A

Imitation Learning: A Survey of Learning Methods

Ahmed Hussein, School of Computing Science and Digital Media, Robert Gordon University
Mohamed Medhat Gaber, School of Computing and Digital Technology, Birmingham City University
Eyad Elyan, School of Computing Science and Digital Media, Robert Gordon University
Chrisina Jayne, School of Computing Science and Digital Media, Robert Gordon University

Imitation learning techniques aim to mimic human behavior in a given task. An agent (a learning machine) is trained to perform a task from demonstrations by learning a mapping between observations and actions. The idea of teaching by imitation has been around for many years, however, the field is gaining attention recently due to advances in computing and sensing as well as rising demand for intelligent applications. The paradigm of learning by imitation is gaining popularity because it facilitates teaching complex tasks with minimal expert knowledge of the tasks. Generic imitation learning methods could potentially reduce the problem of teaching a task to that of providing demonstrations; without the need for explicit programming or designing reward functions specific to the task. Modern sensors are able to collect and transmit high volumes of data rapidly, and processors with high computational power allow fast processing that maps the sensory data to actions in a timely manner. This opens the door for many potential AI applications that require real-time perception and reaction such as humanoid robots, self-driving vehicles, human computer interaction and computer games to name a few. However, specialized algorithms are needed to effectively and robustly learn models as learning by imitation poses its own set of challenges. In this paper, we survey imitation learning methods and present design options in different steps of the learning process. We introduce a background and motivation for the field as well as highlight challenges specific to the imitation problem. Methods for designing and evaluating imitation learning tasks are categorized and reviewed. Special attention is given to learning methods in robotics and games as these domains are the most popular in the literature and provide a wide array of problems and methodologies. We extensively discuss combining imitation learning approaches using different sources and methods, as well as incorporating other motion learning methods to enhance imitation. We also discuss the potential impact on industry, present major applications and highlight current and future research directions.

CCS Concepts: •**General and reference** → **Surveys and overviews;** •**Computing methodologies** → **Learning paradigms; Learning settings; Machine learning approaches;** Cognitive robotics; Control methods; Distributed artificial intelligence; Computer vision;

General Terms: Design, Algorithms

Additional Key Words and Phrases: Imitation learning, learning from demonstrations, intelligent agents, learning from experience, self-improvement, feature representations, robotics, deep learning, reinforcement learning

ACM Reference Format:
Ahmed Hussein, Mohamed M. Gaber, Eyad Elyan, and Chrisina Jayne, 2016. Imitation Learning: A Survey of Learning Methods. *ACM Comput. Surv.* V, N, Article A (January YYYY), 35 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

Author's addresses: A. Hussein, E. Elyan, and Chrisina Jayne School of Computing Science and Digital Media, Robert Gordon University, Riverside East, Garthdee Road, Aberdeen AB10 7GJ, United Kingdom
M. M. Gaber, School of Computing and Digital Technology, Birmingham City University, 15 Bartholomew Row, Birmingham B5 5JU, United Kingdom

DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In recent years, the demand for intelligent agents capable of mimicking human behavior has grown substantially. Advancement in robotics and communication technology have given rise to many potential applications that need artificial intelligence that can not only make intelligent decisions, but is able to perform motor actions realistically in a variety of situations. Many future directions in technology rely on the ability of artificial intelligence agents to behave as a human would when presented with the same situation. Examples of such fields are self-driving vehicles, assistive robots and human computer interaction. For the latter especially, opportunities for new applications are growing due to recent interest in consumer virtual reality and motion capture systems[1]. In these applications and many robotics tasks, we are faced with the problem of executing an action given the current state of the agent and its surroundings. The number of possible scenarios in a complex application is too large to cover by explicit programming and so a successful agent must be able to handle unseen scenarios. While such a task may be formulated as an optimization problem, it has become widely accepted that having prior knowledge provided by an expert is more effective and efficient than searching for a solution from scratch [Schaal 1999] [Schaal et al. 1997] [Bakker and Kuniyoshi 1996] [Billard et al. 2008]. In addition, optimization through trial and error requires reward functions that are designed specifically for each task. One can imagine that even for simple tasks, the number of possible sequences of actions an agent can take grows exponentially. Defining rewards for such problems is difficult, and in many cases unknown.

One of the more natural and intuitive ways of imparting knowledge by an expert is to provide demonstrations for the desired behavior that the learner is required to emulate. It is much easier for the human teacher to transfer their knowledge through demonstration than to articulate it in a way that the learner will understand [Raza et al. 2012]. This paper reviews the methods used to teach artificial agents to perform complex sequences of actions through imitation.

Imitation learning is an interdisciplinary field of research. Existing surveys focus on different challenges and perspectives of tackling this problem. Early surveys review the history of imitation learning and early attempts to learn from demonstration [Schaal 1999] [Schaal et al. 2003]. In [Billard et al. 2008] learning approaches are categorized as engineering oriented and biologically oriented methods, [Ijspeert et al. 2013] focus on learning methods from the viewpoint of dynamical systems, while [Argall et al. 2009] address different challenges in the process of imitation such as acquiring demonstrations, physical and sensory issues as well as learning techniques. However, due to recent advancements in the field and a surge in potential applications, it is important at this time to conduct a survey that focuses on the computational methods used to learn from demonstrated behavior. More specifically, we review artificial intelligence methods which are used to learn policies that solve problems according to human demonstrations. By focusing on learning methods, this survey addresses learning for any intelligent agent, whether it manifests itself as a physical robot or a software agent (such as games, simulations, planning, etc.). The reviewed literature addresses various applications, however, many of the methods used are generic and can be applied to general motion learning tasks. The learning process is categorized into: creating feature representations, direct imitation and indirect learning. The methods and sources of learning for each process are reviewed as well as evaluation metrics and applications suitable for these methods.

[1]In the last two years the virtual reality market has attracted major technology companies and billions of dollars in investment and is still rapidly growing. http://www.fastcompany.com/3052209/tech-forecast/vr-and-augmented-reality-will-soon-be-worth-150-billion-here-are-the-major-pla?partner=engadget

Imitation learning refers to an agent's acquisition of skills or behaviors by observing a teacher demonstrating a given task. With inspiration and basis stemmed in neuroscience, imitation learning is an important part of machine intelligence and human computer interaction, and has from an early point been viewed as an integral part in the future of robotics [Schaal 1999]. Another popular paradigm is learning through trial and error; however, providing good examples to learn from expedites the process of finding a suitable action model and prevents the agent from falling into local minima. Moreover, a learner could very well arrive on its own at a suitable solution, i.e one that achieves a certain quantifiable goal, but which differs significantly from the way a human would approach the task. It is sometimes important for the learner's actions to be believable and appear natural. This is necessary in many robotic domains as well as human computer interaction where the performance of the learner is only as good as a human observer's perception of it. It is therefore favorable to teach a learner the desired behavior from a set of collected instances. However, it is often the case that direct imitation of the expert's motion doesn't suffice due to variations in the task such as the positions of objects or inadequate demonstrations. Therefore imitation learning techniques need to be able to learn a policy from the given demonstrations that can generalize to unseen scenarios. As such the agent learns to perform the task rather than deterministically copying the teacher.

The field of imitation learning draws its importance from its relevance to a variety of applications such as human computer interaction and assistive robots. It is being used to teach robots of varying skeletons and degrees of freedom (DOF) to perform an array of different tasks. Some examples are navigational problems, which typically employ vehicle-like robots, with relatively lower degrees of freedom. These include flying vehicles [Sammut et al. 2014] [Abbeel et al. 2007] [Ng et al. 2006], or ground vehicles [Silver et al. 2008] [Ratliff et al. 2007a] [Chernova and Veloso 2007a] [Ollis et al. 2007]. Other applications focus on robots with higher degrees of freedom such as humanoid robots [Mataric 2000a] [Asfour et al. 2008][Calinon and Billard 2007a] or robotic arms [Kober and Peters 2010][Kober and Peters 2009b][Mülling et al. 2013]. High DOF humanoid robots can learn discrete actions such as standing up, and cyclic tasks such as walking [Berger et al. 2008]. Although the majority of applications target robotics, imitation learning applies to simulations [Berger et al. 2008] [Argall et al. 2007] and is even used in computer games [Thurau et al. 2004a] [Gorman 2009] [Ross and Bagnell 2010].

Imitation learning works by extracting information about the behavior of the teacher and the surrounding environment including any manipulated objects, and learning a mapping between the situation and demonstrated behavior. Traditional machine learning algorithms do not scale to high dimensional agents with high degrees of freedom [Kober and Peters 2010]. Specialized algorithms are therefore needed to create adequate representations and predictions to be able to emulate motor functions in humans.

Similar to traditional supervised learning where examples represent pairs of features and labels, in imitation learning the examples demonstrate pairs of states and actions. Where the state represents the current pose of the agent, including the position and velocities of relevant joints and the status of a target object if one exists (such as position, velocity, geometric information, etc.). Therefore, Markov decision processes (MDPs) lend themselves naturally to imitation learning problems and are commonly used to represent expert demonstrations. The Markov property dictates that the next state is only dependent on the previous state and action, which alleviates the need to include earlier states in the state representation [Kober et al. 2013]. A typical imitation learning work flow starts by acquiring sample demonstrations from an expert which are then encoded as state-action pairs. These examples are then used to train

a policy. However, learning a direct mapping between state and action is often not enough to achieve the required behavior. This can happen due to a number of issues such as errors in acquiring the demonstrations, variance in the skeletons of the teacher and learner (correspondence problem) or insufficient demonstrations. Moreover, the task performed by the learner may slightly vary from the demonstrated task due to changes in the environment, obstacles or targets. Therefore, imitation learning frequently involves another step that requires the learner to perform the learned action and re-optimize the learned policy according to its performance of the task. This self-improvement can be achieved with respect to a quantifiable reward or learned from examples. Many of these approaches fall under the wide umbrella of reinforcement learning.

Figure 1 shows a workflow of an imitation learning process. The process starts by capturing actions to learn from, this can be achieved via different sensing methods. The data from the sensors is then processed to extract features that describe the state and surroundings of the performer. The features are used to learn a policy to mimic the demonstrated behavior. Finally the policy can be enhanced by allowing the agent to act out the policy and refine it based on its performance. This step may or may not require the input of a teacher. It might be intuitive to think of policy-refinement as a post learning step, but in many cases it occurs in conjunction with learning from demonstrations.

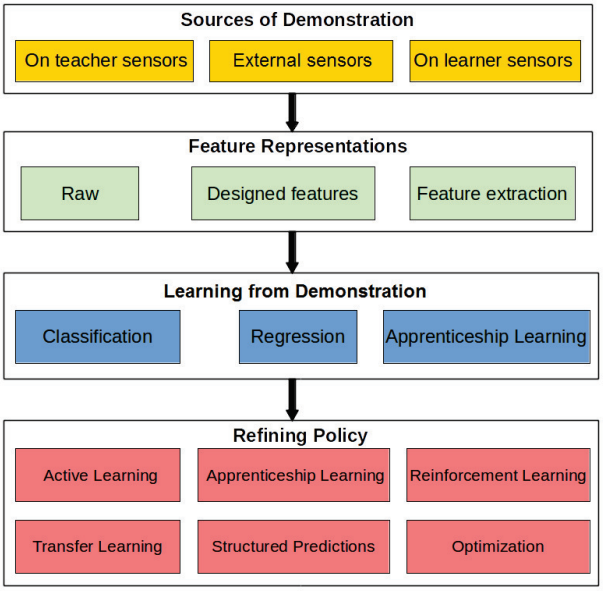

Fig. 1. Imitation learning flowchart

Imitation learning applications face a number of diverse challenges due to their interdisciplinary nature:

— Starting with the process of acquiring demonstrations, whether capturing data from sensors on the learner or the teacher, or using visual information, the captured signals are prone to noise and sensor errors. Similar problems arise during execution,

when the agent is sensing the environment. Noisy or unreliable sensing will result in erroneous behavior even if the model is adequately trained. [Argall et al. 2009] survey the different methods of gathering demonstrations and the challenges in each approach.

— Another issue concerning demonstration is the correspondence problem [Dautenhahn and Nehaniv 2002]. Correspondence is the matching of the learner's capabilities, skeleton and degrees of freedom to that of the teacher. Any discrepancies in the size or structure between the teacher and learner need to be compensated for during training. Often in this case a learner can learn the shape of the movement from demonstrations, then refine the model through trial and error to achieve its goal.
— A related challenge is the problem of observability where the kinematics of the teacher are not known to learner [Schaal et al. 2003]. If the demonstrations are not provided by a designated teacher, the learner may not be aware of the capabilities and possible actions of the teacher; it can only observe the effects of the teacher's actions and attempt to replicate them using its own kinematics.
— The learning process also faces practical problems as traditional machine learning techniques do not scale well to high degrees of freedom [Kober and Peters 2010]. Due to the real-time nature of many imitation learning applications, the learning algorithms are restricted by computing power and memory limitations; especially in robotic applications that require on-board computers to perform the real-time processing.
— Moreover, complex behaviors can often be viewed as a trajectory of dependent micro actions which violates the independent and identically distributed (i.i.d.) assumption adopted in most machine learning practices. The learned policy must be able to adapt its behavior based on previous actions and make corrections if necessary.
— The policy must also be able to adapt to variations in the task and the surrounding environment. The complex nature of imitation learning applications dictate that agents must be able to reliably perform the task even under circumstances that vary from the training demonstration.
— Tasks that entail human-robot interaction pose a new set of challenges. Naturally safety is a chief concern in such applications [De Santis et al. 2008] [Ikemoto et al. 2012], and measures need to be taken to prevent injury of the human partners and insure their safety. Moreover, other challenges concern the mechanics of the robot, such as its ability to react to the humans' force and adapt to their actions.

