

TU DELFT

LITERATURE REVIEW

---

# Towards Off-Policy Corrective Imitation Learning

---

*Author:*

Irene LÓPEZ BOSQUE (5051487)

*Supervisors:*

Jens KOBER

Rodrigo PÉREZ-DATTARI

Carlos CELEMIN

February 22 2021



# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Reinforcement Learning</b>	<b>3</b>
2.1	Markov Decision Process (MDP)	5
2.2	On-policy and off-policy Reinforcement Learning	5
2.3	Online and Offline	6
2.4	Experience Replay	7
<b>3</b>	<b>Imitation Learning</b>	<b>8</b>
3.1	Interactive Imitation Learning	8
3.2	On-policy and off-policy Imitation Learning	9
3.3	Imitation Learning algorithms	11
3.4	Classification of Imitation Learning algorithms as on-policy or off-policy	18
<b>4</b>	<b>Towards off-policy Corrective Imitation Learning</b>	<b>18</b>
4.1	Corrective Imitation Learning	19
4.2	COACH: Corrective Advice Communicated by Humans	19
4.3	D-COACH: Deep COACH	20
4.4	Objective: Off-policy D-COACH	21
<b>5</b>	<b>Conclusions</b>	<b>22</b>
<b>A</b>	<b>Artificial Neural Networks</b>	<b>23</b>
A.1	Feedforward Neural Networks (FNNs)	24
A.2	Recurrent Neural Networks (RNNs)	24
A.3	Convolutional Neural Networks (CNNs)	25
A.4	Autoencoders	26

## Abstract

This document presents the preliminary literature review for the master thesis *Towards Off-Policy Corrective Imitation Learning* whose objective is to improve the *experience replay* implementation of the D-COACH (Deep COorrective Advice Communicated by Humans) algorithm; D-COACH is a framework in the field of Corrective Imitation Learning (CIL), where an agent is trained with feedback that a teacher provides in the form of relative corrections. Experience replay improves the sample efficiency by allowing already collected data to be used multiple times for training. This is particularly important when training neural networks (NN) because it increases their stability helping to preserve old knowledge and thus reducing catastrophic forgetting. Furthermore, experience replay provides uncorrelated data to train the neural network which helps to generalize and to minimize overfitting. In order to use the experience replay technique, it is necessary for the algorithm to be, as it is known in reinforcement learning (RL), *off-policy*. The current version of D-COACH is *on-policy* and thus, the need to transform it into an off-policy version to fully leverage the benefits of experience replay. In this literature review, we first study the relevant theoretical framework, providing a general overview of reinforcement learning and what the terms on-policy and off-policy mean in that specific field. Then, as the main part of this review, a research in the literature is conducted for a clear definition of the terms on-policy and off-policy in imitation learning, a field where they are not as widely used as in reinforcement learning. To contribute in this aspect to the literature, two new definitions are presented and several imitation learning algorithms are classified according to these definitions. Finally, D-COACH is explained in more detail together with the proposal of how to improve the experience replay implementation by adding a new model that will predict the teacher’s feedback.

## 1 Introduction

Robots are increasingly present in our society; from factories to operating theatres, the demand for robots that are able to quickly adapt to changing environments does not stop growing. This necessity of fast adaptation to new situations makes hard-coding control strategies less suitable, and requires other types of flexible control approaches [Nardi and Stachniss, 2016]. Reinforcement learning (RL) is one of these flexible approaches where an intelligent agent tries to learn how to perform a task by maximizing a sum of rewards in a trial and error manner [Sutton and Barto, 1998]. RL techniques have been applied in successful cases such as [Mnih et al., 2013], [Silver et al., 2016], [OpenAI et al., 2019], however, these examples tend to happen in simulated environments with very specific learning tasks. This fact greatly differs from real-world situations such in robotics where, in order to learn a good policy, a robot agent requires a huge amount of data, for which it will have to perform thousands of trials [Kober et al., 2013]. This is very costly both in terms of time and the probable physical damages caused to the robot while learning [Knox and Stone, 2009]. Furthermore, many real problems are easier to demonstrate than to design a reward function for applying reinforcement learning [Kostrikov et al., 2019a]. In fact, the incorporation of human knowledge in the learning process results in dramatically more efficient methods compared to autonomous learning techniques [Attia and Dayan, 2018]. Imitation learning (IL) is the name of the field that leverages the knowledge of a teacher to improve the learning process. The simplest form of IL is known as behavioural cloning (BC) where an expert provides initial demonstrations of a desired task and then, an agent tries to imitate that behaviour via supervised learning. Behavioural cloning has two main drawbacks: First, it requires demonstrations from an expert teacher which limits the possibilities of who can train the agent. And secondly, it suffers from distribution mismatch, a problem that initiates at the moment the agent deviates from the expert trajectory causing a cascade of errors that will probably make the agent fail the task.

Interactive imitation learning (IIL) is a branch of imitation learning [Hoque et al., 2021] that deals with the aforementioned issues by allowing a teacher to supervise and teach an agent *during* its training. The nature of the feedback varies between frameworks; feedback in the form of evaluations (e.g., the TAMER framework [Knox and Stone, 2009]) inform the agent how good or bad was the action taken. This kind of evaluative feedback is easier to implement than to define a reward function and allows a faster convergence than pure autonomous learning. However, the informativeness of this kind of evaluative feedback is still limited [Najar and Chetouani, 2020], and one way to improve it is to use corrections. Corrective imitation learning (CIL) is a branch of IIL that improves the informativeness of evaluative feedback, by allowing the teacher to inform the agent whether the value of a taken action should be increased or decreased [Celemin and Ruiz-Del-Solar, 2019] and it requires less exploration compared to evaluative feedback [Najar and Chetouani, 2020].

The goal of this master thesis is to create an extension of D-COACH (Deep CORrective Advice Communicated by Humans) [Perez-Dattari et al., 2020], a CIL algorithm designed for non-expert humans that uses neural networks to approximate the policy. This extension focuses on the improvement of a key component in several deep learning algorithms, experience replay [Mnih et al., 2013], a technique that uses experiences stored in a replay buffer for training. ER endows algorithms with two main advantages, these being a higher data efficiency and the ability to train with uncorrelated data [Zhang and Sutton, 2018]. These benefits are especially useful when policies are approximated with neural networks because already collected experiences can be reused multiple times and the neural network gets more robust against locally overfitting to the most recent trajectories, a phenomenon known as *catastrophic forgetting*. The incorporation of ER into an algorithm requires that said algorithm is what in RL is known as off-policy [Sutton and Barto, 1998]. According to Sutton and Barto, off-policy learning occurs when a policy is updated with data gathered “off” that policy. The existing version of D-COACH is on-policy, because the policy is updated with data that depends on its most recent version. Despite being on-policy, current D-COACH uses experience replay for data-efficiency purposes. Given that the algorithm is on-policy, the size of the replay buffer has to be small, which works under the assumption that the data stored in the replay buffer is still valid for the current version of the policy, even if it was collected by an older version of itself. As the size of the replay buffer starts to increase, this assumption does not hold anymore and the training of the policy will most likely fail. If we want to have a more general framework, this restriction has to be removed and this is precisely the objective of this master thesis.

This literature review is organized in the following manner: In section 2, the basics of reinforcement learning are presented, a theoretical background that is necessary for later better understand imitation learning and the terms on-policy and off-policy. In section 3, after explaining what is imitation learning, several IL algorithms are gathered and classified according to our proposed definitions of on-policy and off-policy imitation learning. Lastly, section 4 focuses on D-COACH and the proposal of how to transform it into an off-policy version to better leverage the benefits of experience replay.

## 2 Reinforcement Learning

Reinforcement learning is a machine learning technique where a learner interacts with an environment with the objective of finding which of the actions it can take will maximize the cumulative

reward or *return* it will receive in the long run [Sutton and Barto, 1998]. The learner or decision maker is known as the *agent* and everything with which the agent interacts is the *environment*. How the agent behaves is defined by the *policy* normally referred as  $\pi$ . The policy that the agent follows specifies which *actions*,  $a$ , the agent takes depending on the situation of the environment and, as the learning process goes on, the policy tends to improve if the problem is well formulated. A policy can be as simple as a deterministic lookup table where for each situation the table provides an action. On the other hand, more complex stochastic policies provide a probability for each action that can be taken at each situation.

