INTERACTIVE AND UNCERTAINTY-AWARE IMITATION LEARNING: THEORY AND APPLICATIONS

Manfred Ramon Diaz Cabrera

A thesis
in
The Department
of
Engineering and Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

September 2018

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Manfred Ramon Diaz Cabrera**

Entitled: **Interactive and Uncertainty-aware Imitation Learning: Theory and Applications**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Yuhong Yan (Chair)
Dr. Tien Bui (Examiner)
Dr. Charalambos Poullis (Examiner)
Dr. Thomas Fevens (Supervisor)
Dr. Liam Paull (Co-supervisor)

Approved _____
Lata Narayanan, Chair of Department

August 24th, 2018 _____
Dean Mourad Debbabi
Faculty of Engineering and Computer Science

Abstract

Interactive and Uncertainty-aware Imitation Learning: Theory and Applications

Manfred Ramon Diaz Cabrera

Living entities have an innate ability to replicate others' behaviour. As this mechanism helps with overcoming time, mobility and resources constraints in learning new abilities, it is not surprising then that the Imitation Learning framework has played a vital role in many AI systems. In the context of machine learning, Imitation Learning algorithms have been used to infer optimal behaviour for a task using traces of the execution performed by another expert agent. This paradigm has the potential to apply to any setting where an expert's demonstrations are available.

The first part of the present work develops an example of a system where imitation learning principles were applied to the problem of visually impaired people guidance at intersections. As an indirect learning method to transfer skills among intelligent agents, imitation learning techniques helped with capturing the knowledge of sighted individuals into a solution for helping blind individuals with the task of intersection crossing. A system of this kind has the potential to change the lives of its users as it aids their mobility and exploration capabilities.

However, in order to deploy a system of this kind, it is required to guarantee that a policy derived from machine learning-based methods can consistently perform in familiar environments and safely react to the unknown. Hence, the second part of this work is devoted to the development of a theoretical and experimental framework to improve safety on the Imitation Learning process through interactivity and uncertainty estimation. Uncertainty-aware Interactive Imitation Learning algorithms will help the derivation of policies that can guarantee safety in AI systems, thus expanding the range of areas where they can be applied.

Acknowledgments

Through the extension of my Master's studies and before, I have received the help and encouragement of numerous people.

I have been lucky enough to find the mentorship of two incredible supervisors in Professor Thomas Fevens and Professor Liam Paull. First and foremost, I am eternally grateful to both for trusting in my abilities more than I do it myself. To any graduate student feeling trust and confidence from his advisors is an invaluable asset. At an academic level, Professor Paull and Professor Fevens have provided me with careful guidance even when they have given me the precious opportunity to explore subjects beyond their areas of comfort. I have to acknowledge their patience and their willingness to discuss many non-earth-grounded ideas that I have come up over these years. They have always offered me mature and adequate advice that will shape my future as a researcher.

Being here today is also the outcome of my years of work at Gomentr. My colleagues and friends there supported me and encouraged me to pursue this Master's degree selflessly. At Gomentr, I had the satisfaction to work with incredibly talented people like Brad Gaulin and Wayne Morris who impregnated in me the Canadian way of behaving through their daily dedication and passion. More importantly, I would like to thank my mentor, colleague and friend there Maurice Tadros to whom I am forever grateful for all the thoughtful pieces of advice he has given me over these years.

I have been blessed enough to have researched in two world-class labs. I would like to acknowledge the contribution of all my colleagues from Shared Reality Lab at McGill's Centre for Intelligence Machines lead by Professor Jeremy Cooperstock. Professor Cooperstock gave me not only the opportunity to conduct research at his lab but also the possibility to meet incredible researchers and persons like Roger Girgis and Pascal Fortin. With the little time for social life I have had over the course of these two years, I am lucky enough to call Pascal and Roger friends. With Roger, I shared the blessing to start the unclear path of Machine Learning research at the same time. I would like to believe that we have helped each other in this endeavour. These acknowledgements have given me the opportunity to praise them both for making me a better researcher and person. During my visiting period in the Montreal Institute for Learning Algorithms at the University of Montreal, I would like to thanks to all the members of the Robotics Group. Vincent, Krishna, Breandan, Maxime, Florian, Nithin, Bhairav and Zihan have provided me with their insightful comments and ideas. These ideas have nurtured and polished my work over the past eight months, and I am sure I will continue having them as colleagues for the years to come.

None of this would have been possible without the education and the support that my parents

Virginia and Ramon, and all my family, have given me my entire life. I want to dedicate this thesis and the Master's degree to their perseverance and their encouragement to always shoot to the moon.

Finally, I would like to dedicate this thesis to my wife, Krysia:

I would like to thank you for being my sanity and my strength in these two long years, for always placing my interest over yours, my comfort over your sacrifice, and there will not be enough paragraphs, pages, or books I can write to express all my gratitude for what you have done.

Thank you, Pelusita!

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Imitation plays a significant role in the development of skills in the early stages in the lives of humans, and animals in general [17, 16]. Learning by imitation is a high-level social ability. Living entities develop, or have innate, a behavioural skill to replicate others' behaviour that leads to a similarity in the ways two or more individuals act in response to a particular stimulus originated by the world that surrounds them [35]. Copying others is a mechanism that helps with overcoming time, mobility and resources constraints in learning new abilities. While imitating, self-experiencing a task is not a precondition to absorb a new set of abilities through the observation and reproduction of others' conduct. Thus, learning by imitation is an indirect and efficient mechanism that speeds up the absorption of a new ability.

Moreover, imitative behavioural skills serve as a medium to decrease the *learner*'s uncertainty while acting under novel or uncertain contexts in the environment, a reduction explained by the presence of a trusted source of knowledge: the *expert*, an agent that holds the knowledge to act optimally (or near-optimally) in a particular task.

IL has proven to be an efficient method to find optimal (or near-optimal) control policies when a formal specification of a task is challenging to design [3]. Learning behaviour from an expert's demonstrations has expedited transferring to autonomous systems the knowledge of tasks that, although trivial from a human perspective, are particularly difficult to specify in control terms. While earlier systems relied on mathematically-developed routines [3], the paradigm of deriving control parameters from raw sensorial inputs has significantly reduced the amount of engineering behind intelligent systems [83, 13, 63], easing their adoption in an increasing number of domains.

1.1 Contribution

An example of how Imitation Learning applies to real-life problems is presented in Chapter 3, where the problem of guiding visually impaired individuals while crossing street intersections is tackled through Imitation Learning techniques. As an indirect learning method to transfer skills among intelligent agents, imitation learning techniques helped with capturing the knowledge of sighted individuals into a solution for helping blind individuals with the task of intersection crossing.

Previous methods rely on geometric assumptions or on the detection of specific features that could not be reliably guaranteed to exist in all intersections. Also, the results were obtained from single monocular images from a smartphone's camera, contrasting with previous approaches to the problem that used a wide array of sensors and computing capabilities that render those earlier systems impractical in realistic scenarios. A system of its kind has the potential to change the lives of its users as it aids their mobility and exploration capabilities in unfamiliar environments.

However, guaranteeing safety while interactively training or deploying systems of this kind, requires ensuring that a policy derived by the machine learning-based method can consistently perform in familiar environments and safely react to the unknown. This limitation motivates the research presented in Chapter 4. To successfully disseminate IL approaches, it is necessary to develop of a theoretical and experimental framework that incorporates safety in the Imitation Learning process. The Uncertainty-Aware Policy Mixing and Sampling (UPMS) algorithm proposed here is an initial attempt in this direction. Through interactivity and uncertainty estimation, it is possible to ensure that training Interactive Imitation Learning algorithms is constrained into safety guarantees defined by the expert's safety boundaries. Additionally, UPMS reduces the number of required expert queries (or interventions) when compared with state-of-the-art algorithms, which further alleviates the burden current IL methods pose over the demonstrator.

It is the position of this work that, Uncertainty-aware Interactive Imitation Learning algorithms are a fundamental step towards the derivation of policies that can guarantee safety in AI systems not only during training but also while deployed. This will make them fully applicable in safety-critical applications.

1.2 Outline

The present work is divided as follow. Chapter 2 provides an overview on the theoretical foundations of Imitation Learning through the framework of Markov Decision Processes while clearly defining the problem, the solution methods and challenges IL has faced for his application to intelligent systems. This chapter also offers an in-depth overview of Direct Policy Derivation methods in Imitation Learning (Section 2.2), a central body of work required for the understanding of several applications of IL (like the one presented in Chapter 3), and the UPMS algorithm. Chapter 3 presents a published work [37] submitted to the *5th International Workshop on Assistive Computer Vision and Robotics* at the International Conference on Computer Vision (ICCV) 2017. This work is an integral part of the research conducted by the author while visiting the Center for Intelligent Machines at McGill University and developed jointly with Roger Girgis, Jeremy Cooperstock and Thomas Fevens. Chapter 4 is an unpublished manuscript and constitutes the results of the author's work while visiting the Montreal Institute for Learning Algorithms (MILA) under the supervision of Professor Liam Paull and was developed jointly with Professors Liam Paull and Thomas Fevens.

Chapter 2

Imitation Learning

Imitative skills as a cognitive mechanism have not always been well understood even in the areas of ethology and psychology that initially originated the notion of Imitation Learning, and that have been studying the idea for the past 40 years [36]. Billard [19] shows one of the earliest attempts to structure the theory behind system specification through imitation learning by the decomposition into three aspects: attentional, functional and representational. Each of these items defines how and what the *learner* should observe, what is the proper level of abstraction for specifying the *teacher*'s behaviour, and which model should be used to represent the mapping from the sensorial stimuli to the agent actions. It has been this decomposition into a multi-step approach what has allowed to successfully apply this paradigm to various fields. As an in-depth literature survey is outside of the scope of this chapter, the reader may find in [13][63] two comprehensive and contemporary surveys on the state-of-the-art theory and applications of the subject. Both reviews assert how the field has evolved with the rise of novel machine learning models (e.g., Deep Neural Networks) as state-of-the-art methods and algorithms have integrated this novel learning paradigms into offering formal guarantees on the autonomy of the *learner*.

This chapter formalizes the problem of *Imitation Learning (IL)* and describes state-of-the-art models, algorithms, and challenges that Imitation Learning application has faced for the last three decades.

2.1 Problem Statement

Before establishing a mathematically formal definition of the problem of imitation learning, it is imperative to disambiguate its use. The term *imitation learning* has been widely but not evenly used across the literature [13][63] to refer to the process of extracting optimal behaviour of a task using traces of the execution performed by another agent. Here, concepts like *behavioral cloning* or *apprenticeship learning* [89] are considered variations of the IL method. Thus, and henceforth, the term *imitation learning* encompasses direct and indirect methods of learning control policies from demonstrations and subsumes under its scope the application of supervised learning and reinforcement learning techniques to perform this derivation. More precisely, the reference to *imitation*

learning used in the present work is closer to the notion of *learning from demonstrations* introduced by [13] than to any other previously used in the literature. Maybe, the term *demonstration* emphasizes the intentionality behind the expert's behaviour to demonstrate a task while imitative learning encompasses learning from the expert whether its sequence of actions is intentionally demonstrating the task or not.

Similarly, to model the imitation learning problem, two assumptions are necessarily made. Firstly, both the *learner* and the *expert* are decision-making agents observing or interacting with their environments (usually shared). In the same way, the optimality (near-optimality) of the expert's actions: *the observed behaviour is the result of the expert trying, but not necessarily succeeding (non-optimal expert), to maximize a measurement of its efficiency (a reward signal).* The problem of learning to imitate a *teacher* behaviour requires to establish a mathematical abstraction capable of modelling the interactions of the agents with the environment. As discussed before, as IL has mainly been applied to learn robotic control parameters (at different granularity levels) from the perception pipeline (e.g., cameras, laser scanners, radars) [63][133], a desirable characteristic of a model for imitative behavior has to be the ability to handle the notions of states (or observations) and actions at different levels of abstraction.

2.1.1 Markov Decision Processes

Markov Decision Processes (MDP) have played an instrumental role in modelling sequential decision-making problems when the outcomes are uncertain [102]. As a modelling tool, MDP allows the representation of a variety of tasks at a level of abstraction that has been flexible enough to be applied in a variety of domains like finance and investment, epidemics control, or sports, to name a few [132]. MDP are also extensively used in stochastic optimal control to specify the interaction of a goal-oriented agent with its environment under uncertainty. Regarding learning algorithms, MDP is fundamental to the development of the theory of *reinforcement learning*, where it serves as the basis to describe the agent-environment interface [125]. Thus, the theoretical formulation that will result in this section bases its procedure on the specification of Imitation Learning as an MDP structural estimation problem.

Modelling the expert's behaviour and its interaction with the environment requires the definition of the boundaries between these two entities. In a Sequential Decision-making Process (SDP), decisions are made by an agent after receiving information about the environment state. The points in time where decisions are made are usually called *decision epochs* and represented by the variable t that identifies the decision epoch under analysis. The time in an MDP is usually treated as a discrete quantity although it can also be considered a continuous one.

The cardinality of the set of decision epochs T defines an essential property: the *horizon* of the decision problem. A task is called to have *infinite horizon* if $|T| = \infty$ or *finite horizon* if $|T| = n$. Characterizing the horizon plays a significant role in solution methods and their guarantees of convergence. However, as this property holds no substantial relationship to IL methods, an in-depth analysis of these methods for infinite and finite horizon tasks is out of the scope of the present work ([38] [102]). The notion of discrete time induces a stepwise characterization of the interaction

between the environment and an agent.

Known in robotics literature as the *sense-think-act* cycle [121] [127] (Figure 2.1a), the interaction between an agent and its environment have two major components: states (s_t) and actions (a_t). In MDP, this cycle is generally described by a set that records this interaction through time called *history* and that is denoted by H_t [75]. The *history* constitutes a sufficient statistic to explain both the agent's and the environment's successive behaviour. However, the dimensionality of this set makes it impractical to use it in the representation of any real-life problem. Hence, a set of realistic assumptions are required to reduce the complexity of the analysis of any MDP. The history of the system describes what happens at each decision epoch: the agent senses the environment and extracts relevant information to make a decision.

An essential component of this interaction is the expression of an agent decision: the *action*. Agent actions show the intent an agent has to influence the dynamics of the environment -Definition 2.1-, an intent that is intrinsically related to the agent's goal. Consequently, the set of possible actions A expresses the abilities the agent could have to impact the environment.

Definition 2.1. *An action $a_t \in A$ is the control signal an agent is capable of emitting to influence the dynamics of the environment as it evolves.*

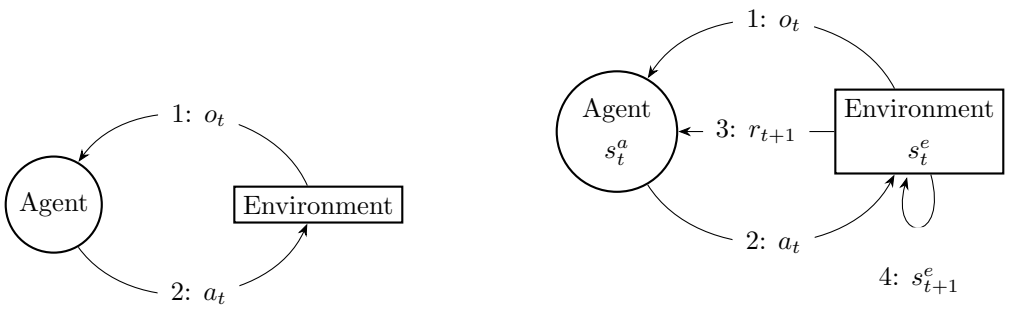
During this interaction, both the agent and the environment hold internal representations that ultimately defines its behaviour. The concept of *state* is introduced to define the data each of them use to pick their next reactions to changes. In the Markov Decision Process formulation, it is important to separate the concepts of *environment state* (s_t^e) and *agent state* (s_t^a). The *environment state* (s_t^e) defines the internal representation the environment uses to define how it transitions after the agent executes an action. The decision processes where an agent observation of the environment is equal to the environment internal state $o_t = s_t^e = s_t^a$, the decision processes are commonly called fully-observable. By contrast, when $o_t \neq s_t^e$ the decision processes are called partially-observable decision processes.

Then, an agent's state $s_t^a \in S$ encloses the information an agent perceives from the environment and it is an abstraction that only contains the relevant information the agent can observe and which it uses in its interaction with the environment. The agent state representation is usually created from a function of the history $s_{t+1}^a = f(H_t)$ as long as this function constitutes sufficient statistics for any decision. As mentioned above, as it is usually impractical to maintain the history in its entirety, some assumptions must be made to reduce the complexity of the SDP. Commonly, the *Markov assumption* is used to handle the representation of the states. When a state has sufficient information to predict the dynamics at the next decision epoch it is called an *information state* or *Markov state* -Definition 2.2.

Definition 2.2. *A state s_t is an information state (a.k.a Markov state) if and only if:*

$$p(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0) = p(s_{t+1}|s_t, a_t) \tag{1}$$

In a nutshell, the Markov property of the history and the agent state implies the future is independent of the past given the present. This assumption relaxes the complexity of a sequential decision-making process by reducing it to a more tractable probabilistic model.



(a) The *sense-think-act* cycle.

(b) The *sense-think-act* cycle in an optimal control formulation.

Figure 2.1: The percept-act cycle between an agent and its environment.

In this agent-environment interaction, the environment is usually capable of sending the agent an evaluative measure of its performance. This scalar signal is called the *reward* (or the cost). Hence, Direct Learning Methods (Reinforcement Learning) assume that the goal of an agent is to maximize the cumulative reward it obtains over the horizon of the task [125]:

$$G_t = R_1 + R_2 + \ldots + R_T \tag{2}$$

where R_t is the return after every decision epoch t. For infinite and continuing tasks, to obtain the formulation described in Equation 3 a polynomial-based on a *discount rate* γ is usually introduced to Equation 3 to weight earlier returns higher than delayed ones:

$$G_t = R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots + \gamma^{T-1} R_T \tag{3}$$

All in all, for any goal-oriented task at hand, the rewards represent the agent behaviour's objective. This property makes the reward generation function the most succinct representation of the task [113].

With all its elements being defined, a Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ is a finite set of states and $\mathcal{A}$ represents a finite set of actions. The system dynamics are modelled through $\mathcal{P}$, an state transition probability distribution of the form $\mathcal{P}^a_{s\hat{s}} = \mathbb{P}[S_{t+1} = \hat{s} | S_t = s, A_t = a]$. Also, an MDP includes a reward function $\mathcal{R}$ as the expected discounted return $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right]$, a discount controlled by $\gamma \in [0, 1]$ the discount factor.

The notion of expected reward at a certain state associates this state with a function known as a *value function* of the state defined by:

$$v(s) = \mathbb{E}\big[G_t | S_t = s\big] \tag{4}$$

This value defines how good the state is concerning the expected future reward obtained from the state by following specific behaviour. It is this behaviour that an agent must determine while learning how to solve an MDP.

A policy $\pi : S \to A$ is then a decision rule that specifies the actions an agent executes at each decision epoch. By having $v(s)$ as a measure of the value of the state an agent transitions to, it is possible to define a partial ordering among the policies over the policy space Π on an MDP. Thus, a policy $\pi \geq \pi'$ (is better) if $\forall s \in S, v_\pi(s) \geq v_{\pi'}(s)$. Then, it is guaranteed that there exists in any MDP a policy $\pi^* \in \Pi$ that it better or equal to all other policies in Π. This policy is usually called an *optimal policy* and determining its structure is the objective of an MDP solution algorithm.

2.1.2 Learning by Imitation

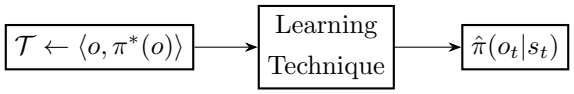
As discussed in the introduction of this chapter, imitation learning is a problem in which a learner agent tries to find an optimal or near-optimal control policy for a task given a set of trajectories of an *expert-on-the-task* behaviour. From the perspective of MDP, imitation learning attempts to solve the control problem on an MDP where one or more components of the decision process are not accessible. From this follows the definition of *imitation learning* attained for the remainder of this work:

Definition 2.3. *Learning by imitation is the process to derive an optimal or near-optimal control policy from a provided set of traces (trajectories) of the expert's policy.*

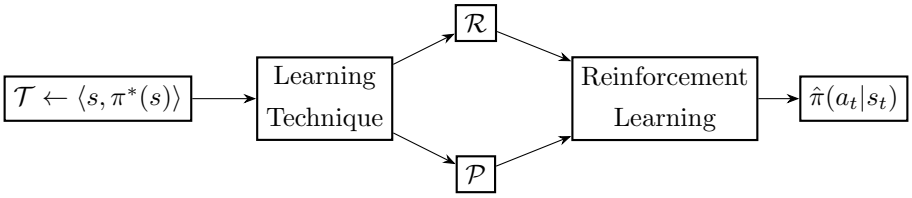
If a finite set $\mathcal{T}$ of traces of the expert's policy π^* execution is defined by $\mathcal{T} = \{(o_0, a_0), \ldots, (o_N, a_N)\}$, the goal is to find the policy $\pi(a_t|o_t)$ so that this policy optimally or nearly-optimally solves a given MDP. It is important to note that, if all components of the MDP were known, the problem of imitation learning would reduce to a Reinforcement Learning problem. In this scenario, the optimal policy can directly be recovered by interacting with the environment given that, as discussed before, the reward is the most succinct representation of the task at hand [113].

Hence, the application of imitation learning methods is relevant when either or both, the reward or the dynamics (transition model) of the environment are not known [3]. From this, three main scenarios can be identified. A Markov Decision Process without Reward (MDP-$\mathcal{R}$), a Markov Decision Process without Dynamics Model (MDP-$\mathcal{P}$) and a Markov Decision Process without Dynamics and Reward (MDP-$\mathcal{RP}$). This distinction has characterized the evolution of the solution methods for imitation learning. Learning algorithms for deriving control policies from trajectories of the optimal policy can be divided into two major subgroups: direct policy derivation (Figure 2.2a) and indirect policy derivation (Figure 2.2b) methods.

Direct Policy Derivation methods (DPD) have traditionally used a supervised learning approach for deriving the optimal policy without taking into account the internal representation of the problem (MDP-$\mathcal{RP}$). The expert's policy is recovered directly from the observation space and mapped into the learner's action space. Historically, DPD was employed to solve highly-complex tasks, like driving [101], even though no formal guarantees of their performance were given at the time. This issue limited their application to real-life, completely autonomous systems, for almost a decade. The work presented by Ross and Bagnell [107] offered for the first time a formal cost-based analysis of using purely-supervised learning as a technique to directly derive the expert's policy. This seminal paper served as the basis for exploring the range of methods that are referred here as DPD.



(a) Direct Policy Derivation



(b) Indirect Policy Derivation

Figure 2.2: Direct and Indirect Policy Derivation schemas based on the framework presented in [13].