In the next section we formally present the problem of imitation learning and discuss different ways to formulate the problem. Section 3 describes the different methods for creating feature representations. Section 4 reviews direct imitation methods for learning from demonstrations. Section 5 surveys indirect learning techniques and presents the different approaches to improve learned policies through optimizing reward functions and teachers' behaviors. The paradigms for improving direct imitation through indirect learning are also discussed. Section 6 reviews the use of imitation learning in multi-agent scenarios. Section 7 describes different evaluation approaches and Section 8 shows the potential applications to utilize imitation learning. Finally, we present a conclusion and discuss future directions in Section 9.

2. PROBLEM FORMULATION

In this section we formalize the problem of imitation learning and introduce some preliminaries and definitions.

DEFINITION 1. *The process of imitation learning is one by which an agent uses instances of performed actions to learn a policy that solves a given task.*

DEFINITION 2. *An agent is defined as an entity that autonomously interacts within an environment towards achieving or optimizing a goal [Russell and Norvig 2003]. An agent can be thought of as a software robot; it receives information from the environment by sensing or communication and acts upon the environment using a set of actuators.*

DEFINITION 3. *A policy is a function that maps a state (a description of the agent, such as pose, positions and velocities of various parts of the skeleton, and its relevant surrounding) to an action. It is what the agent uses to decide which action to execute when presented with a situation.*

Policies can be learned from demonstration or experience. The demonstrations may come from a designated teacher or another agent; the experience may be the agent's own or another's. The difference between the two types of training instances is that demonstrations provide the optimal action to a given state, and so the agent learns to reproduce this behavior in similar situations. This makes demonstrations suited for direct policy learning such as supervised learning methods. While experience shows the performed action, which may not be optimal, but also provides the reward (or cost) of performing that action given the current state, and so the agent learns to act in a manner that maximizes its overall reward. Therefore reinforcement learning is mainly used to learn from experience. More formally, demonstrations and experiences can be defined as follows.

DEFINITION 4. *A demonstration is presented as a pair of input and output* (x, y). *Where* x *is a vector of features describing the state at that instant and* y *is the action performed by the demonstrator.*

DEFINITION 5. *An experience is presented as a tuple* (s, a, r, s') *where* s *is the state,* a *is the action taken at state* s, r *is the reward received for performing action* a *and* s' *is the new state resulting from that action.*

It is clear from this formulation that learning from demonstration doesn't require the learner to know the cost function optimized by the teacher. It can simply optimize the error of deviating from the demonstrated output such as the least square error in supervised learning. More formally, from a set of demonstrations $D = (x_i, y_i)$ an agent learns a policy π such that:

$$u(t) = \pi(x(t), t, \alpha) \tag{1}$$

Where u is the predicted action, x is the feature vector, t is the time and α is the set of policy parameters that are changed through learning. While the time parameter t is used to specify an instance of input and output, it is also input to the policy π as a separate parameter.

DEFINITION 6. *A policy that uses* t *in learning the parameters of the policy is called a non-stationary policy (also known as non-autonomous [Schaal et al. 2003]) i.e the policy takes into consideration at what stage of the task the agent is currently acting.*

DEFINITION 7. *A stationary policy (autonomous) neglects the time parameter and learns one policy for all steps in an action sequence.*

One advantage of stationary policies is the ability to learn tasks where the horizon (the time limit for actions) is large or unknown [Ross and Bagnell 2010]. While non stationary policies are more naturally situated to learn motor trajectories i.e actions that occur over a period of time and are comprised of multiple motor primitive executed sequentially. However, these policies are difficult to adapt to unseen scenarios

and changes in the parameters of the task [Schaal et al. 2003]. Moreover, this failure to adapt to new scenarios, at one point in the trajectory, can result in compounded errors as the agent continues to perform the remainder of action. In light of these drawbacks, methods for learning trajectories using stationary policies are motivated. An example is the use of structured predictions [Ross et al. 2010] where the training demonstrations are aggregated with labeled instances at different time steps in the trajectory – so time is encoded in the state. Alternatively, [Ijspeert et al. 2002a] learns attractor landscapes from the demonstrations, creating a stationary policy that is attracted to the demonstrated trajectory. This avoids compounded errors as the current state is considered by the policy before executing each state of the trajectory.

Learning from experience is commonly formulated as a Markov decision process (MDP). MDPs lend themselves naturally to motor actions, as they represent a state-action network and are therefore suitable for reinforcement learning. In addition the Markov property dictates that the next state is only dependent on the previous state and action, regardless of earlier states. This timeless property promotes stationary policies. There are different methods to learn from experience through reinforcement learning that are out of the scope of this paper. For a survey and formal definitions of reinforcement learning methods for intelligent agents, the reader is referred to [Kober et al. 2013]. Note that both learning paradigms are similarly formulated with the exception of the cost function; the feature vector $x(t)$ corresponds to s, $u(t)$ corresponds to a and $x(t+1)$ corresponds to the resulting state s'. It is therefore not uncommon (especially in more recent research) to combine learning from demonstrations and experience to perform a task.

We now consider the predicted action $u(t)$ in equation 1.

DEFINITION 8. *An action $u(t)$ can often represent a vector rather than a single value. This means that the action is comprised of more than one decision executed simultaneously; such as pressing multiple buttons on a controller or moving multiple joints in a robot.*

Actions can also represent different levels of motor control: low level actions, motor primitives and high level macro actions [Argall et al. 2009].

DEFINITION 9. *Low level actions are those that execute simple commands such as move forward and turn in navigation tasks, or jump and shoot in games.*

These low level actions can be directly predicted using a supervised classifier. Low level actions also extend to regression when the predicted actions have continuous values rather than a discrete set of actions (see learning motion).

DEFINITION 10. *Motor primitives are simple building blocks that are executed in sequence to perform complex motions. An action is broken down into basic unit actions (often concerning one degree of freedom or actuator) that can be used to make up any action that needs to be performed in the given problem.*

These primitives are then learned by the policy. In addition to being useful in building complex actions from a discrete set of primitives, motor primitives can represent a desired move in state space, since they can be used to reach any possible state. As in MDPs described above, the transition from one state to another state based on which action is taken is easily tracked when using motor primitives. In this case the output of the policy in equation 1 can represent the change in the current state [Schaal et al. 2003] as follows:

$$\dot{x}(t) = \pi(x(t), t, \alpha) \tag{2}$$

DEFINITION 11. *High level macro actions are decisions that determine the immediate plan of the agent. It is then broken down to lower level action sequences. Examples of high level decisions are grasp object or perform forehand.*

For a thorough formalization of learning from demonstrations, we refer the reader to [Schaal et al. 2003].

3. FEATURE REPRESENTATIONS

Before learning a policy it is important to represent the observable state in a form that is adequate and efficient for training. This representation is called a feature vector. A feature may include information about the learner, its environment, manipulable objects and other agents in the experiment. Training features need to be adequate, meaning that they convey enough information to form a solid policy to solve the task. It is also important that the features can be extracted and processed efficiently with respect to the time and computational restriction of imitation learning applications.

When capturing data, the important question is: what to imitate? In most real applications, the environment is often too complicated to be represented in its totality, because it usually has an abundance of irrelevant or redundant information. It is therefore necessary to consider which aspects of the demonstrations we want to present to the learner.

3.1. Handling Demonstrations

Even before feature extraction stages, dealing with demonstrations poses a number of challenges. A major issue is the correspondence problem introduced in section 1. Creating correspondence mappings between teacher and learner can be computationally intensive, but some methods attempt to create such correspondence in real-time. In [Jin et al. 2016] a projection of the teacher's behavior to the agent's motion space is developed online by sparsely sampling corresponding features. Neural network can also be utilized to improve the response time of inverse kinematics (IK) based methods [Hwang et al. 2016] and find trajectories appropriate for the learner's motion space based on the desired state of end-effectors. However, IK methods place no further restrictions on the agent's behavior as long as the end effector is in the demonstrated position [Mohammad and Nishida 2013]. This poses a problem for trajectory imitation applications such as gesture imitation. To alleviate this limitation [Mohammad and Nishida 2013] propose a close-form solution to the correspondence problem based on optimizing external (mapping between observed demonstrations and expected behavior of learner) and learner mapping (mapping between state of the learner and its observed behavior). While most approaches store learned behaviors after mapping to the learner's motion space, in [Bandera 2010] human gestures are captured, identified and learned in the motion space of the teacher. While that requires learning a model for the teacher's motion, it allows perception to be totally independent of the learner's constraints and facilitates human motion recognition. The learned motions are finally translated to the robot's motion space before execution. A different approach is to address correspondence in reward space where corresponding reward profiles for the demonstrators and the robot are built [González-Fierro et al. 2013]. The difference between them is optimized with respect to the robots internal constraints to ensure the feasibility of the developed trajectory. This enables learning from demonstrations by multiple human teachers with different physical characteristics.

Another challenge concerning demonstrations is incomplete or inaccurate demonstrations. Statistical models can deal with sensor error or inaccurate demonstrations, however, incomplete demonstrations can result in suboptimal behavior [Khansari-Zadeh and Billard 2011]. In [Khansari-Zadeh and Billard 2011] it is noted that a robot

provided only with demonstrations starting from the right of the target, will start by moving to the familiar position if initialized in a different position. In [Kim et al. 2013] a method for learning from limited and inaccurate data is proposed, where the demonstrations are used to constraint a reinforcement learning algorithm that learns from trial and error. Combining learning from demonstrations and experience is extensively investigated in subsection 5.1. As an alternative way to cope with the lack of good demonstrations, [Grollman and Billard 2012] learn a policy from failed demonstrations where the agent is trained to avoid repeating unsuccessful behavior.

Demonstrations need not be provided by a dedicated teacher but can instead be observed from another agent. In this case an important question is what to imitate from the continuous behavior being demonstrated. In [Mohammad and Nishida 2012] learning from unplanned demonstrations is addressed by segmenting actions from the demonstrations and estimating the significance of the behavior in these segments to the investigated task according to 3 criteria: (1) Change detection is used to discover significant regions of the demonstrations, (2) constrained motif discovery identifies recurring behaviors and (3) change-causality explores the causality between the demonstrated behavior and changes in the environment. Similarly, in [Tan 2015] acquired recordings of a human hand are segmented into basic behaviors before extracting relevant features and learning behavior generation. The feature extraction and behavior generation are performed with respect to 3 attributes: (1) Preconditions, which are environmental condition required for the task. (2) Internal constraints, which are characteristics of the agent that restrict its behavior. (3) Post results, which represent the consequences of a behavior.

Regardless of the source of the signal, captured data may be represented in different ways. We categorize representations as: raw features, designed or engineered features, and extracted features. Figure 2 Shows the relations between different feature representations.

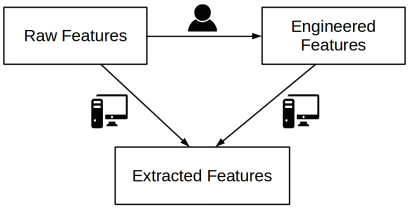

Fig. 2. Features relation diagram

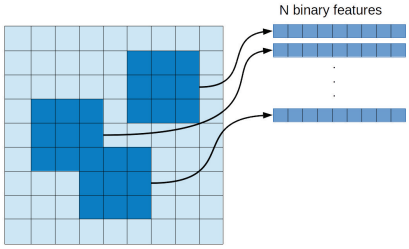

Fig. 3. Extracting binary features from an image

3.2. Raw Features

Raw data captured from the source can be used directly for training. If the raw features convey adequate information and are of an appropriate number of dimensions, they can be suitable for learning with no further processing. This way no computational overhead is spent in calculating features.

In [Ross and Bagnell 2010] the agent learns to play 2 video games: Mario Bros, a classic 2D platformer, and Super TuX Kart, a 3D kart racing game. In both cases, the input is a screenshot of the game at the current frame with the number of pixels reduced. In Super TuX Kart a lower dimensional version of the image is directly input

to the classifier without extracting any designed features. The image is down-sampled to 24 $\times$ 18 pixels to avoid the complications that come with high dimensional feature vectors. An image of that size with three color channels yields a 1,296 feature vector.

3.3. Manually Designed Features

These are features that are extracted using specially designed functions. These methods incorporate expert knowledge about the data and application to determine what useful information can be derived by processing the raw data. [Torrey et al. 2005] extract features from the absolute positions of players on a soccer field. More meaningful information such as relative distance from a player, distance and angle to the goal, and length and angles of player triangles are calculated. The continuous features are discretized into overlapping tiles. This transformation of features from a numeric domain to binary tiles is reported to significantly improve learning.