The different situations of the environment in which the agent may find itself, are the *states*. A state,  $s$ , is a sufficient summary of what is going on in the environment and when it is possible to access to all this sufficient information, the environment is said to be *fully observable*. At every time step, the agent perceives an *observation* that is a consequence of the state of the environment at that moment. Sometimes, just from the available observation, it is not possible to extract the state underneath and then, the environment is said to be *partially observable*. The goal of the agent is to select those actions that will maximize the expected discounted return  $G_t$  after  $k$  time steps, see equation 1.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (1)$$

In previous equation 1,  $\gamma$  is the *discount rate*, a parameter in the range  $0 \leq \gamma \leq 1$  that represents the value that future rewards have at the current time step  $t$ . A  $\gamma$  closer to 0 indicates that the agent favours immediate rewards whereas a  $\gamma$  closer to 1 shows an agent that strongly takes into account future rewards. At each time step  $t$ , the environment sends a *reward signal* to the agent that indicates how well the agent is performing in the *immediate* sense. On the other hand, the *value* of a state indicates how desirable is a state in the *long-run*. In other words, the value of a state  $s$ ,  $v_{\pi}(s)$ , is the expected discounted return  $G_t$  that an agent will receive over the future if it starts interacting at that state  $s$  and continues following its policy  $\pi$ . This is captured by the *value function* shown in equation 4:

$$v_{\pi}(s) = E_{\pi}[G_t \mid S_t = s] \quad (2)$$

Similarly, the *action-value function* or *Q-function*, denoted as  $q_{\pi}$  and shown in equation 3, tells us how good it is for the agent to take an action at a given state while following a policy  $\pi$ . In other words, it gives us the *value of an action* under  $\pi$ .

$$q_{\pi}(s, a) = E_{\pi}[G_t \mid S_t = s, A_t = a] \quad (3)$$

The output of the Q-function for any given state-action pair is called the *Q-value*.

When a reinforcement learning agent starts learning in an unknown environment it has to *explore* new states to discover where the best rewards are. At the same time it has to *exploit* what it already knows to choose the best actions. This is known as the *exploration-exploitation* dilemma and it is a key challenge in reinforcement learning [Sutton and Barto, 1998].

## 2.1 Markov Decision Process (MDP)

Reinforcement learning makes use of Markov decision processes (MDP) as a framework for sequential decision making problems. This framework is a simple way of representing the interaction between the agent and the environment in terms of states, actions and rewards [Sutton and Barto, 1998]. MDPs satisfy the *Markov property* that says that for a sequence of states  $s_0, s_1, \dots, s_t$ , at any time  $t$ , the next state  $s_{t+1}$  only depends on the current state  $s_t$  [Osa et al., 2018].

An MDP is defined by a tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$  where  $\mathcal{S}$  is the set of states,  $\mathcal{A}$  is the set of actions,  $\mathcal{T}(s, a, s')$  is the *transition function* and  $\mathcal{R}(s, a)$  is the *reward function*.

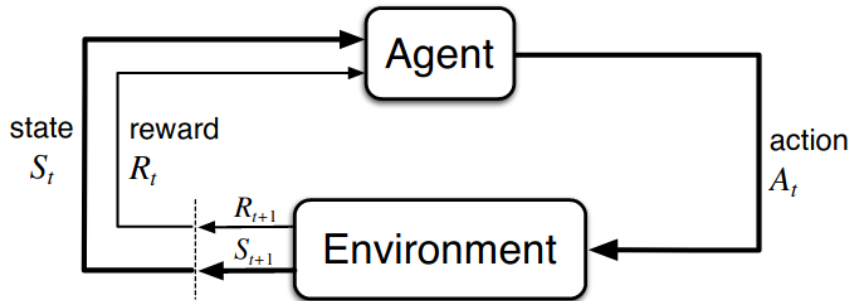


Figure 1: Reinforcement Learning

Figure 1 shows the interaction between an agent and the environment in a MDP. At time step  $t$ , the agent finds itself at state  $s_t$  and performs an action  $a_t$  on the environment. Then, the environment evolves to the next state  $s_{t+1}$  where the agent receives the reward  $r_{t+1}$ . As the learning process continues, the agent and the environment produce a sequence or *trajectory*  $\tau$  of states, actions and rewards:

$$\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \dots \quad (4)$$

## 2.2 On-policy and off-policy Reinforcement Learning

One of the objectives of this literature review is to establish a clear definition of the concepts on-policy and off-policy in the field of imitation learning. Before that, in this section we will explain their meaning of these terms in reinforcement learning, a field where they are widely used and well known. To the best of our knowledge, the first mention of the terms on-policy and off-policy in reinforcement learning is found in the book *Introduction to Reinforcement Learning* by [Sutton and Barto, 1998].

According to Sutton and Barto, on-policy methods use a single policy, the same policy that is evaluated or improved, is the one used to generate behavior. On the other hand, off-policy methods use two policies: They evaluate or improve a policy, the target policy, while using a different policy to generate the data, the behavior policy. In this case we say that learning is from data “off” the target policy, and the overall process is termed off-policy learning. [Sutton and Barto, 1998].

These definitions are the most accepted and used in reinforcement learning and therefore we will use them as reference further in this review. To better understand the terms, we are going to briefly explore two popular RL algorithms, Q-Learning and SARSA to understand what makes the former off-policy and the later on-policy.

## Sarsa and Q-learning

SARSA and Q-learning are Temporal Difference (TD) algorithms that update the knowledge of the agent at every time step by bootstrapping which means that the update is partly based on other learned estimates, without waiting to know the final outcome [Sutton and Barto, 1998]. Equation 5 shows the general update rule of these TD methods [Heitzinger, 2019], here *step size* is equivalent to learning rate.

$$\text{New estimate} \leftarrow \text{Old estimate} + \text{Step size} * [\text{Target} - \text{Old estimate}] \quad (5)$$

Q-Learning was first introduced by [Watkins, 1989] and it is considered an off-policy method because it follows a behaviour policy  $\beta$  while evaluating another policy  $\pi$ . Equation 6 shows the update rule for this method where  $s_t$  and  $a_t$  are the state and action at time step  $t$ ,  $R$  is the reward obtained for taking  $a_t$  at state  $s_t$ ,  $\alpha$  is the step size or learning rate and  $\gamma$  is the discount rate. The term  $\max_a Q(s_{t+1}, a)$  represents the best estimated value for the next state  $s_{t+1}$  and it is independent of the policy being followed.

$$Q^{\text{new}}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ R + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (6)$$

SARSA algorithm was first introduced by [Rummery and Niranjan, 1994] with the name “Modified Connectionist Q-Learning (MCQ-L)” and it was Richard Sutton who suggested, in the same publication of Rummery and Niranjan, the name SARSA, as you need to know State-Action-Reward-State-Action before performing an update. Unlike Q-learning, the policy that SARSA evaluates, is the same with which it generates samples. Equation 7 shows its update rule.

$$Q^{\text{new}}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [R + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (7)$$

When comparing both updating rules in equations 6 and 7, we can see that in Q-learning the agent learns the optimal policy by using a greedy policy when updating the Q-value (see the *max* in equation 6). This is what makes Q-learning off-policy, the fact that it can use a policy for generating behaviour different from the greedy policy with which it updates the Q-values. This is key to understand why Q-learning can make use of the experience replay technique as we will see in section 2.4.

## 2.3 Online and Offline

Reinforcement learning algorithms can also be classified according to how data is collected. Traditional RL algorithms are active or online frameworks [Levine, 2020], where an agent iteratively

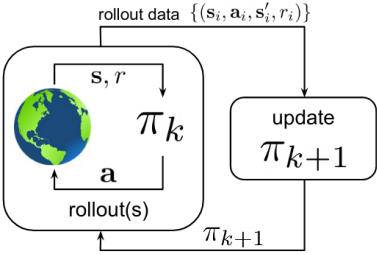
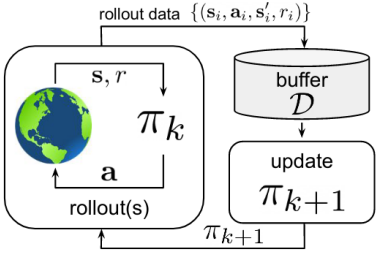
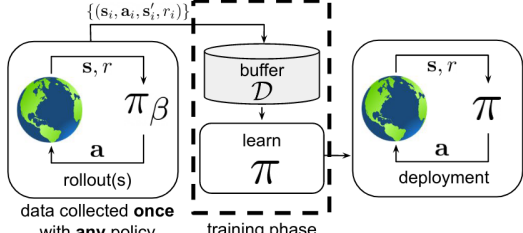
	Only use data collected by current agent	Use data collected by other agents
Data collection using current agent	<p>Online, on-policy RL</p> 	<p>Online, off-policy RL</p> 
Fixed dataset (no additional data collection)	-	<p>Offline (fully off-policy) RL</p> 

Table 1: Classification of RL algorithms, [Agarwal, 2020], [Levine, 2020] and [Levine et al., 2020].

interacts in its environment collecting experience and using it to update its policy. Online RL algorithms can be on-policy, if they use data exclusively collected by the current agent, or off-policy, see table 1; off-policy RL algorithms are able to use previous data but the agent continues interacting with the environment collecting more data that is added to a buffer and then uses the data from the buffer to improve its policy. This online approach works well in very specific and simulated environments however, for real world settings, online learning is impractical because the agent still needs to collect a diverse and huge data set at each iteration.

Offline reinforcement learning, see table 1, also called batch RL, data-driven RL or fully off-policy RL, addresses the aforementioned problem. The key idea is that a large and diverse dataset is previously collected by some behaviour policy and added into a buffer. Then using only that data, the agent has to learn the best possible policy without further interacting with the environment. With this offline framework, it is possible to apply RL to real world domains like robotics where the agent, the robot, could easily get damaged while collecting data iteratively in an online manner. The downside of offline learning resides in collecting a suitable dataset for training which for many real-world applications, data can be very expensive to acquire [Wu et al., 2017].