In contrast, Indirect Policy Derivation (IPD) algorithms focus on the structural estimation of MDP [113]. IPD find the missing components of the decision process (MDP-$\mathcal{R}$ or MDP-$\mathcal{P}$) to solve the MDP control problem and later uses Reinforcement Learning to recover the expert's policy. The first formal definition of *Inverse Reinforcement Learning*(IRL) is given by Russell [113] and posteriorly extended by Ng and Russell [89]. Notably, Russell [113] offers an initial argument against the DPD methods by stating that the reward of an MDP is a more compact, robust, learnable and transferable representation of the task at hand. However, the exploration of algorithms for Inverse Reinforcement Learning has shown that even though IRL as a method is a theoretically sound approach, obtaining the reward function via IRL is not trivial.

From both methods, DPD has been more successfully applied to real intelligent and physical systems, including the work presented in Chapter 3. Hence, the following section presents an in-depth analysis of DPD methods, their formal performance guarantees, and the limitations they present for practical applications.

2.2 Direct Policy Estimation

As discussed earlier, a control policy in an MDP can be a deterministic function in the form $\pi(s) : S \rightarrow A$ that maps each state to an optimal action in such state. If presented with traces of an optimal policy π^* in the form of pairs $(s, \pi^*(s))$, directly estimating a parametric representation of such a mapping function is a straightforward approach to estimate a policy given traces of its execution. In consequence, the application of supervised learning has been one of the most successful and widely used techniques to learn by imitation.

2.2.1 Imitation via Supervised Learning

The supervised learning setting describes a standard machine learning scenario where the learner receives a set of training data points $D = \{(x_0, y_0), ..., (x_N, y_N)\}$ such that each pair (x_i, y_i) contains an input vector and its corresponding label. The task of a supervised learning algorithm is to find a function $h : \mathcal{X} \mapsto \mathcal{Y}$ usually called predictor, hypothesis or classifier [20][118] over a hypothesis

space $\mathcal{H}$ of such mapping functions. In this context, $\mathcal{X}$ represents a random variable called the *input space*, and $\mathcal{Y}$ also represents a random variable commonly called the *output space*. A measure of the success of a predictor is computed by the accordance of the predictions it yields with respect of the supervision given signals. Equation 5 defines a 0-1 error function used to measure the performance of classification tasks that will serve as the basis for future analysis of imitation learning algorithms, measuring the accordance of the predicted mapping with the expert's actions.

$$e_h(x_i) = \begin{cases} 1 & \text{if } h(x_i) = y_i \\ 0 & \text{if } h(x_i) \neq y_i \end{cases} \tag{5}$$

No matter which error measure is used, a supervised learning algorithm should output the hypothesis that minimizes the expected empirical error (or empirical risk $\mathbb{E}[e_h(D)]$) over the training examples. This principle is called Empirical Risk Minimization (ERM) [118] and is given by:

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \mathbb{E}_{x \sim D}[e_h(x)] \tag{6}$$

Posing the imitation learning problem as supervised learning is uncomplicated. For this, the following equivalences can be stated as in equation 7. Also, the hypothesis space of the predictor will determine the complexity of the parametrization of the policy space Π, and they will also be equivalent ($\mathcal{H} \equiv \Pi$).

$$\begin{aligned} D &\equiv \mathcal{T} && \textit{(traces dataset equivalent to training dataset)} \\ \mathcal{X} &\equiv \mathcal{S} && \textit{(state space equivalent to input space)} \\ \mathcal{Y} &\equiv \mathcal{A} && \textit{(action space equivalent to output space)} \end{aligned} \tag{7}$$

Having established these equivalences, algorithm 2.1 depicts the process of directly deriving the policy from a set of traces of an optimal policy π^* using supervised learning. The algorithm receives as input access to the expert's policy π^* and needs to have defined the complexity of the parametrization of the policy (hypothesis) beforehand. The traces dataset are obtained by unrolling the optimal policy, and they are later used as input to train a classifier.

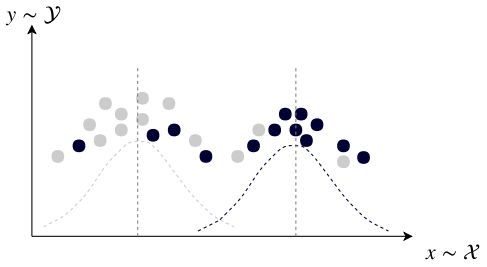
Algorithm 2.1 SUPERVISED LEARNING

Require: π^* : expert's policy, $\mathcal{H}$: policy class

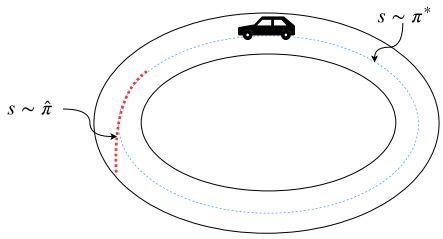
1: **algorithm** SUPERVISED($\mathcal{H}$)
2: **sample** $\mathcal{T} \leftarrow \langle s, \pi^*(s) \rangle$ ▷ Get $\mathcal{T} \sim \pi^*$: traces of π^*
3: **learn** $\mathcal{H}(\mathcal{T}) : \pi_h = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}}[e_\pi(s)]$ ▷ Train $\mathcal{H}$ to minimize $\mathbb{E}[e_\pi(s)]$
4: **return** π_h

The Problem of Distributional Shift

For many years, supervised learning was the *de facto* algorithm to derive policies from human or other expert controller demonstrations [13] [63]. However, this approach suffers from known deficiencies originated by the violation of fundamental assumptions of statistical learning theory. Statistical



(a) Probabilistic view of the distributional shift. Gray dots represent training samples, black dots test samples.



(b) In a driving scenario, a mistake made by a learned policy induces a distributional shift.

Figure 2.3: The distributional shift problem in supervised learning.

learning methods assume that the samples in the dataset are independent of each other and that the training and test sets are identically distributed. These assumptions are commonly known as the *i.i.d assumptions* [57][59]. Initially noted on the experimental results obtained in [101], the execution of a supervised-learned policy in a robotics control problem cannot guarantee the independence of the samples nor the same distribution of the training and test set distributions.

An identifiable scenario of such violation is shown in Figure 2.3b in the context of learning to drive from demonstrations. If at some time step during the execution of the learned policy $\hat{\pi}$ the predicted action deviates from the one seen during training, the systems could end up in a region of the state space not previously presented to the algorithm while training. Hence, the underlying distribution $p(\mathcal{A}|\mathcal{S})$ is shifted as the probability distribution of the state space $p(\mathcal{S})$ changes from train to test phases. This phenomenon is usually referred to in the statistical learning literature as *distributional shift* and it is common in plenty of real-life applications of machine learning algorithms [67] to structured prediction problems (sequential predictions).

A formal analysis of the implication of the distributional shift problem with traditional supervised learning techniques applied to imitation learning settings is offered by Ross and Bagnell [107]. Theorem 2.1 supports the empirical findings in earlier works: once the learner makes a mistake, the changes on the state visitation distribution d_{π^*} induced by the optimal policy begin a composition of errors that grows quadratically in the size of the horizon T of the task at hand.

Theorem 2.1. *Let $\hat{\pi}$ be such that $\mathbb{E}_{s\sim d_{\pi^*}}[e_{\hat{\pi}}(s)] \leq \epsilon$. Then $J(\hat{\pi}) \leq J(\pi^*) + T^2\epsilon$. [107]* [1]

The convention proposed in [107] is maintained here as it has been homogeneously used in derivative work. Hence, $J(\pi) = T\mathbb{E}_{s\sim d_\pi}[e_\pi(s)]$ represents the expected cost incurred by the policy π over the horizon of the task T, having $e_\pi(s) = \mathbb{E}_{a\sim\pi_s}[e(s,a)]$ where $e(s,a) = \mathbb{I}(a = \pi^*(s))$ is a 0-1 loss that measures the agreement between the predictor's and the expert's actions (imitation loss).

The success of the family of DPD algorithms is significantly due to the incorporation of distinctive traits that have been collectively applied to deal with the problem of distributional shift

[1] *The proof of this theorem initially offered in [107] is known to have inconsistencies. See [105], Chapter 2, to find a corrected version of it.*

discussed before. There are at least four identifiable properties that characterize this family of algorithms: interactive learning, policy mixing, on-policy data aggregation and random exploration. The following section provides a thorough inspection of each of these traits and will try to explain how each of them has been crucial in obtaining excellent and robust predictors for imitation learning [105].

2.2.2 Reduction to No-Regret Online Learning

The introduction of online learning is the first of a series of efforts to reduce the impact the i.i.d assumptions have over statistical learners [105]. Online learning algorithms do not require any assumptions concerning the order in which the samples are presented to the learner. Instead of assuming that the data is generated from a stochastic process (distribution), online learning relaxes any assumptions that maybe be made about the data generating process which could be either deterministic, stochastic, or adversarial [27] [117].

In contrast with common offline batch learning approaches, online learning is a process where the training data is streamed to the learner one sample at a timestep. Algorithm 2.2 depicts the general structure of an online learning algorithm. At each timestep t, the learner receives a sample x_t to predict an output value $\hat{y}_t$. Then, the correct prediction y_t is given to the learner from a best-fixed predictor $h^* \in \mathcal{H}$ which usually is in the same hypothesis class of the learner h (*realizability assumption*) [117]. From this correct prediction, the learner suffers a loss $\mathcal{L}(\hat{y}_t, y_t)$ accumulated throughout the horizon of the task.

Algorithm 2.2 Online Learning

Require: $\mathcal{H}$: hypothesis, $h^* \in \mathcal{H}$: best fixed predictor, $\mathcal{L}$: loss function, T : rounds

1: **for** $t = 1...T$ **do**
2: **receive** $x_t \in \mathcal{X}$ ▷ Receive instance for prediction.
3: **predict** $\hat{y}_t \in \mathcal{Y}$ ▷ Predict value of the instance.
4: **receive** $y_t \in \mathcal{Y} = h^*(x_t)$ ▷ Receive ground truth.
5: **suffer** $\mathcal{L}(\hat{y}_t, y_t)$ ▷ Incur in a loss.

The sequential nature of this learning process renders notions like Empirical Risk Minimization or Structural Risk Minimization unusable in this context. Instead, an online learner's performance is measured in terms of *regret* $R_T(h)$:

$$\mathrm{R}_T(h) = \sum_{t=1}^{T} \mathcal{L}(h_t(x_t), y_t) - \sum_{t=1}^{T} \mathcal{L}(h^*(x_t), y_t) \tag{8}$$

The notion of *regret* expresses the difference in the exceeding amount of loss when compared with the loss the best-fixed hypothesis h^* in his class would have suffered over the same time horizon of T. Hence, the online learning theory goal is to find *low regret* algorithms capable of obtaining a regret that grows sub-linearly to the learning task horizon T.

Online Imitation Learning

The introduction of online learning as a method to solve policy estimation in imitation learning does not only help by mitigating the effects of the i.i.d assumptions over supervised policy learning but, as is discussed in the following sections, allows an interactive setting where the expert and the learner policy execution can be interwoven during the training regime. Furthermore, the application of online learning to imitation learning has helped to bound the regret (in terms of the horizon) of a policy π to the optimal policy π^* in a particular policy class Π [107].

The main contribution due to [107] to DPD algorithms is the reduction of imitation learning problems to a *no-regret* online learning process. Firstly, it is essential to define that the regret of a policy π accounts for the T-step cost incurred while learning a task with horizon T. The regret for π is defined by:

$$\begin{aligned} \mathrm{R}_\Pi &= J(\pi) - \min_{\pi' \in \Pi} J(\pi') \\ \mathrm{R}_\Pi &= J(\pi) - J(\pi^*) \end{aligned} \tag{9}$$

where the optimal policy $\pi^* \in \Pi$ regret $\mathrm{R}_T(\pi^*)$ is constant $O(1)$ through the task's horizon T. Then, [107] proposed *forward training* -Algorithm 2.3- as an improvement over basic supervised learning approaches to policy derivation by bounding the regret to grow linearly in T. This performance is equal to that expected from supervised learning when applied to non-structured prediction problems.

Algorithm 2.3 FORWARD TRAINING [107]

Require: π^* : expert's policy, T : task horizon, $\mathcal{H}$: classifier

1: **algorithm** FORWARD($\pi^*, T, \mathcal{H}$)
2: $\pi^0 = (\pi_1^0 \sim \pi^*, \pi_2^0 \sim \pi^*, ..., \pi_T^0 \sim \pi^*)$ ▷ Initialize π_i^0 to query the expert
3: **for** $i \leftarrow 1...T$ **do**
4: **sample** $S_{s_i^k \sim d_{\pi^{i-1}}} = \bigcup_{k=1}^{N} \{(s_0^k, ..., s_T^k)\}$ ▷ Sample N T-step trajectories following π^{i-1}
5: **query** $\mathcal{T} \leftarrow \{(s, \pi^*(s))\} : s \in S_{s_i^k \sim d_{\pi^{i-1}}}$ ▷ All state-action pairs taken by expert
6: **learn** $\mathcal{H}(\mathcal{T}) : \pi_h = \mathrm{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^{i-1}}}[e_\pi(s)]$ ▷ Train $\mathcal{H}$ to minimize $\mathbb{E}[e_\pi(s)]$
7: $\pi^i = (\pi_1^{i-1}, ..., \pi_i^i \sim \pi_h, ...\pi_T^{i-1})$
8: **return** π^T

The reader may notice that the dataset recollection and the learning procedure happen simultaneously over the horizon of the tasks. Thus, the policy is derived online, and the classifier is trained each time over a subset of the input space generated by the execution of the policy π^{i-1} obtained in previous iterations. This behaviour is going to be a distinctive trait for all the DPD algorithms presented from now on.

Interestingly, the guarantees offered by *forward training* are based on numerous artifacts that are impractical in most settings. Firstly, the policy obtained is a non-stationary (time-dependent) policy as the algorithm maintains a solution policy for each timestep through the task's horizon (steps 2 and 7). This issue has a twofold implication. First, the space complexity of the policy is $O(T)$; thus this requirement is mostly unsatisfiable for infinite or large-finite horizon tasks. Furthermore,

a non-stationary policy may introduce instabilities on the system overall performance (e.g., behave as a bang-bang controller). Also, the T-step cost analysis offered in [107] assumes that the optimal policy π^* can quickly recover from an unknown state (stability property).

Those are potentially the main reasons why there are no explicit references in the literature to any practical applications of *forward training* as a DPD method. Nevertheless, this method has the indisputable merit of having introduced two central ideas to the imitation learning theory: online learning (step 6) and mixed policy execution (steps 2 and 7).

2.2.3 Policy Mixing

The problem of the distributional shift in the application of supervised learning for policy derivation can be explained in simplest terms by stating that, for most tasks, the expert's observed behaviour rarely includes recovery actions from unexplored states, e.g., a human driver rarely departs from the middle of the lane. Consequently, the derived policy has no training samples from which to infer the required action under such circumstances. That is why the interwoven execution of the agent's policy and the experts' produces a more robust policy with stronger guarantees. The state distribution d_π obtained by executing the learner's non-converged and error-prone policy induces a non-intentional exploration of the state space as the learner's erroneous predictions may be out of the distribution induced by π^*. This makes $s \sim d_\pi$ closer to the real distribution $s \sim p(\mathcal{S})$ of the task being learned.

Algorithm 2.4 SMILe: Stochastic Mixing Iterative Learning [107]

Require: π^* : expert's policy, N : iterations, $\mathcal{H}$: classifier, α: mixing coefficient

1: **algorithm** SMILe($\pi^*, N, \mathcal{H}, \alpha$)
2: $\quad\pi^0 \sim \pi^*$ ▷ Initialize π^0 to query the expert
3: $\quad$**for** $i \leftarrow 1...N$ **do**
4: $\quad\quad$**sample** $S_{s\sim d_{\pi^{i-1}}} = \{(s_0, ..., s_T)\}$ ▷ T-step trajectories following π^{i-1}
5: $\quad\quad$**query** $\mathcal{T} \leftarrow \{(s, \pi^*(s))\} : s \in S_{s_i^k \sim d_{\pi^{i-1}}}$ ▷ All state-action pairs taken by expert
6: $\quad\quad$**learn** $\mathcal{H}(\mathcal{T}) : \pi_h = \text{argmin}_{\pi\in\Pi} \mathbb{E}_{s\sim d_{\pi^{i-1}}}[e_\pi(s)]$ ▷ Train $\mathcal{H}$ to minimize $\mathbb{E}[e_\pi(s)]$
7: $\quad\quad$**mix** $\pi_h^i = (1-\alpha)^i \pi^* + \alpha \sum_{j=1}^{i} (1-\alpha)^{j-1} \pi_h^j$ ▷ π^i is stochastically mixed
8: $\quad$**remove** $\pi^* : \pi_N = \frac{\pi_h^N - (1-\alpha)^N \pi^*}{1-(1-\alpha)^N}$ ▷ Remove expert's query from final policy
9: $\quad$**return** π_N

Algorithm 2.4 describes the steps of SMILe (Stochastic Mixing Interactive Learning): the first DPD algorithm that takes full advantage of the strategy of *policy mixing* initially presented in [107], and it is an immediate derivation from *forward training*. The immediate benefit it offers over *forward training* is due to the replacement of the non-stationary policy by one that stochastically mixes the expert's policy execution and a weighted average of the policies learned in past iterations with probabilities $(1-\alpha)^i$ and α respectively (step 7). If $\alpha \in O(\frac{1}{T^2})$ and the number of iterations $N \in O(T^2 \log T)$ then SMILe formally guarantees that the regret of the learner is bound by $J(\pi_N) \in O(T)$ ([107], Theorem 4.1).

Although interesting from the perspective as an improvement over *forward training*, SMILe is considered to have two main limitations, both related to the mixture of policies obtained as a final result. In an approximately similar setting to the one offered by *foward training*, the weighted mixture of policies space complexity is in the order of $O(N)$. Given that the formal guarantees of linear regret are offered when $N \in O(T^2 \log T)$ then, a space complexity that is linear on the number of iterations grows approximately quadratically on the task's horizon T and may render this approach prohibitively costly for infinite or large-finite horizon tasks. Similarly, regarding stability, as the policies in the mixtures are weighted as a function of the iteration step, they are trained on and not a function of the predictive performance, there may exist highly weighted policies with inferior performance than some with lower weight and better performance in the mixture. This situation has an impact in the final policy π_N stability, a situation that can be observed from the graphical evidence referred in [107] [2].

On-Policy Data Aggregation

Among the benefits of mixing the learner's policy with the expert's policy during training has brought to the imitation learning theory is the notion of *dataset aggregation*. The expert-learner policy mixing presented in SMILe was not initially considered as a data acquisition strategy in [107]. Nevertheless, as its advantages were numerous, mixing the learner and the expert policy execution in a setting that resembles that of active learning [116] -where the learner can influence the state distribution explored during training- has been the cornerstone for Dataset Aggregation (DAgger) [110].

Algorithm 2.5 DAgger: Dataset Aggregation [110]

Require: π^* : expert's policy, N : iterations, $\mathcal{H}$: classifier, α: mixing coefficient

1: **algorithm** DAgger($\pi^*, \mathcal{H}$, N, α)
2: $\quad\mathcal{T} \leftarrow \emptyset$, $\pi_1 \in \Pi$
3: $\quad$**for** $i \leftarrow 1...N$ **do**
4: $\quad\quad\pi_i = \alpha_i \pi^* + (1 - \alpha_i)\pi_i$ $\quad\triangleright\ \alpha_i = p^{i-1}$: π^* execution decays over time.
5: $\quad\quad$**sample** $\mathcal{T}_i \leftarrow \{s, \pi^*(s)\} : s \sim d_{\pi_i}$
6: $\quad\quad$**aggregate** $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_i'$
7: $\quad\quad$**learn** $\mathcal{H}(\mathcal{T}) : \pi_h = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi_i}}[e_\pi(s)]$
8: $\quad$**return** π_h best on validation.

Dataset Aggregation -Algorithm 2.5- is the first DPD algorithm that implicitly exploits learner's prediction errors as a method for extensive exploration on the boundaries of the state space that are not induced by the expert policy. DAgger iteratively refines the learner's policy (an online-trained classifier) by composing an on-policy dataset through the aggregation of pairs $\{s, \pi^*(s)\}$ (step 5) where the state space distribution d_{π_i} is induced by a mixture of policies between π^* and π_i at every time step (step 4). The analysis of performance for DAgger presented in [110]

[2] There exist a supplementary video of a SMILe learned policy for Super Tux Kart game that shows the controller instability.

is based on the guarantees offered by no-regret online learning algorithms like Follow-The-Leader [117]. Assuming that $e_\pi(s)$ is convex with respect to the parametrization of the classifier π_h, DAgger formally guarantees a linear regret of $O(T)$ over the task's horizon T after N iterations in the order of $O(T^2)$.

Theorem 2.2. *For* DAgger *if* N *is* $O(u^2T^2\log(1/\delta))$ *and* m *is* $O(1)$ *then with probability 1 there exist a policy* $\hat{\pi} \in \hat{\pi}_{1:N}$ *s.t.* $J(\hat{\pi}) \leq J(\pi^*) + uT\hat{\epsilon}_N + O(1)$ *[110].*

The convex loss assumption limited the validity of this finding when the cost functions used to train state of the art machine learning models (e.g., neural networks) are rarely convex with respect to their parameters [15]. Nevertheless, DAgger has been successfully applied to real-life problems like in [93, 111].

2.2.4 Explicit Exploration

The DPD algorithms presented so far have offered an implicit exploration step as part of the policy mixing strategy directed and provoked by potential mistakes in the learner's policy that can induce a different state distribution than the one offered by the expert's policy. However, implicit exploration is not a particularly good strategy under the context that while the learner's policy performance becomes closer to that of the expert's, the exploration stages become more and more sparse. The benefit of exploration steps as part of learning for optimal control has been extensively studied in Reinforcement Learning [125] as a method for discovering novel information about the environment in which an agent acts. This exploration strategy is what potentially enables an IL agent to surpass a non-optimal demonstrator performance.

Algorithm 2.6 AggreVaTe: Aggregate Values to Imitate [109]

Require: π^* : expert's policy, N : iterations, $\mathcal{H}$: classifier, α: mixing coefficient

1: **algorithm** AggreVaTe($\pi^*, \mathcal{H}$, N, α)
2: $\mathcal{T} \leftarrow \emptyset$, $\pi_1 \in \Pi$
3: **for** $i \leftarrow 1...N$ **do**
4: $\pi_i = \alpha_i\pi^* + (1-\alpha_i)\pi_i$ ▷ $\alpha_i = p^{i-1}$: π^* execution decays over time.
5: **sample** $\mathcal{T}_i$ = AggreVaTeSampling() ▷ see Algorithm 2.7
6: **aggregate** $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_i$
7: **train** $\mathcal{H}(\mathcal{T})$: cost-sensitive *or* **train** $\mathcal{H}(\mathcal{T}_i)$: online learner.
8: **return** π_i best on validation.