Manually designed features are popular with learning from visual data. They play an important part in computer vision methods that are used to teach machines by demonstration. Computer vision approaches are popular from an early point in teaching robots to behave from demonstrations [Schaal 1999]. These approaches rely on detecting objects of interest and tracking their movement to capture the demonstrated actions in a form that can be used for learning. In [Demiris and Dearden 2005] a computer vision system is employed to create representations that are used to train a bayesian belief network. Training samples are generated by detecting and tracking objects in the scene; which is performed by clustering regions of the image according to their position and movement properties. [Billard and Matarić 2001] imitate human movement by tracking relevant points on the human body. A computer vision system is used to detect human arm movement and track motion based on markers worn by the performer. The system only learns when motion is detected by observing change in the marker positions. In a recent study [Hwang et al. 2016], humanoid robots are trained to imitate human motion from visual observation. Demonstrations are captured using a stereo-vision system to create a 3D image sequence. The demonstrator's body is isolated from the sequence and a set of predetermined points on the upper and lower body are identified. Subsequently the extracted features are used to estimate the demonstrator's posture along the trajectory. A variation of inverse-kinematics that employs neural networks is used to reproduce the key posture features in the humanoid robot.

For the Mario bros game in [Ross and Bagnell 2010], the source signal is the screenshot at the current frame. The image is divided into 22 $\times$ 22 equally sized cells. For each cell, 14 binary features (the value of each feature can be 0 or 1) are generated; each signifying whether or not the cell contains a certain object such as enemies, obstacles and/or power-ups. As such, each cell can contain between 0 and 14 of the predefined objects. A demonstration is made up of the last 4 frames (so as to contain information about the direction in which objects are moving) as well as the last 4 chosen actions. [Ortega et al. 2013] use a similar grid to represent the environment, but add more numerical features and features describing the state of the character.

Figure 3 illustrates dividing an image to sub-patches and extracting binary features.

3.4. Extracted Features

Feature extraction techniques automatically process raw data to extract the features used for learning. The most relevant information is extracted and mapped to a different domain usually of a lower dimensionality. When dealing with high DOF robots, describing the posture of the robot using the raw values of the joints can be ineffective due to the high number of dimensions. This is more pronounced if the robot only uses a limited number of joints to perform the action rendering most of the features

irrelevant. This issue also applies to visual information. If the agent observes its surroundings using visual sensors, it is provided with high dimensional data in the form of pixels per frame. However, at any given point, most of the pixels in the captured frame would probably be irrelevant to the agent or contain redundant information.

Principal component analysis (PCA) can be used to project the captured data onto orthogonal axes and represent the state of the agent in lower dimensions. This technique has been widely used with high DOF robots [Ikemoto et al. 2012] [Berger et al. 2008] [Vogt et al. 2014] [Calinon and Billard 2007b]. In [Curran et al. 2015], PCA is used to extract features in a Mario game. Data describing the state of the playable character, dangers, rewards and obstacles is projected onto as few as 4 dimensions. It is worth mentioning that the three types of features (raw, designed and extracted) were used in the literature to provide representations for the same Mario task.

Deep learning approaches [Bengio 2009] can also be used to extract features without expert knowledge of the data. These approaches find success in automatically learning features from high dimensional data; especially when no established sets of features are available. In a recent study [Mnih et al. 2015], Deep Q Learning (DQN) is used to learn features from high dimensional images. The aim of this technique is to enable a generic model to learn a variety of complex problems automatically. The method is tested on 49 Atari games, each with different environments, goals and actions. Therefore, it is beneficial to be able to extract features automatically from the captured signals (in this case screenshots of the Atari games at each frame) rather than manually design specific features for each problem. A low resolution (84×84) version of the colored frames is used as input to a deep convolutional neural network (CNN) that is coupled with Q based reinforcement learning to automatically learn a variety of different problems through trial and error. The results in many cases surpass other AI agents and in some cases are comparable to human performance. Similarly, [Koutník et al. 2013] use deep neural networks to learn from video streams in a car racing game. Note that these examples utilize deep neural networks with reinforcement learning, without employing a teacher or optimal demonstrations. However the feature extraction techniques can be used to learn from demonstrations or experience alike. Since the success of DQN, several variations of deep reinforcement learning have emerged that utilize actor-critic methods [Mnih et al. 2016] [Lillicrap et al. 2015] which allow for potential combinations with learning from demonstrations. In [Guo et al. 2014] learning from demonstrations is applied on the same Atari benchmark [Bellemare et al. 2012]. A supervised network is used to train a policy using samples from a high performing but non real-time agent. This approach is reported to outperform agents that learn from scratch through reinforcement learning. In [Levine et al. 2015] deep learning is used to train a robot to perform a number of object manipulation tasks using guided policy search (see section on reinforcement learning).

Automatically extracted features have the advantage of minimizing the task specific knowledge required for training agents. Which allows the creation of more generic learning approaches that can be used to learn a variety of tasks directly from demonstrations with minimal tweaking. Learning with extracted features is gaining popularity due to recent advancements in deep learning. The success of deep learning methods in a variety of applications [Ciresan et al. 2012] [Krizhevsky et al. 2012] promotes learning from raw data without designing what to learn from the demonstrations. That being said, deep learning can also be used to extract higher level features from manually selected features. This approach allows for the extraction of complex features while limiting computation by manually selecting relevant information from the raw data. In recent attempts to teach an agent to play the board game 'Go' [Clark and Storkey 2015] [Silver et al. 2016], the board is divided into a 19×19 grid. Each cell in the grid consists of a feature vector describing the state of the game in this partition of

the board. This state representation is input into a deep convolutional neural network that extracts higher level features and maps the learned features to actions.

4. LEARNING MOTION

We now address the different methods for learning a policy from demonstrations. After considering what to learn, this process is concerned with the question how to learn? The most straight forward way to learn a policy from demonstrations is direct imitation. That is to learn a supervised model from the demonstration, where the action provided by the expert acts as the label for a given instance. The model is then capable of predicting the appropriate action when presented with a situation. Supervised learning methods are categorized into classification and regression.

4.1. Classification

Classification is a popular task in machine learning where observations are automatically categorized into a finite set of classes. A classifier $h(x)$ is used to predict the class y to which an independent observation x belongs; where $y \in Y$, $Y = \{y_1, y_2 \dots y_p\}$ is a finite set of classes, and $x = \{x_1, x_2 \dots x_m\}$ is a vector of m features. In supervised classification, $h(x)$ is trained using a dataset of n labeled samples $(x^{(i)}, y^{(i)})$, where $x^{(i)} \in X$, $y^{(i)} \in Y$ and $i = 1, 2 \dots n$.

Classification approaches are used when the learner's actions can be categorized into discrete classes [Argall et al. 2009]. This is suitable for applications where the action can be viewed as a decision such as navigation [Chernova and Veloso 2007b] and flight simulators [Sammut et al. 2014]. In [Chernova and Veloso 2007b], a Gaussian mixture models (GMM) is trained to predict navigational decisions. Meta classifiers are used in [Ross and Bagnell 2010] to learn a policy to play computer games. The base classifier used in this paper is a neural network. In The Kart racing game the analog joystick commands are discretized into 15 buckets, reducing the problem to a 15 class classification problem. So the neural network used had 15 output nodes. The Mario Bros game uses a discrete controller. Actions are selected by pressing one or more of 4 buttons. So in the neural network, the action for a frame is represented by 4 output nodes. This enables the classifier to choose multiple classes for the same instance. Although the results are promising, it is argued that using an Inverse Optimal Control (IOC) technique [Ratliff et al. 2007b] as the base classifier might be beneficial. In [Ross et al. 2010] the experiments are repeated this time using regression (see regression section) to learn the analog input in Super Tux Kart. For Mario Bros, 4 Support Vector Machine (SVM) classifiers replace the neural network to predict the value of each of the 4 binary classes. Classification can also be used to make decisions that entail lower level actions. In [Raza et al. 2012] high level decision are predicted by the classifier in a multi-agent soccer simulation. Decisions such as 'Approach ball' and 'Dribble towards goal' can then be deterministically executed using lower level actions. An empirical study is conducted to evaluate which classifiers are best suited for the imitation learning task. Four different classifiers are compared with respect to accuracy and learning time. The results show that a number of classifiers can perform predictions with comparable accuracy, however, the learning time relative to the number of demonstrations can vary greatly [Raza et al. 2012]. Recurrent neural networks (RNN) are used in [Rahmatizadeh et al. 2016] to learn trajectories for object manipulation from demonstrations. RNNs incorporates memory of past actions when considering the next action. Storing memory enables the network to learn corrective behavior such as recovery from failure given that the teacher demonstrates such a scenario.

4.2. Regression

Regression methods are used to learn actions in a continuous space. Unlike classification, regression methods map the input state to a numeric output that represents an action. Thus they are suitable for low level motor actions rather than higher level decisions. Especially when actions are represented continuous values, such as input from a joystick [Ross et al. 2010]. The regressor $I(x)$ maps an independent sample x to a continuous value y rather than a set of classes. Where $y \in \mathbb{R}$, the set of real numbers. Similarly the regressor is trained using a set of labeled samples $(x^{(i)}, y^{(i)})$, where $y^{(i)} \in \mathbb{R}$ and $i = 1, 2 \ldots n$.

A commonly used technique is locally weighted regression (LWR). LWR is suitable for learning trajectories, as these motions are made up of sequences of continuous values. Examples of such motions are batting tasks [Kober and Peters 2009c] [Ijspeert et al. 2002b] (where the agent is required to execute a motion trajectory in order to pass by a point and hit a target); and walking [Nakanishi et al. 2004] where the agent needs to produce a trajectory that results in smooth stable motion. A more comprehensive application is table tennis. [Mülling et al. 2013] use Linear Bayesian Regression to teach a robot arm to play a continuous game of table tennis. The agent is required to move with precision in a continuous 3D space in different situations, such as when hitting the ball, recovering position after a hit and preparing for the opponent's next move. Another paradigm commonly used for regression is artificial neural networks (ANN). Neural networks differ from other regression techniques in that they are demanding in training time and training samples. Neural network approaches are often inspired by biology and neuroscience studies, and attempt to emulate the learning and imitation process in humans and animals [Billard et al. 2008]. The use of regression with a dynamic system of motor primitives has produced a number of applications for learning discrete and rhythmic motions [Ijspeert et al. 2002a] [Schaal et al. 2007] [Kober et al. 2008], though most approaches focus on direct imitation without further optimization [Kober and Peters 2009a]. In such applications, a dynamic system represents a single degree of freedom (DOF) as each DOF has a different goal and constraints [Schaal et al. 2007].

Dynamic systems can be combined with probabilistic machine learning methods to reap the benefits of both approaches. This allows the extraction of patterns that are important to a given task and generalization to different scenarios while maintaining the ability to adapt and correct movement trajectories in real time [Calinon et al. 2012]. In [Calinon et al. 2012] the estimation of dynamical systems' parameters is represented as a Gaussian mixture regression (GMR) problem. This approach has an advantage over LWR based approaches as it allows learning of the activation functions along with the motor actions. The proposed method is used to learn time-based and time-invariant movement. In [Rozo et al. 2015] a similar GMM based method is used in a task-parametrized framework which allows shaping the robot's motion as a function of the task parameters. Human demonstrations are encoded to reflect parameters that are relevant to the task at hand and identify the position, velocity and force constraints of the task. This encoding allows the framework to derive the state in which the robot should be, and optimize the movement of the robot accordingly. This approach is used in a Human Robot Collaboration (HRC) context and aims to optimize human intervention as well as robot effort. Deep learning is combined with dynamical systems in [Chen et al. 2015]. Dynamic movement primitives (DMP) are embedded into autoencoders that learn representations of movement from demonstrated data. Autoencoders non-linearly map features to a lower dimensional latent space in the hidden layer. However, in this approach, the hidden units are constrained to DMPs to limit the hidden layer into representing the dynamics of the system.

In both classification and regression methods, a design decision can be made regarding the learning models resources. Lazy learners such as kNN and LWR do not need training but need to retain all training samples when performing predictions. On the other hand, trained models such as ANN and SVM require training time, but once a model is created the training samples are no longer needed and only the model is stored, which saves memory. These models can also result in very short prediction times.

