rate* from their models, which usually rely on generative models, such as forward dynamics models. However, these metrics do not convey useful information regarding the agent's performance in a task. An example is the accuracy of inverse dynamics models. Although they can have high accuracy in their demonstration datasets, they might have poor accuracy in the teacher's observation since it usually follows other distributions. Therefore, accuracy might give the wrong impression that the model is classifying each transition correctly when it is not. Model-specific metrics are used during training to understand whether a model can learn the desired task. Additionally, some work also uses these metrics to interpret when a model becomes biased toward a set of actions [38].

On the other hand, some work [5, 57] use other model metrics to have intuition behind the model's learning. In these work, the authors use dimensionality reduction techniques, such as *t-SNE* [109], to interpret how good their models are in clustering different actions. Although these techniques are promising to give some qualitative intuition behind the models' performance in clustering information, these metrics are often used when the work focuses outside imitation learning, for example, transfer learning. Moreover, when using these dimensionality reduction techniques, one should consider that: (i) they are highly sensitive to hyperparameters, as pointed out by van der Maaten and Hinton [109]; (ii) interpreting them might be challenging [113], which makes them less suitable as an explainable method; and (iii) some of these techniques lack robustness and might not provide consistent results for similar datasets [10].

Saliency maps are also used as qualitative metrics in imitation learning literature [38, 79] as model-domain metrics. They display in the model's input where does are the most active gradients. It gives some intuition on what the model focuses on when predicting an action. Fig. 11 displays an example where the authors use the input image (Fig. 10a) to display the attention map (Fig. 10b) to understand what the agent may be using to classify the action. On the other hand, saliency maps share three of the major drawbacks of dimensionality reduction techniques, which we understand we need to focus on: (i) subjectivity; (ii) unreliability; and (iii) interpretability. The first comes from the fact that saliency maps are influenced by the choice of the saliency model and parameters used in their generation. Different models may produce varying results for the same input, and there is no consensus on which model is the best [14]. Given the subjectivity of saliency maps, it can be sensible to assume that such behaviour might cause unreliable results [1, 52]. Although these models can produce impressive results when connecting labels with raw input data, connecting more complex knowledge representations might be difficult, especially when connecting internal knowledge representations from neural networks to humans [45].

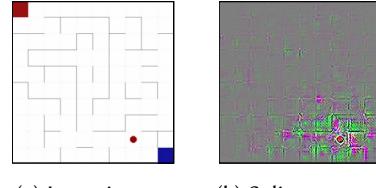
Finally, saliency maps might be challenging to interpret [68]. When interpreting these results, one should remember that there are different possible conclusions given a combination of input and label.

Therefore, researchers should consider using these model metrics to provide an intuition of what imitation learning agents learned or how they behave, but not as concrete proof. Furthermore, a combination of these metrics with domain-specific and behaviour-specific metrics, which show a more accurate representation of how optimal the agent is, should be present for a more complete conclusion.

Algorithm 1 Default evaluation protocol

Require: $D \leftarrow \{(s_0, a_0); \dots; (s_n, a_n)\}$

- 1: train π_θ with \mathcal{T}
- 2: $AER_{\pi_\theta} \leftarrow 0$
- 3: $\mathcal{P}_{\pi_\theta} \leftarrow 0$
- 4: **for** $n \leftarrow 1$ to N **do**
- 5: $AER_{\pi_\theta} \leftarrow AER_{\pi_\theta} + AR(\pi_\theta)/N$
- 6: $\mathcal{P}_{\pi_\theta} \leftarrow \mathcal{P}_{\pi_\theta} + \mathcal{P}(\pi_\theta)/N$



(a) Input image. (b) Saliency map.

Fig. 11. Saliency map example from Gavenski et al.'s work [38].

6.4 Evaluation protocol

All work presented in Section 4 follow in some form the structure in Algorithm 1 when evaluating their method. They train a policy π_θ , given a set of teacher demonstrations D (Line 1). For simplicity, here we consider that all samples are demonstrations (have action notations) (Line 0), but one should assume that this is only true for learning from demonstration scenarios since learning from observation does not require action information during training. Algorithm 1 also only considers trained policies since it refers to the final evaluation protocol, so we remove any iterative training notation. Finally, researchers report using some metrics presented in this section. The number of episodes from an agent in an environment (N) varies between 5 to 100. Although imitation learning work rarely inform how the seeds are selected. It is sensible to assume that the seeds used to collect teacher demonstrations are absent from the evaluation process. Unfortunately, this assumption is not always true since the newest version from OpenAI Gym [107] removes the option for users to set a specific seed for the environment. Therefore, some work use the same seed for both teacher and agent, which can lead to unfair comparisons.

Moreover, some methods employ other experiments to evaluate their performance in scenarios with different amounts of samples (sample efficiency) [40, 51, 70, 122]. In these scenarios, researchers reduce the number of teacher episodes the agent has access to and perform the same evaluation protocol in Algorithm 1. Other work experiment with different degrees of optimality and measure each method's robustness [39]. In it, they create teacher samples with different degrees of optimality and perform the evaluation protocol in Algorithm 1.

Finally, to compare their work, other approaches run the same steps from Algorithm 1 for all baselines selected. Researchers frequently select the traditional behavioural cloning method as a benchmark alongside newer approaches in the same category as the proposed method. For example, learning from observation methods usually compare their approach with behavioural cloning since it has access to the teacher's actions and works as a ground-truth comparison. Moreover, adversarial learning methods compare their approach with Ho and Ermon's work [44], even though they are outdated.

6.5 Metrics Discussion

In this section, we listed different metrics with different contexts for imitation learning work. We divided them into three different groups: behaviour, domain and model. When studying these metrics, we found 50 different ones (differentiating them by context and equation and not only by name). As discussed in Section 6.1, most metrics

focused on quantitative analysis, with the most common one using reward to measure how optimal the agent is. As with the environments, the metrics also present many that appear once. However, unlike environments, comparing all methods with a new metric presents fewer unfair comparison problems if researchers compare all methods using the same metric. Additionally, we observe an increase in performance usage, which considers policies from random and teachers agents to measure optimality.

Qualitative metrics are rare in most novel work, which is understandable since the cost of creating questionnaires or running Turing tests might make some research unfeasible. However, the absence of qualitative metrics in the literature is problematic since it does not allow researchers to understand how human-like the agent is. In this literature review, there are 109 uses of quantitative metrics over 41 different metrics. In comparison, there are only 10 uses of 9 different qualitative metrics.