## 2.4 Experience Replay

Experience Replay is a replay memory technique in which the transitions,  $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ , gathered by an agent at every time step  $t$  are stored into a memory or *replay buffer*. The idea of experience replay is to randomly sample mini-batches of experience from the buffer to update the policy.



Experience replay provides several benefits. First, it is an efficient manner of taking advantage of the experience that has been already collected and use it to learn multiple times [Zhang and Sutton, 2018]. This is particularly important in the case of neural networks because it increases their stability helping to preserve old knowledge and thus reducing catastrophic forgetting [Rolnick et al., 2019]. Furthermore, experience replay provides uncorrelated data to train the neural network which helps it to generalize and to minimize overfitting to the most recent trajectories. [Klaas, 2019].

To be able to use this experience replay technique, it is necessary to have an off-policy algorithm. Off-policy RL methods continue estimating the optimal Q-value,  $Q^*$ , even if the policy that they follow changes from  $\pi_1$  to  $\pi_2$ . However, an on-policy method following a policy  $\pi_1$  will yield a Q-value  $Q_{\pi_1}$  and if the policy changes to  $\pi_2$ , it will yield  $Q_{\pi_2}$ ; the optimal Q-value in this last situation won't be guaranteed. It is important to remark that even if the experiences were collected with a single policy  $\pi$ , because the policy evolves over time, the same policy at time step  $t$ ,  $\pi_t$ , is not equal to that same policy in a later time step,  $\pi_N$ ; they are considered experiences gathered by *different policies* and therefore only off-policy methods are applicable.

### 3 Imitation Learning

Imitation learning is a problem in machine learning where an agent learns a task leveraging the knowledge of a teacher which can be a human or a more experienced agent [Torabi et al., 2019]. Imitation learning is more useful with respect to reinforcement learning when it is easier for a teacher to demonstrate or provide feedback in order for the agent to learn rather than to specify a reward function which would lead to the desired behaviour. The simplest imitation learning algorithm is called behavioural cloning where a teacher provides some initial demonstrations and the agent tries to learn a policy via supervised learning.

#### 3.1 Interactive Imitation Learning

Interactive imitation learning is a branch of imitation learning [Hoque et al., 2021], where the teacher interacts with the agent *during* its training providing feedback to improve its behaviour, see figure 2. Some authors [Osa et al., 2018] consider these interactive techniques as a special part of behavioural cloning but to make a more clear distinction it is preferable to consider them as a branch of IL.

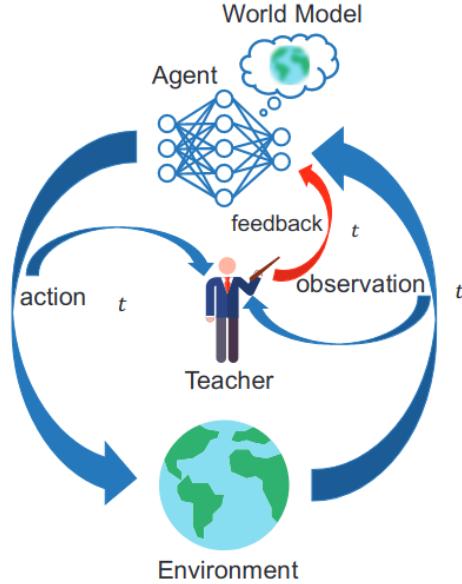


Figure 2: Interactive Imitation Learning [Pérez-Dattari et al., 2018]

The nature of the feedback varies between frameworks: Feedback in the form of evaluations (e.g., the TAMER framework [Knox and Stone, 2009]) inform the agent how good or bad was the action taken. This kind of evaluative feedback is easier to implement than to define a reward function and allows a faster convergence than pure autonomous learning. However, the informativeness of this kind of evaluative feedback is still limited [Najar and Chetouani, 2020], and one way to improve it is to use corrections. Feedback as relative corrections [Celemin and Ruiz-Del-Solar, 2019] gives name to corrective imitation learning, a branch of IIL. Corrective feedback improves the informativeness of evaluative feedback, by allowing the teacher to inform the agent whether the value of a taken action should be increased or decreased [Celemin and Ruiz-Del-Solar, 2019]. This requires less exploration compared to evaluative feedback [Najar and Chetouani, 2020]. Section 4.1 provides more details about corrective imitation learning.

### 3.2 On-policy and off-policy Imitation Learning

Section 2.2 explained the meaning of the terms on-policy and off-policy in the field of reinforcement learning. When researching the literature it is clear that, for RL, the definition provided by Sutton and Barto in 1998 is the most used. However things are not that obvious when talking about imitation learning. According to the paper *An Algorithmic Perspective on Imitation Learning* [Osa et al., 2018] the first mention to the terms on-policy and off-policy regarding imitation learning is due to [Laskey et al., 2017b]. These definitions are:

“In Off-Policy imitation learning, an agent observes demonstrations from a supervisor and tries to recover the behavior via supervised learning, an example of off-policy IL is behavioral cloning. On-policy imitation learning methods sample trajectories from the agent’s current distribution and update the model based on the data received. A common on-policy algorithm is DAgger.”

Similar definitions are found in [Lee et al., 2019a] and [Balakrishna et al., 2020]:

“On-policy imitation learning involves executing the current agent’s policy  $\pi_{\theta_i}$  in the environment allowing it to make errors and observe new states and then soliciting feedback from a supervisor on the visited states to update  $\pi_{\theta_i}$ . This is in contrast to off-policy imitation learning algorithms where policy learning is performed entirely on states from the supervisor’s trajectory distribution.”

While we agree with the previous definition of on-policy IL, regarding off-policy IL, we consider that it is easier to interpret and relate its meaning to RL, simply as the opposite of on-policy IL. Therefore our reinterpretation of the definitions is:

- **On-Policy imitation learning:** “On-Policy IL methods sample trajectories using the current agent’s policy and use that data to update that same policy.”
- **Off-Policy imitation learning:** “Off-Policy IL methods sample trajectories using a policy different from the current agent’s one.”

In [Laskey et al., 2017b], [Lee et al., 2019a] and [Osa et al., 2018] the first example of on-policy imitation learning given is DAgger (Dataset Aggregation) [Ross et al., 2011]; however, according to their own definitions we consider important to make a clarification regarding this classification. Algorithm 1 shows DAgger’s *simplest form*. For the first iteration, DAgger uses the expert’s policy  $\pi^*$  (line 2), to collect a dataset of trajectories  $D$  and then, it trains a policy  $\hat{\pi}_2$  that best mimics the expert on those trajectories. Then at iteration  $n$ , it uses  $\hat{\pi}_n$  to collect more trajectories and adds those trajectories to the dataset  $D$ . The next policy  $\hat{\pi}_{n+1}$  is the policy that best mimics the expert on the whole dataset  $D$ . In summary, DAgger uses its *current policy* to collect a dataset at each iteration and trains the next policy under the aggregate of all collected datasets [Ross et al., 2011].

---

**Algorithm 1** Simplest version of DAgger algorithm

---

```

1: Initialize  $\mathcal{D} \leftarrow \emptyset$ 
2: Initialize  $\hat{\pi}_1 \leftarrow \pi^*$ 
3: for  $i = 1$  to  $N$  do
4:   Let  $\pi_i = \hat{\pi}_i$ 
5:   Sample T-step trajectories using  $\pi_i$ 
6:   Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$  and actions given by expert
7:   Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ 
8:   Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$ 
9: end for
10: Return best  $\hat{\pi}_i$  on validation

```

---

This simple version of DAgger fits in the definition of on-policy imitation learning found in the literature because as it can be seen in lines 4 and 5 of algorithm 1, the policy that gathers the trajectories is the current agent’s policy.

However, and to better leverage the knowledge of the expert, the authors *optionally* allow the algorithm to use a modified policy  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ . Depending on  $\beta$ , which has values within the range  $[0, 1]$ , the expert policy  $\pi^*$ , is able to collect part of the next dataset by itself. If  $\beta = 1$ ,

the expert has full control when collecting the data opposite to  $\beta = 0$  that means that all the data is gathered by the current agent’s policy. The parameter  $\beta$  can be equal to  $\beta_i = p^{i-1}$  representing a decaying rate over time of the usage of the expert to collect the data.

This more generic version represented in algorithm 2 wouldn’t fit into the definition of on-policy imitation learning given by [Osa et al., 2018], [Laskey et al., 2017b], [Lee et al., 2019a] and [Balakrishna et al., 2020] due to the fact that during the first iterations, the expert’s policy has more weight than the agent’s policy when collecting the data.

---

**Algorithm 2** DAgger algorithm with stochastic mixing of the agent and the supervisor policies

---

```

1: Initialize  $\mathcal{D} \leftarrow \emptyset$ 
2: Initialize  $\hat{\pi}_1 \leftarrow \pi^*$ 
3: for  $i = 1$  to  $N$  do
4:   Let  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ 
5:   Sample T-step trajectories using  $\pi_i$ 
6:   Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$  and actions given by expert
7:   Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ 
8:   Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$ 
9: end for
10: Return best  $\hat{\pi}_i$  on validation

```

---

To summarize this section, if the teacher intervenes somehow in the gathering of the data, we consider this imitation algorithm to be off-policy.

### 3.3 Imitation Learning algorithms

In this section, we gather and briefly describe some Imitation Learning algorithms with the objective of later classify them according to the proposed definitions from the previous subsection.