Aggregate Values To Imitate (AggreVaTe) [109] -Algorithm 2.6- is the first DPD method to suggest the incorporation of an explicit exploration as part of its trajectory sampling methodology. Built incrementally over the basis of DAgger, AggreVaTe defines in explicit terms a strategy to interleave the learner's policy execution, an *explicit exploration* step and the expert's policy as shown in Algorithm 2.7 (steps 3-6). By uniformly sampling an instant t in the task's horizon during training, AggreVaTe induces the learner's to potentially unexplored parts of the state space from which to learn recovery actions from the expert's corrective behaviour afterwards.

Algorithm 2.7 AGGREVATESAMPLING: SAMPLING

1: $D \leftarrow \emptyset$
2: **for** $m \leftarrow 1..M$ **do**
3: **sample** $t \sim U(1, T)$
4: **execute** π_i from $1..t-1$
5: **explore** (s_t, a_t)
6: **execute** π^* from $t+1...T$ and get cost-to-go $\hat{Q}$
7: $D \leftarrow D \cup \langle s, t, a, \hat{Q} \rangle$
8: **return** D

This algorithm also leverages the notion of cost-to-go $\hat{Q}$ to characterize how difficult is for the expert to recover from the states induced by implicit and explicit exploration mechanisms. Hence, the dataset it constructs afterwards is formed by tuples $(s, t, a, \hat{Q})$ suitable for learning with a cost-sensitive classifier (one that is aware of the cost of its decisions) [41]. An alternative to a cost-sensitive classifier –as the measurements of the cost of actions and states may not be available for some tasks– is the application of any incremental no-regret online learning algorithm over the each $\mathcal{T}_i$ dataset with no cost incorporated.

As in DAgger, the formal guarantees obtained for AggreVaTe bound its regret linearly to the task horizon [109] (Theorem 2.1, page 4). The assumption of the use of convex loss as a training signal for the classifier was also one limitation inherited from DAgger, an issue that has been addressed by the work of Sun et al. [124]. Sun et al. [124] offers an end-to-end differentiable improvement of AggreVaTe called Differentiable AggreVaTe (AggreVaTeD) that introduces structural changes to the initial algorithm presented in [109] to add a gradient-based policy estimation [125] for what the authors called *differentiable imitation learning*. This addition represents an exciting modernization of DPD algorithms by bringing them into the spectrum of deep neural networks and deep learning creating a sub-field usually referred to as *Deep Imitation Learning*.

Most modern applications of DPD algorithms to robotics [23, 34, 37, 82, 93, 97, 136] now rely on implementing parametric and gradient-optimized representations of the learner's policy by the introduction of neural networks as a representationally more powerful model to cope with the Direct Policy Derivation problem. This class of hypotheses has broadened the spectrum of problems to which Imitation Learning can be applied to, but has also brought a different set of challenges that compromises the traditional guarantees of performance.

One of these challenges is *interpretability*. Interpretability has been an issue in Machine Learning algorithms well before the rise of DNN [39]. However, the non-linearity and the dimension of the number of parameters current models employ to solve any problem have increased the awareness over this particular issue as DNN has been applied to an increasing number of problems. Hence, when a DNN model is selected as policy parametrization for DPD algorithms this decision not only increases the complexity of the policy space that can be explored but also significantly decreases the opportunity to verify the correctness of the derived policy formally. This deficiency leads to a second issue: *safety*.

As it will be discussed in Chapter 4, the algorithmic solutions to the DPD discussed earlier impose some concerns over the safeness of the learning process. Solving safety issues during learning is a first step towards ensuring IL systems can be confidently trained and probably a foundational stone for future research that could guarantee safety at test time for IL algorithms. If this is achieved, IL algorithms will become more widespread than they are nowadays as the number of the task from which expert (human) demonstrations can be recovered are numerous.

Chapter 3

Intersection Crossing from Experts' Demonstrations

Contribution of Authors

Originally published as Diaz, M., Girgis, R., Fevens, T., and Cooperstock, J. To Veer or Not to Veer: Learning from Experts How to Stay Within the Crosswalk. In *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017* (2018), vol. 2018-Janua, this work constitutes the result of Manfred Diaz (M.D) collaboration with Roger Girgis (R.G), Prof. Jeremy R. Cooperstock and Prof. Thomas Fevens while visiting the Centre for Intelligent Machines at McGill University.

M. D and R. G. conceived of the presented idea. R. G. performed the state of the art research. M. D. developed the theory under the framework of Learning from Demonstrations. M.D performed training data recollection and M.D, and R.G performed testing data recollection. M. D. developed the annotation tool. M. D and R. G annotated data for training and testing. R. G trained CNN models. M. D and R. G. designed the prototype application. M. D took the lead in developing the prototype application. M. D and R. G conceived, planned and carried out the experiments.

M.D and R.G took the lead in writing the manuscript. All authors provided critical feedback and helped shape and improve the final manuscript.

3.1 Introduction

Independent navigation of a city is a significantly challenging task for individuals with visual impairment. This challenge is further exacerbated when the environment they are navigating is unknown. For this reason, they tend to remain in known environments [77], as they learn the intersection characteristics of those routes. Some of the problems they face include determining whether the intersection is one- or two-way, orienting to the correct direction for crossing, obtaining the status of the pedestrian signal, and detecting veering during the crossing phase. With regard to the last

of these, mobility training focuses on techniques to keep the individual walking as straight as possible while maintaining a safe distance from parallel traffic, i.e., remaining within the marked lines designating pedestrian crossings. Unfortunately, even after training, detection of veering remains difficult [58]. It has been noted that regardless of visual impairment, in the absence of environmental cues, humans tend to walk in circles [122], with diameters as small as 20 meters. This has obvious implications to crossing at intersections, which can be of similar length.

Various assistive devices are available to help the visually impaired explore a city, including talking GPS systems, and those providing information about points of interest around the user [94]. However, these do not solve the problem of safe crossing at intersections, which is generally agreed to be the most difficult and risky aspects of independent travel for visually impaired individuals [119].

Accessible pedestrian signal (APS) systems provide indications of *when* it is safe to cross [114], and in certain cases, offer auditory cues that help the user determine orientation. Unfortunately, these auditory cues are often masked by background noise. More problematically, due to their high cost, estimated at over $25k per new installation, and approximately $8k at intersections with existing poles [91], APS deployment remains limited. For example, according to the Montreal Association for the Blind, the city of Montreal, Canada, with 1875 intersections [43], reportedly has only 133 installed APS systems [1].

A potential alternative, explored by several research efforts, considers the use of embedded sensors, such as accelerometers or gyroscopes found in typical smartphones [95, 104, 58] to provide the feedback necessary to prevent veering. However, sensor instability and the potential need for frequent re-calibration pose obstacles to such efforts. Furthermore, while these solutions may reduce veering behavior, they do not help with the initial alignment of the user in the correct direction at the start of crossing.

Relying instead on visual information provided by the smartphone camera represents an attractive alternative. This is especially the case considering that non-visual understanding of the environment is not only less effective and efficient, but also potentially dangerous, compared to scanning the surrounding using vision [11]. However, processing of the wide variety of street scenes to extract the appropriate features, if present, needed for such guidance has long been a daunting challenge. Fortunately, the recent explosion of capability of deep learning systems offers a potential solution. In particular, the convolutional neural network (CNN) architecture has been shown to outperform all other methods for image recognition and classification tasks [61, 33, 62, 65, 80]. Previous work [134] has shown that the features these CNN models learn can be transfered to tackle a different problem. In this paper, we combine such pre-trained models with Learning from Demonstrations [64] techniques to provide real-time feedback to visually impaired individuals before and during the crossing of an intersection, helping both initial alignment and maintenance of a straight path.

3.2 Related Work

Many systems have been developed in an attempt to tackle the veering and intersection crossing problem encountered by the visually impaired community. While some of these systems are commercially available, many systems remain limited to an academic setting, and are still either in the experimental phase or would be too expensive for widespread commercial deployment. These systems typically lie within two main categories of systems: sensor-based and vision-based systems. In this Section, we present some of these systems and how they motivated the approach presented in this work.

3.2.1 Sensor-Based Systems

The first category of systems focuses on employing sensors typically mounted on the user. One such orientation and way-finding interface system was proposed in [104] where they explore three different interfaces. The system is comprised of a computer placed in a backpack, to be worn by the user, with an array of speakers placed against the back used to vibrate the direction to follow, a digital compass mounted either on the shoulder or in a hat, and a pair of ear buds mounted on the hat providing stereo audio beeping. The authors found a 31% significant improvement in veering performance when compared to the baseline veer. It is important to note that the authors initially attempted to augment the orientation signal from the digital compass by installing a pedestrian signal system at test intersections which would communicate with the backpack computer. However, this could not be accomplished due to state laws and difficulties with the installation and maintenance of such a system. This further demonstrates the difficulty that one would face in deploying such a system at intersections in a given city. Another drawback with this system is the need to recalibrate the digital compass after every intersection. This makes such an application highly impractical for the intended user group. Finally, the system also assumes that the user is properly oriented at the onset of crossing.

Guth [58] proposes the *Anti-Veering Training Device (AVTD)* which employs a solid state gyroscope to measure the user's cumulative rotation as they walk along a path. The gyroscope also provides tilt and temperature compensation adding robustness to the system. The user is presented with veering correction speech cues and feedback about performance. However, it is not apparent how the system's effectiveness and accuracy were evaluated. Paneels et al. [95] build on this work with their *Walking Straight* application which also uses the gyroscope to measure body sway and orientation. This work also focuses on the feedback modality based on typical mobility training for the blind. The experiment consisted of walking in a straight line towards a 15 m target after initially being positioned in the correct orientation. It was conducted in a controlled outdoor environment, and not at an actual intersection. They find that the system reduced veering to half that encountered during the control condition. Another important result from their experiment was that the most effective method for providing veering feedback was a continuous beep rendered in the ear opposite to the veering direction. As such, the current proposed application uses a continuous beep stimulus.

While these systems can be effective in an ideal setting, sensor stability can prove problematic when the system is continuously used. This is due to the need of recalibrating the sensors, making such systems risky in the intersection crossing task. Additionally, these systems assume that the user was initially properly oriented. However, this can be a difficult task when taking into account the complex sound environment at typical intersections, as discussed in [11]. Moreover, neither of these systems has the capability of providing information regarding the pedestrian signalization status.

3.2.2 Vision-Based Systems

In recent years, many systems have been proposed that instead employ computer vision techniques. Shen et al. [119] developed a prototype on a Nokia 6681 mobile phone, utilizing the camera for detection. The system detected zebra-crossings using segmentation of the edges of strips in the pattern. However, the authors explain that the algorithm performed worse than an earlier version that was implemented using a desktop computer and higher resolution camera. With the advances in mobile devices, Ahmetovic et al. build on the work in [119] by building a two-part system comprising of a *Zebralocalizer* [5] and *ZebraRecognizer* [4]. The former is used to interface between the user and an iOS application while the latter uses a 5-step process, computing the position of the zebra-crossing by using a combination of the camera and the device accelerometer as inputs. Furthermore, they transform the problem into three stages: an approaching stage, an aligning stage and a crossing stage. The user detects and crosses an intersection by holding the mobile phone (an iPhone 4) parallel to the ground, with the camera "looking" for the zebra-crossing. The results were positive, with all the subjects successfully capable of crossing a 6-meter road in an average three to five seconds.

One of the major limitations of these systems is the need for a zebra-crossing pattern, as its presence is infrequent in many cities. As a result, users would still lack the ability to fully and autonomously navigate through an unknown area. Later work by Ahmetovic et al. [6] has been done to address the *zebracrossing* scarcity. In this work, the researchers designed a system that mines existing image databases (e.g. Google Street View images) to plan a route that ensures all intersections have *zebracrossings*. While this method offers an elegant solution, it still doesn't provide users with an independent experience. Another limitation of such a system is its inability to deal with occlusions. As a result, these can cause users to move in wrong directions leading to potentially dangerous situations.

Ivanchenko et al. [66] proposed a system that detects the more common two-stripe crosswalk instead of zebra-crossings, removing the reliance on this pattern. The authors develop the *Crosswatch* application running on a Nokia N95 mobile phone. Additionally, they use accelerometer readings to estimate the direction of gravity, making it easier to position the camera in the correct orientation. In addition, they utilize a 3D analysis technique as an attempt to ensure the subject remains within the two-stripes. To perform this analysis, they use the focal length of the camera lens and estimate an average height of 1.5-m for adults. Finally, the system used high-pitched tones to inform the user if their feet were inside the two-lane corridor. The preliminary experiments required that the blind user correctly identify the location of a crosswalk. However, the experiments did not include task of

crossing the intersection and, therefore, does not allow for evaluation in effectiveness. Additionally, it not clear how such a system would handle partially or fully occluded stripes.

Moreover, Poggi et al. [100, 99] proposed the use of a pocket-sized device with and embedded CPU, coupled with a custom RGBD camera attached to wearable glasses. This device uses the dense disparity map from the RGBD camera to determine the ground plane, which serves as a reliable way to discriminate between the ground and the rest. Furthermore, they train a CNN model, similar to [80], which takes as input a wrapped image of the ground and, if a crosswalk is present, determines its orientation. The authors report a near-perfect accuracy on their test set, testimonial to the power of these models. However, it is important to note that this system uses custom hardware (e.g. custom RGBD camera) designed by the authors. It is not clear how it would be deployed for the general public in an efficient and cost-effective manner. In addition, the authors only report testing results on a computer in an off-line setting. Thus, it is not clear how this system would perform in a real world experiment with blind participants. Finally, this system was designed for the initial orientation part of the intersection crossing task. We are not aware if it can easily be adapted to provide users with real-time veering feedback.

Both of these types of systems rely on the presence of a zebra-crossing or a two-stripe crosswalk. However, as we have experienced through our investigation of the problem, zebra-crossings are not always present at intersections and the lines in the two-lane corridors are, in many cases, faded or obscured. In such intersections, these systems would be incapable of assisting users in their everyday travels. The system proposed in the present work does not depend on any particular structure at intersection crosswalks as it utilizes recent advances in machine learning to detect veering problems.

3.3 Street Crossing from Demonstrations

Despite the recent surge of work in intelligent robotics, to our knowledge, the results from this research have scarcely been applied to alleviate sensorial, motor and cognitive impairments in humans [12]. We believe that such research, in particular, the technique of Learning from Demonstration, is well suited to addressing the problem of veering during street crossing.

3.3.1 Motivation

Learning from Demonstration (LfD) is based on the idea of transferring human behavior to intelligent agents [64] [14]. An agent's policy is a function $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maps every state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. Conceptually, algorithms in the LfD domain aim to acquire the optimal policy π^* for a task from a series of demonstrations $\mathcal{D} = \{d_1, d_2, ..., d_m\}$ that can guide an agent while autonomously performing the task. Following this methodology, we can gather the knowledge of sighted "experts" on the intersection-crossing task, and transfer these to an intelligent assistive agent for the visually impaired.

To completely formulate an LfD solution, one must establish the structure of the world states $s \in \mathcal{S}$ that the agent may reach, the actions $a \in \mathcal{A}$ that the agent is capable of performing, and a transition function $\mathcal{T}(s'|s, a)$ that expresses the probability of landing in $s' \in \mathcal{S}$ given that the

agent executes action a from state s. In most real-life scenarios, the state is not fully observable. Most LfD models handle this uncertainty by relying on the agent's observations of the world, $z \in \mathcal{Z}$, instead of the complete representation of its internal structure [14]. Therefore, our LfD method must determine the optimal policy, $\pi^* : \mathcal{Z} \rightarrow \mathcal{A}$ using demonstrations $d_i = (z_i, a_i) \in \mathcal{D} : \mathcal{Z} \times \mathcal{A}$. We now describe the application of this approach to the street crossing domain.

3.3.2 Action Space Design

In LfD, a transition $t \in \mathcal{T}$ between states occurs when an agent executes the actions specified by its policy. We choose to discretize the space of possible actions by dividing the agent's field of view into 12 evenly spaced vertical bins as presented in Figure 3.1, following a similar approach taken in previous research [108, 24, 73]. Each bin, $v \in \mathcal{V}, \mathcal{V} = \{v_1, v_2, ..., v_{12}\}$, is an action in $\mathcal{A}$ an expert would recommend to execute given an observed state in the street crossing task. The bins are intended to capture the heading of the goal relative to the expert's field of view, with bin v_1 corresponding to the agent having to veer maximally to the left, and bin v_{12} representing having to veer maximally to the right.

For situations where an expert could not identify the bin including the goal, for example, in the scenario shown in Figure 3.2a, our problem model also included an action *unknown* $\in \mathcal{A}$. As we will discuss in Section 3.4, this representation allowed us to experiment with different levels of granularity for the action space.

3.3.3 Task Demonstrations

Once the problem design space was defined, the structure of the demonstration set had to be specified. We divided our collection of demonstrations into two steps: *(i)* demonstrations acquisition and *(ii)* expert's knowledge extraction.

Each demonstrator was asked to stand at the corner of an intersection, holding a smartphone at chest level, and capture, from a first-person perspective, the sequence of actions required to cross the intersection. The motivation for this particular position of the smartphone is the outcome of previous experiments carried out with visually impaired users [94, 95]. As our interpretation of the street crossing task also included an initial orientation phase to the correct direction towards the goal, demonstrators were asked to record the procedure of rotating within a range of $\pm 45^o$ about the appropriate heading from the starting corner to the goal corner.

Furthermore, as suggested by previous work [101, 129, 108], the high sensitivity of LfD techniques to the quality of demonstrations greatly impacts their generalization ability. A comprehensive set of samples $(z, a) \in D$ should capture not only the optimal behavior of the task, but also states that could only be reachable by some suboptimal action sequence. To ensure that this was the case, the demonstrators were asked to include suboptimal behaviors in their crossings, along with the corresponding corrective actions.

Our demonstrators recorded 215 videos of approximately 25 s each from street intersections in downtown Montreal, Canada, registering the sequence of states transitioned by sighted individuals performing the task. As a compromise between data quantity and a desire to minimize redundancy

of frames at a high framerate of 30 frames per seconds (fps), we extracted frames from the collected videos at a rate of 2 fps, which resulted in a total of 8125 observations.

Figure 3.1: Action space discretization into vertical bins $\mathcal{V} = \{v_1, .., v_{12}\}$ from left to right.

3.3.4 Experts' Knowledge Extraction

As our method did not incorporate a technique to capture the demonstrators' actions on-site, we relied on three experts' knowledge to extract optimal behavior from those observations, in a post-demonstrations procedure. For this, each expert was presented with frames randomly sampled from the observations, in a structure similar to the one depicted on Figure 3.1. They were then asked to select the bin $v \in \mathcal{V}$ that contained the position of the goal.

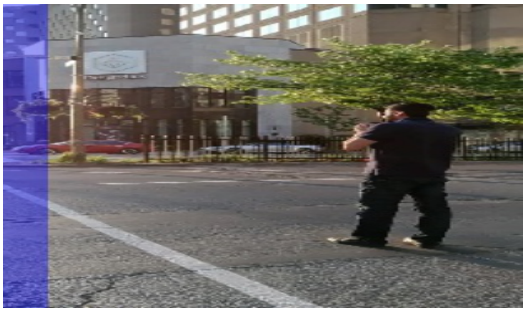
To ensure some resiliency to occlusions in the derived policy, we instructed the experts to choose the bin closest to the presumed goal position in scenarios in which the goal was occluded or otherwise not visible, provided that its location could be assumed (e.g., Figure 3.2b). We expected that under most conditions, a sighted individual could quickly estimate the relative orientation towards the goal from a single observation. For those exceptional cases where it was not possible to infer the target position, the experts were asked to assign *unknown* as the recommended action (e.g., Figure 3.2a).

By virtue of symmetry, we were able to mirror each image around its central vertical axis and associate the flipped image with the corresponding inverse action (i.e., swapping left-to-right with right-to-left). This allowed us to create a set of synthetic observations which, combined with the demonstration examples gathered, doubled the size of $\mathcal{D}$ and ensured a balance between the states explored and the optimal behavior observed.

3.3.5 Policy Derivation Technique

The literature on LfD suggests the existence of three categories of policy derivation methods: direct learning, indirect learning, and execution plans, only differentiated by how much understanding of the environment each algorithm requires while inferring a policy [64, 14]. The algorithms contained in the Direct Learning category are mostly independent of beliefs about the internal state of the environment, thus easier to implement. Then, the family of direct policy learning algorithms was our preference to solve the street crossing veering problem.

(a) total occlusion of the environment



(b) optimal actions inferred even if the goal is not visible

Figure 3.2: Examples of demonstrations' frames including corner cases in our dataset.

Based on the discretization of our actions space and the reduction of the observations to features, we choose to implement our policy extraction strategy as an image classification problem. A classification problem is one where a classifier $c(x) : X \rightarrow Y$ is used to predict the class y of an instance x, having $y \in Y$, $Y = \{y_1, y_2, ..., y_m\}$ a discrete set of classes. Usually, $x \in X$ is a vector $\vec{f} = \{f_1, f_2, ..f_n\}$ of features that reduce the dimensionality of the samples in X. In a supervised learning setting, the classifier is trained using a dataset $\mathcal{N}$ of samples in the form $(\vec{f_i}, y_i)$. Thus, we established the equivalence: $\mathcal{D} \equiv \mathcal{N}, \mathcal{Z} \equiv X, Y \equiv \mathcal{A}$ where the classifier $c(x) : \mathcal{Z} \rightarrow \mathcal{A}$, a CNN model, was trained to infer our policy π^* directly from samples on $\mathcal{D}$.

CNN for Classification Tasks

As an image usually contains irrelevant and redundant information for the resolution of visual tasks, it is better to deal with a condensed representation of such knowledge. Computer Vision techniques often rely on the extraction of salient attributes as a way to minimize the dimensionality of the information contained in an image. Manual extraction of those features requires a comprehensive understanding of the environment and the task at hand. The appearance of Convolutional Neural Networks (CNN) has come to alleviate this need while achieving human-level performance on computer-assisted visual tasks.

Notably, CNN architectures have eliminated the prerequisite of hand-crafted feature extraction algorithms by learning the required features and the task at hand, simultaneously [18]. Since ImageNet Large Scale Visual Recognition Challenge 2012 [112], CNN have obtained state of the art results [76] on benchmark datasets in image classification, segmentation or object detection like ImageNet or PASCAL Visual Object Classes Challenge (VOC) [42].