Finally, we analyze different protocols for evaluating imitation learning methods. Most work follows the same guidelines when evaluating novel methods and baselines. In its essence, the presented protocol offers a straightforward way to evaluate imitation learning methods. However, researchers fail to show how their method compared outside of the optimal scenario by not applying the same protocol to measure other characteristics outside the overall performance, such as efficiency and effectiveness. We conclude that following protocols presented in Section 6.4, such as our protocol where we reduce the degree of optimality of experts [39], might alleviate some of the issues from the original protocol but will not solve them.

7 Reflections

In this section, we reflect on the insights we had while writing this survey. These reflections can help researchers better understand imitation learning and its challenges, which we discuss further at the end of this section. Importantly, we aim to help new researchers design imitation learning agents while avoiding common pitfalls.

For this section, we consider that under the loss function ℓ , an agent learns how to maximise the return of ℓ to perform a task under certain conditions. ℓ might be the accumulated value of M 's reward function ($\sum_i \gamma^i r_{i+1}$) or capture some other specific desired behaviour. For example, ℓ may want to minimise the number of vertices in the solution of a path-finding problem, or instead compute the overall length of the path according to the weight of the edges between vertices. Training a policy π with parameters θ and loss function ℓ yields the value function $v_\ell(\pi_\theta, s)$, which measures the quality of π for each state s . A policy π with parameters θ_o is *optimal* for ℓ when for all states s and all parameters β , $v_\ell(\pi_{\theta_o}, s) \geq v_\ell(\pi_\beta, s)$. Moreover, π is *theoretically optimal*, when for all loss functions h and all parameters β , $v_\ell(\pi_{\theta_o}, s) \geq v_h(\pi_\beta, s)$. In other words, even though an agent may be optimal under a given function ℓ , it may still under-perform under another loss function h .

7.1 Imitation Learning Evaluation

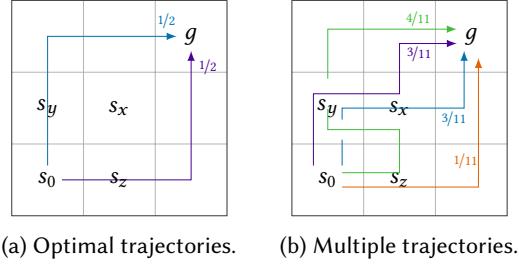
When we consider IL systems, we are likely to compare them to machine learning (ML) and reinforcement learning (RL) approaches. After all, IL methods are a relaxation of the RL approach into an ML supervised classification problem. However, this comparison comes with some preconceptions that we want to address: (i) disparities over testing processes; (ii) divergence in benchmarking; and (iii) the main goal of IL approaches.

Traditional ML applications divide their data into three different group sets: training, validation and test sets. They assume that all sets share the same distribution over data, and the test set should be only used at the end to evaluate how the model will perform under a ‘real-world’ scenario. This allows a well-defined and clear evaluation. Conversely, IL approaches shy away from using test sets. Since the typical application for IL approaches is agent-based, it is more practical to test the final learned policy via simulation of the environment. This testing approach is similar to how RL methods are evaluated. However, the training processes for RL and IL agents are different. Training RL agents involves the use of exploration and exploitation phases without a training set. During these phases, agents learn the different values for states outside an optimal trajectory. Thus, even though a RL agent

might not visit a state often, the agent will have some information about how to act on it. The training of IL agents is limited to the states in the training set, which means they have no information about states outside this set. For example, consider the grids in Fig. 12, where the objective is for the agent to reach square g . Suppose the training set of an IL agent consists only of the two optimal trajectories in purple and blue given in Fig. 12a. After training, the agent will have acquired some knowledge about how to get to g from any square except for s_x . Furthermore, the probability of reaching s_x is 0% from the learned policy's perspective since the teacher samples prevent the agent from reaching s_x from any state. Now consider the training set in Fig. 12b, containing 4 green, 3 purple, 3 blue, and 1 orange trajectories. The learned policy will not be successful for 50% of the non-goal states, because the states s_0 , s_x , s_y , and s_z lead to an infinite loop. If the agent starts in s_0 , it will go to s_y , since $P(\text{'up'} | s_0) = 6/11$. If it starts in s_z , then it will go to s_x , since $P(\text{'up'} | s_z) = 4/5$. Once in s_x or s_y , it will loop because from s_x , it will go to s_y , since $P(\text{'left'} | s_x) = 4/10$ (highest) and from s_y , it will go to s_x , since $P(\text{'right'} | s_y) = 6/10$. Therefore, it is important to understand how the simulations used for testing relate to the samples used in the training since particular initialisations of the tests might involve states and behaviours not present in the training data. In the example given in Fig. 12, it is possible to understand the looping behaviour from a manual inspection alone. However, this may be impossible in environments with a high number of states, such as continuous environments or for higher dimensions. Moreover, other considerations are needed to understand the evaluation results of IL agents, such as poor performance due to drastic state variance from training and testing, or good performance due to lower variance and *data leakage*, which occurs when the training data is present in the test set.

Secondly, with the popularity increase of ML approaches, benchmarking datasets have helped to evaluate how different methodologies perform under the same conditions and some applications hide test data from researchers to maintain evaluation integrity [31, 88]. Moreover, using the same dataset (and the same proportion of training/testing split) allows researchers to validate each other's results more efficiently, and simplifies the evaluation process because there is no need to run every relevant baseline over the data again. RL work uses the same benchmarking approach, with the performance of RL methods evaluated on popular common environments [83]. Unfortunately, the same uniformity in evaluation has not yet been employed in IL methods, which usually create customised training data and compare performance with baselines over the data for which the other methods were not initially created. Therefore, when comparing a new approach, a teacher dataset must be created, available reliable code found (with hyperparameters or requiring a vast search if wanting to compare with the best results for each method), and every baseline must be run over the generated data, leaving much room for issues, such as lack of data or diversity. This divergence in benchmarking further corroborates the main problems from the disparities in the test process [121].

Finally, we argue that IL's ultimate goal is to learn how to accomplish a task using the teacher's data and not simply to copy the actions taken by the teacher. In other words, the IL agent should act as the teacher would in previously seen states, but it should also somehow learn how the teacher would act in unseen states. For example, in Fig. 12a, given state s_x , 'up' and 'right' are optimal actions *ideally* to be taken by the IL agent. If that is not possible, the agent should at least attempt to choose an action that takes it back to an already-seen trajectory. Analogously, the problem highlighted in Fig. 12b stems from the fact that the IL alternates between competing trajectories without ever completing either.