4.3. Hierarchical Models

Rather than using a single model to reproduce human behavior, a hierarchical model can be employed that breaks down the learned actions into a number of phases. A classification model can be used to decide which action or sub-action, from a set of possible actions, should be performed at a given time. A different model is then used to define the details of the selected action, where each possible low-level action has a designated model. [Bentivegna et al. 2004] use a hierarchical approach for imitation learning on two different problems. The first is Air Hockey which is played against an opponent, and the objective is to shoot a puck into the opponent's goal while protecting your own. The second game is marble maze; the maze can be tilted around different axis to move the ball towards the end of the maze. Each task has a set of low level actions called motor primitives that make up the playing possibilities for the agent (e.g., Straight shot, Defend Goal, and Roll ball to corner). In the first stage, a nearest neighbor classifier is used to select the action to be performed. By observing the state of the game the classifier searches for the most similar instances in the demonstrations provided by the human expert, and retrieves the primitive selected by the human at that point. The next step is to define the goal of the selected action, for example the velocity of the ball or the position of the puck when the primitive is completed. The goal is then used in a regression model to find the parameters of the action that would optimize the desired goal. The goal is derived from the k nearest neighbor demonstrations found in the previous step. The goals in those demonstrations are input in a local weighted regression model to perform the primitive. In a similar fashion, [Chernova and Veloso 2008] use a classifier to make decisions in a sorting task consisting of the following macro actions (wait, sort left, sort right and pass). Each macro action entails temporal motor actions such as picking up a ball, moving and placing the ball.

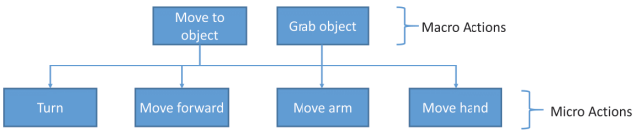

Fig. 4. Example of hierarchical learning of actions

Table I shows a list of methods used for direct imitation in the literature. Along with the year, the domain in which imitation learning was used, and whether additional learning methods where used to improve learning from demonstrations. Popular applications in robotics are given their own category, such as navigation or batting (which are applications where a robot limb moves to make contact with an object, such as table tennis). More diverse or generic tasks are listed as robotics. The table shows that robotics and games are popular domains for imitation learning. They cover a wide variety of applications where an intelligent agent acts in the real world and in simulated environments respectively. Robotics is an attractive domain for AI research due to the

huge potential in applications that can take advantage of sensing and motor control in the physical world. While video games can be attractive because they alleviate many challenges such as data capturing and sensor error; and thus allow development of new complex learning methods in a controlled and easily reproducible environment. Moreover, games have built-in scoring measures that can facilitate evaluation and even designing the learning process in reinforcement learning approaches.

5. INDIRECT LEARNING

In this section, we discuss indirect ways to learn policies that can complement or replace direct imitation. The policy can be refined from demonstrations, experience or observation to be more accurate or to be more general and robust against unseen circumstances.

It is often the case that direct imitation on its own is not adequate to reproduce robust, human-like behavior in intelligent agents. This limitation can be attributed to two main factors: (1) errors in demonstration, and (2) poor generalization. Due to limitations of data acquisition techniques, such as correspondence problem, sensor error and physical influences in kinesthetic demonstrations [Argall et al. 2009], direct imitation can lead to inaccurate or unstable performance, especially in tasks that require precise motion in continuous space such as reaching or batting. For example, in [Berger et al. 2008] a robot attempting to walk by directly mimicking the demonstrations would fall because the demonstrations do not accurately take into consideration the physical properties involved in the task such as the robot's weight and center of mass. However, refinement of the policy through trial and error would take these factors into account and produce a stable motion.

While generalization is an important issue in all machine learning practices, a special case of generalization is highlighted in imitation learning applications. It is common that human demonstrations are provided as sequence of actions. The dependence of each action on the previous part of the sequence violates the 'iid' assumption of training samples that is integral to generalization in supervised learning [Ross and Bagnell 2010]. Moreover, since human experts provide only correct examples, the learner is unequipped to handle errors in the trajectory. If the learner deviates from the optimal performance at any point in the trajectory (which is expected in any machine learning task), it would be presented with an unseen situation that the model is not trained to accommodate for. A clear example is provided in [Togelius et al. 2007] where supervised learning was used to learn a policy to drive a car. Given that human demonstrations contained only 'good driving' with no crashes or close calls, when error occurs and the car deviates from demonstrated trajectories, the learner does not know how to recover.

5.1. Reinforcement Learning

Reinforcement learning (RL) learns a policy to solve a problem via trial and error.

DEFINITION 12. *In RL, an agent is modeled as a Markov Decision Process (MDP) that learns to navigate in a state space. A finite MDP consists of a tuple* (S, A, T, R)*, where* S *is a finite set of states,* A *is the set of possible actions,* T *is the set of state transition probabilities and* R *is a reward function.* TP_sa *contain a set of probabilities where* P_sa *is the probability of arriving at state* s *given action* a *and where* $a \in A$ *and* $s \in S$*. The reward function* $R(s_k, a_k, s_{k+1})$ *returns an immediate reward for taking an action in a given state and ending up in a new state* $a_k \rightarrow s_{k+1}$ *where* k *is the time step. This reward is discounted over time by a discount factor* $\gamma \in [0, 1)$ *and the goal of the agent is to maximize the expected discounted reward at each time step.*

Table I. Direct Learning Methods.

Paper	Domain	Learning Method	Self-improvement
[Lin 1992]	navigation	Artificial Neural Networks (ANN)	✓
[Pook and Ballard 1993]	object manipulation	Hidden Markov Model (HMM), K-Nearest Neighbor (KNN)	✗
[Mataric 2000b]	robotics	ANN	✗
[Billard and Matarić 2001]	robotics	ANN	✗
[Ijspeert et al. 2002b]	robotics	Local Weighted Regression (LWR)	✗
[Geisler 2002]	video game	Naive Bayes (NB), Decision Tree (DT), ANN	✗
[Oztop and Arbib 2002]	object manipulation	ANN	✗
[Nicolescu and Mataric 2003]	object manipulation	graph based method	✓
[Dixon and Khosla 2004]	navigation	HMM	✗
[Ude et al. 2004]	robotics	optimization	✗
[Nakanishi et al. 2004]	robotics	LWR	✗
[Bentivegna et al. 2004]	robotics	KNN, LWR	✓
[Thurau et al. 2004b]	games	bayesian methods	✗
[Aler et al. 2005]	soccer simulation	PART	✓
[Torrey et al. 2005]	soccer simulation	Rule based learning	✓
[Saunders et al. 2006]	navigation, object manipulation	KNN	✗
[Chernova and Veloso 2007b]	navigation	Gaussian Mixture Model (GMM)	✓
[Guenter et al. 2007]	object manipulation	GMR	✓
[Togelius et al. 2007]	games/driving	ANN	✗
[Schaal et al. 2007]	batting	LWR	✗
[Calinon and Billard 2007b]	object manipulation	GMM, GMR	✗
[Berger et al. 2008]	robotics	direct recording	✓
[Asfour et al. 2008]	object manipulation	HMM	✗
[Coates et al. 2008]	aerial vehicle	Expectation Maximization (EM)	✗
[Mayer et al. 2008]	robotics	ANN	✓
[Kober and Peters 2009c]	batting	LWR	✓
[Munoz et al. 2009]	games/driving	ANN	✗
[Cardamone et al. 2009]	games/driving	KNN, ANN	✗
[Ross et al. 2010]	games	Support Vector Machine (SVM)	✓
[Muñoz et al. 2010]	games/driving	ANN	✗
[Ross and Bagnell 2010]	games	ANN	✓
[Geng et al. 2011]	robot grasping	ANN	✗
[Ikemoto et al. 2012]	assistive robots	GMM	✓
[Judah et al. 2012]	benchmark tasks	linear logistic regression	✓
[Vlachos 2012]	structured datasets	online passive-aggressive algorithm	✓
[Raza et al. 2012]	soccer simulation	ANN, NB, DT, PART	✓
[Mülling et al. 2013]	batting	Linear Bayesian Regression	✓
[Ortega et al. 2013]	games	ANN	✓
[Niekum et al. 2013]	robotics	HMM	✓
[Rozo et al. 2013]	robotics	HMM	✓
[Vogt et al. 2014]	robotics	ANN	✗
[Droniou et al. 2014]	robotics	ANN	✗
[Brys et al. 2015b]	benchmark tasks	Rule based learning	✓
[Levine et al. 2015]	object manipulation	ANN	✓
[Silver et al. 2016]	board game	ANN	✓

RL starts off with a random policy and modifies its parameters based on rewards gained from executing this policy. Reinforcement learning can be used on its own to learn a policy for a variety of robotic applications. However, if a policy is learned from demonstration, reinforcement learning can be applied to fine tune the parameters. Providing positive or negative examples to train a policy helps reinforcement learning by reducing the search space available [Billard et al. 2008]. Enhancing the policy using RL is sometimes necessary if there are physical discrepancies between the teacher and the learner or to alleviate errors in acquiring the demonstrations. RL can also be useful to train the policy for unseen situations that are not covered in the demonstrations. Applying reinforcement learning to the learned policy instead of a random one can significantly speed up the RL process and avoids the risk of the policy from converging into a local minimum. Moreover RL can find a policy to perform a task that does not look normal to the human observer. In applications where the learner interacts with a human, it is important for the user to intuitively recognize the agent's actions. This is common in cases where robots are introduced into established environments (such as homes and offices) to interact with untrained human users [Calinon and Billard 2008]. By applying Reinforcement learning to a policy learned from human demonstrations the problem of unfamiliar behavior can be avoided. In imitation learning methods, reinforcement learning is often combined with learning from demonstrations to improve a learned policy when the fitness of the performed task can be evaluated.

In early research, teaching using demonstrations of successful actions was used to improve and speed up reinforcement learning. In [Lin 1992], reinforcement learning is used to learn a policy to play a game in a 2D dynamic environment. Different methods for enhancing the RL policy are examined. The results demonstrate that teaching the learner with demonstrations improves its score and helps prevent the learner from falling in local minima. It is also noted that the improvement from teaching increases with the difficulty of the task. Solutions to simple problems can be easily inferred without requiring demonstrations from an expert. But as the complexity of the task increases the advantage of learning from demonstrations becomes more significant, and even necessary for successful learning in more difficult tasks [Lin 1991].

In [Guenter et al. 2007] Gaussian mixed regression (GMR) is used to train a robot on an object grasping task. Since unseen scenarios such as obstacles and the variable location of the object are expected in this application, reinforcement learning is used to explore new ways to perform the task. The trained system is a dynamic system that performs damping on the imitated trajectories. This allows the robot to smoothly achieve its target and prevents reinforcement learning from resulting in oscillations. Using damping in dynamic systems is a common approach when combining imitation learning and reinforcement learning [Kober and Peters 2010][Kober et al. 2013].

An impressive application of imitation and reinforcement learning is training an agent to play the board game 'Go' that rivals human experts [Silver et al. 2016]. A deep convolutional neural network is trained using past games. Then reinforcement learning is used to refine the weights of the network and improve the policy.

A different approach to combine learning from demonstrations with reinforcement learning is employed in [Brys et al. 2015a]. Rather than using the demonstrations to train an initial policy, they are used to derive prior knowledge for reward shaping [Ng et al. 1999]. A reward function is used to encourage sub-achievements in the task, such as milestones reached in the demonstrations. This reward function is combined with the primary reward function to supply the agent with the cost of its actions. This paradigm of using expert demonstrations to derive a reward function is similar to inverse reinforcement learning approaches [Abbeel and Ng 2004].

Policy search methods are a subset of reinforcement learning that lend themselves naturally to robotic applications as they scale to high dimensional MDPs [Kober et al.

2013]. Therefore policy search methods are a good fit to integrate with imitation learning methods. A policy gradient method is used in [Kohl and Stone 2004] to improve an existing policy that can be created through supervised learning or explicit programming. A similar approach [Peters and Schaal 2008] is used within a dynamic system that was previously used for supervised learning from demonstrations [Ijspeert et al. 2002b]. This led to a series of work that utilizes the dynamic system in [Ijspeert et al. 2002b] to learn from demonstrations and subsequently use reinforcement learning for self-improvement [Kober and Peters 2010] [Kober and Peters 2014] [Buchli et al. 2011] [Pastor et al. 2011]. This framework is used to teach robotic arms a number of applications such as ball in cup, ball paddling [Kober and Peters 2010] [Kober and Peters 2009c] and playing table tennis [Mülling et al. 2013]. Rather than using reinforcement learning to refine a policy trained from demonstrations, demonstrations can be used to guide the policy search. In [Levine and Koltun 2013] differential dynamic programming is used to generate guiding samples from human demonstrations. These guiding samples help the policy search explore high reward regions of the policy space.

Recurrent neural networks are incorporated into guided policy search in [Zhang et al. 2016] to facilitate dealing with partially observed problems. Past memories are augmented to the state space and are considered when predicting the next action. A supervised approach uses demonstrated trajectories to decide which memories to store while reinforcement learning is used to optimize the policy including the memory state values.

A different way to utilize reinforcement learning in imitation learning is to use RL to provide demonstrations for direct imitation. This approach does not need a human teacher as the policy is learned from scratch using trial and error and then used to generate demonstrations for training. One reason for generating demonstrations and training a supervised model rather than using the RL policy directly is that the RL method does not act in real-time [Guo et al. 2014]. Another situation is when the RL policy is learned in a controlled environment. In [Levine et al. 2015] reinforcement learning is used to learn a variety of robotic tasks in a controlled environment. Information such as the position of target objects is available during this phase. A deep convolutional neural network is then trained using demonstrations from the RL policy. The neural network learns to map visual input to actions and thus learns to perform the tasks without the information needed in the RL phase. This mimics human demonstrations as humans utilize expert knowledge – that is not incorporated in the training process – to provide demonstrations.