#### Behavioural Cloning

Behavioural cloning (BC) [Pomerleau, 1991], one of the oldest Imitation Learning frameworks, consists on directly learning a policy, via supervised learning, given a data set of demonstrations provided by a teacher. The main problem with BC is known as *distribution mismatch* and it happens because at the moment that the learner agent deviates from the expert trajectory, it cannot recover from that failure and go back to the expert trajectory. This provokes a cascade of errors that keep growing because the agent doesn’t know how to act in those states that haven’t been visited by the expert [Attia and Dayan, 2018].

#### DART

DART (Disturbances for Augmenting Robot Trajectories) [Laskey et al., 2017a] is an algorithm close to behavioural cloning [Pomerleau, 1991]. According to its authors is an off-policy imitation

learning algorithm because, after the first expert’s demonstration, it does not require to query the expert anymore. This method works by injecting noise into the supervisor’s policy while demonstrating the task.

## **SQIL**

SQIL (Soft Q Imitation Learning) [Reddy et al., 2019] is a variant of behavioural cloning that incentivises the agent to match human demonstrations over a long horizon. SQIL uses reinforcement learning but doesn’t learn a reward function, instead, when the agent matches a demonstrated action in a demonstrated state, it receives a reward of "1" and "0" for every other behaviour.

## **ValueDICE**

ValueDICE (Distribution Correction Estimation) [Kostrikov et al., 2019b], is an off-policy imitation learning [Laskey, 2018] that minimize the KL-divergence between the expert distribution and the distribution induced by the agent when interacting with the environment in a off-policy manner.

## **Dagger**

Dagger (Data Aggregation) [Ross et al., 2011] is one of the most well-known imitation learning algorithms and it was explained in-depth in section 3.2. Many variants of Dagger exist and some of them are going to be commented on next.

### **Dagger by coaching**

Dagger by coaching [He et al., 2012], is a version of Dagger [Ross et al., 2011] where the human teacher executes actions that are within learner’s ability. This means that when the agent is at a state far from the desired state, the teacher won’t try to correct directly that difference but instead it will try to redirect the agent gradually [Attia and Dayan, 2018].

## **AggreVaTe**

AggreVaTe (Aggregate Values to Imitate) [Ross and Bagnell, 2014], is a version of Dagger [Ross et al., 2011], that learns to choose actions that minimize the cost-to-go of the expert, [Attia and Dayan, 2018]. The cost-to-go  $Q$  is the cost of executing an action in a state and continue executing a policy for the next steps. The first iteration is the same as in Dagger, then for the next iterations, at a time  $t$ , the cost-to-go  $Q$  of taking an action in a state is observed and added to the the initial dataset  $D$  together with the action and the state.

## AggreVaTeD

AggreVaTeD [Sun et al., 2017] is a differentiable version of the AggreVaTe algorithm introduced by [Ross and Bagnell, 2014] that does not perform any data aggregation. It includes two gradient update procedures: An online gradient descent and a natural gradient update similar to exponential gradient descent.

## SafeDagger

SafeDagger [Zhang and Cho, 2016], is a version of DAgger [Ross et al., 2011], that aims to reduce the burden of constantly querying the human teacher by incorporating a safety policy. This safety policy predicts the discrepancy between the teacher and the learner and it is only on those states where the discrepancy is high, that the teacher is queried.

## EnsembleDagger

EnsembleDagger [Menda et al., 2019], is an extension of DAgger where the learning agent only acts when two measures are compliant with two preset thresholds: The first measure is the *discrepancy*, which is the same as in SafeDagger [Zhang and Cho, 2016]; discrepancy represents the deviation between the agent and the expert. The second measure is the *doubt* which is the variance that indicates the familiarity of the agent with its current state; the doubt constrains the learner to only act in familiar states.

## SHIV

SHIV (Svm-based reduction in Human InterVention) [Laskey et al., 2016], is a method similar to DAgger [Ross et al., 2011] that differs from this one in the fact that the human teacher is not queried to provide labels for all the visited states but only when it is risky. The risk is described as distance to a boundary defined by a one class support vector machine.

## Hierarchical Guidance DAgger

Hierarchical guidance [Le et al., 2018] is a framework for hierarchical problems where low level sub-tasks can be identified by an expert. The authors present two settings: In the first one, hierarchical imitation learning, the teacher provides high-level feedback and only provides low-level feedback when needed. The second setting is a hybrid imitation–reinforcement learning where the teacher provides only high-level feedback and the learner uses reinforcement learning at the low level.

## **BAgger**

BAgger (Bayesian dataset Aggregation) [Cronrath et al., 2018], is an extension of DAgger [Ross et al., 2011] that aims to increase safety and reduce expert burden by querying the expert only when there is risk of not being able to imitate the expert. BAgger trains a Bayesian neural network or a Gaussian process to predict the expected error between teacher and learner.

## **SAIL**

SAIL (Safety-Aware Imitation Learning) [Xiong et al., 2019], is an extension of DAgger that aims to reduce the number of queries to the expert. Only when the confidence level of the learning agent is lower than a threshold, is the expert queried. The expert continues providing labels until the confidence in all states is again higher than a threshold. Then, an uncertainty based approach decides whether or not continue querying the expert.

## **Retrospective DAgger**

Retrospective DAgger [Song et al., 2019], is a version of the DAgger [Ross et al., 2011] algorithm, designed for combinatorial search spaces. The policy learns from its mistakes by learning from retrospective inspections of its own roll-outs.

## **HG-DAgger**

HG-DAgger (Human-Gated Data Aggregation) [Kelly et al., 2019], is a version of DAgger [Ross et al., 2011] that includes a gating function controlled by the expert; when the expert teacher detects that the agent is at a unsafe region of the state space, the expert takes control and leads the learning agent to a safe region of the state space.

## **EIL**

EIL (Expert Intervention Learning) [Spencer et al., 2020], is an algorithm similar to HG-Dagger, that allows the teacher to take the control of the agent when needed. In HG-Dagger, the labels that the human provides when taking control (explicit feedback) are used to update the data set. On the other hand, EIL not only uses this explicit feedback but also the implicit feedback which are those actions that the human does not correct because the agent is already behaving correctly. Finally, EIL also takes into account the timing, meaning when the feedback was provided.

## **FIRE**

FIRE (Failure Identification to Reduce Expert burden) [Ablett et al., 2020], is an algorithm similar to HG-Dagger that incorporates the ability of notifying the expert when the agent is at a unsafe

state. FIRE has a failure predictor that classifies expert and non-expert data and predicts when a failure may occur. When this happens, the policy stops and if the human agrees with the prediction, he/she teleoperates the agent back to a safe state.

## **IWR**

IWR (Intervention Weighted Regression) [Mandlekar et al., 2020] is an algorithm similar to FIRE, that focuses on *bottlenecks* regions, that is, those states where it is necessary a sequence of precise actions. IWR keeps two different data sets, one with data provided by the human when intervening in a trajectory (mostly during bottleneck regions), and another one for the rest of the trajectory when the human does not intervene. During training, the two data sets are equally sampled which re-weights the data distribution to reinforce those labels provided by the human during bottleneck regions, while the data sampled from the non-intervention data set keeps the policy close to previous policy iterations [Mandlekar et al., 2020] .

## **DA-RB**

DA-RB (DAgger Replay Buffer) [Prakash et al., 2020], is an extension of the DAgger algorithm [Ross et al., 2011] that incorporate an additional replay buffer with the aim of controlling, in the data set, the proportion of data provided by the human and data gathered by the agent. This buffer is said to help the policy to focus on its weaker behaviours.

## **COACH**

COACH (COrrective Advice Communicated by Humans) [Celemin and Ruiz-del-Solar, 2015], is an algorithm designed for non-expert humans where feedback is provided as a binary signal interpreted as an increase or decrease of the value of an action. The feedback is immediately executed which makes it easy for the teacher to observe the change in the behaviour of the agent and continue providing further corrections. When a sequence of corrections have the same sign, it indicates that the correction to be made has a large magnitude. Opposite, if the signal alternates signs, the teacher is trying to make smaller corrections around a certain state. This algorithm, and more specifically its deep version [Pérez-Dattari et al., 2018] are the main focus of this master thesis.

## **D-COACH**

D-COACH (Deep COrrective Advice Communicated by Humans) [Pérez-Dattari et al., 2018] is the deep learning version of the COACH algorithm [Celemin and Ruiz-del-Solar, 2015]. It uses artificial neural networks to represent the policy and includes a buffer to replay recent experiences. As already mentioned, this algorithm is the starting point of the present master thesis.



## **Advise**

Advise [Griffith et al., 2013], is a policy shaping approach where the human teacher feedback is interpreted as direct policy labels; example of feedback could be “this is right” or “this is wrong” given an action taken by the agent. Advise also takes into account that the human feedback can be inconsistent and that correct feedback is provided with a probability  $C$ . Advise uses this probability and the Bayes rule to represent the human feedback policy [Zhang et al., 2019].

## **I-SABL**

I-SABL (Inferring Strategy-Aware Bayesian Learning) [Loftin et al., 2016], which is most similar to Advise [Griffith et al., 2013] uses expectation-maximization to calculate the best action. With this method, if the learning agent takes an optimal action, the human teacher provides positive feedback, otherwise, the human provides negative feedback [Zhang et al., 2019].