7.2 Expert optimality feasibility

Imitation learning always assumes that training samples come from the execution of a task by an *expert* [47]. While it is fair to assume that the source is proficient, the implicit presumption of the *theoretical* optimality of the demonstration itself is problematic. Firstly, the more complex the environment, the harder it is to ascertain whether the samples are optimal. For example, while it might be easy to identify which episodes are near optimality in the CartPole environment [9] (where performance is considered optimal when the accumulated rewards match the number of steps), it is much more challenging to evaluate how ‘good’ an agent is in autonomous driving scenarios, such as SUMO [60]. In particular, some important aspects of the driver’s behaviour (e.g., *situational awareness*) may be difficult to fully measure. Moreover, in highly dynamic systems such as in real-world driving, how can one be sure that the samples collected fully capture all aspects of the optimality of the execution? In practice, even though the teacher might be *proficient*, the samples collected may be corrupt or not cover all the experiences necessary for the learner to achieve the same level of optimality. Although it is possible to evaluate performance in both cases, it might be impractical to do so. Since reproducibility is crucial, we expect an increasing focus on presenting more data about the teacher’s behaviour, an aspect that is lacking in most work, except for [11, 39].

Secondly, we argue that the demonstrations may not necessarily fully capture the intended optimal behaviour. For example, they may not necessarily achieve other secondary goals, such as fairness and safety, that may not play a part in the scenarios available for the learning. We note that safe and fair agents are not inherently suboptimal, but as in the first scenario, it might be impossible to create a perfect fitness function to account for the desired behaviour. In these cases, samples may unintentionally be used where the teacher does not fully constrain actions, resulting in less than theoretical optimal agents (with respect to e.g., safety requirements). For example, suppose an agent is trained to drive a car and uses samples from a human driver, who may drive incautiously. In this case, the agent will learn like its teacher, which is optimal for some driving scenarios but not optimal in situations such as emergencies where the driver must act with extreme caution. We therefore believe that only striving for optimal agents is not always sufficient.

Furthermore, consideration for aspects beyond the agent’s overall reward with a focus on human feedback is central to ‘Reinforcement Learning from Human Feedback’ [20, 33]. Here, theoretical optimality is sacrificed for other four aspects. Firstly, humans can be more nuanced than machines when taking action and safety and ethical considerations override overall performance. Secondly, humans have intrinsic notions of solution quality that are difficult to capture in the design of reward functions. Human feedback reduces the need for intricate designs. Thirdly, human guidance in the agent’s exploration phase helps to reduce the training time. Lastly, human feedback allows for *curriculum learning*, where the agent starts with human-curated samples and gradually transitions to more autonomous learning paradigms. Since imitation learning, by definition, relies on human examples to guide the entire learning process, we argue that IL agents should also consider these qualitative aspects, which are possible avenues for future research.

Therefore, the trade-off between expert and teacher agents has to be understood when developing a new agent. In some cases, such as robotics for manufacturing, it is desirable to employ expert policies because they are guaranteed to be most efficient. In others, such as driving, optimal behaviour may be difficult to demonstrate, and teacher policies are acceptable. We argue that by understanding the role of experts and teachers in a task, researchers can make more informative decisions about the most appropriate type of demonstrations and what the performance evaluation really measures.

7.3 Expert Necessary Requirements

As we saw in the previous section, it is desirable and perhaps inevitable that IL agents sometimes learn from non-optimal sources. If an environment follows an MDP (as described in Section 3), it has a universe of all possible policies Π (that either succeed or fail to solve the Markovian problem). Some successful policies are also

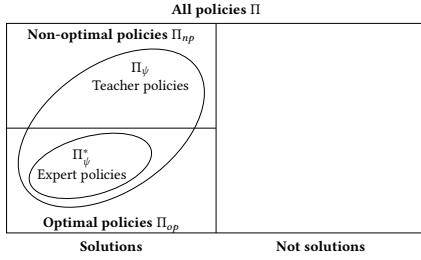


Fig. 13. Diagram of teacher, optimal and expert policy sets.

Table 1. Necessary requirements for expert validation.

Aspect	Description
(i) State space	Finite / Discrete
(ii) Action space	Finite / Discrete
(iii) State observability	Fully observable
(iv) MDP sample coverage	All MDP states

optimal (Π_{op}). The training data does not usually contain all optimal policies, restricting itself to a subset of these (the ‘expert’ policies Π_{ψ}^*). In this scenario, researchers should strive to acquire all optimal policies for training the agent, such that the expert policy set is *complete* ($\Pi_{\psi}^* = \Pi_{op}$). In general, it is impossible or impractical to do so, resulting in incomplete coverage. Furthermore, in practice, the successful policies used to teach a task or skill also contain policies that are not optimal (the ‘teacher’ policies Π_{ψ} , which we also call *complete* when $\Pi_{\psi} = \Pi_{op} \cup \Pi_{np}$). Including non-optimal policies in Π_{ψ} can happen because researchers sometimes provide their own demonstrations, and it may not be possible to check them for optimality. Therefore, we argue that for clarity, we should call the policies used for training simply as *teachers*, reserving the term ‘experts’ for policies whose optimality has been ascertained. Fig. 13 illustrates the relationship between all policies.

We now look at the role played by the teachers’ policies. Table 1 presents a set of necessary requirements all of which need to be fulfilled for the source data to be reliably considered an expert. (i) and (ii) relate to the state and action space from environments. Most often, work that use continuous (infinite) states and actions spaces in their problem formulation impose strict assumptions, such as injective MDPs [122] and trajectories with the same initial states [51]. These assumptions are not applicable in most practical scenarios or require extensive modifications. Therefore, we can assume that work whose environments have infinite state and action spaces cannot guarantee optimality of the training data. (iii) refers to the observability of the environments. Optimality of policies for environments with partial or faulty observations [119] cannot be guaranteed. (iv) considers how much of the state space the samples collected cover. We argue that in cases where the samples do not fully cover the Markovian problem, we cannot guarantee that the agent fully learns the expert’s behaviour. Hence, the output from training might be an optimal policy but does not guarantee optimal behaviour in unseen states. Nevertheless, we observe that although researchers might be able to fulfil these conditions for optimality, they should not be used as the sole criteria for the evaluation of the overall performance of the learning approach, since they are too restrictive. In real-world applications, it is often the case that one of more conditions need to be relaxed for practical use.