Yosinksi et al. [135] analyzed why CNN has performed remarkably well on visual tasks and concluded that the way convolutional filters are organized explains this success in part. In a CNN, each convolutional filter learns to search for specific patterns in an image. Filters on first layers of these models learn to detect low-level characteristics (e.g., edges), while filters in deeper layers are fine-tuned to compose the low-level patterns into high-level features (e.g., the shape of a flower),

according to a hierarchical structure. Therefore, we used CNN architectures to convert our z component of the demonstrations $d_i = (z_i, a_i)$ to a vector $z : \vec{f} = \{f_1, f_2, ..., f_n\}$ of features and to map these features into our discrete action space $\mathcal{A}$, thus generating an optimal policy π^*.

Transfer Learning

Training a CNN for classification using randomly initialized filters, or even with traditional heuristics [56], is usually a challenging and time-consuming task as the space of the models' hyper-parameters has to be explored. Moreover, our dataset had significantly fewer instances that the ImageNet dataset (8725 vs. 1.2 million instances) and the dimensionality of the classification task is significantly lower (13 vs. 1000 classes). Consequently, the direct application of models designed for ImageNet could lead to overfitting our dataset and to the loss of generality on the predicted actions.

In this regard, the notion of transfer learning helped us to overcome those obstacles. The theory of transfer learning establishes that the knowledge on a source problem space $\mathcal{P}_s$ of a learned task $\mathcal{T}_s$ could help improve the learning of a target task $\mathcal{T}_t$ on a target problem space $\mathcal{P}_t$. How much knowledge is transferable from one domain/task to the another is directly associated with the amount of overlap between the problem areas in both [92].

Therefore, there exists a proven transferability property between features of a CNN trained on different visual tasks [134]. Although the overlapping between our demonstrations and the training samples on the ImageNet dataset is not clear, we still relied on models pre-trained on the latter as a starting point for fine-tuning different classifiers. Consequently, the high-level features of our problem were built upon the low-level features in the pre-trained models by re-training the appropriate deeper layers in each model.

Interestingly, the derivation of policies with supervised learning has presented some weakness in the past when the independence and identical distribution of the samples collected on $\mathcal{D}$ cannot be guaranteed (i.i.d principle)[108]. To guarantee such independence, each frame and the corresponding expert's action was considered a self-contained demonstration. Recent applications of LfD and CNN to navigation problems in robotics [24, 55, 73] have disregarded the sequential interpretation of a go-to-goal process thus inferring a stationary (time independent) policy. Moreover, the observations presented to the experts for labeling were randomized, ensuring their action (class) recommendation was independent of a sequential analysis of the frames.

3.4 Results and Discussion

3.4.1 Training the Agent

The accuracy of CNN models has significantly improved in recent years relative to their computational complexity [26]. However, state of the art results remain dependent on models relying on high-performance hardware, especially Graphics Processing Units (GPUs) to carry out their inference within adequate time constraints for real-life or real-time applications. Recent work [65, 62, 103, 96] has explored CNN architectures that aim to achieve a balance between the human-level accuracy

results of their predecessors and the prediction time, thus making the application of deep learning techniques to a real-time problem, such as street crossing, feasible.

In this work, we experimented with four state of the art CNN architectures. Firstly, Resnet50 [61] and Xception [33] have a reported top-5 accuracy over 90% on the ImageNet dataset. This motivated our exploration of their potential as policy extractors. Moreover, we were curious to investigate the performance of network models that have been designed specifically to achieve a balance between classification accuracy and training/inference time. Thus, we selected Squeezenet [65] and Mobilenet [62] as our testbed for a mobile deployable solution.

Our transfer learning approach was based on the fine-tuning of each model by removing the latest layers, containing high-level features, and training our custom structure from scratch. In the cases of Xception, Mobilenet and Squeezenet, after removing those high-level-feature layers from each model, we added a $3 \times 3 \times 32$ convolutional layer, followed by a $1 \times 1 \times |\mathcal{A}|$ convolutional layer, both activated with ReLUs [88]. Finally, we added a softmax activation layer with a size of $|\mathcal{A}|$. Because of the particular structure of residual networks [61], we could only add to Resnet50 an extra fully connected layer converging to the number of actions and, similarly to the models above, this layer was followed by a softmax activation layer.

After introducing these modifications, we fine-tuned the models, while holding the pre-trained layers constant, and only trained the final layers we added. Each model was trained with a small learning rate (0.0002), using the RMSprop optimizer [128] ($\rho = 0.9, \epsilon = 1 \times 10^{-8}, \delta = 0.0$) and a categorical cross-entropy loss. The values of these hyper-parameters were selected empirically. With this configuration, we aimed to ensure the stability of the pre-trained values of each model.

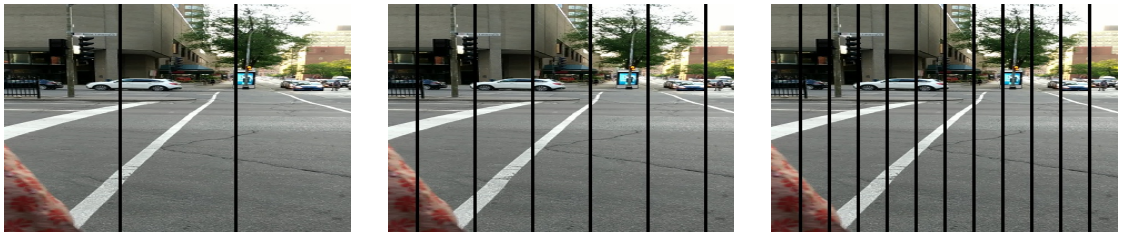
We then experimented with reducing the dimensionality of the action space. Starting from the arrangement of 12 bins, we generated the following three configurations:

- 4-actions space: $\mathcal{V}_1$, by combining $\{v_1, ..., v_4\}$, $\{v_5, ..., v_8\}$ and $\{v_9, ..., v_{12}\}$ into $\{v_{\text{left}}, v_{\text{straight}}, v_{\text{right}}\}$ respectively, plus the *unknown* action, as shown in 3.3a.
- 8-actions space: $\mathcal{V}_2$, by combining $\{v_2, v_3\}$, $\{v_4, v_5\}$,..., $\{v_{10}, v_{11}\}$, reserving bins $\{v_1\}$ and $\{v_{12}\}$ for those situations when the goal is not visible but its position can be inferred, as shown in Figure 3.3b
- 13-actions space: $\mathcal{V}_3$, retaining the full configuration of as shown in Figure 3.3c

For each of these configurations, we modified the associated Softmax layer to accord with the sizes of $\mathcal{A}_1 = \mathcal{V}_1, \mathcal{A}_2 = \mathcal{V}_2, \mathcal{A}_3 = \mathcal{V}_3$, and added $v_0 = unknown$. We then trained the CNN classifiers and evaluated their performance.

3.4.2 Testing the Agent

To evaluate the generalization of the learned policy, we created a second demonstration dataset from different intersections that were not included in the training set. Following the procedures described in Section 3.3, a supplementary collection of 51 videos was acquired, resulting in a new set $\mathcal{O} : \mathcal{Z}_o \times \mathcal{A}_o$ of 1170 observations. The optimal action for those samples was crowd-sourced



(a) 4-actions configuration (b) 8-actions configuration (c) 13-actions configuration

Figure 3.3: All actions-space configurations experimented while training and testing the agent.

to another ten experts who labeled each sample at least five times. The conditions described for labeling the initial set $\mathcal{D}$ were also followed here.

Table 1 presents the accuracy of the derived policy, applied over the observations on $\mathcal{O}$. These results are computed based on the best-predicted action of the classifier compared to the action that received the most votes from our experts. However, we note that best-action accuracy metrics are not meaningfully indicative of the model's actual performance on a practical task. Instead, Table 2 presents the mean absolute error in the agent's predicted action, a measurement computed by taking the absolute difference between the index of the action inferred by the policy from an *observation* and the index of the winning vote from the experts. Considering that our distribution of the action space is dependent on the spatial arrangement of the bins, we excluded the results of the action *unknown* in this calculation.

Model	**4-Action**	**8-Action**	**13-Action**
ResNet-50	0.746	0.635	0.503
Xception	0.822	0.615	0.526
Squeezenet	0.775	0.483	0.393
Mobilenet	0.822	0.599	0.467

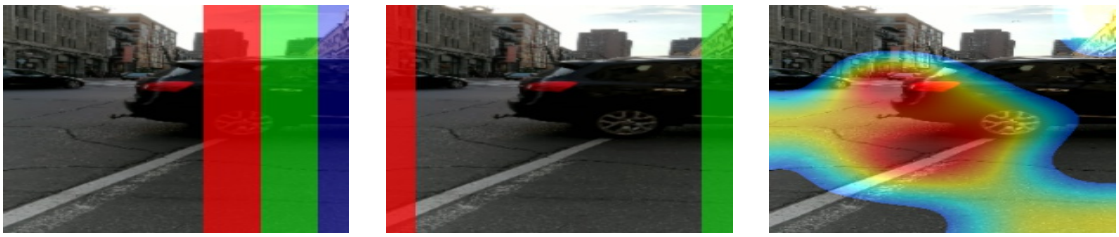
Table 1: Accuracy of each model in predicting the correct action, compared to the experts' optimal action.

Model	**4-Action**	**8-Action**	**13-Action**
ResNet-50	0.27 ± 0.03	0.61 ± 0.07	1.14 ± 0.12
Xception	0.20 ± 0.03	0.59 ± 0.07	1.05 ± 0.12
Squeezenet	0.26 ± 0.03	0.83 ± 0.07	1.37 ± 0.12
Mobilenet	0.20 ± 0.03	0.71 ± 0.08	1.24 ± 0.12

Table 2: Each model's mean absolute difference between predicted action and the experts' optimal action, presented with the corresponding 95% confidence margin.

As can be seen, relying solely on the accuracy metric would suggest that the agent exhibits poor

performance. However, given the mean absolute error reported—typically within a difference of a single bin—the average performance of the system is actually satisfactory across all model types and action space configurations. This can be verified by analysis of the confusion matrix for each action-space configuration. One can observe in Figure 3.5 a strong tendency around the diagonal in all four models, indicating that errors in the agent's prediction are most often the result of confusion with an adjacent, i.e., very similar, action. Thus, a mean absolute error metric is more appropriate than a simple correctness percentage score to characterize the performance of the model. Although we only present here the 8-actions configuration, similar behavior was exhibited for the other action-spaces tested.



(a) expert's prediction (b) agent's prediction (c) activation maps

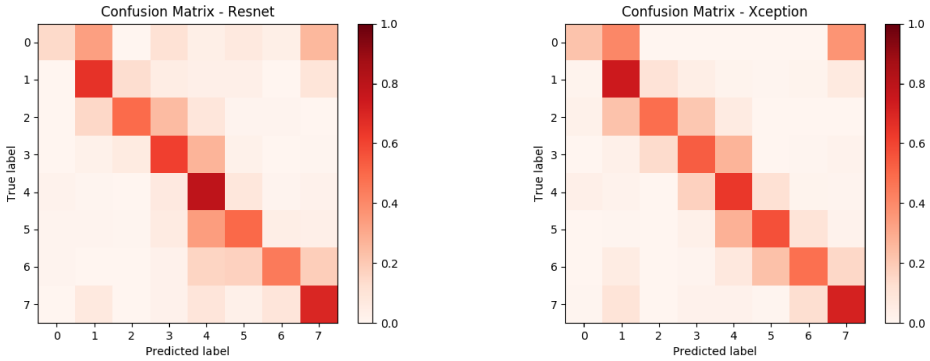
Figure 3.4: Mobilenet top-3 predictions ($blue, green, red$) vs. experts' predictions on the 8-action-space. A missing bin corresponds to *unknown*. (c) shows the CNN activation maps [115].

It is also interesting to note that for situations where the experts' optimal action was *unknown* (i.e., the correct label is 0), the agent would most often confuse it with the extreme veering conditions (i.e., actions 1 and 7). This suggests that when the expert is unsure of the required action, the agent's predictions recommend rotation. We suspect that this behavior is related to the way experts chose the optimal action in the training demonstrations; when the goal was not seen, the expert would choose the edge column that they guessed was the best direction to which one should rotate. Figures 3.4 and 3.5 make it evident that the agent has also learned this behavior. Although some perfect agreements between the policy and the expert's judgment are represented in the first row of Figure 3.6, there are still scenarios in which the goal is occluded and the policy is not capable of inferring the correct behavior, as shown in the third row of Figure 3.6.

3.4.3 Mobile Prototype

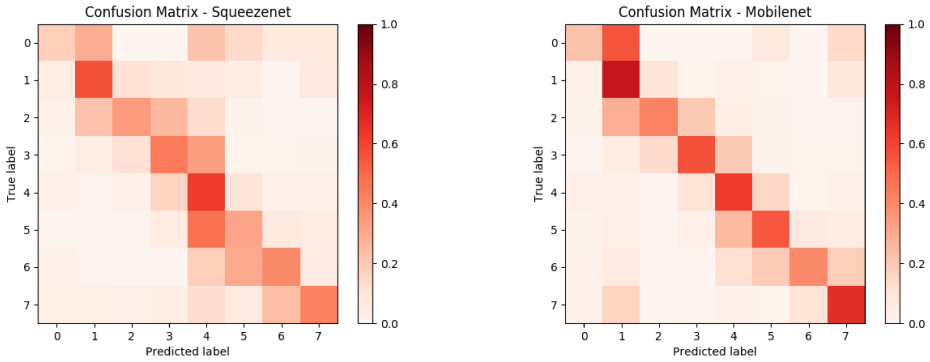
To evaluate the potential of our solution, we developed a prototype application, initially for the Android platform, presuming our users would possess nothing more than a smartphone and bone-conduction headphones as an aid to complete the task. While designing this prototype, we decided not to pursue an evaluation of factors such as energy efficiency, traffic data, or inference times. Instead, we implemented a mobile-only approach to gauge the feasibility of such an implementation empirically.

Our prototype performs three high-level tasks. First, using the front-facing camera of the smartphone, the system captures and pre-processes video frames. Next, a batch of these frames is fed into the pre-trained network for inference, based on Google's Tensorflow API for Android [2], and the



(a) Resnet50

(b) XCeption



(c) Squeezent

(d) MobileNets

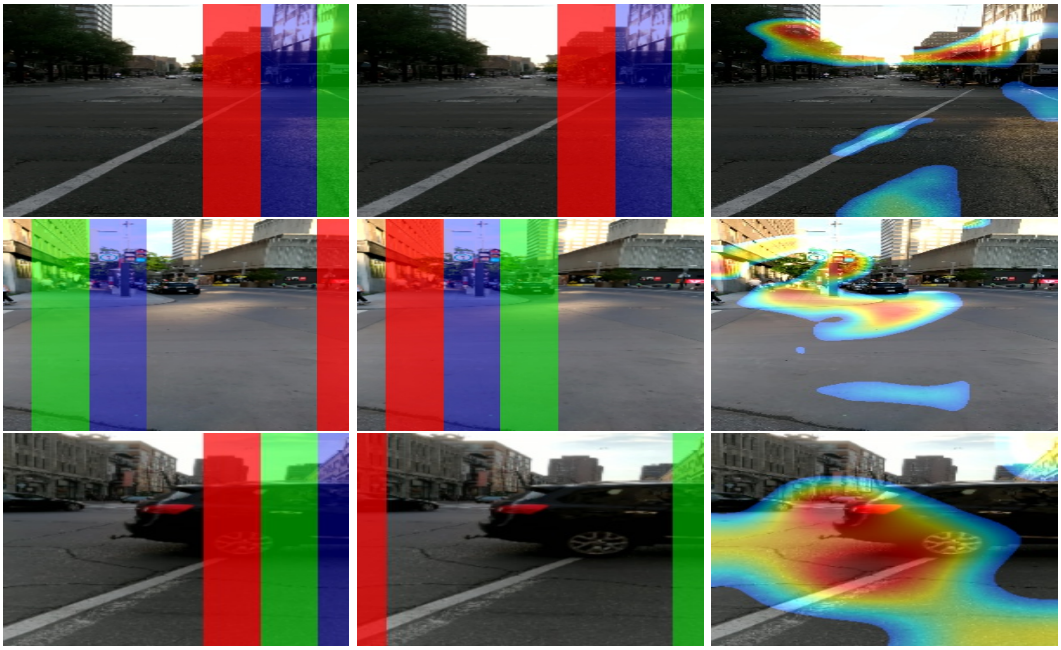
Figure 3.5: Confusion Matrix for each model trained on the 8-action-space configuration.

results are collected. The relative position of the goal is extracted from the policy contained in the model, and converted to an angle from the subdivision of the 90^o region in front of the user.

Based on those results, a stereo panning audio signal is rendered, representing a beacon the user should follow to reach the goal. This approach of rendering is intended to mimic the general assistive behavior of the Accessible Pedestrian Signal. Testing of this system with human subjects is expected to begin shortly.

3.5 Conclusion and Future Work

This chapter presented a basis for the construction of an intelligent assistive agent with the aim of helping the visually impaired with safely crossing road intersections. As discussed in Section 3.4, the LfD method implemented has effectively eliminated the need of extracting specific features, such as zebra stripes, from the environment (e.g. second row of Figure 3.6). Instead, it uses the advances in deep learning to extract the necessary features to derive an optimal policy. In all model configurations, we observed that the agent was capable of predicting close to optimal actions (when compared to experts), as seen by the confusion matrices in Figure 3.5.



(a) expert's prediction (b) agent's prediction (c) activation maps

Figure 3.6: Mobilenet top-3 predictions ($blue, green, red$) vs. experts' predictions on the 8-action-space. A missing bin corresponds to *unknown*. (c) shows the CNN activation maps [115].

However, as the analysis of the results showed, a lot of work still remains. Firstly, our method would greatly benefit from the collection of a larger number of demonstrations for both the training and testing processes. This increase in amount of data would undoubtedly help the generation of a robust agent, more capable of handling usual roads configurations. Another interesting problem would be to explore human-computer interaction aspect.

That is, how should one render the agent's outputs with each of the action-space configurations we presented. To answer this question, future work should aim at developing a smartphone application, built on these trained models. A user study with visually impaired individuals should then be conducted to evaluate the effectiveness of the various rendering possibilities. Moreover, it would be interesting to benchmark each model's battery consumption statistics and inference times.

Chapter 4

Interactive Imitation Learning via Uncertainty-aware Direct Policy Derivation

4.1 Introduction

The use of machine learning algorithms in robotics is becoming mainstream with applications in a rich and an increasing number of domains. As initially discussed in Chapter 2, safety is a critical component in the expansion of the domains to which imitation learning techniques can be applied. Today, there are numerous examples of learning algorithms controlling a self-driving car [23], helping people with disabilities [37], aiding in diagnosis and healthcare treatments [31], among many others areas. As this list keeps expanding the need for safety guarantees in learning algorithms continues growing.

A summary of the potential threats of the application of learning algorithms, and AI in general, to a real-life problem is presented by Amodei et. al. [10] where the authors identify at least five major sources of failure in AI methods and their underlying assumptions: *negative side effects*, *reward hacking*, *scalable oversight*, *safe exploration*, and *robustness to distributional shift*.

Chapter 2 presented a general overview of the problem of imitation learning, state of the art algorithms and its performance guarantees (contrasted with the expert's). The methods presented there provided solutions to the known limitations of the reduction of imitation learning to supervised learning: distributional shift. These algorithms introduced learning a policy interactively through the strategies of policy mixing, data aggregation and randomized exploration to solve the problem of the distributional shift, a process from that will be called from now on *Interactive Imitation Learning* (IIL) [105]. To clarify what the scope of IIL is, Algorithm 4.1 serves as a template for describing the general steps of IIL as it summarizes each of the strategies mentioned before.

Despite the formal guarantees offered by the algorithmic analysis presented in [107, 109, 110, 124], from the perspective of safety one can question whether the stochasticity added by implicit or explicit

execution of the learner's policy in earlier stages of training introduces safety concerns in an imitation learning algorithm. For instance, both DAgger [110] and AggreVaTe [109] analysis have assumed that π^* is able to *recover* from the change in the state space visitation distribution d_{π_i} induced by executing π_i ([107] Section 2.2 and [109] Section 2.2). Thus, it remains unclear whether random policy mixing and exploration strategies -like the one presented in Algorithm 4.2- could guarantee the safeness of an IIL learning procedure. As a thought exercise one may question what would happen in large-finite or infinite horizon tasks when uniformly sampling a time step (Algorithm 4.2, step 3) for exploiting a non-converged learner's policy or for execution of an exploration step that gives an amount of execution time t to any of those steps that disregard the global status of learning procedure.

Algorithm 4.1 General Interactive Imitation Learning

Require: π^* : expert's policy, N : iterations, $\mathcal{H}$: classifier, α: mixing coefficient

1: **algorithm** GIIL($\pi^*, \mathcal{H}$, N, α)
2: $\quad\mathcal{T} \leftarrow \emptyset$, $\pi_1 \in \Pi$
3: $\quad$**for** $i \leftarrow 1...N$ **do**
4: $\quad\quad$**mix** $\pi_i = \alpha_i \pi^* + (1 - \alpha_i)\pi_i$ ▷ $\alpha_i = p^{i-1}$: π^* execution decays over time.
5: $\quad\quad$**sample** $\mathcal{T}_i =$ RandomSampling() ▷ see Algorithm 4.2
6: $\quad\quad$**aggregate** $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_i$ ▷ optionally, if use a cost-sensitive classifier
7: $\quad\quad$**learn** $\mathcal{H}(\mathcal{T}) : \pi_i = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi_i}}[e_\pi(s)]$ ▷ $\mathcal{H}(\mathcal{T})$ cost-sensitive *or* $\mathcal{H}(\mathcal{T}_i)$ online learner.
8: $\quad$**return** π_i best on validation.

Algorithm 4.2 Random Sampling on Interactive Imitation Learning

1: $D \leftarrow \emptyset$
2: **for** $m \leftarrow 1..M$ **do**
3: $\quad$**sample** $t \sim U(1, T)$ ▷ randomly sample a time step t.
4: $\quad$**execute** π_i from $1..t-1$ ▷ execute in-training policy π_i.
5: $\quad$**explore** (s_t, a_t) ▷ random exploration step.
6: $\quad$**execute** π^* from $t+1...T$ ▷ use expert's policy π^* to recover.
7: $\quad D \leftarrow D \cup \langle s, a \rangle$
8: **return** D

Consequently, this chapter aims to establish the impact of considering safety issues over the Interactive Imitation Learning setting.

4.2 Related Work

The problem of safety in Sequential Decision Processes (SDP) has attracted a significant amount of research over the years [49]. This research characterizes the fundamental safety issue on SDP solution

algorithms: the ergodicity assumption. Most methods assume that the state space of the decision problem is *ergodic*: any state is reachable from another after a finite number of decision epochs. Regarding system safety, ergodicity implies that there are no regions or subsets of the state space from which an expert agent cannot recover. Particularly, common exploration techniques encourage inspection of infrequent regions (or subsets) of the state space as a method for policy improvement. Assuming ergodicity is impractical (and dangerous) in real physical systems [87]. While the literature agrees on the dangers of the ergodicity assumption, it disagrees over the approaches that could be implemented to prevent a learning system from reaching dangerous regions of the state space.