## **ABLUF**

ABLUF (Adaptive Bayesian Learning with Uncertain Feedback) [He et al., 2020], is based on expectation maximization algorithms, and it is similar to I-SABL [Loftin et al., 2016]. However, whereas I-SABL assumes that the expert only provides positive feedback when the agent takes an optimal action, ABLUF models the human feedback as a probability distribution, where the probability of providing positive or negative feedback, increases or decreases with respect to the distance between the action taken and the optimal action.

## **TAMER**

In the TAMER (Training an Agent Manually via Evaluative Reinforcement) framework [Knox and Stone, 2009], the teacher is seen as a reward function that maps the actions of the agent to negative, neutral or positive feedback. This kind of feedback is called evaluative feedback because the teacher evaluates how good or bad is the action taken by the agent. This reward function replaces the rewards provided by the environment in a classical reinforcement learning problem [Zhang et al., 2019].

## **Deep TAMER**

Deep TAMER [Warnell et al., 2018] is a version of the TAMER framework [Knox and Stone, 2009] where the policy is represented with a deep neural network.

## **DQN-TAMER**

DQN-TAMER [Arakawa et al., 2018], is a combination of the TAMER framework [Knox and Stone, 2009] with Deep Q-Network (DQN). The original TAMER framework doesn't take into account the environment reward, but DQN-TAMER trains a DQN agent and a TAMER agent, and the final decision policy is a weighted average of the policies from both agents [Zhang et al., 2019].

## **Convergent Actor-Critic by Humans**

Convergent Actor-Critic by Humans [MacGlashan et al., 2017], is an algorithm inspired in TAMER [Knox and Stone, 2009] that differs from this one in that fact that TAMER interprets human feedback as a reward function independent of the agent's current policy [Zhang et al., 2019]. Contrary, Convergent Actor-Critic by Humans assumes that human feedback depends on the agent's current policy and that it should be interpreted as the advantage function that tells how much better or worse when deviating from the agent's current policy.

## **FRESH**

FRESH (Feedback-based REward SHaping) [Xiao et al., 2020], is similar to Deep-TAMER [Warnell et al., 2018] but unlike Deep-TAMER, FRESH takes into account the reward from the environment.

## **LOKI**

LOKI (Locally Optimal search after K-step Imitation) [Cheng et al., 2018], is an algorithm that has two phases: A first imitation learning phase and a second reinforcement learning phase. First, it randomly picks a number within a range and performs that number of online imitation learning steps. Then, in the reinforcement learning phase, it improves the policy with a policy gradient RL method.

## **AOR**

AOR (Adaptive On-Policy Regularization) [Lee et al., 2019b], is an algorithm that uses dynamic regret which measures performance of a policy at each iteration. Dynamic regret compares the current policy against the best it could be on its distribution with respect to the expert [Laskey et al., 2018].

## **Cycle-of-Learning**

Cycle-of-Learning [Waytowich et al., 2018], is a framework that allows to switch between multiple types of human interventions when teaching an agent. Human interactions are divided in three

categories arranged from more human control to less human control: Learning from human demonstration, learning from human intervention and learning from human evaluation. As the learning task progresses in time and the agent improves, less human intervention is required and therefore the algorithm switches to the following type of intervention technique.

### 3.4 Classification of Imitation Learning algorithms as on-policy or off-policy

Table 2 shows the classification of the previous imitation learning algorithms according to the definitions shown in section 3.2.

	<b>On-policy IL: Only data gathered by the current policy is useful for updating it</b>	<b>Off-policy: Data gathered by any agent is useful for updating the policy</b>
<b>Online sampling / Interactive Imitation Learning</b>	TAMER [Knox and Stone, 2009] DAgger (simple version) [Ross et al., 2011] DAgger by coaching [He et al., 2012] COACH [Celemin and Ruiz-del-Solar, 2015] SHIV [Laskey et al., 2016] I-SABL [Loftin et al., 2016] Convergent Actor-Critic by Humans [MacGlashan et al., 2017] Deep TAMER [Warnell et al., 2018] D-COACH [Pérez-Dattari et al., 2018] Hierarchically Guided DAgger [Le et al., 2018] DQN-TAMER [Arakawa et al., 2018] LOKI [Cheng et al., 2018] AOR [Lee et al., 2019b] ABLUF [He et al., 2020] EIL [Spencer et al., 2020] DAgger Replay Buffer [Prakash et al., 2020]	DAgger (version with $\beta$ ) [Ross et al., 2011] Advise [Griffith et al., 2013] AggreVaTe [Ross and Bagnell, 2014] Safe DAgger [Zhang and Cho, 2016] AggreVaTeD [Sun et al., 2017] BAgger [Cronrath et al., 2018] Cycle of Learning [Waytowich et al., 2018] Retrospective DAgger [Song et al., 2019] SAIL [Xiong et al., 2019] HG-DAgger [Kelly et al., 2019] EnsembleDAgger [Menda et al., 2019] FRESH [Xiao et al., 2020] FIRE [Ablett et al., 2020] IWR [Mandlekar et al., 2020]
<b>Offline sampling</b>	-	Behavioural Cloning [Pomerleau, 1991] DART [Laskey et al., 2017a] ValueDICE [Kostrikov et al., 2019b] SQIL [Reddy et al., 2019]

Table 2: Classification of Imitation Learning algorithms

## 4 Towards off-policy Corrective Imitation Learning

Section 4 concludes this review by explaining in detail the objective of this master thesis and the proposal to carry it out. The goal is to transform the corrective imitation learning algorithm D-COACH, which is the deep version of the COACH algorithm, into an off-policy version that will incorporate an improved experience replay without the restrictions present in current version of D-COACH.

## 4.1 Corrective Imitation Learning

As it was presented in section 3.1, there are different types of feedback that a teacher can provide during an interactive imitation learning training session. In this Master Thesis we are going to focus on feedback in the form of relative corrections which is the type of feedback used in D-COACH [Pérez-Dattari et al., 2018]. The usage of corrective feedback names the term corrective imitation learning which we consider as a branch of IIL.

## 4.2 COACH: Corrective Advice Communicated by Humans

COACH [Celemin and Ruiz-del-Solar, 2015] is a CIL framework designed for non-expert humans teachers where the person supervising the learning agent, provides occasional corrections when the agent behaves wrongly. This corrective feedback  $h$  is a binary signal that indicates the direction in which the executed action  $a = \pi_\theta(s)$ , should be modified. The parameters of the policy  $\pi_\theta(s)$ ,  $\theta$  are updated using a stochastic gradient descent (SGD) strategy in a supervised learning manner where the cost function  $J(\theta)$  is the mean squared error between the applied and the desired action [Wout et al., 2019]. Equation 8 shows the general update rule of the parameters  $\theta$ :

$$\theta \leftarrow \theta - \alpha \cdot \nabla_\theta J(\theta) \quad (8)$$

COACH works under the assumption that teachers are non-experts and that therefore they are just able to provide a correction trend that tells the sign of the policy error but not the error itself. To compute the exact magnitude of the error, COACH incorporates a hyperparameter  $e$  that needs to be defined beforehand, resulting in  $\text{error}_t = h_t \cdot e$  [Celemin and Ruiz-del-Solar, 2015]. The error needs to be defined as a function of the parameters in order to compute the gradient in the parameter space of the policy; thus, the error can be also described as the difference between the desired action generated with the teacher’s feedback,  $a_t^{\text{target}} = a_t + \text{error}_t$ , and  $a_t$  is the current output of the policy,  $a_t = \pi_\theta(o_t)$ :

$$\text{error}_t(\theta) = a_t^{\text{target}} - a_t \quad (9)$$

Equation 10 shows the final update rule of COACH obtained from equations 8, 9 and the derivative of the mean squared error:

$$\theta \leftarrow \theta + \alpha \cdot \text{error}_t \cdot \nabla_\theta \pi_\theta \quad (10)$$

Where  $\nabla_\theta \pi_\theta$  is the gradient of the policy,  $\frac{\partial \pi}{\partial \theta}$ , w.r.t the parameters  $\theta$ . Algorithm 3 shows the pseudocode of the basic COACH version:

---

**Algorithm 3** Basic COACH

---

```
1:  $e \leftarrow$  error magnitude
2:  $\alpha \leftarrow$  learning rate
3: while true do
4:    $s \leftarrow$  getStateVec()
5:   execute action  $a_t = \pi_\theta(s_t)$ 
6:   feedback human corrective advice  $h_t$ 
7:   if  $h_t$  is not 0 then
8:      $error_t = h_t \cdot e$ 
9:      $a_{target(t)} = a_t + error_t$ 
10:    update  $\pi$  using SGD with pair  $(o_t, a_t^{target})$ 
11:  end if
12: end while
```

---

### 4.3 D-COACH: Deep COACH

D-COACH [Perez-Dattari et al., 2020] is the “deep” version of the COACH algorithm which represents the policy with an Artificial Neural Network.