7.4 Challenges and Future Directions

Throughout this survey, we have discussed several challenges that imitation learning agents face. This section highlights some of them and discusses possible future directions for imitation learning research.

Safety is a crucial aspect of agent-based systems. Without it, agents can cause harm to themselves, other agents, or the environment. However, for imitation learning agents, safety can be hard to achieve since they learn through a teacher’s perspective and without a direct signal. When learning via observation, the samples might lack information regarding the environment’s dynamics, leading to unsafe behaviour due to the agent’s lack of knowledge. Moreover, the teacher might not behave safely, leading to an unsafe agent. Thus, we believe there

is space to research how imitation learning agents can learn safely from teachers. Possibly, imitation learning agents should learn how to differentiate between safe and unsafe samples and teachers.

Learning efficiency became a trend for imitation learning algorithms [70, 97, 106, 118]. For imitation learning, researchers measure efficiency by the number of samples needed to learn a task. However, we believe that efficiency by itself might be a misleading metric. For example, an agent who learns a task by only observing a single episode from a teacher might be considered efficient. Conversely, suppose the agents fail to generalise to other scenarios or act unsafely. In that case, their efficiency is not as good as it seems. Thus, efficiency should be measured by the number of samples needed to learn a task while measuring other factors, such as safety.

Learning effectiveness is often overlooked by imitation learning researchers. In this work, we define effectiveness as the ability of an agent to learn a task or skill optimally (or close to it) with non-optimal teachers. Effectiveness is as crucial as efficiency since it is not enough to learn a task or skill with few samples if the agent does not learn it well. Moreover, we believe effectiveness is more challenging than efficiency due to how imitation learning agents learn. Considering our premise that is impossible to know how optimal a teacher is in real-world applications, it is essential to develop agents that can learn effectively from non-optimal teachers. Thus, we believe that imitation learning researchers should focus more on effectiveness than efficiency.

Adaptability comes from combining generalisation and *transfer learning*. Generalisation is the ability of an agent to learn a task or skill from a teacher and apply it to other unseen scenarios (similar to effective learning). Transfer learning is the ability of an agent to learn a task or skill from a teacher and apply it to other tasks or skills. More formally, transfer learning in imitation learning refers to the agent's ability to translate knowledge it learned from one Markov Decision Process M to another M' , either regarding the transition function T (from T to T') or in the form of a new immediate reward function r . Considering that imitation learning agents do not have access to r , this form of transfer learning usually appears in the sense of T remaining the same but the goal of the environment changing. Given the nature of the imitation learning agent's training and lack of a direct signal, we believe these two aspects are essential to strive for. By achieving generalisation or transfer learning, imitation learning agents can succeed more in real-world scenarios. However, researchers should focus on learning paradigms outside imitation learning, such as symbolic learning, to achieve these goals.

Multi-agent systems are not a research topic as common as single agents for imitation learning researchers. We believe such behaviour comes from imitation learning being more adequate for learning low-level tasks, which usually multi-agent systems do not have. However, imitation learning can be used to learn high-level tasks, such as coordination and communication, which are essential for multi-agent systems. Imitation learning is a highly adaptable process that can be employed with other techniques, such as planning [81, 118], explorations [38, 51]. Perhaps using imitation learning on more descriptive processes, such as 'Belief Desire and Intention', can help imitation learning agents achieve better results on multi-agent systems scenarios.

8 Conclusions

In this survey, we present a comprehensive review of imitation learning methods, environments and metrics. We provided background knowledge required to understand the field and how different imitation learning methods use data to learn a policy. One of our contributions is the provision of novel taxonomies for methods, environments and metrics.

We first classified *methods* according to new trends in the field, highlighting key aspects. This should help researchers identify new opportunities without supplanting the traditional model-free/model-based dichotomy.

Our new taxonomy for *environments* focuses on how they are used in the evaluation process, i.e., for *validation*, to assess *precision*, or whether they are *sequential* – when some actions can influence the choice of future actions. This focus on purpose should help proponents of new methods choose the most appropriate environment for their methods, allowing for more effective evaluations.

We then systematically reviewed the *metrics* used to evaluate imitation learning methods, partitioning them into the three categories *behaviour*, *domain*, and *model*, which can be further refined into qualitative and quantitative. We highlighted the importance of these categories in the evaluation of the corresponding characteristics of the method's performance. Given the variety of metrics, an open question is whether and how they can be aggregated to rank or better compare methods more generally.

Finally, we showed a lack of standardisation in the evaluation process of the imitation learning field, making it difficult to compare methods uniformly under specific perspectives. To the best of our knowledge, this is the first IL survey to comprehensively and systematically review IL methods, environments and metrics in this way.

While there has been much recent progress, we believe the imitation learning field still shows much potential for future research. This survey provides a solid foundation from which to further develop the area – one that allows the systematic evaluation of new approaches with regard to the state-of-the-art. In particular, by contrasting the different ways in which existing work has been evaluated, we see that there is much to improve in terms of the methods' resulting behaviour beyond the use of the environmental metrics that traditional reinforcement learning employs. More specifically, imitation learning can benefit from work in other research fields that strive for more human-like behaviour, such as reinforcement learning from human feedback.

References

- [1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.), 9525–9536. <https://proceedings.neurips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7fea049b8737-Abstract.html>
- [2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine* 34, 6 (Nov. 2017), 26–38. <https://doi.org/10.1109/msp.2017.2743240>
- [3] Pierre Aumaud. 2016. Maze Gym. GitHub. <https://github.com/MattChanTK/gym-maze>
- [4] Pierre Aumaud. 2020. Jaco Gym. GitHub. <https://github.com/PierreExeter/jaco-gym>
- [5] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. 2018. Playing hard exploration games by watching YouTube. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/35309226eb45ec366ca86a4329a2b7c3-Paper.pdf
- [6] Michael Bain and Claude Sammut. 1999. A Framework for Behavioural Cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*. Oxford University, GBR, 103–129.
- [7] Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. 2017. End-to-End Differentiable Adversarial Imitation Learning. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 390–399. <https://proceedings.mlr.press/v70/baram17a.html>
- [8] Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shible Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2020. The Hanabi challenge: A new frontier for AI research. *Artif. Intell.* 280 (2020), 103216. <https://doi.org/10.1016/j.artint.2019.103216>
- [9] Andrew G Barto, Richard S Sutton, and Charles W Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* 1, 5 (1983), 834–846.
- [10] Etienne Becht, Charles-Antoine Dutertre, Immanuel W. H. Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W. Newell. 2018. Evaluation of UMAP as an alternative to t-SNE for single-cell data. (apr 2018). <https://doi.org/10.1101/298430>
- [11] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. 2023. Data Quality in Imitation Learning. *CoRR* abs/2306.02437 (2023). <https://doi.org/10.48550/ARXIV.2306.02437> arXiv:2306.02437
- [12] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* 47 (jun 2013), 253–279. <https://doi.org/10.1613/jair.3912>
- [13] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (March 2008), 156–172. <https://doi.org/10.1109/tsmcc.2007.913919>
- [14] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Fredo Durand. 2019. What Do Different Evaluation Metrics Tell Us About Saliency Models? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 3 (mar 2019), 740–757. <https://doi.org/10.1109/tipami.2018.2815601>