4.2.1 Safety on MDP

The work in [53, 52, 7, 44] proposes to ensure safety by performing a reachability analysis of the system state space (or its transition probabilities). This method assumes that it is possible to compute safe regions of the state space (*reachability sets*) to where the system can be kept constrained during execution. An issue with applying reachability analysis in many real systems is that it suffers from the curse of dimensionality: it is impractical to be applied to those systems where the dimensionality of the state (or observation) space is high. Nevertheless, the fundamental disagreement in literature relies on the argument of whether reachability sets are sufficient to ensure safety in an MDP. In the framework of Markov Decision Processes (MDP), guaranteeing safety constraints over exploration and exploitation strategies has been reflected by work in [50, 87, 48, 131, 130]. For instance, Moldovan and Abbeel [87] present an example of an MDP where finding safe regions using the partitions of the state space (visited states) –like in [53]– fails to guarantee safety. Instead, the solution Moldovan and Abbeel propose the use of the frequency of those visitations to ensure safety, a strategy equivalent to ensuring safety over the policy space. Interestingly, this perspective seems to be a consequence of the *one-step lookahead* characteristic of Reinforcement Learning (RL) algorithms.

It remains unclear then where either strategy can thoroughly ensure the safety of MDP-formulated machine learning systems. However, despite its MDP formulation, ensuring safety in Imitation Learning algorithms provides a different context. In an SDP (or an MDP), the learning agent starts with no knowledge of the task and it has to execute while learning and exploring. Having no previous knowledge implies the agent operates with no prior over which state-action pairs are safe or unsafe. García and Fernández [48] relates the introduction of demonstrations of expert behaviour as a method to bootstrap and control safe exploitation and exploration in RL, a proposition that further validates that ensuring safety on the IL framework is a more constrained problem: the expert offers an optimally-safe behaviour as it demonstrates a task as long as it can be considered rational and non-adversarial.

4.2.2 Safety on Imitation Learning

As a subset of the general SDP problem, the literature on safety guarantees in Imitation Learning is not as ample as it is for other solution methods. Probably, the Confident Execution framework by

Chernova & Veloso [29] is one of the earliest attempts to circumvent safety issues around IL. Confident Execution is a general purpose method that requires that the parametrization of the learner's policy to be any classifier capable of providing -along with the classification- a measurement of its confidence in the predicted value. Then, it requires an autonomy threshold value that defines when the control of the system is transferred from the learner to the expert and vice-versa. Although initially stated as a method for Active Imitation Learning [120, 68], introducing the learner's uncertainty into the framework ensures an adjustable autonomy setting where it was possible to alternate between autonomous and supervised execution safely.

In derivative work, Chernova and Veloso [30] extended the Confident Execution framework to consider a multi-thresholded approach as single threshold was deemed insufficient for classifiers of higher complexity (Support Vector Machines, k-Nearest Neighbors, and others) than the one initially used in [29] (Gaussian Mixture Model). Remarkably, the analysis of why a single threshold is insufficient reveals that in an interactive setting, as the decision boundaries of the classifiers are being defined while the training progresses, it is difficult to determine a single threshold for the uncertainty of the classifier for all classes in the problem. Furthermore, it is important to note that the selection of a unique threshold is also impacted by the separability of training samples on the state space and the representational capacity of the classifier. The multi-threshold solution proposed required a threshold value for each decision boundary provided one can extract the decision boundaries for each classifier.

Probably, the most comprehensive solution to safety in an Imitation Learning setting has been given by the Confidence-Based Autonomy framework proposed by Chernova & Veloso [28]. This algorithm extends the Confident Execution framework by explicitly incorporating the teacher's intervention at any decision epoch in the learning process. These interventions are considered Corrective Demonstrations and are fed to the classifier as part of the training procedure. The authors found that this technique, in the evaluated settings, was more effective than negative reinforcement while it made the learning agent correct its mistakes faster. One of the goals of the present work is to conciliate the Confidence-Based Autonomy framework with recent developments in Interactive Imitation Learning algorithms as recent developments in this area [72, 136, 86] are considered here as derivations of the Confident Execution framework, and that do not thoroughly ensure the safety of the learning process.

The work in [72, 136, 86] explores metrics for establishing the boundaries between the teacher and the learner agent. For instance, Kim & Pineau [72] (Maximum Mean Discrepancy-IL) establish a bound over the discrepancy between the learner and the expert policy predictions during training and uses a single threshold to determine how to mix these policies during training with DAgger [106]. Meanwhile, Zhang & Cho [136] (SafeDAgger) trained a binary safety classifier that predicts 0 when the error of the learner policy prediction compared to the expert's input is above a certain threshold and 1 when it is below. Then, the learner's non-converged policy is executed when the safety classifier predicts a safe scenario, otherwise, the expert is asked for demonstrations. Finally, Menda et al. [86] (DropoutDAgger) improved over [136] by discarding the safety classifier and estimating the uncertainty of the learner's policy parametrization using Monte Carlo Dropout [47]

as a Bayesian approximation. As will be discussed in the next sections, it will be interesting to analyze if these learner-centric algorithms are robust enough to overcome overconfident predictions of the classifiers and misspecification of the uncertainty threshold.

4.3 Uncertainty-Aware Interactive Imitation Learning

The problem of approximating a control policy $\hat{\pi}$ from a set of traces $\mathbb{T}$ of an expert policy π^* execution can be stated as the unconstrained minimization problem of finding the policy in a policies space Π that minimizes a cost function C that measures the agreement between the expert's and the learner's decisions at every state on S:

$$\hat{\pi} = \underset{\pi \in \Pi}{\arg\min} \quad \mathcal{C}(\pi(s), \pi^*(s)) \quad s \sim d_\pi \tag{10}$$

At first glance, the problem of imitation learning reduces to a supervised offline learning classification or regression problem. In this non-interactive formulation, the expert policy execution generates a distribution of visited states d_π and defines a region S_o of the state space that can be considered safe assuming the expert is always non-adversarial and rational. Thus, it is feasible to consider that offline non-interactive methods are less prone to present safety issues induced by a non-optimal policy execution. However, as discussed in Chapter 2, staying within the boundaries of this region does not yield the best performance results.

State-of-the-art IIL methods propose an interactive setting where the learner's and the expert's policies execution are interwoven and in which exploration steps are encouraged. However, leaving the expert-induced region imposes safety issues on this process. These issues are not only provoked by exploration steps but also by the unbounded execution of a frequently non-optimal and often non-converged policy parametrization.

4.3.1 Safety Boundaries on the State Space

An unusual characteristic of IIL algorithms during training is the distributions of the tasks state space S (e.g., all images of $128 \times 128 \times 3$ pixels) they induce. A detailed analysis of the General Interactive Imitation Learning (GIIL) framework -Algorithms 4.1 and 4.2- results in the division of the input space into at least three major safety regions or subsets of the state space. Each of these sets can be explained by (1) the agent that induces the state visitation distribution of the region and (2) which agent guarantees the safety in the region.

First, an *optimality* region S_o -Definition 4.4- is the region defined by all the samples of the state space visited as a product of the execution of the expert's optimal policy π^* and characterized by the state visitation distribution d_{π^*}.

Definition 4.4. *The optimality region $S_o \subset S$ is a region of the state space where $\forall s_t \in S_o = \{s_0, s_1, ..., s_N\}$, s_t is induced by the execution of the optimal policy π^*.*

Second, there exist a *recoverability* region S_r –Definition 4.5– containing samples induced by the mixture of policy $\alpha\pi^* + (1-\alpha)\hat{\pi}$ and the uniformly sampled timestep strategy. The excess of samples

in S_r (compared to S^o) originate from those instances of the state space visited as a consequence of mistakes made by executing the learner's policy or by random exploration steps (implicit or explicit exploration).

Definition 4.5. *The recoverability region $S_r \subset S$ is a region of the state space where $\forall s \in S_r$ there exist a finite number t of steps such that if we follow π^* for t timesteps then $s_t \in S_o$.*

Moreover, finally, the region defined by all the samples outside the boundaries of the recoverability region S_r but inside the perceptual state space S is from now on called *uncertainty* or *novelty* region S_u. These samples define the boundaries of the expert knowledge of the task (optimal or near optimal states) and constitute final error states from which the task cannot, or is not advisable to, be recovered.

Definition 4.6. *The uncertainty region S_u is composed by all state space samples such that $\forall s \in S_u, s \notin S_r$ and it is also given by the expression $S_u = S \setminus S_r$.*

Even if initially supervised learning approaches tried to estimate the underlying expert policy by only observing traces of execution inside the boundaries of S_o, the introduction of the traits discussed in Chapter 2 increased the exhaustiveness of the state space visitation. This analysis highlights an important issue: IIL algorithms have solved the distributional shift by pushing the boundaries of the subset of the state space where the distributional shift occurs. While these traces have indeed increased the robustness and performance of the learner's policy (knows how to react to a significant number of instances of the state space), this improvement has come at the expense of sacrificing the safety of the learning process.

4.3.2 Safe Interactive Imitation Learning

The existence of an *uncertainty* region is sufficient evidence that the problem of *distributional shift* persists for GIIL. The reachability of the state space instances inside S_u is not currently bounded by any of the learning techniques discussed despite being stochastically and implicitly encouraged by policy mixing, random sampling and exploration strategies in this framework.

Hence, it is essential to guarantee that the steps of policy mixing and random sampling induce ergodic traces in $\tau \in \mathbb{T}$. The ergodicity of the traces has to comply with either the expert's or the learner's knowledge. A *recoverability trace* – Definition 4.7 – crosses the boundaries of the optimal region S_o into the recoverability region S_r and back, provided the implicit or explicit exploration steps selected at random can guarantee the learner remains inside S_r, a situation for which no formal guarantees have been given.

Definition 4.7. *A trace (in state space) $\tau = \{s_0, s_1, ..., s_N\}$ is a recoverable finite subset of the state space induced by an imitation learning algorithm if $s_0, s_N \in S_o$ then $\exists s_t \in \tau$ where $s_t \in S_r$.*

Furthermore, there is a significant issue with the learner's execution at learning and testing phases: *the traces of π^* from which the learner derives $\hat{\pi}$ almost never contain the expert decisions in the uncertainty region S_u.* In simpler words, the state distribution $d_{\hat{\pi}}$ induced by learning $\hat{\pi}$ does

not exhaustively cover the full perceptual state S, and it is far from the real underlying distribution $p(S)$ for high-dimensional perception inputs. Consequently, *robustness to distributional shift* is not an inherent property of *GIIL* or any policy derivation technique that uses a finite number of policy traces over optimal or near-optimal states. Thus, in order to guarantee the safety of a decision-making system where its policy has been derived from demonstrations it is necessary to construct exhaustive policies:

Definition 4.8. *A learned policy $\hat{\pi}$ is considered exhaustive if and only if $\forall s \in S, \exists a$ such that $\hat{\pi}(a|s) > 0$*

Creating an exhaustive policy by solely using traces of the optimal policy is not trivial. There are at least two significant steps that need to be taken for creating such type of policy: (1) to detect samples outside (inside) the recoverability region S_r and (2) to define a suitable action for out-of-distribution samples at test time. Probably, the complexity of both steps is one of many factors making *distributional shift* a pervasive problem on the application of machine learning algorithms [10], not only in the GIIL setting. Then, it is possible to define the safety requirements for the GIIL framework formally:

Definition 4.9. *A General Interactive Imitation Learning process is safe if and only if it is possible to guarantee that:*

(a) RECOVERABILITY*: for all $\tau = \{s_0, s_1, ..., s_N\} \in \mathbb{T}$, the start and final states $\{s_0^\tau, s_N^\tau\}$ are in S_o and should exits $s_t^\tau \in \tau$ such that $s_t^\tau \in S_r$.*

(b) EXHAUSTIVENESS*: $\exists \hat{\pi} \in \Pi$ that guarantees that $\exists a$ so that $\hat{\pi}(a|s) > 0$ for all $s \in S$.*

4.3.3 Safe Imitation Learning via Uncertainty Estimation

One of the strategies that has been successfully employed to guarantee safety in an otherwise utterly stochastic exploration processes is *bounded exploration*. Bounded exploration refers to the set of techniques that constrains the execution of an agent into a region of the state space where even the worst outcomes are recoverable [10]. By its nature, GIIL reduces the bounded exploration task to defining the partition $\{S_o, S_r, S_u\}$ of the state space induced during the learning and testing phases and by guaranteeing that the traces generated during these phases are, at least, *recoverability* traces. In GIIL, *bounded exploration* setting is slightly different to a general MDP learning approach (reinforcement learning) while the expert's knowledge of π^* and the induced state visitation distribution could define the recoverability set S_r of the state space. Furthermore, the term exploration does not only refer to implicit exploration steps included in the framework but also to the execution of the non-converged learner's policy.

Recent literature [8, 32, 69, 86, 81] has proposed uncertainty as a measure for modelling an agent's knowledge (or lack of it) in an IIL process. Generally, the learner's policy is parametrized by machine learning models with sufficient capacity for the task at hand (e.g., neural networks). Hence, referring to the uncertainty of the learner's policy reduces to estimate the uncertainty of the underlying model used to parametrize the policy. Numerous sources can cause uncertainty in

a model prediction: noisy measurements, misspecification of the hypothesis class (inductive bias), uncertainty in the hypothesis parameters, or -at test time- out-of-distribution samples [46].

The uncertainty of a machine learning model (or any parametric estimation of a phenomenon) is further divided into two major types according to the factors that origin them: *aleatoric* and *epistemic* [90, 47, 46]. Aleatoric uncertainty describes the inherent variability associated with the physical system or the environment under consideration. Epistemic uncertainty is a potential inaccuracy in any phase or activity of the modelling process that is due to the *lack of knowledge*. Although this distinction is customary in general machine learning settings, in GIIL, of more relevance to estimate are: (1) the uncertainty of the teacher $\mathcal{U}_{\pi^*}(s)$ and (2) the uncertainty of the learner $\mathcal{U}_{\pi_i}(s)$ over a particular state of the environment. With access to any function of the form $\mathcal{U}(s) : S \mapsto \mathbb{R}_{\geq 0}$, it is feasible to pose the Safe GIIL as a minimization problem. Bounding the states visited during the learning process with the expert's and the learner's uncertainty constraints the subset of policies Π_{safe} the optimization process can find.

Then, it is possible to devise at least three different optimization problems constrained by the uncertainties available during learning time. Firstly, it is desirable to design a Safe Interactive Imitation Learning with Expert Uncertainty minimization problem where, for all visited states of the state space S and having the expert's uncertainty is below certain threshold δ^* the process of deriving a safe policy π_{safe} is defined by:

$$\begin{aligned} \hat{\pi}_{safe} = \underset{\pi \in \Pi_{safe}}{\arg\min} \quad & \mathcal{C}(\pi(s), \pi^*(s)) \quad s \sim d_\pi \\ \text{s.t.} \quad & \mathcal{U}_{\pi^*}(s) < \delta^*, \quad s \sim d_\pi \end{aligned} \tag{11}$$

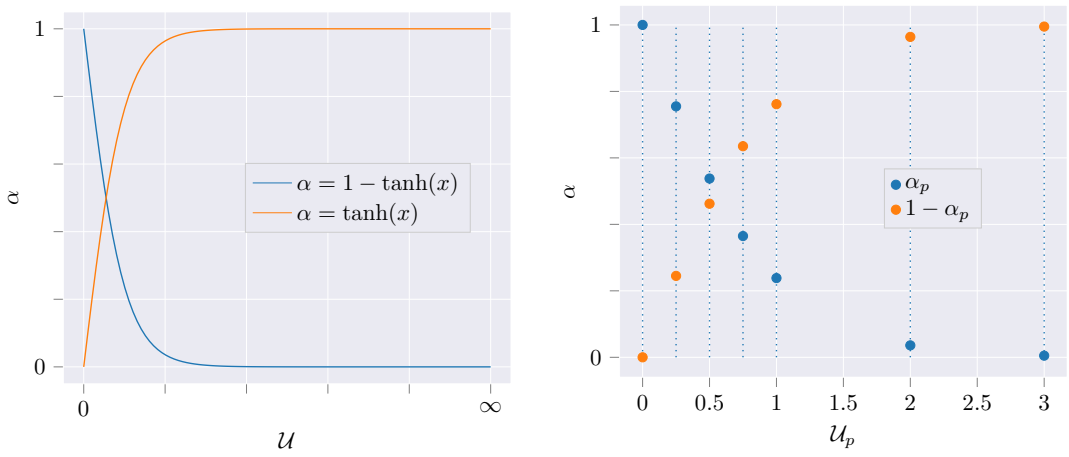
Similarly, the converse problem, *Safe Interactive Imitation Learning with Learner Uncertainty* is the minimization problem that uses the learner's uncertainty to define the constrained region of the state space where the uncertainty function $\mathcal{U}_{\pi_i}$ at timestep i is bounded by a user-defined value δ and it is defined by the expression:

$$\begin{aligned} \hat{\pi}_{safe} = \underset{\pi \in \Pi_{safe}}{\arg\min} \quad & \mathcal{C}(\pi(s), \pi^*(s)) \quad s \sim d_\pi \\ \text{s.t.} \quad & \mathcal{U}_{\pi_i}(s) < \delta, \quad i = 1 \dots N, \quad s \sim d_\pi \end{aligned} \tag{12}$$

Also, it is desirable to devise a third problem inspired by the notion of shared trust: *shared uncertainty*. In this setting, Safe Interactive Imitation Learning with Shared Uncertainty merges the expert's and the learner's uncertainty estimation functions such that:

$$\begin{aligned} \hat{\pi}_{safe} = \underset{\pi \in \Pi_{safe}}{\arg\min} \quad & \mathcal{C}(\pi(s), \pi^*(s)) \quad s \sim d_\pi \\ \text{s.t.} \quad & \mathcal{U}_c(s) < \delta_c, \quad i = 1 \dots N, \quad s \sim d_\pi \end{aligned} \tag{13}$$

where $\mathcal{U}_c = \alpha\, \mathcal{U}_{\pi^*}(s) + (1-\alpha)\, \mathcal{U}_{\pi_i}(s)$. Then, the problem is to incorporate these constraints into the GIIL framework described by Algorithm 4.1 and to investigate how each of the strategies of *policy mixing* and *random sampling* are affected by those changes.



(a) Mixture value α as a function of the uncertainty $\mathcal{U}$, e.g,: the uncertainty function $\mathcal{U}$ is composed with $tanh(x)$.

(b) Behavior of *Preferential Policy Mixing* (PPM) $\alpha_p\pi_p + (1 - \alpha_p)\pi_s$. As $\mathcal{U}_p(s) \to 0$ the primary policy π_p takes control, conversely $\mathcal{U}_p(s) \to \infty$ cedes control.

Figure 4.1: Uncertainty-aware policy mixing.

4.3.4 Uncertainty-Aware Policy Mixing

As explained before, the policy mixing step is one of the most distinctive characteristics of the GIIL framework:

$$\textbf{mix}\ \pi_i = \alpha_i\pi^* + (1 - \alpha_i)\pi_i \quad (\alpha_i = p^{i-1})$$

That is one of the reasons why algorithms like MMD-IL [72], SafeDAgger [136] or DropoutDAgger [86] have implemented their uncertainty-aware safety mechanism around this decision rule by enforcing the execution of π^* only when $\mathcal{U}_{\hat{\pi}}$ is above a certain threshold. These types of methods are related to the problem Safe Interactive Imitation Learning with Learner Uncertainty and they are (along with the expert's uncertainty variant) special cases of the formal specification of Uncertainty-aware Preferential Policy Mixing (UPPM).

Uncertainty-Aware Preferential Policy Mixing

Generally, UA-IIL systems design requires to express in the policy mixing (or the IIL system in general) a preference for either $\mathcal{U}_{\pi^*}$ or $\mathcal{U}_{\hat{\pi}}$ as the measurement of uncertainty driving the system. In order not to lose generality, π_p denotes the policy our system expresses preference for and $\mathcal{U}_p(s) : S \mapsto \mathbb{R}^+$ a function that assigns to each state (or decision) an estimate of π_p's uncertainty. The problem is to find a mixing coefficient $\alpha_p : f(U_p)$, $0 \geq \alpha_p \geq 1$ expressed as a function of the $\mathcal{U}_p$ that ensures preference over π_p. It is required then that α_p guarantees the following conditions:

1. π_p asymptotically takes full-control of the if $\mathcal{U}_{\pi_p} \to 0$.

2. π_p asymptotically transfers control to a secondary policy π_s as $\mathcal{U}_p \to \infty$.

3. π_p and π_s are executed at random with probability α_p and $1-\alpha_p$ respectively.

This implicates that, for a linear interpolation $\alpha_p\pi_p + (1-\alpha_p)\pi_s$ the preferential policy π_p would be executed with probability 1 when $\mathcal{U}_p$ reaches its minimum or would not be executed at all when it reaches a maximum (or a saturation point). This can be mathematically stated as in Equation 14, and presented in Figure 4.1b:

$$\begin{aligned} \lim_{\mathcal{U}_p \to 0^+} \alpha_p\pi_p + (1-\alpha_p)\pi_s = \pi_p \\ \lim_{\mathcal{U}_p \to \infty} \alpha_p\pi_p + (1-\alpha_p)\pi_s = \pi_s \end{aligned} \tag{14}$$

Hence, if preference (or access to $\mathcal{U}_{\hat{\pi}}$) over the learner's uncertainty must be expressed while designing a system, it is possible to clearly state the form of the mixture with the following expression:

$$\pi_i = (1-\alpha_p)\pi^* + \alpha_p\pi_i \tag{15}$$

If rather, one must express a preference (or access to $\mathcal{U}_{\pi^*}$) over the expert's uncertainty, the policy mixture would take the form of:

$$\pi_i = \alpha_p\pi^* + (1-\alpha_p)\pi_i \tag{16}$$

Equation 15 encases the preference expressed by algorithms like MMD-IL [72], SafeDAgger [137] or DropoutDAgger [86] where the system control policy is selected by following values of the learner's uncertainty function. As discussed, uncertainty-aware learner-driven systems have the potential to be exposed to overconfident predictions or under(over)estimation of the uncertainty and do not explicitly offer the expert's an opportunity to intervene in such cases. Hence, this form of preferential mixing with learner's uncertainty does not completely guarantee the Recoverability property enunciated in Section 4.3.2. On the other hand, there is no previous reference in the literature of algorithms that implement preferential policy mixing with expert's uncertainty -Equation 16. Even if the strategy of expressing a preference over the expert policy does permit the teacher intervention at any time t, it would also be hard to offer any safety guarantees that are not correlated to determine expert attention over the horizon of the task.