---

**Algorithm 4** Deep COACH

---

```
1: Require: error magnitude  $e$ , buffer update interval  $b$ 
2: Init:  $\mathcal{B} = [ ]$  # initialize memory buffer
3: for  $t = 1, 2, \dots$  do
4:   observe state  $o_t$ 
5:   execute action  $a_t = \pi_\theta(o_t)$ 
6:   feedback human corrective advice  $h_t$ 
7:   if  $h_t$  is not 0 then
8:      $error_t = h_t \cdot e$ 
9:      $a_{target(t)} = a_t + error_t$ 
10:    update  $\pi$  using SGD with pair  $(o_t, a_t^{target})$ 
11:    update  $\pi$  using SGD with a mini-batch sampled from  $\mathcal{B}$ 
12:    append  $(o_t, a_t^{target})$  to  $\mathcal{B}$ 
13:  end if
14:  if  $\text{mod}(t, b)$  is 0 then
15:    update  $\pi_\theta$  using SGD with a mini-batch sampled from  $\mathcal{B}$ 
16:  end if
17: end for
```

---

The current version of D-COACH already implements experience replay to be more data efficient and it does this by incorporating a memory buffer  $\mathcal{B}$  (see pseudocode 4) which the basic version of COACH lacks. During learning, tuples of recent corrections,  $(o_t, a_t^{target})$ , are stored in said memory buffer and then they are replayed. In section 2.4 it was mentioned that ER requires off-policy algorithms to work properly, but current D-COACH is on-policy because the policy is updated with data that depends on its most recent version. In this case, the replay buffer  $\mathcal{B}$  works by assuming that *recent* feedback is still being valid to update the most recent version of the policy.

Due to this assumption, the size of the buffer that D-COACH implements needs to be drastically reduced in comparison with normal ER buffers, otherwise old corrections could update the policy in undesired directions of the policy’s parameter space. On the other hand, a very small ER buffer will provoke an overfitting of the policy to data generated in the most recent trajectories, which limits the current version of D-COACH to work in short horizon problems.

#### 4.4 Objective: Off-policy D-COACH

In order to truly leverage the advantages of experience replay it is necessary to develop a novel off-policy D-COACH algorithm. To do so, it is proposed to incorporate a model that learns to predict the teacher’s feedback and include it in the D-COACH framework. This new model, that we call the Human Model and which is represented by a neural network, should learn which feedback is required for a given state/action pair. The Human Model will be updated with batches of states and actions stored in a replay buffer. Then, the model will output a batch of feedback predictions that, together with the correspondent batch of feedback provided by the human from the buffer, will update the Human Model’s weights via supervised learning, see figure 3.

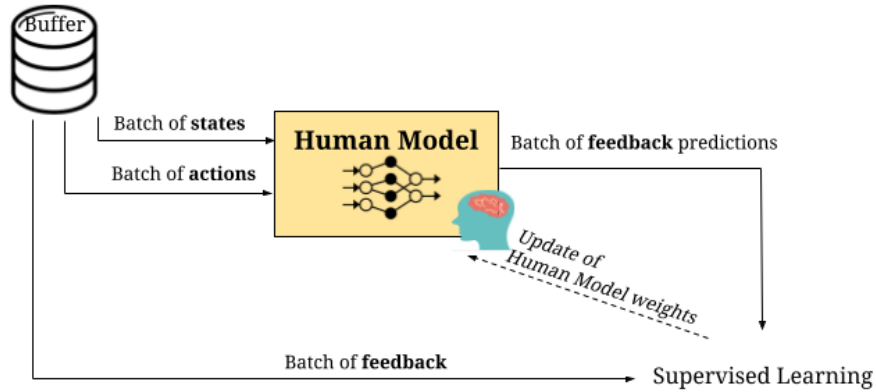


Figure 3: Update of the Human Model

Figure 4 depicts how the agent’s policy will be updated with off-policy corrections indirectly through the Human Model. The Human Model will be able to provide feedback to the newest version of the agent’s policy from a batch of states sampled from the replay buffer. This batch of states will be pass to both the agent’s policy and the human model. Then, the agent’s policy will output a batch of actions that will be fed to the human model. This batch of actions together with the output of the Human Model, will serve to update the weights of the agent’s policy. Both models, the agent and the human ones could be learned in parallel.

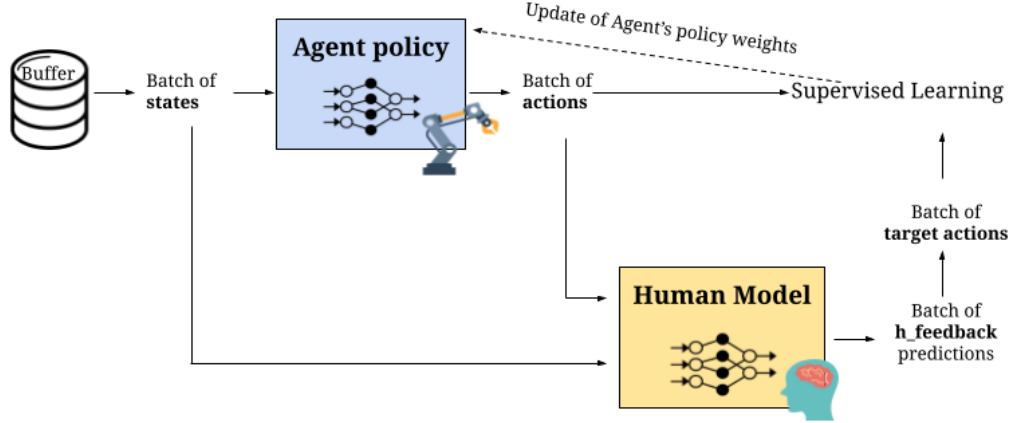


Figure 4: Update of the agent's policy

## 5 Conclusions

Sections 2, 3 and 4 present the theoretical relevant background for this master thesis. Specifically, section 2 introduces reinforcement learning and how it makes use of MDPs as a framework for sequential decision making problems. The review continues explaining what the terms on-policy and off-policy mean in RL with the help of SARSA and Q-learning algorithms. This section ends with the concept of experience replay, a technique that is a critical part of this thesis. In section 3 we introduce the concept of imitation learning, a machine learning method that leverages human knowledge in the learning process being more efficient than pure autonomous learning approaches in complex real-world problems. We present the definitions found in the literature of the terms on-policy and off-policy in imitation learning together with our interpretation with which several IL algorithms are classified. This literature review ends with section 4 where we explain in more detail the algorithm COACH and its deep version D-COACH. There, it is explained why D-COACH needs to be transformed into an off-policy algorithm to fully leverage experience replay. Finally, we present the proposal of how to carry on this transformation by introducing a new model in the D-COACH framework that will predict the human's feedback.

## A Artificial Neural Networks

An Artificial Neural Network or ANN is a computational model formed by a connection of artificial neurons arranged in structures called layers, [Graupe, 2013]. Every ANN possesses three types of layers, the input layer, the hidden layer(s) and the output layer, see figure 5.

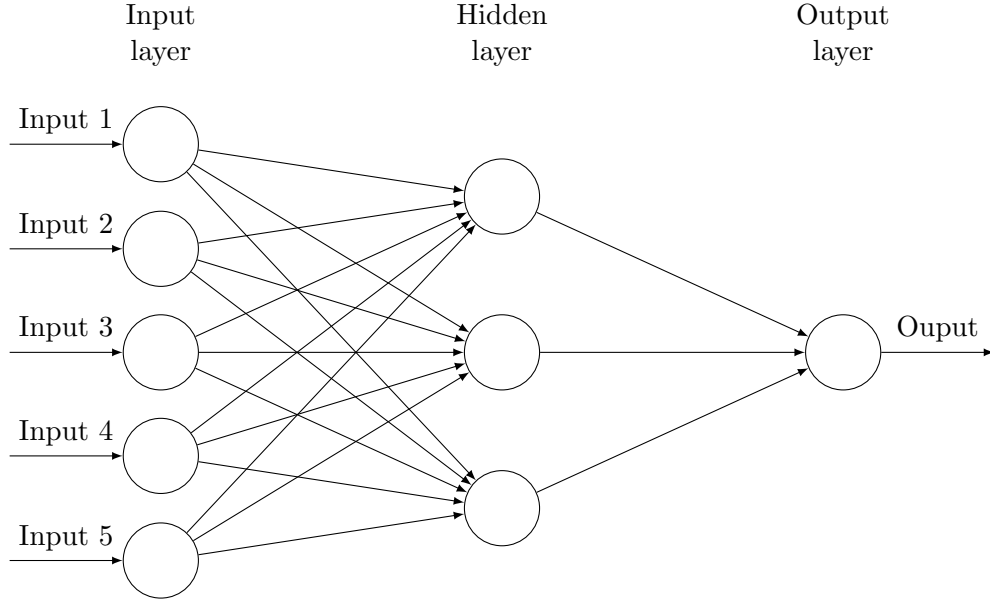


Figure 5: Diagram of an Artificial Neural Network, [Medina, 2013]

An artificial neuron, see figure 6, is the simplest element of an ANN. Similar to a biological neuron, an artificial neuron has input connections through which it receives external stimuli, the input data  $x$ , [Graupe, 2013]; With these inputs, the neuron makes a computation and generates an output value. This computation is a weighted sum of the input data where the weights  $w$  are the parameters of the model that have to be adjusted in order for the model to learn. Furthermore, there is an additional input connection to the neuron, the parameter bias  $b$  that also gets added to the weighted sum,  $wx + b$ . The final element of the artificial neuron is the activation function  $f$  that takes as input the previous weighted sum and distorts it by adding non-linear deformations,  $f(wx + b)$ .