- [15] Sylvain Calinon and Aude G. Billard. 2007. What is the teacher's role in robot programming by demonstration? *Interaction Studies* 8, 3 (oct 2007), 441–464. <https://doi.org/10.1075/is.8.3.08cal>
- [16] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the Utility of Learning about Humans for Human-AI Coordination. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf
- [17] Wei-Di Chang, Juan Camilo Gamboa Higuera, Scott Fujimoto, David Meger, and Gregory Dudek. 2022. IL-flOw: Imitation Learning from Observation using Normalizing Flows. 2205.09251 (2022). arXiv:2205.09251v1 [cs.LG]
- [18] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Learning by Cheating. In *Proceedings of the Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 100)*, Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (Eds.). PMLR, 66–75. <https://proceedings.mlr.press/v100/chen20a.html>
- [19] S. Chernova and M. Veloso. 2008. Teaching collaborative multi-robot tasks through demonstration. In *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*. IEEE. <https://doi.org/10.1109/ichr.2008.4755982>
- [20] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf
- [21] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. 2019. Quantifying Generalization in Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 1282–1289. <https://proceedings.mlr.press/v97/cobbe19a.html>
- [22] Rémi Coulom. 2002. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control. (Apprentissage par renforcement utilisant des réseaux de neurones, avec des applications au contrôle moteur)*. Ph.D. Dissertation. Grenoble Institute of Technology, France. <https://tel.archives-ouvertes.fr/tel-00003985>
- [23] Shreyansh Daftary, J. Andrew Bagnell, and Martial Hebert. 2017. Learning Transferable Policies for Monocular Reactive MAV Control. In *Springer Proceedings in Advanced Robotics*. Springer International Publishing, 3–11. https://doi.org/10.1007/978-3-319-50115-4_1
- [24] David B. D'Ambrosio and Kenneth O. Stanley. 2013. Scalable multiagent learning through indirect encoding of policy geometry. *Evolutionary Intelligence* 6, 1 (jan 2013), 1–26. <https://doi.org/10.1007/s12065-012-0086-3>
- [25] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. 1–16.
- [26] Wei Du and Shifei Ding. 2020. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. *Artificial Intelligence Review* 54, 5 (Nov. 2020), 3215–3238. <https://doi.org/10.1007/s10462-020-09938-y>
- [27] Wanyu Du and Yangfeng Ji. 2019. An Empirical Comparison on Imitation Learning and Reinforcement Learning for Paraphrase Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 6011–6017. <https://doi.org/10.18653/v1/D19-1619>
- [28] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. One-Shot Imitation Learning. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/ba3866600c3540f67c1e9575e213be0a-Paper.pdf
- [29] Ashley D Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L Isbell. 2019. Imitating Latent Policies from Observation. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*. Proceedings of the 36th International Conference on Machine Learning, 1755–1763.
- [30] Tom Erez, Yuval Tassa, and Emanuel Todorov. 2011. Infinite-Horizon Model Predictive Control for Periodic Tasks with Contacts. In *Robotics: Science and Systems VII, University of Southern California, Los Angeles, CA, USA, June 27-30, 2011*, Hugh F. Durrant-Whyte, Nicholas Roy, and Pieter Abbeel (Eds.). <https://doi.org/10.15607/RSS.2011.VII.010>
- [31] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2015. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision* 111, 1 (Jan. 2015), 98–136.
- [32] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=rc8o_j818PX
- [33] Patrick Fernandes, Aman Madaan, Emmy Liu, António Farinhás, Pedro Henrique Martins, Amanda Bertsch, José G. C. de Souza, Shuyan Zhou, Tongshuang Wu, Graham Neubig, and André F. T. Martins. [n. d.]. Bridging the Gap: A Survey on Integrating (Human) Feedback for Natural Language Generation. ([n. d.]). arXiv:2305.00955v2 [cs.CL]
- [34] Henrique Ferrolho, Vladimir Ivan, Wolfgang Merkt, Ioannis Havoutis, and Sethu Vijayakumar. 2021. Inverse dynamics vs. forward dynamics in direct transcription formulations for trajectory optimization. In *2021 IEEE International Conference on Robotics and*