For a comprehensive survey of reinforcement learning in robotics, the reader is referred to [Kober et al. 2013]

5.2. Optimization

Optimization approaches can also be used to find a solution to a given problem.

DEFINITION 13. *Given a cost function $f : A \rightarrow \mathbb{R}$ that reflects the performance of an agent, where A is a set of input parameters and $\mathbb{R}$ is the set of real numbers, optimization methods aim to find the input parameters x_0 that minimize the cost function. Such that $f(x_0) \leq f(x)\ \forall x \in A$*

Similar to reinforcement learning, optimization techniques can be used to find solutions to problems by starting with a random solution and iteratively improving to optimize the fitness function. Evolutionary algorithms (EA) are popular optimization methods that have extensively been used to find motor trajectories for robotic tasks [Nolfi and Floreano 2000]. EAs are used to generate motion trajectories for high and low DOF robots [Rokbani et al. 2012] [Min et al. 2005]. Popular swarm intelligence methods such as Particle Swarm Optimization (PSO) [Zhang et al. 2015] and Ant

Colony Optimization (ACO) [Zhang et al. 2010] are used to generate trajectories for unmanned vehicle navigation. These techniques simulate the behavior of living creatures to find and optimal global solution in the search space. As is the case with reinforcement learning, evolutionary algorithms can be integrated with imitation learning to improve trajectories learned by demonstration or to speed up the optimization process.

In [Berger et al. 2008] a genetic algorithm (GA) is used to optimize demonstrated motion trajectories. The trajectories are used as a starting population for the genetic algorithm. The recorded trajectories are encoded as chromosomes constituted of genes representing the motor primitives. The GA searches for the chromosome that optimizes a fitness function that evaluates the success of the task. Projecting the motor trajectories to lower dimension illustrates the significant change between the optimized motion and the one learned directly from kinesthetic manipulation [Berger et al. 2008].

Similarly in [Aler et al. 2005] evolutionary algorithms are used after training agents in a soccer simulation. A possible solution (chromosome) is represented as a set of *if-then* rules. The rules are finite due to the finite permutations of observations and actions. A weighted function of the number of goals and other performance measures is used to evaluate the fitness of a solution. Although the evolutionary algorithm had a small population size and did not employ crossover, it showed promising results over the rules learned from demonstrations.

[Togelius et al. 2007] also used evolutionary algorithms to optimize multiple objectives in a racing game. The algorithms evolve an optimized solution (controller) from an initial population of driving trajectories. Evaluation of the evolved controllers found that they stay faithful to the driving style of the players they are modeled after. This is true for quantitative measures such as speed and progress, and for subjective observations such as driving in the center of the road.

[Ortega et al. 2013] treat the weights of a neural network as the genome to be optimized. The initial population is provided by training the network with demonstrated samples to initialize the weights. The demonstrations are also used to create a fitness value corresponding to the mean squared error distance from the desired outputs (human actions).

In [Sun et al. 2008] Particle Swarm Optimization (PSO) is used to find the optimal path for an Unmanned Aerial Vehicle (UAV) by finding the best control points on a B-spline curve. The initial points that serve as the initial PSO particles are provided by skeletonization. A social variation of PSO is introduced in [Cheng and Jin 2015], inspired by animals learning in nature from observing their peers. Each particle starts with a random solution and a fitness function is used to evaluate each solution. Then imitator particles (all except the one with the best fitness) modify their behavior by observing demonstrator particles (better performing particles). As in nature an imitator can learn from multiple demonstrators and a demonstrator can be used to teach more than one imitator. Interactive Evolutionary algorithms (IEA) [Gruau and Quatramaran 1997] employ a different paradigm. Rather than use human input to start an initial population of solutions and the optimize them, IEA uses human input to judge the fitness of the solutions. To avoid the user evaluating too many potential solutions, a model is trained on supervised examples to estimate the human user's evaluation. In [Bongard and Hornby 2013] fitness based search is combined with Preference-based Policy Learning (PPL) to learn robot navigation. The user evaluations from PPL guide the search away from local minima while the fitness based search searches for a solution. In similar spirit [Lin et al. 2011] train a robot to imitate human arm movement. The difference in degrees of freedom (DOF) between the human demonstrator and the robot obstructs using the demonstrations as an initial population. However, rather than use human input to subjectively evaluate a solution, the similarity of the robot movement to human demonstrations is quantitatively evaluated. A sequence-

independent joint representation for the demonstrator and the learner is used to form a fitness function. PSO is used to find the joint angles to optimize this similarity measure. A different method of integrating demonstrations is proposed in [El-Hussieny et al. 2015]. Inspired by Inverse Reinforcement Learning (see section on apprenticeship learning), an Inverse Linear Quadratic Regulator(ILQR) framework is used to learn cost function optimized by the human demonstrator. PSO is then employed to find a solution for the learned function instead of gradient methods.

5.3. Transfer Learning

Transfer learning is a machine learning paradigm where knowledge of a task or a domain is used to enhance learning of another task.

DEFINITION 14. *Given a source Domain D_s and task T_s, transfer learning is defined as improving the learning of a target task T_t in domain D_t using knowledge of D_s and T_s; where $D_s \neq D_t$ or $T_s \neq T_t$. A domain $D = \{\chi, P(X)\}$ is defined as a feature space χ and a marginal probability distribution $P(X)$, Where $X = \{x_1, ...x_n\} \in \chi$. The condition $D_s \neq D_t$ holds if $\chi_s \neq \chi_t$ or $P_s(X) \neq P_t(X)$ [Pan and Yang 2010].*

A learner can acquire various forms of knowledge about a task from another agent such as useful feature representations or parameters for the learning model. Transfer learning is relevant to imitation learning and robotic applications because acquiring samples is difficult and costly. Utilizing knowledge of a task that we already invested to learn can be efficient and effective.

A policy learned in one task can be used to advice (train) a learner on another task that carries some similarities. In [Torrey et al. 2005] this approach is implemented on two robocup soccer simulator tasks, the first is to keep the ball from the other team, and second to score a goal. It is obvious that skills learned to perform the first task can be of use in the later. In this case advice is formulated as a rule concerning the state and one or more action. To create advice the policy for the first task is learned using reinforcement learning. The learned policy is then mapped by a user (to avoid discrepancies in state or action spaces) into the form of advice that is used to initiate the policy for the second task. After receiving advice the learner continues to refine the policy through reinforcement learning and can modify or ignore the given advice if it proves through experience to be inaccurate or irrelevant.

Often in transfer learning, human input is needed to map the knowledge from one domain to another, however, in some cases the mapping procedure can be automated [Torrey and Shavlik 2009]. For example, in [Kuhlmann and Stone 2007] a mapping function for general game playing is presented. The function automatically maps between different domains to learn from previous experience. The agent is able to identify previously played games relevant to the current task. The agent may have played the same game before or a similar one and is able to select an appropriate source task to learn from without it being explicitly designated. Experiments show that the transfer learning approach speeds up the process of learning the game via reinforcement learning (compared to learning from scratch) and achieves a better performance after the learning iterations are complete. The results also suggest that the advantage of using transfer learning is correlated with the number of training instances transferred from the source tasks. Even if the agent encounter negative transfer [Pan and Yang 2010] for example from overfitting to the source task, it can recover by learning through experience and rectifying its model in the current task to converge in appropriate time [Kuhlmann and Stone 2007].

Brys et al [Brys et al. 2015b] combine reward shaping and transfer learning to learn a variety of benchmark tasks. Since reward shaping relies on prior knowledge to influence the reward function, transfer learning can take advantage of a policy learned for

one task and perform reward shaping for a similar task. In [Brys et al. 2015b] transfer learning is applied from a simple version of the problem to a more complex one (e.g 2D to 3D mountain car and a Mario game without enemies to a game with enemies).

5.4. Apprenticeship Learning

In many artificial intelligence applications such as games or complex robotic tasks, the success of an action is hard to quantify. In that case the demonstrated samples can be used as a template for the desired performance. In [Abbeel and Ng 2004], apprenticeship learning (or inverse reinforcement learning) is proposed to improve a learned policy when no clear reward function is available; such as the task of driving. In such applications the aim is to mimic the behavior of the human teachers under the assumption that the teacher is optimizing an unknown function.

DEFINITION 15. *Inverse reinforcement learning (IRL) uses the training samples to learn the reward function being optimized by the expert and use it to improve the trained model.*

Thus, IRL obtains performance similar to that of the expert. With no reward function the agent is modeled as an MDP/R (S,A,T). Instead the policy is modeled after feature expectations μ_E derived from expert's demonstrations. Given n trajectories $\{s_0^{(i)}, s_1^{(i)}, \dots\}_{i=1}^{m}$ the empirical estimate for the feature expectation of the expert's policy $\mu_E = \mu(\Pi_E)$ is denoted as:

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)}). \tag{3}$$

Where γ is a discount factor and $\phi(s_t^{(i)})$ is the feature vector at time t of demonstration i. The goal of the RL algorithm is to find a policy $\bar{\pi}$ such that $||\mu(\bar{\pi}) - \mu_E||_2 \leq \epsilon$ where $\mu(\bar{\pi})$ is the expectation of the policy [Abbeel and Ng 2004].

[Ziebart et al. 2008] employ a maximum entropy approach to IRL to alleviate ambiguity. Ambiguity arises in IRL tasks since many reward functions can be optimized by the same policy. This poses a problem when learning the reward function, especially when presented with imperfect demonstrations. The proposed method is demonstrated on a task of learning driver route choices where the demonstrations may be suboptimal and non-deterministic. This approach is extended to a deep-learning framework in [Wulfmeier et al. 2015]. Maximum entropy objective functions enable straightforward learning of the network weights, and thus the use of deep networks trained with stochastic gradient descent [Wulfmeier et al. 2015]. The deep architecture is further extended to learn the features via Convolution layers instead of using pre-extracted features. This is an important step in the route to automate the learning process. One of the main challenges in reinforcement learning through trial and error is the requirement of human knowledge in designing the feature representations and reward functions [Kober et al. 2013]. By using deep learning to automatically learn feature representations and using IRL to infer reward functions from demonstrations, the need for human input and design is minimized. The inverse reinforcement learning paradigm provides an advantage over other forms of learning from demonstrations in that the cost function of the task is decoupled from the environment. Since the objective of the demonstrations is learned rather than demonstrations themselves, the demonstrator and learner do not need to have the exact skeleton or surroundings, thus alleviating challenges such as the correspondence problem. Therefore, it is easier to provide demonstrations that are generic and not tailor-made for a specific robot or environment.

In addition, IRL can be employed rather than traditional RL even if a reward function exists (given that demonstrations are available). For example, in [Lee et al. 2014] apprenticeship learning is used to derive a reward function from expert demonstrations in a Mario game. While the goals in a game such as Mario can be pre-defined (such as score from killing enemies and collecting coins or the time to complete the level), it is not known how an expert user prioritizes these goals. So in an effort to mimic human behavior, a reward function extracted from demonstrations is favored to a manually designed reward function.

5.5. Active Learning

Active learning is a paradigm where the model is able to query an expert for the optimal response to a given state, and use these active samples to improve its policy.

DEFINITION 16. *A classifier $h(x)$ is trained on a labeled dataset $D_K(x^{(i)}, y^{(i)})$ and used to predict the labels of an unlabelled dataset $D_U(x^{(i)})$. A subset $D_C(x^{(i)}) \subset D_U$ is chosen by the learner to query the expert for the correct labels $y^{*(i)}$. The active samples $D_C(x^{(i)}, y^{*(i)})$ are used to train $h(x)$ with the goal of minimizing n : the number of samples in D_C.*

Active learning is a useful method to adapt the model to situations that were not covered in the original training samples. Since imitation learning involves mimicking the full trajectory of a motion, an error may occur at any step of the execution. Creating passive training sets that can avoid this problem is very difficult.

One approach to decide when to query the expert is using confidence estimations to identify parts of the learned model that need improvement. When performing learned actions, the confidence in this prediction is estimated and the learner can decide to request new demonstrations to improve this area of the application or to use the current policy if the confidence is sufficient. Alternating between executing the policy and updating it with new samples, the learner gradually gains confidence and obtains a generalized policy that after some time does not need to request more updates. Confidence based policy improvement is used in [Chernova and Veloso 2007b] to learn navigation and in [Chernova and Veloso 2008] for a macro sorting task.

In [Judah et al. 2012] active learning is introduced to enable the agent to query expert at any step in the trajectory, given all the past steps. This problem is reduced to iid active learning and is argued to significantly decrease the number of required demonstrations.

[Ikemoto et al. 2012] propose active learning in human-robot cooperative tasks. The human and robot physically interact to achieve a common goal in an asymmetric task (i.e the human and the robot have different roles). Active learning occurs between rounds of interaction and the human provides feedback to the robot via a graphical user interface (GUI). The feedback is recorded and is added to a database of training samples that is used to train the Gaussian mixture model that controls the actions of the robot. The physical interaction between the human and robot results in mutually dependent behavior. So with each iteration of interaction, the coupled actions of the two parties converge into a smoother motion trajectory. Qualitative analysis of the experiments show that if the human adapts to the robots actions, the interaction between them can be improved; and that the interaction is more significantly improved if the robot in turn adapts to the human's action with every round of interaction.