Nevertheless, both approaches fail to guarantee the rational selection of a policy under uncertainty as they do not take into account the uncertainty of the secondary policy π_s to transfer control, – e.g., what if the expert is not attentive, what if the learner is uncertain, or both cases happen. This problem violates the principle of Exhaustiveness asserted in Section 4.3.2 that establishes that a secure IIL system must guarantee safe execution across the state space of the task.

Uncertainty-Aware Rational Policy Mixing

Having analyzed the UA-PPM framework before, a desirable property of an IIL system is then it should not express in its design preference for either the learner's or the expert's uncertainty estimation. Instead, it should establish a collaborative approach where the level of autonomy is adjustable by alternating autonomous execution with an expert demonstration. This proposition is

aligned with the framework of Confidence-Based Autonomy proposed in [30] that was discussed in the Related Work (Section 4.2).

For the design of this system, it is required to assume that π_p, π_q (and to ensure the generality of the analysis) are the policies in the mixture, and that it is possible to obtain access to both $\mathcal{U}_{\pi_p},\ \mathcal{U}_{\pi_q} : S \mapsto \mathbb{R}_{\geq 0}$ as uncertainty estimation functions for each policy. Furthermore, to express the neutrality of the system $\alpha_{\pi_p} : f(U_{\pi_p}),\ \alpha_{\pi_q} : f(U_{\pi_q})$ are used as mixing coefficients for each policy and a function of their respective uncertainties. As before, the problem is to find the mixing coefficients $\alpha_{\pi_p}, \alpha_{\pi_p}$ as a function of uncertainty estimation such that the behavior of mixture of policies with the form $\pi_{RM} = \alpha_{\pi_p}\pi_p + \alpha_{\pi_q}\pi_q$ ensures the following conditions are met:

1. Consistency or the ability to maintain probabilistic consistent coefficients $\alpha_{\pi_p} + \alpha_{\pi_p} = 1$.
2. Rationality or the ability to asymptotically select the best policy under uncertainty.
3. Neutrality or the ability to asymptotically assign equal probabilities to equally uncertain policies.
4. Impossibility or the ability to detect an impossible decision under uncertainty where no policy can be confidently selected.

These four conditions can be expressed in terms of the behaviour of the mixture through the asymptotic limits of each uncertainty estimation function $\mathcal{U}_p,\ \mathcal{U}_q$ and can be described with the following mathematical expressions:

$$\begin{aligned}
\alpha_{\pi_p} + \alpha_{\pi_q} &= 1 && \text{(consistency)} \\
\lim_{\mathcal{U}_p \to 0,\ \mathcal{U}_q \to \infty} \pi_{RM} &= \pi_p && \text{(rationality)} \\
\lim_{\mathcal{U}_p \to \infty,\ \mathcal{U}_q \to 0} \pi_{RM} &= \pi_q && \text{(rationality)} \\
\lim_{\mathcal{U}_p \to 0,\ \mathcal{U}_q \to 0} \pi_{RM} &= \frac{1}{2}\pi_p + \frac{1}{2}\pi_q && \text{(neutrality)} \\
\lim_{\mathcal{U}_p \to \infty,\ \mathcal{U}_q \to \infty} \pi_{RM} &: \text{undefined} && \text{(impossibility)}
\end{aligned} \tag{17}$$

Under these conditions, a rational mixture of policies π_{RM} guarantees that a more certain policy has a higher probability of being selected *(rationality)*, a behaviour that ensures the system can consistently respond over any state, at any decision epoch (Exhaustiveness). For those cases when no policy can be rationally selected *(impossibility)* an undefined scenario is created and left open to the algorithm designer to determine how to act. Another important feature is that any reasonable selection of the policy must not show, across the IIL training phase, any preference *(neutrality)* for any of the policies involved $(\pi^*, \hat{\pi})$. The neutrality of the mixture typically allows a level of cooperation equal to the one presented in the Confidence-Based Autonomy framework.

4.3.5 Uncertainty-Aware Rational Sampling

Random sampling -Algorithm 4.2- is another characteristic of GIIL algorithms that needs to be analyzed from the optics of its impact on safety issues. Generally, this strategy is used to collect

demonstrations from the expert's policy by randomly (uniformly) sampling a timestep t between the beginning and horizon of the task $t \sim U(1, T)$. This time step defines three partitions of the sampling procedure where:

1. From $1..t-1$, the system exploits π_i.
2. At timestep t, the system explores (s, a).
3. From $t+1...T$, the system executes π^* (receives corrective demonstrations).

This sampling strategy does not match the mixed iterative control objective of a Safe Interactive Imitation Learning algorithm. When $t \sim U(1, T)$ results in values of $t \geq T/2$, it is impossible to control the system at early stages of training as the non-converged policy receives an amount of time that is not proportional to its uncertainty (or training error). This phenomena could happen with probability $U(t \geq T/2) = 0.5$, rendering the learning system unsafe if π_i parametrization is non-converged. If executing π_i guarantees expert's intervention at any time -e.g., using Uncertainty-Aware Rational Policy mixing, it is possible to remove the corrective step (π^* execution) in the random sampling procedure. Then, it is only required to find a function $t : f(\mathcal{U})$ of the learner's uncertainty $\mathcal{U}_{\hat{\pi}}$ such that balances the exploitation (execution of π_i) and exploration (execution of an exploration strategy $\mathcal{E}$). Such a function should guarantee:

> RATIONALITY or the ability to asymptotically select exploitation or exploration under uncertainty.

In this context, rationality enforces that the exploration strategy $\mathcal{E}$ is only executed in low uncertainty contexts and that the execution of the mixture policy π_i is under high uncertainty contexts. This behaviour can be stated as:

$$\begin{aligned} \lim_{\mathcal{U}_{\hat{\pi}} \to 0} t &= 0 \quad \text{(explore)} \\ \lim_{\mathcal{U}_{\hat{\pi}} \to \infty} t &= T \quad \text{(exploit)} \end{aligned} \tag{18}$$

After each time step, this strategy now partitions the horizon T into segments for exploitation and exploration:

1. From $1..t-1$, the system executes π_i.
2. From $t..T$, the system executes $\mathcal{E}$.

For this to work, it is necessary to re-compute $t = f(U_{\pi_i})$ on every step as the uncertainty estimation of the learner evolves through training. Another way to present this trade-off between the execution of π_i and $\mathcal{E}$ is by representing the uncertainty-aware partition of the sampling as Uncertainty-Aware Preferential Policy Mixture (PPM) π_a between π_i and $\mathcal{E}$ expressing a preference for π_i:

$$\pi_a = \alpha_{\pi_i} \pi_i + (1 - \alpha_{\pi_i}) \mathcal{E}, \quad (\alpha_{\pi_i} = f(\mathcal{U}_{\hat{\pi}})) \tag{19}$$

From this expression, it is easier to detect a significant drawback of this method: it is highly susceptible to underestimation of the learner's uncertainty. In such cases, the exploration strategy $\mathcal{E}$ would be probably selected more frequently. This behaviour would impede the expert's intervention through π_i. Hence, it is desirable to express $\mathcal{E}$ in a way that can allow expert's policy execution during exploration. A straightforward solution to this problem could be to formulate an exploration strategy $\mathcal{E}'$ through a PPM with the expert policy π^*, and that expresses preference over π^*:

$$\mathcal{E}' = \alpha_{\pi^*}\pi^* + (1 - \alpha_{\pi^*})\mathcal{E} \tag{20}$$

This analysis completes the formalization of the Uncertainty-Aware Rational Sampling mechanism that can now be re-stated as:

$$\pi_{RS} = \alpha_{\pi_i}\pi_i + (1 - \alpha_{\pi_i})\mathcal{E}', \quad (\alpha_{\pi_i} = f(\mathcal{U}_{\hat{\pi}})) \tag{21}$$

4.3.6 Uncertainty-aware Rational Policy Mixing and Sampling

Having established a mechanism to incorporate uncertainty measurement into the steps of *policy mixing* and *random sampling*, it is possible to devise modification to the GIIL framework that can guarantee safety during the learning process. Algorithm 4.3 depicts Uncertainty-Aware Policy Mixing and Sampling (UPMS), a method that by modifying the mixing and sampling strategies introduces expert-bounded safety into the GIIL framework.

Algorithm 4.3 UNCERTAINTY-AWARE POLICY MIXING AND SAMPLING

Require: π^* : expert's policy, N : iterations, T : horizon, $\mathcal{H}$: classifier, $f(U_{\pi^*})$: expert's uncertainty, $f(U_{\hat{\pi}})$: learner's uncertainty, $\mathcal{E}$: exploration strategy

algorithm UPMS($\pi^*, \mathcal{H}$, N, $f(U_{\pi^*})$, $f(U_{\hat{\pi}})$)
  $\mathcal{D} \leftarrow \emptyset$, $\pi_1 \in \Pi$
  for $i \leftarrow 1...N$ **do** ▷ Execute algorithm through N iterations.
    compute $\alpha_{\pi^*} \leftarrow f(U_{\pi^*})$, $\alpha_{\hat{\pi}} \leftarrow f(U_{\hat{\pi}})$
    mix $\pi_{RM} = \alpha_{\pi^*}\pi^* + \alpha_{\hat{\pi}}\hat{\pi}$ ▷ Mix learner's and expert's policies.
    sample $\mathcal{D}_i$ = RATIONALSAMPLING() ▷ see Algorithm 4.4
    aggregate $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ ▷ optionally, aggregate data for offline learning.
    learn $\mathcal{H}(\mathcal{D}) : \hat{\pi} = \text{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi_{RM}}}[e_\pi(s)]$ ▷ $\mathcal{H}(\mathcal{D})$ *or* $\mathcal{H}(\mathcal{D}_i)$ online.
  return $\hat{\pi}$ best on validation.

Following the discussion in the preceding sections, this algorithm does guarantee expert intervention during the complete execution of the learning system. As it is composed by rationally mixing and sampling, Algorithm 4.3 can ensure that the RECOVERABILITY, and EXHAUSTIVENESS requirements enunciated in Section 4.3.2 are maintained throughout the training phase. However, yet to be proved are the practical applications of the changes introduced.

Algorithm 4.4 Uncertainty-Aware Rational Sampling

$D \leftarrow \emptyset$
for $m \leftarrow 1..T$ **do** ▷ executes Rational Sampling through the horizon T.
  $\alpha_{\pi_i} \leftarrow f(\mathcal{U}_{\hat{\pi}})$ ▷ compute mixing coefficient.
  execute $\pi_{RS} = \alpha_{\pi_i}\pi_i + (1 - \alpha_{\pi_i})\mathcal{E}'$
  $D \leftarrow D \cup (s, a^*)$ ▷ add (s, a^*) pairs when π^* is executed.
return D

Consequently, it is *de rigueur* to experimentally verify if these modifications impact the learning procedure of the GIIL framework and if they indeed guarantee safety in the process. By contrasting UPMS with previous methods on tasks for which they have excelled before, it would be possible to the determine the reach of these changes and how to possibly mitigate negative impacts they may have over the training phase.

4.3.7 Uncertainty Estimation

The applicability of uncertainty-aware algorithms to IIL is highly conditioned on the methods used to estimate the uncertainty of the learner's (e.g., DropoutDAgger) and also the teacher's (like in UPMS). Hence, it is essential to define how to compute the uncertainty functions $\mathcal{U}_{\hat{\pi}}$ and $\mathcal{U}_{\pi^*}$.

Estimate the expert's uncertainty when the expert is human is a difficult task. In the case of the lane following task, the uncertainty of a human driver could be related to different factors including attention, confidence in the learner's autonomy, confidence in self-ability, and several others. As modelling, such a complex behaviour may prove to be challenging, here a simplified version of this function is implemented. For these experiments, it is safe to assume that the following function measures the expert's uncertainty:

$$\mathcal{U}_{\pi^*} = \begin{cases} 0 & \text{(if providing input)} \\ \infty & \text{(otherwise)} \end{cases} \tag{22}$$

provided the expert is always attentive during the learning process. Although simple, this function suffices for the analysis of UPMS -the only of the algorithms investigated that requires teacher's uncertainty- and it would be left to future work to investigate whether a more complex specification of this function improves the overall performance of UPMS.

An Overview of Model Uncertainty

The selection of a CNN as the policy parametrization has the undisputed advantage of increasing the complexity of the policy space that can be explored. However, the representational power they bring comes attached with the difficulty of estimating the uncertainty of these type of models. Learning a parametrization of a policy $\pi_\theta(a|s)$ from a set of traces $\mathbb{T} : S \times A$ can be computed by posing learning as an optimization process of the form:

$$\hat{\theta} = \underset{\theta \in \Theta}{\arg\min} \quad \mathcal{C}(\pi, \pi^*; \theta) \tag{23}$$

where $\hat{\theta}$ is the best fixed point estimate of the parameters. However, this fixed point estimate alone does not suffice for computing the uncertainty of a model. Instead, it is possible to formulate the learning process as inference in a Bayesian setting [46] by treating the parameters θ as a random variable and establishing a prior assumption over its distribution $p(\theta)$:

$$p(\theta|s,a) = \frac{p(a|s,\theta)p(\theta)}{p(a|s)} \tag{24}$$

The distribution $p(\theta|s,a)$ represents the most probable parameters for the state and action pairs given as input to our model. In Equation 24, $p(\theta)$ represent our prior assumption over the parameters and $p(a|s)$ is a probability distribution given by the observation of the traces $\mathbb{T}$. Then, if one aims to compute the uncertainty of any parametrization, it is necessary to compute the posterior proposed in Equation 24. As this posterior cannot be computed analytically [46, 51], another parametrized distribution $q_\omega(\theta)$ is used to approximate it, given the following optimization procedure:

$$q^*_\omega(\theta) = \underset{q_\omega(\theta)}{\operatorname{argmin}} \, KL_D(q_\omega(\theta)||p(\theta|s,a)) \tag{25}$$

where KL_D is the Kullback-Leibler divergence (relative entropy) and serves as a asymmetric measure of how a probability distribution diverges from another. Even the value of $KL_D(q_\omega(\theta)||p(\theta|s,a))$ in Equation 25 is not tractable by analytic methods as it depends on the joint probability of $p(s,a)$. To solve this equation, a set of techniques called *variational inference* [22] is usually employed. There are at least three major groups of methods to solve the variational inference problem: mean-field and stochastic variational inference, stochastic gradients of the Evidence Lower BOund (ELBO) and non-field methods. As a detailed analysis of the algorithms of these groups is outside the scope of the present work, the reader may find in [21] a recent survey in this literature.

Unobtrusive Uncertainty Estimation

Even if Bayesian Machine Learning (BML) models are ideal candidates to learn tasks where the estimation of the uncertainty of the model is crucial, there are several shortcomings of these methods that need to be addressed. Firstly, state-of-the-art BML models do not scale well. When the dimensionality of the parameter space Θ is high (the curse of dimensionality), just imposing a Gaussian distribution of the form $\mathcal{N}(\mu, \sigma; \theta)$ as a prior over the parameters of a model would increase their memory consumption and reduce their inference performance. Furthermore, changes of this type would affect existing models that have already obtained significant performance gains by using a large number of parameters (e.g., DNN, CNN). These are probably the main reasons why most uncertainty estimation methods developed to date that can demonstrate practical applications try to find approximations to the Bayesian modelling framework from an unobtrusive perspective.

This unobtrusive perspective has been implemented through the design of methods that leverage techniques already used in Deep Learning. To date, it is possible to account for modifications introduced on stochastic regularization techniques [47], ensembles of networks [78], weight perturbation

through optimization methods [71], and batch normalization [126] to be used to estimate uncertainty unobtrusively over DNN. These algorithms aim to simulate the behaviour of Bayesian models by introducing a reduced and manageable number of modifications to existing architectures.

From all these methods, uncertainty estimation via stochastic regularization techniques is probably the most widely applied method with demonstrated success in practical applications. The method -presented first in [47]- proposes a connection between Gaussian Processes and the use of Dropout [123] in Neural Networks by placing this stochastic regularizer after every layer of a DNN. Stochastic regularization is commonly used in learning DNN with the aim to regularize these highly parametrized models, that tend to overfit the training data. Thus, Monte Carlo Dropout -as this method is also known- allows the computation of the predictive mean and variance of a parametrized h_θ model evidence $p(y|x)$ by reducing its computation to T stochastic (with dropout probability less than 1 at test time) forward computations of its prediction [47]:

$$\begin{aligned} p(y|x) &= \frac{1}{T}\sum_{t=1}^{T} h_\theta^t(x) \\ \sigma(y|x) &= \frac{1}{T}\sum_{t=1}^{T} h_\theta^t(x)^2 - \left(\sum_{t=1}^{T} h_\theta^t(x)\right)^2 + \frac{1}{T}\sum_{t=1}^{T}\sigma(x) \end{aligned} \tag{26}$$

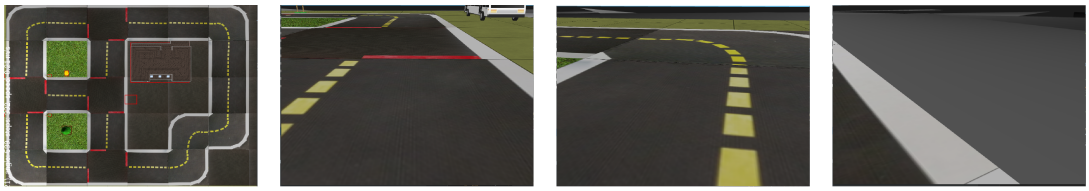
In the initial work and subsequent contributions, the authors have demonstrated the applicability of this method to Convolutional Neural Networks and Recurrent Neural Networks [45] architectures. Furthermore, the method has been successfully implemented in practical applications to estimating uncertainty in computer vision models (classification, semantic segmentation) [70] or as an uncertainty method for autonomous driving pipelines (semantic segmentation, object detection) [85]. Here, due to these properties and along with its use in *DropoutDAgger*, Monte Carlo Dropout is the method selected to compute the learner's policy parametrization uncertainty $\mathcal{U}_{\tilde{\pi}}$. To obtain the results described in the next section, the predictive mean and variance were estimated with $T = 16$ forward passes of the CNN with a *Dropout* probability of 0.5.

4.4 Experiments

This section presents the application of UPMS to the problem of lane following to explore the impact of using Uncertainty-aware Policy Mixing and Sampling (UPMS) in the autonomous driving domain. Using a single monocular camera as input and a simulated environment, the experiments also develop a comparative analysis of the safety, performance and learning efficiency of UPMS against other state-of-the-art IIL algorithms.

4.4.1 Duckietown OpenAI Gym Environments

The Duckietown OpenAI Gym Environments (DOGE)[84] is an OpenAI Gym[25]-style simulation platform for autonomous driving. It provides a toolkit for developing and benchmarking autonomous driving solutions in a simulated world that closely resembles the structure of a real Duckietown [98] configuration. The iterative process of training and testing algorithms in this simulator makes



(a) Duckietown OpenAI Gym Environments: UdeM 1 map.

(b) Agent in-lane position, receives a $R_t \approx 1$.

(c) Agent out-of-lane position, receives a $R_t \approx -1$.

(d) Agent close to world bounds, receives a $R_t = -1000$.

Figure 4.2: Duckietown OpenAI Gym Environments: agent's observations in different states and their approximate rewards.

it a suitable tool for benchmarking UPMS and the GIIL framework. The *sense-think-act* cycle implemented in this simulation allows fine-grained control commands (actions) to be sent to the environment to which it responds with an observation (a 120x160 colour image), a reward and a series of control flags. DOGE also allows the interaction with the environment via different control interfaces (joystick or keyboard) suitable for providing human demonstrations of autonomous driving tasks.

Regarding safety research, training and testing in a simulated environment before deploying to real-world scenarios are fundamental. Two characteristics make this simulator especially suitable for exploring safety issues in an autonomous driving setting: (1) its reward shape and (2) the control flags provided. The reward per timestep t in DOGE is a linear combination of features with the following form:

$$\mathcal{R}_t = \begin{cases} ||\vec{v}||_2^2 \cdot dir(\vec{v}) - 10d - 40c & \text{if inside the world bounds} \\ -1000 & \text{otherwise} \end{cases} \tag{27}$$

where $\vec{v}$ is the velocity vector, $dir(\vec{v})$ computes the orientation of $\vec{v}$ with respect to the lane direction, d is the distance from the middle of the lane, and c is a collision penalty [1] Hence, this reward shape highly penalizes riskier state of the environment: wrong direction in the lane, high deviation from the center of the lane, and higher negative reward for leaving the boundaries of the world map -Figure 4.2. Other essential features of DOGE are the behaviour of the control flags *done* and $info$. Through these control flags, it is possible to detect when an episode finished before the end of the horizon of the task by either the agent went out of bounds or when a collision is detected. Both capabilities were remarkably useful in detecting safety issues with GIIL algorithms.

4.4.2 Experiments Setting

To ensure fairness across all the algorithms evaluated, it is necessary to precisely define the protocol for gathering expert's demonstrations and interactively train and test IIL algorithms for the lane following task. Across all the execution, the horizon of this task was estimated to 512 simulation steps

[1] In the lane following task, the agent does not incur in any collision penalty $c = 0$ as the environment does not contain any obstacles.

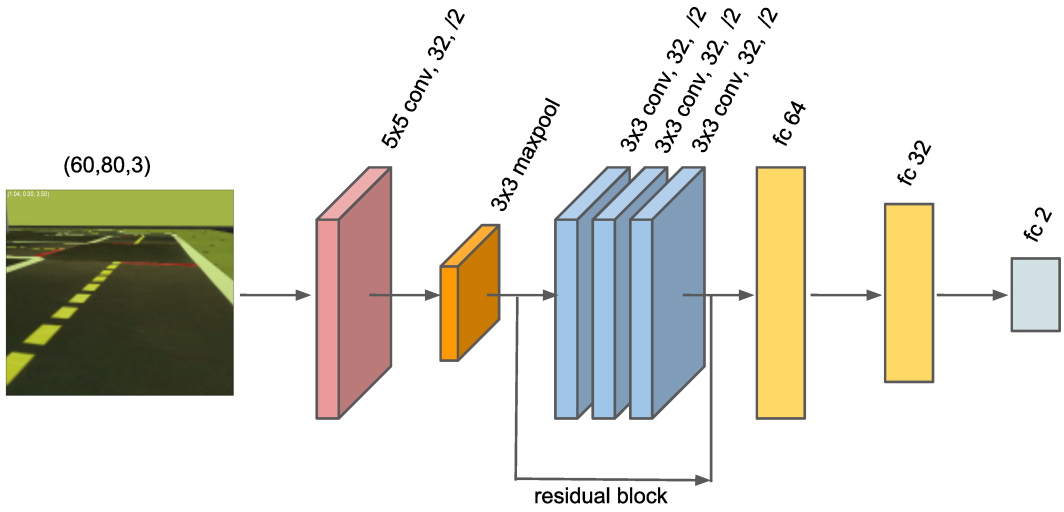


Figure 4.3: Convolutional Neural Network architecture used as learner's policy parametrization, based on Residual Networks [60].

which are approximately equivalent to one lap around the outer lane of this map. Each algorithm was trained for ten episodes with pre-computed starting points that were chosen at random and that represented a correct position and orientation in the lane. The demonstrations were collected from a human expert through the DOGE joystick interface to simplify the training procedure. The samples were obtained from driving around the outer lane of the *UdeM-1* map (Figure 4.2a).