- Automation (ICRA)*. IEEE, 12752–12758.
- [35] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 49–58. <https://proceedings.mlr.press/v48/finn16.html>
 - [36] Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. 2021. panda-gym: Open-source goal-conditioned environments for robotic learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS* (2021). arXiv:2106.13687v2 [cs.LG]
 - [37] Tanmay Gangwani, Yuan Zhou, and Jian Peng. 2021. Imitation Learning from Observations under Transition Model Disparity. In *Deep RL Workshop NeurIPS 2021*. <https://openreview.net/forum?id=7sVWxL1s3Ii>
 - [38] Nathan Gavenski, Juarez Monteiro, Roger Granada, Felipe Meneguzzi, and Rodrigo C. Barros. 2020. Imitating Unknown Policies via Exploration. In *International British Machine Vision Virtual Conference*. 1–8. https://www.bmvc2020-conference.com/conference/papers/paper_0774.html
 - [39] Nathan Gavenski, Juarez Monteiro, Adilson Medronha, and Rodrigo C. Barros. 2022. How Resilient Are Imitation Learning Methods to Sub-optimal Experts?. In *Intelligent Systems*, João Carlos Xavier-Junior and Ricardo Araújo Rios (Eds.). Springer International Publishing, Cham, 449–463.
 - [40] Nathan Schneider Gavenski. 2021. *Self-supervised imitation learning from observation*. Master's thesis. Pontifícia Universidade Católica do Rio Grande do Sul.
 - [41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf>
 - [42] Xiaoxiao Guo, Shiyu Chang, Mo Yu, Gerald Tesauro, and Murray Campbell. 2019. Hybrid Reinforcement Learning with Expert State Sequences. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (jul 2019), 3739–3746. <https://doi.org/10.1609/aaai.v33i01.33013739>
 - [43] Joerg Henrichs. 2006. SuperTuxKart. <https://supertuxkart.net/>.
 - [44] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in neural information processing systems*. Advances in neural information processing systems, 4565–4573.
 - [45] Lars Holmberg, Carl Johan Helgstrand, and Niklas Hultin. 2022. More Sanity Checks for Saliency Maps. In *Foundations of Intelligent Systems - 26th International Symposium, ISMIS 2022, Cosenza, Italy, October 3–5, 2022, Proceedings (Lecture Notes in Computer Science, Vol. 13515)*, Michelangelo Ceci, Sergio Flesca, Elio Masciari, Giuseppe Manco, and Zbigniew W. Ras (Eds.). Springer, 175–184. https://doi.org/10.1007/978-3-031-16564-1_17
 - [46] Weiming Hu, Xu Li, Jinchao Hu, Dong Kong, Yue Hu, Qimin Xu, Yan Liu, Xiang Song, and Xuan Dong. 2023. A Safe Driving Decision-making Methodology based on Cascade Imitation Learning Network for Automated Commercial Vehicles. *IEEE Sensors Journal* (2023), 1–1. <https://doi.org/10.1109/jsen.2023.3256704>
 - [47] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)* 50, 2 (Jun 2017), 1–35.
 - [48] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The Malmo Platform for Artificial Intelligence Experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 4246–4247. <http://www.ijcai.org/Abstract/16/643>
 - [49] Christian Kauten. 2018. Super Mario Bros for OpenAI Gym. GitHub. <https://github.com/Kautenja/gym-super-mario-bros>
 - [50] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. 2016. ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. In *IEEE Conference on Computational Intelligence and Games*. IEEE, Santorini, Greece, 341–348. <https://doi.org/10.1109/CIG.2016.7860433> The Best Paper Award.
 - [51] Rahul Kidambi, Jonathan Chang, and Wen Sun. 2021. MobileLE: Model-Based Imitation Learning From Observation Alone. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 28598–28611. <https://proceedings.neurips.cc/paper/2021/file/f06048518ff8de2035363e00710c6a1d-Paper.pdf>
 - [52] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2019. The (Un)reliability of Saliency Methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller (Eds.). Lecture Notes in Computer Science, Vol. 11700. Springer, 267–280. https://doi.org/10.1007/978-3-030-28954-6_14
 - [53] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. 1997. RoboCup: A Challenge Problem for AI. *AI Mag.* 18, 1 (1997), 73–85. <https://doi.org/10.1609/aimag.v18i1.1276>
 - [54] Jonathan Wilder Lavington, Sharun Vaswani, and Mark Schmidt. 2022. Improved Policy Optimization for Online Imitation Learning. In *Proceedings of The 1st Conference on Lifelong Learning Agents (Proceedings of Machine Learning Research, Vol. 199)*, Sarath Chandar, Razvan Pascanu, and Doina Precup (Eds.). PMLR, 1146–1173. <https://proceedings.mlr.press/v199/lavington22a.html>

- [55] Geoffrey Lee, Min Luo, Fabio Zambetta, and Xiaodong Li. 2014. Learning a Super Mario controller from examples of human play. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. <https://doi.org/10.1109/cec.2014.6900246>
- [56] Ruiping Li and Xiang Cheng. 2019. DIVINE: A Generative Adversarial Imitation Learning Framework for Knowledge Graph Reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 2642–2651. <https://doi.org/10.18653/v1/D19-1266>
- [57] Yunzhu Li, Jiaming Song, and Stefano Ermon. 2017. InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.
- [58] Zhen Liang, Taoran Wu, Changyuan Zhao, Wanwei Liu, Bai Xue, Wenjing Yang, and Ji Wang. 2023. Repairing Deep Neural Networks Based on Behavior Imitation. *CoRR* abs/2305.03365 (2023). <https://doi.org/10.48550/arXiv.2305.03365> arXiv:2305.03365
- [59] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. 2018. Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. <https://doi.org/10.1109/icra.2018.8462901>
- [60] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic Traffic Simulation using SUMO, In The 21st IEEE International Conference on Intelligent Transportation Systems. *IEEE Intelligent Transportation Systems Conference (ITSC)*.
- [61] Xingzhou Lou, Jiaxian Guo, Junge Zhang, Jun Wang, Kaiqi Huang, and Yali Du. 2023. PECAN: Leveraging Policy Ensemble for Context-Aware Zero-Shot Human-AI Coordination. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2023-01-16)*. arXiv:2301.06387v4 [cs.AI]
- [62] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)* (2017).
- [63] Matteo Lucchi, Friedemann Zindler, Stephan Mühlbacher-Karrer, and Horst Pichler. 2007. robo-gym – An Open Source Toolkit for Distributed Deep Reinforcement Learning on Real and Simulated Robots. (2007). arXiv:2007.02753v2 [cs.RO]
- [64] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. 2020. Learning Latent Plans from Play. In *Proceedings of the Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 100)*, Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (Eds.). PMLR, 1113–1132. <https://proceedings.mlr.press/v100/lynch20a.html>
- [65] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. 2018. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research* 61 (2018), 523–562.
- [66] Zhao Mandi, Homanga Bharadhwaj, Vincent Moens, Shuran Song, Aravind Rajeswaran, and Vikash Kumar. 2022. CACTI: A Framework for Scalable Multi-Task Multi-Scene Visual Imitation Learning. *CoRR* abs/2212.05711 (2022). <https://doi.org/10.48550/arXiv.2212.05711>
- [67] Raphael Memmesheimer. 2023. *On the recognition of human activities and the evaluation of its imitation by robotic systems*. doctoralthesis. Universität Koblenz, Universitätsbibliothek.
- [68] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* 73 (feb 2018), 1–15. <https://doi.org/10.1016/j.dsp.2017.10.011>
- [69] Juarez Monteiro, Nathan Gavenski, Roger Granada, Felipe Meneguzzi, and Rodrigo C. Barros. 2020. Augmented Behavioral Cloning from Observation. In *International Joint Conference on Neural Networks* (Glasgow, Scotland). 1–8. <https://arxiv.org/abs/2004.13529>
- [70] Juarez Monteiro, Nathan Gavenski, Felipe Meneguzzi, and Rodrigo C. Barros. 2023. Self-Supervised Adversarial Imitation Learning. *2020 International Joint Conference on Neural Networks (IJCNN)* (2023). arXiv:2304.10914v1 [cs.LG]
- [71] Andrew William Moore. 1990. *Efficient memory-based learning for robot control*. Ph.D. Dissertation. University of Cambridge.
- [72] Igor Mordatch and Pieter Abbeel. 2017. Emergence of Grounded Compositional Language in Multi-Agent Populations. *arXiv preprint arXiv:1703.04908* (2017).
- [73] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. 2017. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. <https://doi.org/10.1109/icra.2017.7989247>
- [74] Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N. Yannakakis. 2013. Imitating human playing styles in Super Mario Bros. *Entertainment Computing* 4, 2 (apr 2013), 93–104. <https://doi.org/10.1016/j.entcom.2012.10.001>
- [75] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (Oct. 2010), 1345–1359. <https://doi.org/10.1109/tkde.2009.191>
- [76] Antonio Paolillo, Paolo Robuffo Giordano, and Matteo Saveriano. 2023. Dynamical System-based Imitation Learning for Visual Servoing using the Large Projection Formulation. In *ICRA’23-IEEE Int. Conf. on Robotics and Automation*.
- [77] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. 2018. Zero-Shot Visual Imitation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