In [Calinon and Billard 2007b] the teacher initiates the corrections rather than the learner sending a query. The teacher observes the learner's behavior and kinesthetically corrects the position of the robot's joints while it performs the task. The learner tracks its assisted motion through its sensors and uses these trajectories to refine the

model which is learned incrementally to allow for additional demonstrations at any point.

5.6. Structured Predictions

In a similar spirit, DAGGER [Ross et al. 2010] employs sample aggregation to generalize for unseen situations. However, the approach is fundamentally different. DAGGER formulates the imitation learning problem as a structured prediction problem inspired by [Daumé Iii et al. 2009], an action is regarded as a sequence of dependent predictions. Since each action is dependent on the previous state, an error leads to unseen state from which the learner cannot recover, leading to compounded errors. DAGGER shows that it is both necessary and sufficient to aggregate samples that cover initial learning errors. Therefore, an iterative approach is proposed that uses an optimal policy to correct each step of the actions predicted using the current policy, thus creating new modified samples that are used to update the policy. As the algorithm iterates, the utilization of the optimal policy diminishes until only the learned policy is used as the final model.

[Le et al. 2016] propose an algorithm called SIMILE that mitigates the limitations of [Ross et al. 2010] and [Daumé Iii et al. 2009] by producing a stationary policy that doesn't require data aggregation. SIMILE alleviates the need for an expert to provide the action at every stage of the trajectory by providing "virtual expert feedback" that controls the smoothness of the corrected trajectory and converges to the expert's actions.

Considering past actions in the learning process is an important point in imitation learning as many applications rely on performing trajectories of dependent motion primitives. A generic method of incorporating memory in learning is using recurrent neural networks (RNN) [Droniou et al. 2014]. RNNs create a feedback loop among the hidden layers in order to consider the network's previous outputs and are therefore well suited for tasks with structured trajectories [Mayer et al. 2008].

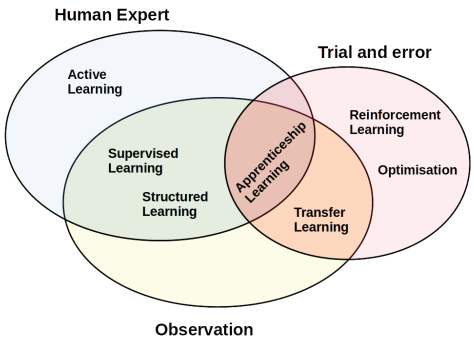

Fig. 5. learning methods from different sources

To conclude this section, Figure 5 shows a Venn diagram outlining the sources of data employed by different learning methods. An agent can learn from dedicated teacher demonstrations, observing other agent's actions or through trial and error. Active learning needs a dedicated oracle that can be queried for demonstrations. While other methods that utilize demonstrations can acquire them from a dedicated expert

or by observing the required behavior from other agents. RL and optimization methods learn through trial and error and do not make use of demonstrations. Transfer learning uses experience from old tasks, or knowledge from other agents to learn a new policy. Apprenticeship learning uses demonstrations from an expert or observation to learn a reward function. A policy that optimizes the reward function can then be learned through experience.

6. MULTI-AGENT IMITATION

Although creating autonomous multi-agents have been thoroughly investigated through reinforcement learning [Shoham et al. 2003] [Busoniu et al. 2008], it is not as extensively explored in imitation learning. Despite the lack of research, imitation learning and multi-agent applications can be a good fit. Learning from demonstrations can be improved in multi-agent environments as knowledge can be transferred between agents of similar objectives. On the other hand, imitation learning can be beneficial in tasks where agents need to interact in a manner that is realistic from a human's perspective. Following we present methods that incorporate imitation learning in multiple agents.

In [Price and Boutilier 1999] implicit imitation is used to improve a learner's RL model. A multi-agent setting enables an agent to learn by observing other agents performing a similar task using similar actions to those possessed by the agent. The mentor agent provides demonstrations by performing the task to optimize its objectives, so there is no need for a designated teacher; and the actions of an agent are unknown to other agents. A learner can observe the state changes resulting from the actions of a mentor agent and accordingly refine its model. This premise is useful for real applications where multiple agents act in the same environment. However, this work assumes that the agents are non-interacting, i.e., the consequences of one agent's actions are independent of other agents. Implicit imitation is closely related to transfer learning as a learner acquires the knowledge learned previously by a different learner. This application corresponds to a transductive transfer learning setting where the task is the same but the domains of the mentor and learner are different [Pan and Yang 2010]. An interesting aspect of this approach is that an agent can learn different skills from different mentors. In one experiment two mentors are acting to achieve different goals; the learner uses observations from both agents to learn a more difficult task. Note however, that the tasks were designed so that a combination of the policies used by the mentors form an optimal policy for the learner's problem. [Price and Boutilier 1999]. This can be considered as an example of hierarchical transfer learning, where learning solutions to multiple problems can help achieve more complex tasks [Torrey and Shavlik 2009]. However, in this case the knowledge of the tasks is learned through observations and imitation.

Multi-agent settings add complexity to the imitation problem in a number of ways. The learner's state space can be significantly expanded to include the status of other agents, as observing the actions of other agents affects its decision. The reward function is also affected if multiple agents compete or collaborate towards the same goal. The complexity of reward functions can increase even if the agents do not share the same goal. In a competitive setting, in addition to maximizing its own reward, an agent aims to minimize its opponents reward. In a cooperative setting, the total reward of the team might be taken into consideration. These new complexities can be incorporated in the learning process at different levels. For example, [Price and Boutilier 1999] exploit the presence of multiple agents to learn from observation, however, robots are non-interacting and act independently of each other. So state space and reward function are not affected. In 'keep away' in robot soccer, the robots collaborate as a team to keep ball from other team. However, the whole team share a Q policy, because they

all perform the same task. When a new action is performed the shared Q policy of all players is updated and the same reward is applied to the entire team. While in [Raza et al. 2012] agents learn different roles that complement each other to maximize the common goal. In a defensive task, one agent tries to recover the ball from the attacking opponents while the other falls back to act as a goal keeper. The roles are interchangeable – so each agent learns both skills as well as when to assume one of the two roles. In [DAmbrosio and Stanley 2013] a team of agents is treated as a pattern of policies rather than individual agents. That is a pattern that connects agents in a team that perform complementary roles. This method enables a team of agents to be scaled significantly after training without requiring retraining of different agents for similar roles.

Multi-agent learning from demonstration can however introduce new challenges in terms of acquiring demonstrations. In [Chernova and Veloso 2008] where active learning is employed in a multi-robot environment, the human expert is required to interact simultaneously with multiple robots. A system is then developed to divide the expert's time among the learners based on their need, by attracting the expert's attention through audio visual cues in order to query information.

7. EVALUATION

Imitation learning is a relatively young field, and evaluation of novel algorithms is challenging due to the lack of standard evaluation criteria and benchmark tasks. It is common that experiments are conducted on specialized robots or simulators. This makes comparison to other approaches difficult for two reasons, firstly, because the designed algorithms are often specialized for a specific task and setup, and secondly, because reproducing the results of an experiment needs special hardware or software that may not be available to other researchers.

One track of imitation learning uses structured prediction datasets as a benchmark for comparison with other techniques [Ross et al. 2010] [He et al. 2012] [Judah et al. 2012]. However, structured prediction tasks do not always correspond to real robotic applications. The dependency of a state on previous states can be limited. Furthermore, simple hamming loss is usually employed, i.e., if a mistake is made in step t the best action for $t + 1$ remains unchanged. This makes such benchmarks less suitable for testing robustness to errors and the ability to recover. Although not widely used to compare different techniques, some tools are readily available to perform and evaluate imitation learning such as video games [Thurau et al. 2004a] [Gorman 2009] [Mnih et al. 2015] [Geisler 2002] [Ross and Bagnell 2010], AI simulators [Raza et al. 2012] [Aler et al. 2005] and reinforcement learning benchmarks [Judah et al. 2012] [Randlov and Alstrom 1998]. Methods for evaluating imitation learning applications can be quantitative or qualitative.

7.1. Quantitative Evaluation

7.1.1. Error Calculation. These are tasks where the score of an executed policy can be directly calculated. For instance a ball in cup task ultimately depends on the ball falling into the cup [Kober and Peters 2009b]. The loss in the performance is the distance of the ball from the cup. Another example is quantifying progress in a Mario game as the distance traveled from left to right [Ross and Bagnell 2010]. In cyclic tasks where the agent is required to maintain an action, time can be inherently used as a score. For example, in Keepaway[Torrey et al. 2005], a robocup soccer simulator task, one team is required to keep the ball from the other team. The game ends if the ball is intercepted or goes out of bounds. Therefore, the duration that the ball is kept until the game ends reflects the performance of the keeping team. Another measure for evaluating cyclic actions is the amount of training needed to obtain a stable performance [Kober and Pe-

ters 2009b]. The error could also be calculated as deviation from the teacher's actions. This is possible if the motor primitives are the same for the teacher and test samples can be reproduced [Schaal et al. 2003]. However, these methods need to account for the fact that small errors could substantially change the action trajectory.

7.1.2. Quantitative Scores. In many applications the error cannot be explicitly calculated. However, some criteria can be used to quantify the quality of an executed policy. The performance is usually quantified by counting the number of desired (or unwanted) statuses achieved during the execution of a policy over a sequence. For example, in a driving simulator an undesired action is hitting an object. Thus, the number of collisions could be used as a criterion for quantifying a driving task [Abbeel and Ng 2004].

7.2. Qualitative Evaluation

In some applications it is as important for the action to appear natural and believable as it is to achieve the target. It is difficult to quantify the believability of a performance, even by comparing it to recorded human performances. In this case it is common for the performance to be subjectively judged by human inspection. As with any subjective analysis it is advisable to use more than one observer and regulate their decisions as comparable scores. [Gorman 2009] use a believability index to assess agents in a multiplayer game. The index is calculated from ratings on a scale of 1-5 of how-human like the performance appears. The ratings are performed by a number of independent observers.

[Ortega et al. 2013] conducted a qualitative study in the form of an online Turing test. The users are presented with a number of pairs of play-through footage of a Mario game. For each pair, the user has to decide which footage belonged to a human player, and which belonged to an AI controller. A similar evaluation is used in [Lee et al. 2014] for the same task. In [Ortega et al. 2013] the AI controllers where distinguishable from human players. However, agents that are designed to imitate human players where more believable than agents designed to perform well in the game. In addition to the Turing test, a qualitative evaluation is performed to test how closely the trained agents resemble human players. One drawback with this approach is that you can only compare an agent's behavior to one human demonstration, as human demonstrations can vary significantly among themselves.

It is clear that quantitative and qualitative analysis focus on different aspects of the agent's performance; and thus have different requirements. Table II summarizes points that help decide whether quantitative or qualitative evaluation is more appropriate for a given task.

Table II. When to use quantitative and qualitative evaluation.

Quantitative	**Qualitative**
Effectiveness in the task is more important than believability	How the agent acts is more important than what it achieves
Distance from the desired goal could be calculated	Subjects are available to perform subjective analysis
Scoring criteria could be designed for the task	Subjective analysis could be formalized(e.g via questionnaire)
Deviation from the optimal action could be calculated	A Turing test could be conducted (the performer is hidden from the observers)

In addition to evaluating the quality of an agent at performing a task, it is important for researchers to evaluate the value of the various self-improvement methods available. In transfer learning, three measures are used to quantify the advantage

of using prior knowledge form other tasks [Torrey and Shavlik 2009]. The first is to compare a model that has learned only from previous tasks to an untrained one. This measure demonstrates if the skills learned in the source task are of any use in the target task, without any subsequent learning. The second measure compares the time to completely learn the target task from scratch against a learner initialized using transfer learning. This is useful to find out if transfer learning can speed up the learning process for the target task. And finally to evaluate the performance of the initialized agent after subsequently learning from the target task against one that learned the target task from scratch. Comparison of the final performance represents the ability of transfer learning to improve over a fully trained learner. A graph plotting the performance of an agent learning from scratch against one that uses transfer learning over time can illustrate all three measures [Torrey and Shavlik 2009] [Kuhlmann and Stone 2007]

7.3. Evaluating the Teaching Process

While most research focus on evaluating the performance of the learner, [Calinon and Billard 2007b] raise the issue of evaluating the teaching process as part of evaluating an imitation learning system. A number of benchmarks are proposed to evaluate different teaching methods and their educational value. The paper also discusses recommendations for demonstrating techniques that provide better training examples.

8. APPLICATIONS

Advances in imitation learning open the door for a variety of potential applications. Though commercialization and finished products of such applications may not be realized yet, we present some of the directions taken in the literature and their potential applications.