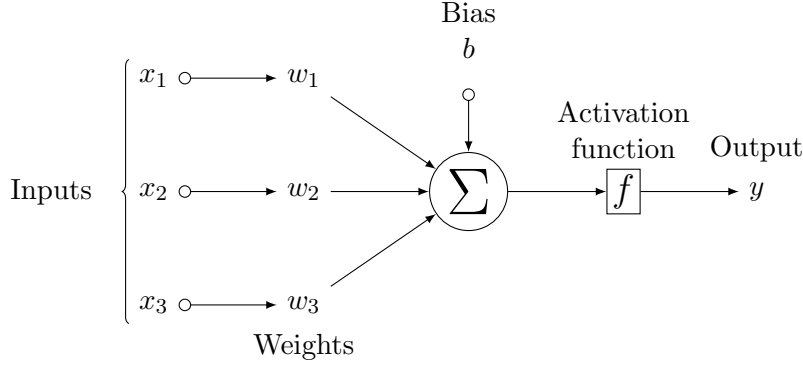


Figure 6: Neuron diagram, [Medina, 2013]

ANNs can be classified according to multiple taxonomies, one of them refers to the direction in which the data is propagated through the layers, giving as a result two main groups: feedforward neural networks (FNN) and recurrent neural networks (RNN), [Ciaburro and Venkateswaran, 2017].

### A.1 Feedforward Neural Networks (FNNs)

A feedforward neural network (FNN) is a type of artificial neural network where information flows in only one direction from the input layer to the output layer without going through any loop.

### A.2 Recurrent Neural Networks (RNNs)

Opposite to feedforward neural networks, recurrent neural networks (RNNs) propagate data both forwards and backwards through the layers endowing the model with memory. RNNs are specially useful when dealing with sequential or time dependent where some information underlays hidden, e.g., temporal information.

#### 1. Vanilla RNN

Vanilla RNNs are the simplest version of recurrent networks (Figure 7 shows an unrolled Vanilla RNN cell). The hidden state  $h$  is a parameter whose dimension is defined by the user and it is this parameter  $h$  the one that forms the recurrent connection within the cell. The hidden state at time  $t$ ,  $h_t$  is computed by adding the input data at that time,  $x_t$ , plus the hidden state from the previous time step  $h_{t-1}$ , see equation 12. It is precisely the fact of adding information from previous states, which makes the model able to remember.

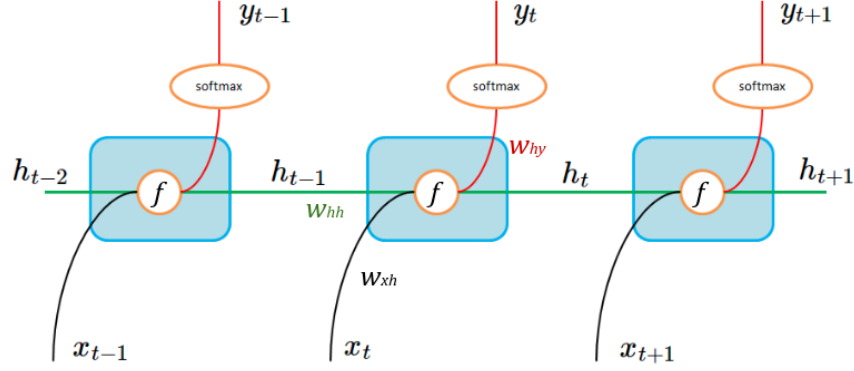


Figure 7: Vanilla Recurrent Neural Network, [Tran, 2020]

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1}) \quad (11)$$

$$h_t = \tanh\left(\begin{pmatrix} W_{xh} & W_{hh} \end{pmatrix} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}\right) \quad (12)$$

Vanilla RNNs suffer from a problem called vanishing/exploding gradient that makes them unable to work properly with big temporal horizons. Here is where long short-term memory (LSTM) come to play.

## 2. Long Short-Term Memory (LSTM)

LSTMs are an improved version of the Vanilla RNN; They are able to learn long term dependencies by maintaining not only the hidden state  $h$  but also a cell state  $c$  which is the key of these networks. Furthermore, an LSTM cell includes four gates that control the flow of information to the cell state.

## A.3 Convolutional Neural Networks (CNNs)

A convolutional neural network is a special type of ANN mostly used when the input data are images from which we want to extract information. To achieve this, filters (also called kernels) convolve on the input data producing feature maps. An example of how this convolution is done, can be seen in figure 8. The filter (blue) slides over the input image (red) and at every location, a matrix multiplication takes place giving as a result a feature map (green). The objective of these CNNs is to learn the filters in order to detect patterns in the input images.

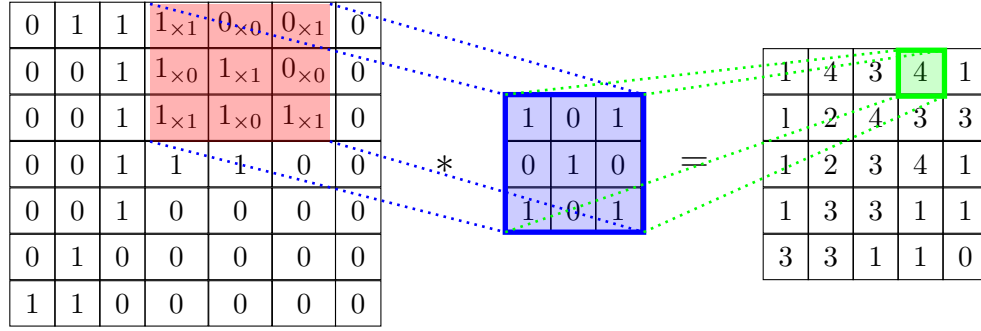


Figure 8: Example of kernel in a CNN, [Jaramillo, 2019]

## A.4 Autoencoders

An autoencoder, see figure 9 is neural network technique whose objective is to learn how to reduce the dimensionality of the input data, normally images, without losing their most important features. To do this, an autoencoder has three main parts: An encoder  $e$ , a latent space  $L$  and a decoder  $d$ . The encoder  $e(x)$ , takes the input data and encodes it into a latent space  $L$ , a space of lower dimensionality. Then, the decoder takes this  $L$  as input, and tries to reconstruct it so the output of the decoder is as similar as possible to the input  $x$ . The more similar is  $\hat{x}$  to  $x$ , the better the relevant features have been captured into the latent space.

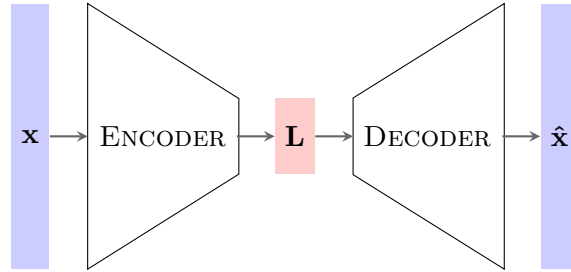


Figure 9: Autoencoder, [Allingham, 2020]

## References

- [Ablett et al., 2020] Ablett, T., Marić, F., and Kelly, J. (2020). Fighting failures with fire: Failure identification to reduce expert burden in intervention-based learning.
- [Agarwal, 2020] Agarwal, R. (2020). An optimistic perspective on offline reinforcement learning.
- [Allingham, 2020] Allingham, J. (2020). Latex-tikz-diagrams.
- [Arakawa et al., 2018] Arakawa, R., Kobayashi, S., Unno, Y., Tsuboi, Y., and ichi Maeda, S. (2018). Dqn-tamer: Human-in-the-loop reinforcement learning with intractable feedback.
- [Attia and Dayan, 2018] Attia, A. and Dayan, S. (2018). Global overview of imitation learning.
- [Balakrishna et al., 2020] Balakrishna, A., Thananjeyan, B., Lee, J., Li, F., Zahed, A., Gonzalez, J. E., and Goldberg, K. (2020). On-policy robot imitation learning from a converging supervisor.
- [Celemin and Ruiz-del-Solar, 2015] Celemin, C. and Ruiz-del-Solar, J. (2015). Coach: Learning continuous actions from corrective advice communicated by humans. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 581–586.
- [Celemin and Ruiz-Del-Solar, 2019] Celemin, C. and Ruiz-Del-Solar, J. (2019). An interactive framework for learning continuous actions policies based on corrective feedback. *J. Intell. Robotics Syst.*, 95(1):77–97.
- [Cheng et al., 2018] Cheng, C.-A., Yan, X., Wagener, N., and Boots, B. (2018). Fast policy learning through imitation and reinforcement.
- [Ciaburro and Venkateswaran, 2017] Ciaburro, G. and Venkateswaran, B. (2017). *Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles*. Packt Publishing.
- [Cronrath et al., 2018] Cronrath, C., Jorge, E., Moberg, J., Jirstrand, M., and Lennartson, B. (2018). Bagger: A bayesian algorithm for safe and query-efficient imitation learning.
- [Graupe, 2013] Graupe, D. (2013). *Principles Of Artificial Neural Networks (3rd Edition)*. Advanced Series In Circuits And Systems. World Scientific Publishing Company.
- [Griffith et al., 2013] Griffith, S., Subramanian, K., Scholz, J., Isbell, C., and Thomaz, A. (2013). Policy shaping: Integrating human feedback with reinforcement learning. *Advances in Neural Information Processing Systems*.
- [He et al., 2012] He, H., III, H., and Eisner, J. (2012). Imitation learning by coaching. *Advances in Neural Information Processing Systems*, 4:3149–3157.
- [He et al., 2020] He, X., Chen, H., and An, B. (2020). Learning behaviors with uncertain human feedback.
- [Heitzinger, 2019] Heitzinger, C. (2019). Reinforcement learning: Algorithms and convergence.
- [Hoque et al., 2021] Hoque, R., Balakrishna, A., Putterman, C., Luo, M., Brown, D. S., Seita, D., Thananjeyan, B., Novoseller, E., and Goldberg, K. (2021). Lazydagger: Reducing context switching in interactive imitation learning.