- [78] Brahma S. Pavse, Faraz Torabi, Josiah Hanna, Garrett Warnell, and Peter Stone. 2020. RIDM: Reinforced Inverse Dynamics Modeling for Learning from a Single Observed Demonstration. *IEEE Robotics and Automation Letters* 5, 4 (oct 2020), 6262–6269. <https://doi.org/10.1109/lra.2020.3010750>
- [79] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. 2019. Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=HyxPx3R9tm>
- [80] Dean A. Pomerleau. 1988. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *International Conference on Neural Information Processing Systems*. 305–313.
- [81] Henry Pulver, Francisco Eiras, Ludovico Carozza, Majd Hawasly, Stefano V. Albrecht, and Subramanian Ramamoorthy. 2021. PILOT: Efficient Planning by Imitation Learning and Optimisation for Safe Autonomous Driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*. IEEE, 1442–1449. <https://doi.org/10.1109/IROS51168.2021.9636862>
- [82] Rodrigo Pérez-Dattari and Jens Kober. 2023. Stable Motion Primitives via Imitation and Contrastive Learning. *arXiv* (2023). arXiv:2302.10017v1 [cs.RO]
- [83] Antonin Raffin. 2018. RL Baselines Zoo. <https://github.com/araffin/rl-baselines-zoo>.
- [84] Dripta S. Raychaudhuri, Sujoy Paul, Jeroen Vanbaar, and Amit K. Roy-Chowdhury. 2021. Cross-domain Imitation from Observations. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8902–8912. <https://proceedings.mlr.press/v139/raychaudhuri21a.html>
- [85] Saleha Raza, Sajjad Haider, and Mary-Anne Williams. 2012. Teaching coordinated strategies to soccer robots via imitation. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. <https://doi.org/10.1109/robio.2012.6491170>
- [86] Saleha Raza, Sajjad Haider, and Mary-Anne Williams. 2012. Teaching Coordinated Strategies to Soccer Robots via Imitation. In *International Conference on Robotics and Biomimetics (ROBIO 2012)*. 1434–1439.
- [87] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 627–635.
- [88] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Ziheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. <https://doi.org/10.1007/s11263-015-0816-y>
- [89] Stuart Russell and Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach* (4th Edition). Pearson. <http://aima.cs.berkeley.edu/>
- [90] S.J. Russell, S. Russell, and P. Norvig. 2020. *Artificial Intelligence: A Modern Approach*. Pearson.
- [91] Stefan Schaal, Auke Ijspeert, and Aude Billard. 2003. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 358, 1431 (Feb 2003), 537–547.
- [92] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).
- [93] Jinghuan Shang and Michael S. Ryoo. 2021. Self-Supervised Disentangled Representation Learning for Third-Person Imitation Learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. <https://doi.org/10.1109/iros51168.2021.9636363>
- [94] Andy Shih, Stefano Ermon, and Dorsa Sadigh. 2022. Conditional Imitation Learning for Multi-Agent Games. *arXiv preprint arXiv:2201.01448* (2022).
- [95] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. 2018. Multi-Agent Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/240c945bb72980130446fc2b40fb8e0-Paper.pdf
- [96] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. 2017. Third Person Imitation Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=B16dGcqLx>
- [97] Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. 2019. Provably Efficient Imitation Learning from Observation Alone. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6036–6045. <https://proceedings.mlr.press/v97/sun19b.html>
- [98] Richard S. Sutton. 1995. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. In *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, David S. Touretzky, Michael Mozer, and Michael E. Hasselmo (Eds.). MIT Press, 1038–1044.
- [99] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690* 1 (2018), 1–24.
- [100] Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, 4906–4913. <https://doi.org/10.1109/IROS.2012.6386025>
- [101] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. <https://doi.org/10.1109/iros.2012.6386109>