Autonomous vehicles have been a popular concept in AI from its early days. And recently self-driving cars have been gaining a lot of attention from car manufacturers and tech companies alike. The advancement in sensors and onboard computers in modern cars have rekindled the interest in producing driverless vehicles commercially. Early research in imitation learning focused on this problem, proposing a method for learning to fly an aircraft from demonstrations provided via remote control [Sammut et al. 2014] and self-driving road vehicles [Pomerleau 1995]. Since then, many researchers have directed imitation learning research to navigational or driving tasks [Abbeel and Ng 2004] [Chernova and Veloso 2007b] [Dixon and Khosla 2004] [Saunders et al. 2006] [Ross and Bagnell 2010] [Munoz et al. 2009] [Cardamone et al. 2009], not only low level control car route selection and planning [Ziebart et al. 2008]

Assistive robotics aims to provide intelligent robots that can help elderly or recovering individuals in their day to day activities. Generalization is necessary in most applications as the human partner is usually untrained, so the robot must be able to behave robustly in unseen situations. The Human-robot interaction is not limited to physical assistance. Socially assistive robots can offer help with sociological and mental problems [Feil-Seifer and Mataric 2005] [Bemelmans et al. 2012] [Tapus et al. 2009]. For robots to be effective in such a social context, their behavior must be human-like to be intuitively recognized by the human partner. Therefore, imitation learning has the potential to be an integral part in assistive robots. The same argument can be made for teaching infants using interactive robots. [Ikemoto et al. 2012] incorporates interaction in the training process to account for the changing behavior of the human partner. The robot, therefore, can adapt to the human's reactions, as the human naturally makes similar adaptations.

Electronic games is a multi-billion dollar industry, and one in which realism and immersion are important factors. There is an ever growing demand for believable arti-

ficial intelligence to enhance immersion in games [Hingston 2012]. Similar to Human-robot interaction, the possibilities are too great to consider with explicit programming [Geisler 2002], especially in modern games where the player has more freedom and control and the environments are becoming increasingly complex. A number of studies investigate the effectiveness of imitation learning in video games such as First Person Shooters [Geisler 2002] [Gorman 2009] [Thurau et al. 2004a], platformers [Ortega et al. 2013] [Ross and Bagnell 2010] [Ross et al. 2010] and racing[Ross and Bagnell 2010] [Ross et al. 2010] [Munoz et al. 2009] . These examples are applied to relatively simple problems, with limited action options and few features to consider from the environment; however, they show promising performances from imitation agents that could be extended to more complex games in the future. For instance, the study in [Gorman 2009] showed that not only are imitating agents subjectively believable, but that they also outperform other hand crafted agents. Electronic games provide a friendly platform for advancing imitation learning as they do not require additional systems such as sensors and hardware; and the cost of faults is low compared to applications where faults might endanger people or property. They can also be used as a testbed for artificial general intelligence [Schaul et al. 2011].

Humanoid robots is one of the domains most associated with artificial intelligence. It is one of the most relatable domains, because many of its potential applications are quite obvious. The premise of humanoid robots is to replace some of the workload that humans do. These problems range from specific tasks such as physical chores and housework, to generic problem solving. Since most of the required tasks are already performed by humans, humanoid robot applications are inherently suitable for the concept of learning by imitation. Humanoid robots can learn to perform actions that only utilize part of their skeleton [Billard and Matarić 2001] [Ijspeert et al. 2002b] [Vogt et al. 2014] or the entire body [Berger et al. 2008] [Calinon and Billard 2007b] [Ude et al. 2004]. Since humanoid robots are commonly used to interact with humans in a social context, it is important not only to learn how to move but also how to direct its attention to important occurrences. In [Shon et al. 2005] a robot learns where to look by imitating a teacher's gaze and learns to estimate saliency based on the teacher's preferences.

Another field of robotic applications is **automation**.It is different from the aforementioned domains in that automation is more relevant to industrial tasks while the other domains aim to create AI for personal products and services. However, automation can still be useful in domestic applications [Saunders et al. 2006]. While automation is not a new concept, learning by imitation introduces generalization and adaptability to automated tasks [Nicolescu and Mataric 2003]. This means that the robot can act robustly in unseen situations such as the introduction of obstacles or changes in the shape or position of relevant objects. Generalization diminishes the need for supervision and human intervention between small tasks. Imitation learning research done in this direction focus on object manipulation, such as sorting and assembly tasks [Saunders et al. 2006] [Pook and Ballard 1993] [Oztop and Arbib 2002] [Calinon and Billard 2007b] [Nicolescu and Mataric 2003]. Another example of automation is medical procedures. In [Mayer et al. 2008] a robot is trained to automate part of a surgical procedure from surgeons' demonstrations.

9. CONCLUSION AND FUTURE DIRECTIONS

In this survey, we review and discuss the methods and techniques used in the literature to learn by imitation. A key challenge in imitation is generalization to unseen scenarios. Due to the dynamic nature of imitation learning applications, demonstration can only cover a subset of possible states, therefore direct imitation learns from a state distribution that is different to the distribution that faces the agent when performing

the task. It is therefore essential to perform indirect learning that allows the agent to perform its current policy and refine it based on feedback from the environment or the teacher. Optimization approaches that search directly in solution space face similar limitations as they optimize for specific situations. A more adaptable approach is searching in policy space such as using evolutionary algorithms to find weights for neural networks. Indirect learning also addresses another important challenge which is imperfect demonstrations. This may arise from the capturing process or due to discrepancies between the teacher and learner or their respective environments. These challenges motivate a trend of generalizing the learning process. Creating methods that are specific to a particular setup obstructs extending these techniques to different tasks. It also makes it difficult to replicate and compare different techniques. Although imitation learning methods have developed substantially in recent years and demonstrated success in a variety of problems, the above mentioned challenges, among others, present many open research points. We now highlight key directions for future research.

— **General feature representations** can be created by automatically extracting features to represent the current state. Eliminating the need for engineered features brings us a step closer to the goal of a generic task independent learning process. While recent utilization of deep learning have produced success stories in this area, such techniques need to be examined in more realistic dynamic environments.
— **General task learning** follows a similar motivation to learning feature representations. The aim is to create a learning process with minimal expert knowledge of the task. The end goal is to have the ability to train intelligent agents solely by demonstrating a task. Since direct imitation is often no sufficient for robust imitation, apprenticeship learning might play an important role as a self-improvement method that does not require human intervention or knowledge of the task apart from the provided demonstrations.
— **Benchmarking** can greatly benefit any research topic as it allows comparing different techniques and identifying challenges more clearly. Although some benchmarking tools are available, more realistic tasks are needed to serve as standard imitation learning problems.
— **Multi-agent imitation** poses a complex learning situation that requires agents to learn cooperative or competitive strategies. Most of the current research focuses on single agents learning from a single teacher, although most real life applications will require the agent to interact with a number of humans or other agents in a shared environment [Rozo Castañeda et al. 2013].
— **Agent memory** is another area that is scarcely addressed. Memory allows the agent to perceive its actions in context and predict a likely outcome. [Pastor et al. 2013] highlight the need for such awareness and propose Associative Skill Memory (ASM) as a possible extension to learning from demonstrations and experience.

REFERENCES

Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. 2007. An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems* 19 (2007), 1.

Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 1.

Ricardo Aler, Oscar Garcia, and José María Valls. 2005. Correcting and improving imitation models of humans for robosoccer agents. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 3. IEEE, 2402–2409.

Brenna Argall, Brett Browning, and Manuela Veloso. 2007. Learning by demonstration with critique from a human teacher. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*. ACM, 57–64.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. 2008. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics* 5, 02 (2008), 183–202.

Paul Bakker and Yasuo Kuniyoshi. 1996. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*. 3–11.

JP Bandera. 2010. Vision-based gesture recognition in a robot learning by imitation framework. *Malaga: SPICUM* (2010).

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2012. The arcade learning environment: An evaluation platform for general agents. *arXiv preprint arXiv:1207.4708* (2012).

Roger Bemelmans, Gert Jan Gelderblom, Pieter Jonker, and Luc De Witte. 2012. Socially assistive robots in elderly care: A systematic review into effects and effectiveness. *Journal of the American Medical Directors Association* 13, 2 (2012), 114–120.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2, 1 (2009), 1–127.

Darrin C Bentivegna, Christopher G Atkeson, and Gordon Cheng. 2004. Learning tasks from observation and practice. *Robotics and Autonomous Systems* 47, 2 (2004), 163–169.

Erik Berger, Heni Ben Amor, David Vogt, and Bernhard Jung. 2008. Towards a simulator for imitation learning with kinesthetic bootstrapping. In *Workshop Proceedings of Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*. 167–173.

Aude Billard, Sylvain Calinon, RŘdiger Dillmann, and Stefan Schaal. 2008. Robot programming by demonstration. In *Springer handbook of robotics*. Springer, 1371–1394.

Aude Billard and Maja J Matarić. 2001. Learning human arm movements by imitation:: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems* 37, 2 (2001), 145–160.

Josh C Bongard and Gregory S Hornby. 2013. Combining fitness-based search and user modeling in evolutionary robotics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 159–166.

Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. 2015a. Reinforcement learning from demonstration through shaping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. 2015b. Policy Transfer using Reward Shaping. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 181–188.

Jonas Buchli, Freek Stulp, Evangelos Theodorou, and Stefan Schaal. 2011. Learning variable impedance control. *The International Journal of Robotics Research* 30, 7 (2011), 820–833.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 38, 2 (2008), 156–172.

Sylvain Calinon and Aude Billard. 2007a. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*. ACM, 255–262.

Sylvain Calinon and Aude Billard. 2008. A framework integrating statistical and social cues to teach a humanoid robot new skills. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Workshop on Social Interaction with Intelligent Indoor Robots*.

Sylvain Calinon and Aude G Billard. 2007b. What is the teachers role in robot programming by demonstration?: Toward benchmarks for improved learning. *Interaction Studies* 8, 3 (2007), 441–464.

Sylvain Calinon, Zhibin Li, Tohid Alizadeh, Nikos G Tsagarakis, and Darwin G Caldwell. 2012. Statistical dynamical systems for skills acquisition in humanoids. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 323–329.

Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. 2009. Learning drivers for TORCS through imitation using supervised methods. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*. IEEE, 148–155.

Nutan Chen, Justin Bayer, Sebastian Urban, and Patrick Van Der Smagt. 2015. Efficient movement representation by embedding Dynamic Movement Primitives in deep autoencoders. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 434–440.

Ran Cheng and Yaochu Jin. 2015. A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences* 291 (2015), 43–60.

Sonia Chernova and Manuela Veloso. 2007a. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 233.

Sonia Chernova and Manuela Veloso. 2007b. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 233.

Sonia Chernova and Manuela Veloso. 2008. Teaching collaborative multi-robot tasks through demonstration. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*. IEEE, 385–390.

Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 3642–3649.

Christopher Clark and Amos Storkey. 2015. Training deep convolutional neural networks to play go. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 1766–1774.

Adam Coates, Pieter Abbeel, and Andrew Y Ng. 2008. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*. ACM, 144–151.

William Curran, Tim Brys, Matthew Taylor, and William Smart. 2015. Using PCA to Efficiently Represent State Spaces. *arXiv preprint arXiv:1505.00322* (2015).

David B DAmbrosio and Kenneth O Stanley. 2013. Scalable multiagent learning through indirect encoding of policy geometry. *Evolutionary Intelligence* 6, 1 (2013), 1–26.

Hal Daumé Iii, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning* 75, 3 (2009), 297–325.

Kerstin Dautenhahn and Chrystopher L Nehaniv. 2002. *The correspondence problem*. MIT Press.

Agostino De Santis, Bruno Siciliano, Alessandro De Luca, and Antonio Bicchi. 2008. An atlas of physical human–robot interaction. *Mechanism and Machine Theory* 43, 3 (2008), 253–270.

Yiannis Demiris and Anthony Dearden. 2005. From motor babbling to hierarchical learning by imitation: a robot developmental pathway. (2005).

Kevin R Dixon and Pradeep K Khosla. 2004. Learning by observation with mobile robots: A computational approach. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 1. IEEE, 102–107.

Alain Droniou, Serena Ivaldi, and Olivier Sigaud. 2014. Learning a repertoire of actions with deep neural networks. In *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on*. IEEE, 229–234.

Haitham El-Hussieny, Samy FM Assal, AA Abouelsoud, Said M Megahed, and Tsukasa Ogasawara. 2015. Incremental learning of reach-to-grasp behavior: A PSO-based Inverse optimal control approach. In *2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*. IEEE, 129–135.

David Feil-Seifer and Maja J Mataric. 2005. Defining socially assistive robotics. In *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*. IEEE, 465–468.

Benjamin Geisler. 2002. *An empirical study of machine learning algorithms applied to modeling player behavior in a first person shooter video game*. Ph.D. Dissertation. Citeseer.

Tao Geng, Mark Lee, and Martin Hülse. 2011. Transferring human grasping synergies to a robot. *Mechatronics* 21, 1 (2011), 272–284.

Miguel González-Fierro, Carlos Balaguer, Nicola Swann, and Thrishantha Nanayakkara. 2013. A humanoid robot standing up through learning from demonstration using a multimodal reward function. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 74–79.

Bernard Gorman. 2009. *Imitation learning through games: theory, implementation and evaluation*. Ph.D. Dissertation. Dublin City University.