- [Jaramillo, 2019] Jaramillo, H. (2019). Drawing a convolution with tikz.
- [Kelly et al., 2019] Kelly, M., Sidrane, C., Driggs-Campbell, K., and Kochenderfer, M. J. (2019). Hg-dagger: Interactive imitation learning with human experts.
- [Klaas, 2019] Klaas, J. (2019). *Machine Learning for Finance: Principles and practice for financial insiders*. Packt Publishing.
- [Knox and Stone, 2009] Knox, W. B. and Stone, P. (2009). Interactively shaping agents via human reinforcement: The tamer framework. In *The Fifth International Conference on Knowledge Capture*.
- [Kober et al., 2013] Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.
- [Kostrikov et al., 2019a] Kostrikov, I., Nachum, O., and Tompson, J. (2019a). Imitation learning via off-policy distribution matching.
- [Kostrikov et al., 2019b] Kostrikov, I., Nachum, O., and Tompson, J. (2019b). Imitation learning via off-policy distribution matching.
- [Laskey, 2018] Laskey, M. (2018). *On and Off-Policy Deep Imitation Learning for Robotics*. PhD thesis, EECS Department, University of California, Berkeley.
- [Laskey et al., 2017a] Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. (2017a). Dart: Noise injection for robust imitation learning.
- [Laskey et al., 2017b] Laskey, M., Lee, J., Hsieh, W. Y., Liaw, R., Mahler, J., Fox, R., and Goldberg, K. (2017b). Iterative noise injection for scalable imitation learning. *CoRR*, abs/1703.09327.
- [Laskey et al., 2016] Laskey, M., Staszak, S., Hsieh, W. Y., Mahler, J., Pokorný, F. T., Dragan, A. D., and Goldberg, K. (2016). Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 462–469.
- [Laskey et al., 2018] Laskey, M., Tanwani, A., and Goldberg, K. (2018). Stability analysis of on-policy imitation learning algorithms using dynamic regret.
- [Le et al., 2018] Le, H. M., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and au2, H. D. I. (2018). Hierarchical imitation and reinforcement learning.
- [Lee et al., 2019a] Lee, J. N., Laskey, M., Tanwani, A. K., Aswani, A., and Goldberg, K. (2019a). Dynamic regret convergence analysis and an adaptive regularization algorithm for on-policy robot imitation learning.
- [Lee et al., 2019b] Lee, J. N., Laskey, M., Tanwani, A. K., Aswani, A., and Goldberg, K. (2019b). Dynamic regret convergence analysis and an adaptive regularization algorithm for on-policy robot imitation learning.
- [Levine, 2020] Levine, S. (2020). Offline reinforcement learning.
- [Levine et al., 2020] Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems.

- [Loftin et al., 2016] Loftin, R., Peng, B., Macglashan, J., Littman, M. L., Taylor, M. E., Huang, J., and Roberts, D. L. (2016). Learning behaviors via human-delivered discrete feedback: Modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30(1):30–59.
- [MacGlashan et al., 2017] MacGlashan, J., Ho, M. K., Loftin, R., Peng, B., Roberts, D., Taylor, M. E., and Littman, M. L. (2017). Interactive learning from policy-dependent human feedback.
- [Mandlekar et al., 2020] Mandlekar, A., Xu, D., Martín-Martín, R., Zhu, Y., Fei-Fei, L., and Savarese, S. (2020). Human-in-the-loop imitation learning using remote teleoperation.
- [Medina, 2013] Medina, G. (2013). Diagram of an artificial neural network.
- [Menda et al., 2019] Menda, K., Driggs-Campbell, K., and Kochenderfer, M. J. (2019). Ensembledagger: A bayesian approach to safe imitation learning.
- [Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.
- [Najar and Chetouani, 2020] Najar, A. and Chetouani, M. (2020). Reinforcement learning with human advice: a survey.
- [Nardi and Stachniss, 2016] Nardi, L. and Stachniss, C. (2016). Experience-based path planning for mobile robots exploiting user preferences. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1170–1176.
- [OpenAI et al., 2019] OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2019). Learning dexterous in-hand manipulation.
- [Osa et al., 2018] Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J. (2018).
- [Perez-Dattari et al., 2020] Perez-Dattari, R., Celemin, C., Franzese, G., Ruiz-del-Solar, J., and Kober, J. (2020). Interactive learning of temporal features for control: Shaping policies and state representations from human feedback. *IEEE Robotics Automation Magazine*, 27(2):46–54.
- [Pomerleau, 1991] Pomerleau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97.
- [Prakash et al., 2020] Prakash, A., Behl, A., Ohn-Bar, E., Chitta, K., and Geiger, A. (2020). Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11760–11770.
- [Pérez-Dattari et al., 2018] Pérez-Dattari, R., Celemin, C., del Solar, J. R., and Kober, J. (2018). Interactive learning with corrective feedback for policies based on deep neural networks.
- [Reddy et al., 2019] Reddy, S., Dragan, A. D., and Levine, S. (2019). Sqil: Imitation learning via reinforcement learning with sparse rewards.
- [Rolnick et al., 2019] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. (2019). Experience replay for continual learning.

- [Ross and Bagnell, 2014] Ross, S. and Bagnell, J. A. (2014). Reinforcement and imitation learning via interactive no-regret learning. *CoRR*, abs/1406.5979.
- [Ross et al., 2011] Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). A reduction of imitation learning and structured prediction to no-regret online learning.
- [Rummery and Niranjan, 1994] Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning using connectionist systems. CUED/F-INFENG/TR 166, Cambridge University Engineering Department.
- [Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.
- [Song et al., 2019] Song, J., Lanka, R., Zhao, A., Bhatnagar, A., Yue, Y., and Ono, M. (2019). Learning to search via retrospective imitation.
- [Spencer et al., 2020] Spencer, J., Choudhury, S., Barnes, M., Schmittle, M., Chiang, M., Ramadge, P., and Srinivasa, S. (2020). Learning from interventions: Human-robot interaction as both explicit and implicit feedback.
- [Sun et al., 2017] Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. (2017). Deeply aggravated: Differentiable imitation learning for sequential prediction.
- [Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.
- [Torabi et al., 2019] Torabi, F., Warnell, G., and Stone, P. (2019). Recent advances in imitation learning from observation.
- [Tran, 2020] Tran, T. (2020). Creating a text generator using recurrent neural network.
- [Warnell et al., 2018] Warnell, G., Waytowich, N., Lawhern, V., and Stone, P. (2018). Deep tamer: Interactive agent shaping in high-dimensional state spaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- [Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King’s College, Cambridge, UK.
- [Waytowich et al., 2018] Waytowich, N. R., Goecks, V. G., and Lawhern, V. J. (2018). Cycle-of-learning for autonomous systems from human interaction.
- [Wout et al., 2019] Wout, D., Scholten, J., Celemin, C., and Kober, J. (2019). Learning gaussian policies from corrective human feedback.
- [Wu et al., 2017] Wu, Q., Wu, H., Zhou, X., Tan, M., Xu, Y., Yan, Y., and Hao, T. (2017). Online transfer learning with multiple homogeneous or heterogeneous sources. *IEEE Transactions on Knowledge and Data Engineering*, 29(7):1494–1507.
- [Xiao et al., 2020] Xiao, B., Lu, Q., Ramasubramanian, B., Clark, A., Bushnell, L., and Pooven-dran, R. (2020). Fresh: Interactive reward shaping in high-dimensional state spaces using human feedback.

- [Xiong et al., 2019] Xiong, B., Wang, F., Yu, C., Liu, X., Qiao, F., Yang, Y., and Wei, Q. (2019). Learning safety-aware policy with imitation learning for context-adaptive navigation. *CEUR Workshop Proceedings*.
- [Zhang and Cho, 2016] Zhang, J. and Cho, K. (2016). Query-efficient imitation learning for end-to-end autonomous driving. *CoRR*, abs/1605.06450.
- [Zhang et al., 2019] Zhang, R., Torabi, F., Guan, L., Ballard, D. H., and Stone, P. (2019). Leveraging human guidance for deep reinforcement learning tasks. *CoRR*, abs/1909.09906.
- [Zhang and Sutton, 2018] Zhang, S. and Sutton, R. S. (2018). A deeper look at experience replay.