- [102] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral Cloning from Observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (Stockholm, Sweden) (*IJCAI’18*). AAAI Press, 4950–4957. <https://www.ijcai.org/proceedings/2018/0687.pdf>
- [103] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Generative Adversarial Imitation from Observation. *arXiv preprint arXiv:1807.06158* (2018). arXiv:1807.06158v4 [cs.LG]
- [104] Faraz Torabi, Garrett Warnell, and Peter Stone. 2019. Imitation Learning from Video by Leveraging Proprioception. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) (*IJCAI’19*). AAAI Press, 3585–3591.
- [105] Faraz Torabi, Garrett Warnell, and Peter Stone. 2019. Recent Advances in Imitation Learning from Observation. *International Joint Conference on Artificial Intelligence* (2019). arXiv:1905.13566v2 [cs.RO]
- [106] Faraz Torabi, Garrett Warnell, and Peter Stone. 2021. DEALIO: Data-Efficient Adversarial Learning for Imitation from Observation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. <https://doi.org/10.1109/iros51168.2021.936169>
- [107] Mark Towers, Jordan K Terry, Ariel Kwiatkowski, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. 2023. Gymnasium. <https://doi.org/10.5281/ZENODO.8127026>
- [108] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. 2020. dm_control: Software and tasks for continuous control. *Software Impacts* 6 (2020), 100022. <https://doi.org/10.1016/j.simpa.2020.100022>
- [109] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. <http://jmlr.org/papers/v9/vandermaaten08a.html>
- [110] Mudit Verma and Subbarao Kambhampati. 2023. Data Driven Reward Initialization for Preference based Reinforcement Learning. *arXiv* (2023). arXiv:2302.08733v1 [cs.LG]
- [111] Andreas Vlachos, Gerasimos Lampouras, and Sebastian Riedel. 2017. Imitation learning for structured prediction in natural language processing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*. Association for Computational Linguistics, Valencia, Spain. <https://aclanthology.org/E17-5003>
- [112] Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. 2017. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems* 30 (2017).
- [113] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. 2016. How to Use t-SNE Effectively. *Distill* 1, 10 (oct 2016).
- [114] Paweł Wawrzynski. 2009. A Cat-Like Robot Real-Time Learning to Run. In *Adaptive and Natural Computing Algorithms, 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 5495)*, Mikko Kolehmainen, Pekka J. Toivanen, and Bartłomiej Beliczynski (Eds.). Springer, 380–390. https://doi.org/10.1007/978-3-642-04921-7_39
- [115] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (London, England, United Kingdom) (EASE ’14). Association for Computing Machinery, New York, NY, USA, Article 38, 10 pages. <https://doi.org/10.1145/2601248.2601268>
- [116] Bernhard Wyrmann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. 2014. TORCS, The Open Racing Car Simulator. <http://www.torcs.org>.
- [117] Shuo Yang, Wei Zhang, Ran Song, Jiyu Cheng, Hesheng Wang, and Yibin Li. 2023. Watch and Act: Learning Robotic Manipulation From Visual Demonstration. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2023), 1–13.
- [118] Zhao-Heng Yin, Weirui Ye, Qifeng Chen, and Yang Gao. 2022. Planning for Sample Efficient Imitation Learning. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). <https://openreview.net/forum?id=iKKfdIm81Jt>
- [119] Xingrui Yu, Yueming Lyu, and Ivor Tsang. 2020. Intrinsic Reward Driven Imitation Learning via Generative Model. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 10925–10935. <https://proceedings.mlr.press/v119/yu20d.html>
- [120] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. 2018. Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. <https://doi.org/10.1109/icra.2018.8461249>
- [121] Boyuan Zheng, Sunny Verma, Jianlong Zhou, Ivor Tsang, and Fang Chen. 2021. Imitation learning: Progress, taxonomies and opportunities. *arXiv preprint arXiv:2106.12177* (2021).
- [122] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. 2020. Off-Policy Imitation Learning from Observations. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 12402–12413. <https://proceedings.neurips.cc/paper/2020/file/92977ae4d2ba1425a59afb269c2a14e-Paper.pdf>
- [123] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* 109, 1 (Jan. 2021), 43–76. <https://doi.org/10.1109/jproc.2020.3004555>
- [124] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning.. In *Aaaai*, Vol. 8. Chicago, IL, USA, 1433–1438.

A Validation environments

Table 2. Validation environments information. Environments from benchmarks are grouped by their respective suite.

Environments	Domain Type	Action	State	Deterministic	Accessibility	Dynamics
Bandits [94]	Task specific	discrete	discrete	deterministic	partial observable	static
Hanabi [8]	Task specific	discrete	discrete	deterministic	partial observable	dynamic
RoboCup [53]	Task specific	continuous	continuous	deterministic	fully observable	dynamic
Sort [19]	Task specific	discrete	continuous (Image)	deterministic	fully observable	dynamic
Predator-Prey, Keep-Away[62], Particle[72]	Task specific	continuous	discrete	deterministic	fully observable	dynamic
TurtleBot [77], 2D Plane [57]	Domain specific	continuous	continuous	deterministic	fully observable	static
Classic control environments						
Acrobot [98], Pendulum	Task specific	continuous	continuous	deterministic	fully observable	static
CartPole [9], MountainCar [71]	Task specific	discrete	continuous	deterministic	fully observable	static
MuJoCo environments						
Ant [92], HalfCheetah [114], Hopper [30], Humanoid [100], InvertedPendulum [9], InvertedDoublePendulum [9], Pusher, Reacher, Swimmer [22], Walker2D [30]	Task specific	continuous	continuous	deterministic	fully observable	static
DeepMind Control Suite environments [99]						
Ball-in-cup, Cheetah Run, Finger Spin, Hopper Hop, Humanoid Walk, Swingup, Reacher Easy, Walker Walk	Task specific	continuous	continuous	deterministic	fully observable	static
Atari environments [12]						
Alien, Beam Rider, Breakout, Taxi, Kung-Fu Master, MsPacMan, PitFall, Pong, Private Eye, Q*bert, Seaquest, Space Invaders, Montezuma	Task specific	discrete	both (Image or RAM)	deterministic	fully observable	static

B Precision environments

Table 3. Precision environments information. Environments from benchmarks are grouped by their respective suite.

Environments	Domain Type	Action	State	Deterministic	Accessibility	Dynamics
Jaco [4]	Domain specific	continuous	continuous (Image and Vector)	deterministic	fully observable	static
MAV [23]	Domain specific	continuous	continuous	deterministic	fully observable	static
Panda [36]	Both	continuous	continuous (Image and Vector)	deterministic	fully observable	static
MuJoCo environments [101]						
Disc, DoorOpening, PegInsertion, GripperPusher, Point Robot body, Robotic Arm	Domain specific	continuous	continuous	deterministic	fully observable	static
Both	continuous	continuous	deterministic	fully observable	static	

C Sequential environments

Table 4. Sequential environments information. RAM means that states are vector representations of values stored in memory.

Environments	Domain Type	Action	State	Deterministic	Accessibility	Dynamics
Maze [3]	Task specific	discrete	Discrete (Image or Vector)	deterministic	fully observable	static
CoinRun [21]	Task specific	discrete	continuous (Image)	both	both	dynamic
TORCS [116]	Domain specific	both	continuous (Image and RAM)	deterministic	fully observable	dynamic
VizDoom [50]	Domain specific	discrete	continuous (Image)	deterministic	fully observable	dynamic
Blocks World [28]	Domain specific	continuous	continuous (Image)	deterministic	fully observable	static
LunarLander	Task specific	discrete	continuous	deterministic	fully observable	static
CARLA [25], SUMO [60]	Domain specific	continuous	continuous (Image or Vector)	deterministic	fully observable	dynamic
Mario [49], Minecraft [48], Overcooked [16], SuperTuxKart [43]	Task specific	discrete	continuous (Image or RAM)	deterministic	fully observable	dynamic