Daniel H Grollman and Aude G Billard. 2012. Robot learning from failed demonstrations. *International Journal of Social Robotics* 4, 4 (2012), 331–342.

Frederic Gruau and Kameel Quatramaran. 1997. Cellular encoding for interactive evolutionary robotics. In *Fourth European Conference on Artificial Life*. MIT Press, 368–377.

Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. 2007. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics* 21, 13 (2007), 1521–1544.

Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. 2014. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In *Advances in Neural Information Processing Systems*. 3338–3346.

He He, Jason Eisner, and Hal Daume. 2012. Imitation learning by coaching. In *Advances in Neural Information Processing Systems*. 3149–3157.

Philip Hingston. 2012. Believable Bots. (2012).

Chih-Lyang Hwang, Bo-Lin Chen, Huei-Ting Syu, Chao-Kuei Wang, and Mansour Karkoub. 2016. Humanoid Robot's Visual Imitation of 3-D Motion of a Human Subject Using Neural-Network-Based Inverse Kinematics. *IEEE Systems Journal* 10, 2 (2016), 685–696.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. 2013. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation* 25, 2 (2013), 328–373.

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. 2002a. *Learning attractor landscapes for learning motor primitives*. Technical Report.

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. 2002b. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the ieee/rsj int. conference on intelligent robots and systems (iros2002)*. 958–963.

Shuhei Ikemoto, Heni Ben Amor, Takashi Minato, Bernhard Jung, and Hiroshi Ishiguro. 2012. Physical human-robot interaction: Mutual learning and adaptation. *Robotics & Automation Magazine, IEEE* 19, 4 (2012), 24–35.

Shuo Jin, Chengkai Dai, Yang Liu, and Charlie CL Wang. 2016. Motion Imitation Based on Sparsely Sampled Correspondence. *arXiv preprint arXiv:1607.04907* (2016).

Kshitij Judah, Alan Fern, and Thomas G Dietterich. 2012. Active imitation learning via reduction to iid active learning. *arXiv preprint arXiv:1210.4876* (2012).

S Mohammad Khansari-Zadeh and Aude Billard. 2011. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics* 27, 5 (2011), 943–957.

Beomjoon Kim, Amir massoud Farahmand, Joelle Pineau, and Doina Precup. 2013. Learning from limited demonstrations. In *Advances in Neural Information Processing Systems*. 2859–2867.

Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* (2013), 0278364913495721.

Jens Kober, Betty Mohler, and Jan Peters. 2008. Learning perceptual coupling for motor primitives. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 834–839.

Jens Kober and Jan Peters. 2009a. Learning motor primitives for robotics. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2112–2118.

Jens Kober and Jan Peters. 2010. Imitation and reinforcement learning. *Robotics & Automation Magazine, IEEE* 17, 2 (2010), 55–62.

Jens Kober and Jan Peters. 2014. Movement templates for learning of hitting and batting. In *Learning Motor Skills*. Springer, 69–82.

Jens Kober and Jan R Peters. 2009b. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*. 849–856.

Jens Kober and Jan R Peters. 2009c. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*. 849–856.

Nate Kohl and Peter Stone. 2004. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 3. IEEE, 2619–2624.

Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. 2013. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 1061–1068.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

Gregory Kuhlmann and Peter Stone. 2007. Graph-based domain mapping for transfer learning in general games. In *Machine Learning: ECML 2007*. Springer, 188–200.

Hoang M Le, Andrew Kang, Yisong Yue, and Peter Carr. 2016. Smooth Imitation Learning for Online Sequence Prediction. *Proceedings of the 33rd International Conference on Machine Learning* (2016).

Geoffrey Lee, Min Luo, Fabio Zambetta, and Xiaodong Li. 2014. Learning a Super Mario controller from examples of human play. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 1–8.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2015. End-to-End Training of Deep Visuomotor Policies. *arXiv preprint arXiv:1504.00702* (2015).

Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In *Proceedings of The 30th International Conference on Machine Learning*. 1–9.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

Hsien-I Lin, Yu-Cheng Liu, and Chi-Li Chen. 2011. Evaluation of human-robot arm movement imitation. In *Control Conference (ASCC), 2011 8th Asian*. IEEE, 287–292.

Long Ji Lin. 1991. Programming Robots Using Reinforcement Learning and Teaching.. In *AAAI*. 781–786.

Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8, 3-4 (1992), 293–321.

Maja J Mataric. 2000a. Getting humanoids to move and imitate. *IEEE Intelligent Systems* 15, 4 (2000), 18–24.

Maja J Mataric. 2000b. Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In *Imitation in animals and artifacts*. Citeseer.

Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jürgen Schmidhuber. 2008. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics* 22, 13-14 (2008), 1521–1537.

Hua-Qing Min, Jin-Hui Zhu, and Xi-Jing Zheng. 2005. Obstacle avoidance with multi-objective optimization by PSO in dynamic environment. In *2005 International Conference on Machine Learning and Cybernetics*, Vol. 5. IEEE, 2950–2956.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783* (2016).

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and others. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

Yasser Mohammad and Toyoaki Nishida. 2012. Fluid imitation. *International Journal of Social Robotics* 4, 4 (2012), 369–382.

Yasser Mohammad and Toyoaki Nishida. 2013. Tackling the correspondence problem. In *International Conference on Active Media Technology*. Springer, 84–95.

Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. 2013. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research* 32, 3 (2013), 263–279.

Jorge Munoz, German Gutierrez, and Araceli Sanchis. 2009. Controller for torcs created by imitation. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*. IEEE, 271–278.

Jorge Muñoz, German Gutierrez, and Araceli Sanchis. 2010. A human-like TORCS controller for the Simulated Car Racing Championship. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. IEEE, 473–480.

Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. 2004. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems* 47, 2 (2004), 79–91.

Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. 2006. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*. Springer, 363–372.

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

Monica N Nicolescu and Maja J Mataric. 2003. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, 241–248.

Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. 2013. Incremental Semantically Grounded Learning from Demonstration.. In *Robotics: Science and Systems*, Vol. 9.

Stefano Nolfi and Dario Floreano. 2000. Evolutionary robotics. (2000).

Mark Ollis, Wesley H Huang, and Michael Happold. 2007. A bayesian approach to imitation learning for robot navigation. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 709–714.

Juan Ortega, Noor Shaker, Julian Togelius, and Georgios N Yannakakis. 2013. Imitating human playing styles in super mario bros. *Entertainment Computing* 4, 2 (2013), 93–104.

Erhan Oztop and Michael A Arbib. 2002. Schema design and implementation of the grasp-related mirror neuron system. *Biological cybernetics* 87, 2 (2002), 116–140.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on* 22, 10 (2010), 1345–1359.

Peter Pastor, Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, and Stefan Schaal. 2011. Skill learning and task outcome prediction for manipulation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 3828–3834.

Peter Pastor, Mrinal Kalakrishnan, Franziska Meier, Freek Stulp, Jonas Buchli, Evangelos Theodorou, and Stefan Schaal. 2013. From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems* 61, 4 (2013), 351–361.

Jan Peters and Stefan Schaal. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks* 21, 4 (2008), 682–697.

Dean Pomerleau. 1995. Neural network vision for robot driving. (1995).

Polly K Pook and Dana H Ballard. 1993. Recognizing teleoperated manipulations. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 578–585.

Bob Price and Craig Boutilier. 1999. Implicit imitation in multiagent reinforcement learning. In *ICML*. Citeseer, 325–334.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, and Ladislau Bölöni. 2016. Learning Manipulation Trajectories Using Recurrent Neural Networks. *arXiv preprint arXiv:1603.03833* (2016).

Jette Randlov and Preben Alstrom. 1998. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*. 463–471.

Nathan Ratliff, David Bradley, J Andrew Bagnell, and Joel Chestnutt. 2007a. Boosting structured prediction for imitation learning. *Robotics Institute* (2007), 54.

Nathan Ratliff, David Bradley, J Andrew Bagnell, and Joel Chestnutt. 2007b. Boosting structured prediction for imitation learning. *Robotics Institute* (2007), 54.

Saleha Raza, Sajjad Haider, and M-A Williams. 2012. Teaching coordinated strategies to soccer robots via imitation. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*. IEEE, 1434–1439.

Nizar Rokbani, Abdallah Zaidi, and Adel M Alimi. 2012. Prototyping a biped robot using an educational robotics kit. In *Education and e-Learning Innovations (ICEELI), 2012 International Conference on*. IEEE, 1–4.

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*. 661–668.

Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. 2010. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686* (2010).

Leonel Rozo, Danilo Bruno, Sylvain Calinon, and Darwin G Caldwell. 2015. Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 1024–1030.

Leonel Rozo, Pablo Jiménez, and Carme Torras. 2013. A robot learning from demonstration framework to perform force-based manipulation tasks. *Intelligent service robotics* 6, 1 (2013), 33–51.

Leonel Rozo Castañeda, Sylvain Calinon, Darwin Caldwell, Pablo Jimenez Schlegl, and Carme Torras. 2013. Learning collaborative impedance-based robot behaviors. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. 1422–1428.

Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach* (2 ed.). Pearson Education.

Claude Sammut, Scott Hurst, Dana Kedzier, Donald Michie, and others. 2014. Learning to fly. In *Proceedings of the ninth international workshop on Machine learning*. 385–393.

Joe Saunders, Chrystopher L Nehaniv, and Kerstin Dautenhahn. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. ACM, 118–125.

Stefan Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences* 3, 6 (1999), 233–242.

Stefan Schaal and others. 1997. Learning from demonstration. *Advances in neural information processing systems* (1997), 1040–1046.

Stefan Schaal, Auke Ijspeert, and Aude Billard. 2003. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B: Biological Sciences* 358, 1431 (2003), 537–547.

Stefan Schaal, Peyman Mohajerian, and Auke Ijspeert. 2007. Dynamics systems vs. optimal controla unifying view. *Progress in brain research* 165 (2007), 425–445.

Tom Schaul, Julian Togelius, and Jürgen Schmidhuber. 2011. Measuring intelligence through games. *arXiv preprint arXiv:1109.1314* (2011).

Yoav Shoham, Rob Powers, and Trond Grenager. 2003. Multi-agent reinforcement learning: a critical survey. *Web manuscript* (2003).

Aaron P Shon, David B Grimes, Chris L Baker, Matthew W Hoffman, Shengli Zhou, and Rajesh PN Rao. 2005. Probabilistic gaze imitation and saliency learning in a robotic head. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2865–2870.

David Silver, James Bagnell, and Anthony Stentz. 2008. High performance outdoor navigation from overhead data using imitation learning. *Robotics: Science and Systems IV, Zurich, Switzerland* (2008).

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

Tsung-Ying Sun, Chih-Li Huo, Shang-Jeng Tsai, and Chan-Cheng Liu. 2008. Optimal UAV flight path planning using skeletonization and particle swarm optimizer. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 1183–1188.

Huan Tan. 2015. A Behavior Generation Framework for Robots to Learn from Demonstrations. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*. IEEE, 947–953.

Adriana Tapus, Cristian Tapus, and Maja J Mataric. 2009. The use of socially assistive robots in the design of intelligent cognitive therapies for people with dementia. In *Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on*. IEEE, 924–929.

Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. 2004a. Imitation learning at all levels of game-AI. In *Proceedings of the international conference on computer games, artificial intelligence, design and education*, Vol. 5.

Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. 2004b. Learning human-like movement behavior for computer games. In *Proc. Int. Conf. on the Simulation of Adaptive Behavior*. 315–323.

Julian Togelius, Renzo De Nardi, and Simon M Lucas. 2007. Towards automatic personalised content creation for racing games. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*. IEEE, 252–259.

Lisa Torrey and Jude Shavlik. 2009. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* 1 (2009), 242.

Lisa Torrey, Trevor Walker, Jude Shavlik, and Richard Maclin. 2005. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Machine Learning: ECML 2005*. Springer, 412–424.

Aleš Ude, Christopher G Atkeson, and Marcia Riley. 2004. Programming full-body movements for humanoid robots by observation. *Robotics and autonomous systems* 47, 2 (2004), 93–108.

Andreas Vlachos. 2012. An investigation of imitation learning algorithms for structured prediction.. In *EWRL*. Citeseer, 143–154.

David Vogt, Heni Ben Amor, Erik Berger, and Bernhard Jung. 2014. Learning Two-Person Interaction Models for Responsive Synthetic Humanoids. *Journal of Virtual Reality and Broadcasting* 11, 1 (2014).

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. 2015. Maximum Entropy Deep Inverse Reinforcement Learning. *arXiv preprint arXiv:1507.04888* (2015).

Chao Zhang, Ziyang Zhen, Daobo Wang, and Meng Li. 2010. UAV path planning method based on ant colony optimization. In *2010 Chinese Control and Decision Conference*. IEEE, 3790–3792.

Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Learning deep neural network policies with continuous memory states. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 520–527.

Yudong Zhang, Shuihua Wang, and Genlin Ji. 2015. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering* 2015 (2015).

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum Entropy Inverse Reinforcement Learning.. In *AAAI*. 1433–1438.