Since early work by Pomerleau [101], Neural Networks have become the standard machine learning model for applying imitation learning in the autonomous driving domain. Recently, Convolutional Neural Networks (CNN) have been successfully applied before as a policy parametrization for Direct Policy Derivation algorithms for mobile robotics control using monocular images [9, 23, 34, 82, 93, 136]. In these experiments and across each algorithm, the learner's policy was parametrized with the CNN model architecture depicted in Figure 4.3, that is based on the Residual Networks architecture proposed in [60]. Residual Networks offer a modular architecture adjustable to the complexity of the task. As the simulated lane following tasks seems a less complicated scenario (simulation image vs real-world image and number of classes), the capacity of the original architecture was significantly reduced -similarly to [82]- to contain only one *residual block*.

Interactive Training

One of the advantages of the IIL framework is that in theory, it is possible to train the learner while receiving expert input interactively. It is interesting to note then that to date only in [93] it is possible to find traces of Interactive (online) Imitation Learning being applied by prescribed in these methods. So far, all training procedures have relied on offline re-labelling aggregated data recollected while executing policies online [23, 54, 111]. It is arguable then that IIL algorithms have

been exploited to reach their formal guarantees of performance. It could be hypothesized then that this in part due to a poorly defined control boundary between the expert's and the learner's shared control during learning. In the experiments configuration presented here, a fully interactive learning procedure is implemented for all evaluated algorithms with the following rules:

1. The expert policy π^* has control over the first training episode: $\alpha_i\pi^* + (1-\alpha_i)\pi_i = \pi^* \quad (\alpha_0 = 1)$

2. Starting the second episode, the expert intervenes only as per the requirement of the algorithms.

It is important to state that the requirements of the algorithms in (2) refer to either labelling requirements or input of teacher's uncertainty given to each algorithm accordingly.

Also, to ensure fairness and minimize the impact of humans demonstrations in this setting [79], the training process was repeated over **3** iterations and the results obtained were averaged for each algorithm performance. Furthermore, to reduce stochasticity in the environment and improve reproducibility of the experiments, all the random generators were initialized using a precomputed seed that was kept constant across algorithms in the same iteration (e.g., $seed = 1234$ for all algorithms in iteration 1).

The learning samples were feed to the learner's policy parametrization through online batch learning. Online batch learning is a method proposed in [109] where after finishing each training episode –approximately 512 samples in our setting– the state-action pairs collected are used to train the CNN. This method helps alleviate the noise in the gradient used to optimize this model. Regarding the optimization method implemented, it was empirically determined that *AdaGrad* [40] with an initial learning rate of 0.001 obtained the best results during online training while compared with other optimizers like *Adam* [74].

4.4.3 Results and Discussion

The experimental setting described in Section 4.4.2 helped to evaluate the impact of the integration of uncertainty measures in the GIIL framework. The evaluation is based on the implementation of Supervised, DAgger [110], AggreVaTe [109] algorithms as IIL baselines that do not incorporate uncertainty measures into their training procedure. The experiments also included an implementation of DropoutDAgger [86] as a baseline for the incorporation of uncertainty in IIL. The uncertainty threshold (predictive variance) σ^2 was empirically computed for the task and set to $\sigma^2 = 0.1$. This selection also impacted the mixture coefficients in UPMS. To ensure fairness and to make make the threshold point close to the saturation point of $1 - \tanh(\sigma^2) \approx 3$ — see Appendix A.1 for a formal proof that $1 - tanh(\sigma^2)$ is a correct choice for $f(U_\pi)$–, the uncertainty values was multiplied by 30 units.

Also, to isolate the impact of the changes introduced by UPMS, the evaluation procedure included three variations to the original definition of UPMS. UPMS-NE performs no explicit exploration, having $\mathcal{E} = \hat{\pi}$ that continued exploiting instead executing a random exploration mechanism implemented

for UPMS. Meanwhile, UPMS-SL tries to address a possible *data starvation* problem in UPMS by aggregating $\hat{\pi}(s)$ predictions to the training dataset. A *data starvation* problem may exist in the original definition of UPMS as its sampling method only aggregates (s, a^*) when an expert intervention happens (except on the first iteration of training). Finally, UPMS-NE-SL accounted for a combination of the strategies explained above.

Algorithms	Training Reward	Penalties	Out of Bounds	Queries
Supervised (baseline)	3434.85 ± 8.67	9.06 ± 0.36	2	5120
DAgger ($\alpha_1 = 0.99$) [110]	3175.94 ± 14.63	32.88 ± 4.47	1	2924
AggreVaTe ($\alpha_1 = 0.99$) [109]	3154.63 ± 67.03	252.63 ± 43.58	13	4440
DropoutDAgger [86] ($\sigma^2 > 0.1$)	3574.55 ± 10.75	17.87 ± 0.51	5	2826
UPMS ($\alpha = 1 - \tanh(30 \cdot \sigma^2)$)	3807.29 ± 10.83	10.18 ± 0.78	1	994
UPMS-NE (no exploration)	3813.60 ± 11.20	9.18 ± 0.54	1	931
UPMS-SL (self-learning)	3517.04 ± 22.41	12.74 ± 0.62	0	1175
UPMS-NE-SL	3948.06 ± 25.58	3.53 ± 0.34	0	1100

Table 3: Training reward, penalties, outs of world bounds, and number of queries to the expert (interventions) originated from the training phase of each IIL algorithm in DOGE.

Table 3 reflects the results obtained during the training phase of each algorithm on the lane following task in DOGE. As stated earlier, DOGE reward shape and control flags allow a fine-grain analysis of safety issues in the learning process. The training reward reflects the performance of the entire system -teacher and expert mixture of policies- over the total number of episodes (10) it was executed during training. Here, the reward has been re-normalized to exclude the *out of bounds* negative penalty (-1000) given by the simulator in such cases (out of bounds column). Also, it is reasonable to notice the distinction between the positive and negative reward (penalties) received by the system during training. In DOGE simulations, negative values of reward are given by the environment when the learning system invades the opposite lane or moves contrary to the lane direction. That negative signal along with the occurrence of out of bounds events constitute the primary measure of safety for an IIL algorithm in our evaluation.

The analysis of the results presented in Table 3 evidences that uncertainty-aware IIL algorithms (DropoutDAgger and UPMS variants) offer better safety guarantees to learning system while requiring fewer interventions/queries to the expert's policy, a result that reaffirms the findings in [86]. It is also interesting to note that both DropoutDAgger and UPMS and its variants improve the learning system behaviour where these shared control strategies outperform a supervised learning baseline containing only human input. Furthermore, UPMS and its variants significantly exceed DropoutDAgger (and traditional IIL baselines) performance across all the evaluation parameters over the full training steps (5120). As discussed before, one of the main drawbacks of DropoutDAgger -and any leaner-based Preferential Policy Mixing strategy in general- is that it does not entirely address the safety issues in the learning system. Failures in the uncertainty estimation function or misspecification of the uncertainty threshold may provoke the overconfident execution of the learner's

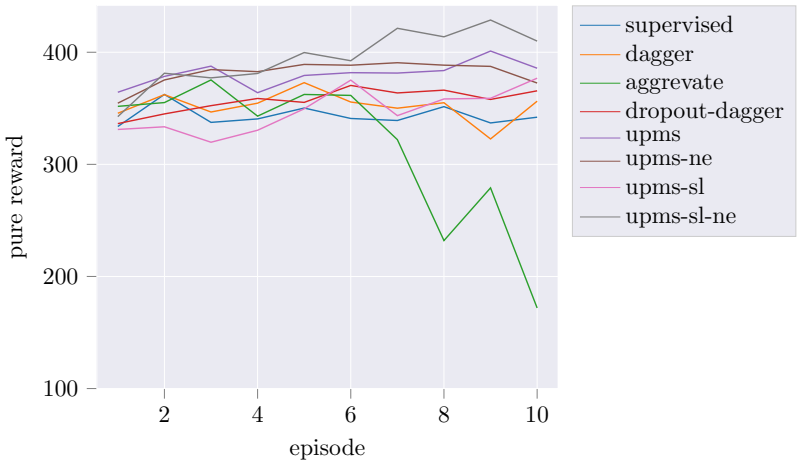


Figure 4.4: Normalized (pure) reward obtained by each algorithm over training episodes.

policy. This phenomenon is reflected by the penalties and out of bounds quantities for this method entry in Table 3.

Most importantly, it is also essential to corroborate the consistency of these results from the perspective of how the learning system evolves over training episodes. Consequently, Figures 4.4, 4.5 display per-episode values of cumulative normalized reward and total number of queries/interventions for all algorithms over the 10 episodes they were executed. Regarding training reward, the performance drop in DAgger and AggreVaTe towards the end of training empirically solidifies the analysis that hypothesis these algorithms suffer from safety issues. This situation arises when the mixture of policies becomes preferential towards the learner's non-converged policy, making the learning system harder to control by expert's input.

An unintended effect of incorporating uncertainty estimation in the ILL framework is the reduction of the number of queries (or interventions) of the expert policy. This effect was first observed in [136] and posteriorly corroborated in [86]. As displayed in Figure 4.5, the number of queries/interventions in DropoutDAgger –which surpasses SafeDAgger [136] in this regard– decreases linearly with the number of episodes. The uncertainty-based shared control strategy proposed here (UPMS and variants) has significantly improved this behaviour. After two episodes, the number of expert's interventions are reduced to less than a 20% of the horizon size and remains approximately constant to the end of training.

Figures 4.6, 4.7 display the values of the penalties received and the number of *out of bounds* events in each training episode. As discussed before, the value of these parameters corroborate the analysis on the normalized reward: IIL algorithms suffer safety issues toward the end of training when the learner's policy is not converged. This behaviour is observed in DAgger and AggreVaTe as the amount of negative reward and *out of bounds* events increases toward the end of training. Uncertainty-aware IIL algorithms seem to be less affected by this phenomenon. As discussed, DropoutDAgger still presents some issues in this regard that may be directly related to the selection of the uncertainty threshold, issues in the uncertainty estimation method, or that the shared control

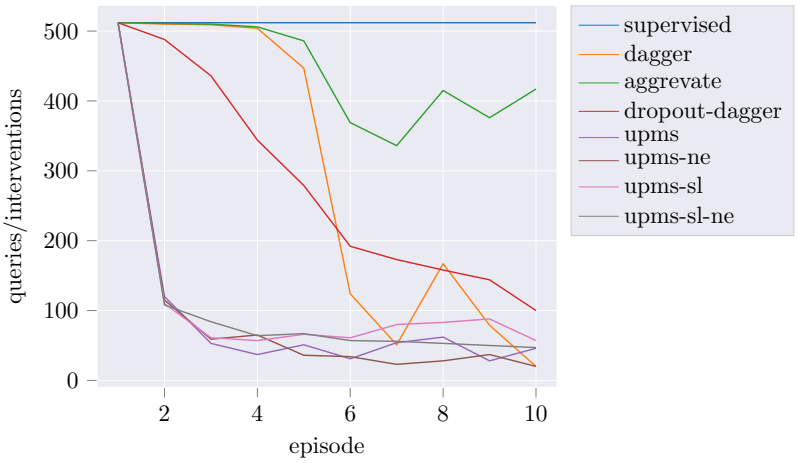


Figure 4.5: Expert's queries/interventions required by each algorithm over training episodes.

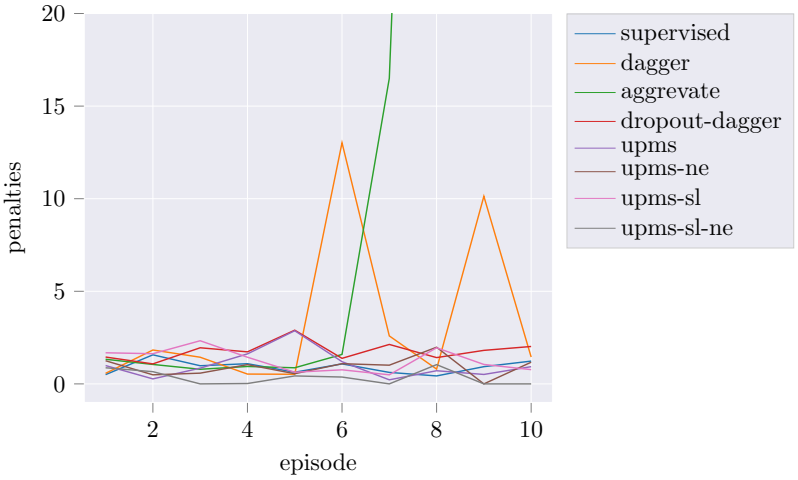


Figure 4.6: Penalty values obtained by each algorithm over training episodes.

strategy is deficient (it was empirically observed that DropoutDAgger does not define a consistent mechanism to share control with a non-algorithmic expert). UPMS and its variants seem to perform consistently in this aspect as the values for penalties remain approximately constant through training and do not incur in *out of bounds* events.

4.5 Conclusion and Future Work

This chapter presented an analysis of the incorporation of safety in Interactive Imitation Learning processes through uncertainty estimation. Here, Uncertainty-Aware Policy Mixing and Sampling (UPMS) is introduced as an algorithm that can guarantee expert-bounded safety and a higher level of interactivity than its predecessors during IIL training phase. Experimental results validate these claims as UPMS performance was contrasted under several safety criteria on the lane following tasks.

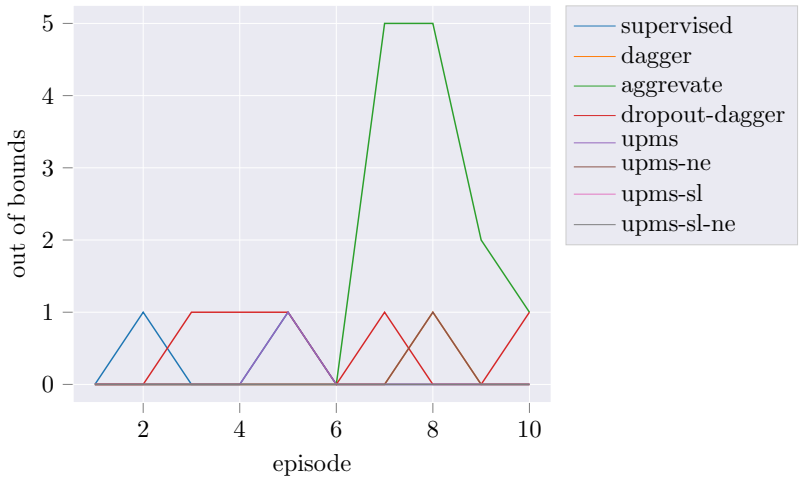


Figure 4.7: Total number of *out of bounds* events (discrete quantity) for each algorithm over training episodes and iterations.

Future work should include an analysis to determine if these changes in the learning procedure have an impact on the learned policy performance at test time, and if any of the policy mixing procedure described here can aid increasing safety on the IIL policies on real physical systems. Also, it would be essential to evaluate the algorithms proposed in a closer to real scenario (e.g., Duckietown instances [98]).

Chapter 5

Conclusions and Future Work

Imitation Learning (IL) has proven to be an efficient method to find optimal (or near-optimal) control policies when a formal specification of a task is challenging to design. Learning behaviour from an expert's demonstrations has expedited transferring to autonomous systems the knowledge of tasks that, although trivial from a human perspective, are particularly difficult to specify in control terms. While earlier systems relied on mathematically-developed routines [3], the paradigm of deriving control parameters from raw sensorial inputs has significantly reduced the amount of engineering behind intelligent systems [83, 13, 63], easing their adoption in an increasing number of domains.

Here, an example of how IL applies to real-life problems was presented in Chapter 3, where the problem of guiding visually impaired individuals while crossing street intersections is tackled through IL techniques. As previous methods rely on geometric assumptions or the detection of specific features, the method presented here proposed a solution closer to realistic deployment scenarios contrasting with those assumptions that could not be reliably guaranteed to hold in all intersections. The IL method implemented effectively eliminated the need for extracting specific features, such as zebra stripes, sidewalks, among others, from the environment. Instead, it uses the advances in CNN to extract the necessary features to derive an assistive policy.

A system of its kind has the potential to change the lives of its users as it aids their mobility and exploration capabilities in unfamiliar environments. However, as the analysis of the obtained results showed, there exist areas that still require further improvements. For instance, the method presented would greatly benefit from the collection of a more significant number of demonstrations for both the training and testing processes. This increased dataset would undoubtedly help the generation of a robust agent, more capable of handling roads configurations. Also, uncertainty estimation techniques should be included to guarantee that the overconfident predictions of the trained classifier do not impact the overall system safety. Another interesting problem would be to explore the Human-Computer Interaction aspect of the system. A user study with visually impaired individuals should be conducted to evaluate the effectiveness of the various rendering signal to communicate to the users the system output.

Another critical aspect of Imitation Learning was discussed in this manuscript. Guaranteeing

safety while interactively training or deploying systems with critical implications to their users, requires ensuring that a policy derived by the machine learning-based method can consistently perform in familiar environments and safely react to the unknown. This limitation motivated the research presented in Chapter 4 that incorporates safety in the IL training process. Through the analysis of the different traits of state-of-the-art Interactive Imitation Learning (IIL) algorithms, an Uncertainty-Aware Policy Mixing and Sampling (UPMS) algorithm that can guarantee expert-bounded safety and a higher level of interactivity than its predecessors during the IIL training phase. Additionally, UPMS reduces the number of required expert queries (or interventions) when compared with state-of-the-art algorithms, which further alleviates the burden that current IIL methods pose over the demonstrator. Experimental results validate these claims as UPMS performance was contrasted under several safety criteria on the lane following tasks. It is the position of this work that, Uncertainty-aware IIL algorithms are a fundamental step towards the derivation of policies that can guarantee safety in AI systems not only during training but also while deployed. This improvement will make IIL fully applicable in safety-critical applications.

Future work in this area should include an analysis to determine if these changes in the learning procedure have an impact on the learned policy performance at test time, and if any of the policy mixing procedure described here can aid increasing safety on the IIL policies on real physical systems. Also, it would be essential to evaluate the algorithms proposed in a closer -to-real scenario (e.g., Duckietown instances [98]). This research also opens up the possible exploration of avenues on the intersection of Imitation Learning with other areas of Machine Learning research like active learning, continual learning or curiosity-driven learning.

Appendices

Appendix A

Mathematical Proofs and Derivations

A.1 Hyperbolic Tangent for Uncertainty-Aware Preferential Policy Mixing

If we select $\alpha_p = 1 - \tanh(\mathcal{U}_{\pi_p})$, we can guarantee the following conditions, initially stated in Equation 14:

$$\lim_{\mathcal{U}_p \to 0^+} \alpha_p \pi_p + (1 - \alpha_p)\pi_s = \pi_p$$

$$\lim_{\mathcal{U}_p \to \infty} \alpha_p \pi_p + (1 - \alpha_p)\pi_s = \pi_s$$

Proof. We need to analyze the asymptotic behavior of $1 - tanh(\mathcal{U}_{\pi_p})$ when $\mathcal{U}_{\pi_p} \to 0$ (minimum uncertainty - maximum certainty) and when $\mathcal{U}_p \to \infty$ (maximum uncertainty - minimum certainty):

$$\begin{aligned}
\lim_{\mathcal{U}_p \to 0} \alpha_p \pi_p + (1 - \alpha_p)\pi_s &= \lim_{\mathcal{U}_p \to 0} (1 - \tanh(\mathcal{U}_p))\pi_p + \tanh(\mathcal{U}_p)\pi_s \\
&= \lim_{\mathcal{U}_p \to 0} \pi_p + \lim_{\mathcal{U}_p \to 0} -\tanh(\mathcal{U}_p)\pi_p + \lim_{\mathcal{U}_p \to 0} \tanh(\mathcal{U}_p)\pi_s \\
&= \pi_p + 0 + 0 \\
&= \pi_p \quad \text{(requirement 1)}
\end{aligned}$$

$$\begin{aligned}
\lim_{\mathcal{U}_p \to \infty} \alpha_p \pi_p + (1 - \alpha_p)\pi_s &= \lim_{\mathcal{U}_p \to \infty} (1 - \tanh(\mathcal{U}_p))\pi_p + \tanh(\mathcal{U}_p)\pi_s \\
&= \lim_{\mathcal{U}_p \to \infty} \pi_p + \lim_{\mathcal{U}_p \to \infty} -\tanh(\mathcal{U}_p)\pi_p + \lim_{\mathcal{U}_p \to \infty} \tanh(\mathcal{U}_p)\pi_s \\
&= \pi_p - \pi_p + \pi_s \\
&= \pi_s \quad \text{(requirement 2)}
\end{aligned}$$

Hence, this proves that $1 - tanh(\mathcal{U}_{\pi_p})$ guarantees that π_p asymptotically takes full control of the learning system as its uncertainty $\mathcal{U}_{\pi_p}$ asymptotically approaches its minimum. Meanwhile, when

$\mathcal{U}_{\pi_p}$ asymptotically approaches its maximum the secondary policy π_s π_p cedes control of the system to π_s. □

A.2 Hyperbolic Tangent for Uncertainty-Aware Rational Policy Mixing

If we select $\alpha_{\pi_p} = 1 - \tanh(U_{\pi_p})$, $\alpha_{\pi_q} = 1 - \tanh(U_{\pi_q})$, we need to prove that selection guarantees the following four conditions of rationality in the system behavior:

$$\begin{aligned}
\alpha_{\pi_p} + \alpha_{\pi_q} &= 1 && \text{(consistency)} \\
\lim_{\mathcal{U}_p \to 0,\, \mathcal{U}_q \to \infty} \pi_{RM} &= \pi_p && \text{(rationality)} \\
\lim_{\mathcal{U}_p \to \infty,\, \mathcal{U}_q \to 0} \pi_{RM} &= \pi_q && \text{(rationality)} \\
\lim_{\mathcal{U}_p \to 0,\, \mathcal{U}_q \to 0} \pi_{RM} &= \frac{1}{2}\pi_p + \frac{1}{2}\pi_q && \text{(neutrality)} \\
\lim_{\mathcal{U}_p \to \infty,\, \mathcal{U}_q \to \infty} \pi_{RM} &: \text{undefined} && \text{(impossibility)}
\end{aligned} \tag{28}$$

Proof. Here, we investigate how the coefficients' selection satisfies each condition:

1. Consistency: as $0 \leq \alpha_{\pi_p} \leq 1$ and $0 \leq \alpha_{\pi_p} \leq 1$, then, we need to re-normalize their values such that $\alpha_{\pi_p} + \alpha_{\pi_q} = 1$:

$$\begin{aligned}
\alpha'_{\pi_p} &= \frac{\alpha_{\pi_p}}{\alpha_{\pi_q} + \alpha_{\pi_p}} \\
\alpha'_{\pi_q} &= \frac{\alpha_{\pi_q}}{\alpha_{\pi_q} + \alpha_{\pi_p}}
\end{aligned}$$

2. Rationality: we need to prove that our selection of function guarantees the selection of the best policy in the mixture under uncertainty:

$$\begin{aligned}
\lim_{\mathcal{U}_p \to 0,\, \mathcal{U}_q \to \infty} \alpha'_{\pi_p}\pi_p + \alpha'_{\pi_q}\pi_q &= \frac{1 - \tanh(\mathcal{U}_{\pi_p})}{2 - \tanh(\mathcal{U}_{\pi_p}) - \tanh(\mathcal{U}_{\pi_q})}\pi_p + \frac{1 - \tanh(\mathcal{U}_{\pi_q})}{2 - \tanh(\mathcal{U}_{\pi_p}) - \tanh(\mathcal{U}_{\pi_q})}\pi_q \\
&= \frac{1}{2 - 1 - 0}\pi_p + \frac{0}{2 - 1 - 0}\pi_q \\
&= \pi_p
\end{aligned}$$

$$\begin{aligned}
\lim_{\mathcal{U}_p \to \infty,\, \mathcal{U}_q \to 0} \alpha'_{\pi_p}\pi_p + \alpha'_{\pi_q}\pi_q &= \frac{1 - \tanh(\mathcal{U}_{\pi_p})}{2 - \tanh(\mathcal{U}_{\pi_p}) - \tanh(\mathcal{U}_{\pi_q})}\pi_p + \frac{1 - \tanh(\mathcal{U}_{\pi_q})}{2 - \tanh(\mathcal{U}_{\pi_p}) - \tanh(\mathcal{U}_{\pi_q})}\pi_q \\
&= \frac{0}{2 - 0 - 1}\pi_p + \frac{1}{2 - 0 - 1}\pi_q \\
&= \pi_q
\end{aligned}$$

3. Neutrality: we need to prove the learning system show no preference for equally uncertainty

policies:

$$\begin{aligned}\lim_{\mathcal{U}_p \to 0,\, \mathcal{U}_q \to 0} \alpha'_{\pi_p}\pi_p + \alpha'_{\pi_q}\pi_q &= \frac{1-\tanh(\mathcal{U}_{\pi_p})}{2-\tanh(\mathcal{U}_{\pi_p})-\tanh(\mathcal{U}_{\pi_q})}\pi_p + \frac{1-\tanh(\mathcal{U}_{\pi_q})}{2-\tanh(\mathcal{U}_{\pi_p})-\tanh(\mathcal{U}_{\pi_q})}\pi_q \\ &= \frac{1}{2-0-0}\pi_p + \frac{1}{2-0-0}\pi_q \\ &= \frac{1}{2}\pi_p + \frac{1}{2}\pi_q\end{aligned}$$

4. Impossibility: we need to prove the system is able to detect an impossible decision under uncertainty where both policy asymptotically approach their maximum:

$$\begin{aligned}\lim_{\mathcal{U}_p \to \infty,\, \mathcal{U}_q \to \infty} \alpha'_{\pi_p}\pi_p + \alpha'_{\pi_q}\pi_q &= \frac{1-\tanh(\mathcal{U}_{\pi_p})}{2-\tanh(\mathcal{U}_{\pi_p})-\tanh(\mathcal{U}_{\pi_q})}\pi_p + \frac{1-\tanh(\mathcal{U}_{\pi_q})}{2-\tanh(\mathcal{U}_{\pi_p})-\tanh(\mathcal{U}_{\pi_q})}\pi_q \\ &= \frac{0}{2-1-1}\pi_p + \frac{0}{2-1-1}\pi_q \\ &= \frac{0}{0}\pi_p + \frac{0}{0}\pi_q \\ &\textit{does not exist}\end{aligned}$$

□

A.3 Hyperbolic Tangent for Uncertainty-Aware Rational Sampling

If we make $t = \lfloor T \tanh(\mathcal{U}_{\hat{\pi}}) \rfloor$, it is possible to demonstrate the system has the ability to asymptotically trade-off exploitation and exploration steps under the learner's uncertainty values:

Proof. Rationality:

$$\begin{aligned}\lim_{\mathcal{U}_\pi \to 0} t &= 0 && \text{(explore when certain)} \\ \lim_{\mathcal{U}_\pi \to 0} \lfloor T \tanh(\mathcal{U}_{\hat{\pi}}) \rfloor &= \lfloor T0 \rfloor \\ &= 0 \\ \lim_{\mathcal{U}_\pi \to \infty} t &= T && \text{(exploit when uncertain)} \\ \lim_{\mathcal{U}_\pi \to \infty} \lfloor T \tanh(\mathcal{U}_{\hat{\pi}}) \rfloor &= \lfloor T1 \rfloor \\ &= T\end{aligned}$$

□

Bibliography

[1] List of audible pedestrian signals installed and to come. `http://www.mabmackay.ca/pages/158/INFO-Accessibility?langue=en`. Accessed: 2017-06-30.

[2] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Abbeel, P. *Apprenticeship Learning and Reinforcement Learning with Application to Control*. PhD thesis, Stanford University, 2008.

[4] Ahmetovic, D., Bernareggi, C., Gerino, A., and Mascetti, S. Zebrarecognizer: Efficient and precise localization of pedestrian crossings. In *ICPR* (2014).

[5] Ahmetovic, D., Bernareggi, C., and Mascetti, S. Zebralocalizer: identification and localization of pedestrian crossings. In *Mobile HCI* (2011).

[6] Ahmetovic, D., Manduchi, R., Coughlan, J. M., and Mascetti, S. Mind your crossings: Mining gis imagery for crosswalk localization. *ACM Transactions on Accessible Computing (TACCESS) 9*, 4 (2017), 11.

[7] Akametalu, A. K., Kaynama, S., Fisac, J. F., Zeilinger, M. N., Gillula, J. H., and Tomlin, C. J. Reachability-Based Safe Learning with Gaussian Processes. In *53rd IEEE Conference on Decision and Control* (Los Angeles, California, USA, 2014), pp. 1424–1431.

[8] Amini, A., Paull, L., Karaman, S., and Rus, D. Learning Safety Bounds for Parallel Autonomous Systems. In *International Conference on Robotics and Automation* (2018).

[9] Amini, A., Soleimany, A., Karaman, S., and Rus, D. Spatial Uncertainty Sampling for End to End Control. In *Bayesian Deep Learning Workshop, NIPS* (2017).

[10] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete Problems in AI Safety. 2016.

[11] Arditi, A., Holtzman, J. D., and Kosslyn, S. M. Mental imagery and sensory experience in congenital blindness. *Neuropsychologia 26 1* (1988), 1–12.

[12] Argall, B. D. Human autonomy through robotics autonomy. `https://youtu.be/od6V1tt0ctc`, 2016.

[13] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems 57*, 5 (may 2009), 469–483.

[14] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems 57*, 5 (2009), 469–483.

[15] Auer, P., Herbster, M., and Warmuth, M. K. Exponentially many local minima for single neurons.

[16] Bandura, A. Observational Learning. *Learning and Memory* (nov 2004), 482–484.

[17] Bandura, A., and Walters, R. H. *Social learning and personality development*. Holt, Rinehart and Winston, 1963.

[18] Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell. 35*, 8 (Aug. 2013), 1798–1828.

[19] Billard, A. Imitation: A Means to Enhance Learning of a Synthetic Protolanguage in Autonomous Robots. In *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, Cambridge, MA, USA, 2002, ch. Imitation:, pp. 281–310.

[20] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[21] Blei, D., Ranganath, R., and Mohamed, S. Variational Inference: Foundations and Modern Methods. *Neural Information Processing Systems Tutorials* (2016), 162.

[22] Blei, D. M., Kucukelbir, A., and Mcauliffe, J. D. Variational Inference: A Review for Statisticians.

[23] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to End Learning for Self-Driving Cars. *arxiv preprint arXiv:1604.07316v1* (2016).

[24] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *CoRR abs/1604.07316* (2016).

[25] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym, 2016.

[26] Canziani, A., Paszke, A., and Culurciello, E. An analysis of deep neural network models for practical applications. *CoRR abs/1605.07678* (2016).

[27] Cesa-Bianchi, N., and Lugosi, G. *Prediction, Learning, and Games*. Cambridge University Press, Edimburg, UK, 2006.

[28] Chernova, S. *Confidence-Based Robot Policy Learning from Demonstration*. PhD thesis, Carnegie Mellon University, 2009.

[29] Chernova, S., and Veloso, M. Confidence-based Policy Learning from Demonstration Using Gaussian Mixture Models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems* (New York, NY, USA, 2007), AAMAS '07, ACM, pp. 233:1—-233:8.

[30] Chernova, S., and Veloso, M. Multi-thresholded Approach to Demonstration Selection for Interactive Robot Learning. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction* (New York, NY, USA, 2008), HRI '08, ACM, pp. 225–232.

[31] Choi, E., Schuetz, A., Stewart, W. F., and Sun, J. Medical Concept Representation Learning from Electronic Health Records and its Application on Heart Failure Prediction. *CoRR* (2017).

[32] Choi, S., Lee, K., Lim, S., and Oh, S. Uncertainty-Aware Learning from Demonstration using Mixture Density Networks with Sampling-Free Variance Modeling. 2017.

[33] Chollet, F. Xception: Deep learning with depthwise separable convolutions. *CoRR abs/1610.02357* (2016).

[34] Codevilla, F., Müller, M., Dosovitskiy, A., López, A., and Koltun, V. End-to-end Driving via Conditional Imitation Learning. *arXiv preprint arXiv:1710.02410* (2017).

[35] Davis, J. M. *Imitation: A Review and Critique*. Springer US, Boston, MA, 1973, pp. 43–72.

[36] Demiris, J. Experiments Towards Robotic Learning By Imitation. In *Association for the Advancement of Artificial Intelligence (AAAI)* (1994), p. 223.

[37] Diaz, M., Girgis, R., Fevens, T., and Cooperstock, J. To Veer or Not to Veer: Learning from Experts How to Stay Within the Crosswalk. In *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017* (2018), vol. 2018-Janua.

[38] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, Belmont, Massachusetts, 2005.

[39] Doshi-Velez, F., and Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. Tech. rep.

[40] Duchi, J., Edu, J. B., Hazan, E., and Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization *. *Journal of Machine Learning Research 12* (2011), 2121–2159.

[41] Elkan, C. The Foundations of Cost-Sensitive Learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)* (2001).

[42] Everingham, M., Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV) 88*, 2 (June 2010), 303–338.

[43] Fernandes, D. *Vehicle-pedestrian Accidents at Signalized Intersections in Montreal*. PhD thesis, McGill University, 2013.

[44] Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., and Tomlin, C. J. A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems. *CoRR* (2018), 16.

[45] Gal, Y., and Ghahramani, Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems* (2016), pp. 1019–1027.

[46] Gal, Y., and Ghahramani, Z. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. *arXiv preprint arXiv:1506.02158* (2016).

[47] Gal, Y., and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning (ICML)* (2016).

[48] García, J., and Fernández, F. Safe Exploration of State and Action Spaces in Reinforcement Learning. *Journal of Artificial Intelligence Research 4512*, 12 (2012), 515–56412.

[49] García, J., and Fernández, F. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research 16* (2015), 1437–1480.

[50] Geibel, P., and Wysotzki, F. Risk-sensitive Reinforcement Learning Applied to Control Under Constraints. *J. Artif. Int. Res. 24*, 1 (jul 2005), 81–108.

[51] Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature 521* (2015), 452–459.

[52] Gillula, J. H., and Tomlin, C. J. Reducing Conservativeness in Safety Guarantees by Learning Disturbances Online: Iterated Guaranteed Safe Online Learning. In *Robotics: Science and Systems* (Sydney, Australia, 2012).

[53] Gillulay, J. H., and Tomlin, C. J. Guaranteed safe online learning of a bounded system. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (sep 2011), pp. 2979–2984.

[54] Giusti, A., Guzzi, J., Cirean, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Caro, G. D., Scaramuzza, D., and Gambardella, L. M. A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robotics and Automation Letters 1*, 2 (2016), 661–667.

[55] Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters 1*, 2 (2016), 661–667.

[56] Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), pp. 249–256.

[57] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

[58] Guth, D. Why does training reduce blind pedestrians veering. *Blindness and brain plasticity in navigation and object perception* (2007), 353–365.

[59] Hastie, T., Tibshirani, R., and Friedman, J. H. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009.

[60] He, H., Daumé Iii, H., Eisner, J., Daume, H., and Daumé, H. Imitation Learning by Coaching. *Advances in Neural Information Processing Systems*, Section 5 (2012), 3158–3166.

[61] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[62] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR abs/1704.04861* (2017).

[63] Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation Learning. *ACM Computing Surveys 50*, 2 (apr 2017), 1–35.

[64] Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR) 50*, 2 (2017), 21.

[65] Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR abs/1602.07360* (2016).

[66] Ivanchenko, V., Coughlan, J., and Shen, H. Staying in the crosswalk: A system for guiding visually impaired pedestrians at traffic intersections. *Assistive technology research series 25*, 2009 (2009), 69.

[67] Joaquin Quinonero-Candela, Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA, USA, 2009.

[68] Judah, K., Fern, A. P., and Dietterich, T. G. Active Imitation Learning: Formal and Practical Reductions to I.I.D. Learning. *Journal of Machine Learning Research 15* (2007), 4105–4143.

[69] Kahn, G., Villaflor, A., Pong, V., Abbeel, P., and Levine, S. Uncertainty-Aware Reinforcement Learning for Collision Avoidance.

[70] Kendall, A., and Gal, Y. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems* (2017), pp. 5580–5590.

[71] Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. Tech. rep., 2018.

[72] Kim, B., and Pineau, J. Maximum Mean Discrepancy Imitation Learning.

[73] Kim, D. K., and Chen, T. Deep neural network for real-time autonomous indoor navigation. *arXiv preprint arXiv:1511.04668* (2015).

[74] Kingma, D. P., Salimans, T., and Welling, M. Variational Dropout and the Local Reparameterization Trick.

[75] Kolobov, A., and Mausam. *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool, 2012.

[76] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems* (USA, 2012), NIPS'12, Curran Associates Inc., pp. 1097–1105.

[77] Lahav, O., Schloerb, D., Kumar, S., and Srinivasan, M. A virtual environment for people who are blind a usability study. *Journal of Assistive Technologies 6*, 1 (2012), 38–52.

[78] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *31st Conference on Neural Information Processing Systems (NIPS 2017)* (Long Beach, CA, USA, 2017).

[79] Laskey, M., Chuck, C., Lee, J., Mahler, J., Krishnan, S., Jamieson, K., Dragan, A., and Goldberg, K. Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations. In *Proceedings - IEEE International Conference on Robotics and Automation* (oct 2017), pp. 358–365.

[80] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[81] Lee, K., Saigol, K., and Theodorou, E. Safe end-to-end imitation learning for model predictive control. *CoRR* (2018).

[82] Loquercio, A., Maqueda, A. I., Del-Blanco, C. R., and Scaramuzza, D. DroNet: Learning to Fly by Driving. *IEEE Robotics and Automation Letters 3*, 2 (2018), 1088–1095.

[83] Mataric, M. J. Learning social behavior. *Robotics and Autonomous Systems 20* (1997), 191–204.

[84] Maxime Chevalier-Boisvert Florian Golemo, Y. C. B. M. L. P. Duckietown Environments for OpenAI Gym. \url{https://github.com/duckietown/gym-duckietown}, 2018.

[85] McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. *IJCAI International Joint Conference on Artificial Intelligence* (2017).

[86] Menda, K., Driggs-Campbell, K., and Kochenderfer, M. J. DropoutDAgger: A Bayesian Approach to Safe Imitation Learning.

[87] Moldovan, T. M., and Abbeel, P. Safe Exploration in Markov Decision Processes. In *Proceedings of the 29th International Coference on International Conference on Machine Learning* (USA, 2012), ICML'12, Omnipress, pp. 1451–1458.

[88] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel* (2010), pp. 807–814.

[89] Ng, A. Y., and Russell, S. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the 17th International Conference on Machine Learning* (2000), pp. 663—-670.

[90] Oberkampf, W. L., Deland, S. M., Rutherford, B. M., Diegert, K. V., and Alvin, K. F. Error and uncertainty in modeling and simulation. *Reliability Engineering and System Safety 75* (2002), 333–357.

[91] of Transportation, N. Y. C. D. Accessible pedestrian signals program status report.

[92] Pan, S. J., and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering 22*, 10 (Oct 2010), 1345–1359.

[93] Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E., and Boots, B. Agile Autonomous Driving using End-to-End Deep Imitation Learning. *arXiv preprint arXiv:1709.07174* (2018).

[94] Panëels, S., Olmos, A., Blum, J., and Cooperstock, J. R. Listen to it yourself! evaluating usability of "what's around me?" for the blind. In *Human Factors in Computing Systems (CHI)* (2013).

[95] Panëels, S. A., Varenne, D., Blum, J. R., and Cooperstock, J. R. The walking straight mobile application: Helping the visually impaired avoid veering. In *Proceedings of ICAD13* (2013), ICAD, pp. 25–32.

[96] Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR abs/1606.02147* (2016).

[97] Patel, S., Griffin, B., Kusano, K., and Corso, J. J. Predicting Future Lane Changes of Other Highway Vehicles using RNN-based Deep Models.

[98] Paull, L., Tani, J., Ahn, H., Alonso-Mora, J., Carlone, L., Cap, M., Chen, Y. F., Choi, C., Dusek, J., Fang, Y., Hoehener, D., Liu, S., Novitzky, M., Okuyama, I. F., Pazis, J., Rosman, G., Varricchio, V., Wang, H., Yershov, D., Zhao, H., Benjamin, M., Carr, C., Zuber, M., Karaman, S., Frazzoli, E., Vecchio, D. D., Rus, D., How, J., Leonard, J., and Censi, A. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (may 2017), pp. 1497–1504.

[99] Poggi, M., and Mattoccia, S. A wearable mobility aid for the visually impaired based on embedded 3d vision and deep learning. In *Computers and Communication (ISCC), 2016 IEEE Symposium on* (2016), IEEE, pp. 208–213.

[100] Poggi, M., Nanni, L., and Mattoccia, S. Crosswalk recognition through point-cloud processing and deep-learning suited to a wearable mobility aid for the visually impaired. In *International Conference on Image Analysis and Processing* (2015), Springer, pp. 282–289.

[101] Pomerleau, D. A. ALVINN, an autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, ch. ALVINN: An, pp. 305–313.

[102] Puterman, M. L., and L., M. *Markov decision processes : discrete stochastic dynamic programming*. Wiley, 1994.

[103] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 779–788.

[104] Ross, D. A., and Blasch, B. B. Wearable interfaces for orientation and wayfinding. In *Proceedings of the fourth international ACM conference on Assistive technologies* (2000), ACM, pp. 193–200.

[105] Ross, S. *Interactive Learning for Sequential Decisions and Predictions*. Ph.d., Carnegie Mellon University, 2013.

[106] Ross, S., and Bagnell, J. A. Efficient Reductions for Imitation Learning Supplementary Material.

[107] Ross, S., and Bagnell, J. A. Efficient Reductions for Imitation Learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), pp. 661—-668.

[108] Ross, S., and Bagnell, J. A. Efficient reductions for imitation learning. In *In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS* (2010).

[109] Ross, S., and Bagnell, J. A. Reinforcement and Imitation Learning via Interactive No-Regret Learning.

[110] Ross, S., Gordon, G. J., and Bagnell, J. A. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* (2011), pp. 627–635.

[111] Ross, S., Melik-Barkhudarov, N., Shaurya Shankar, K., Wendel, A., Dey, D., Andrew Bagnell, J., and Hebert, M. Learning Monocular Reactive UAV Control in Cluttered Natural Environments. Tech. rep.

[112] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[113] Russell, S. Learning Agents for Uncertain Environments (Extended Abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (New York, NY, USA, 1998), COLT' 98, ACM, pp. 101–103.

[114] Scott, A., Barlow, J., Bentzen, B., Bond, T., and Gubbe, D. Accessible pedestrian signals at complex intersections: Effects on blind pedestrians. *Transportation Research Record: Journal of the Transportation Research Board*, 2073 (2008), 94–103.

[115] Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *arXiv preprint arXiv:1610.02391* (2016).

[116] Settles, B. Active Learning Literature Survey. *Machine Learning 15*, 2 (2010), 201–221.

[117] Shalev-Shwartz, S. Online Learning and Online Convex Optimization. *Foundations and Trends in Machine Learning 4*, 2 (2011), 107–194.

[118] Shalev-Shwartz, S., and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[119] Shen, H., Chan, K.-Y., Coughlan, J., and Brabyn, J. A mobile phone system to find crosswalks for visually impaired pedestrians. *Technology and disability 20*, 3 (2008), 217–224.

[120] Shon, A. P., Grimes, D. B., Baker, C. L., and Rao, R. P. N. A Probabilistic Framework for Model-Based Imitation Learning Imitation learning in animals and machines.

[121] Siegwart, R., and Nourbakhsh, I. R. *Introduction to Autonomous Mobile Robots*. MIT Press, Cambridge, MA, USA, 2004.

[122] Souman, J. L., Frissen, I., Sreenivasa, M. N., and Ernst, M. O. Walking straight into circles. *Current Biology 19*, 18 (2009), 1538–1542.

[123] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research 15* (2014), 1929–1958.

[124] Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction. *Proceedings of the 34th International Conference on Machine Learning 70* (2017), 3309–3318.

[125] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[126] Teye, M., Azizpour, H., and Smith, K. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. Tech. rep., 2018.

[127] Thrun, S., Burgard, W., and Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[128] Tieleman, T., and Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[129] Togelius, J., Nardi, R. D., and Lucas, S. M. Towards automatic personalised content creation for racing games. In *2007 IEEE Symposium on Computational Intelligence and Games* (April 2007), pp. 252–259.

[130] Turchetta, M., Berkenkamp, F., and Krause, A. Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Advances in Neural Information Processing Systems* (2016), pp. 4312–4320.

[131] Wachi, A., Sui, Y., Yue, Y., and Ono, M. Safe Exploration and Optimization of Constrained MDPs using Gaussian Processes.

[132] White, D. J. A Survey of Applications of Markov Decision Processes. *The Journal of the Operational Research Society 44*, 11 (1993), 1073–1096.

[133] Wulfmeier, M. On Machine Learning and Structure for Mobile Robots. Tech. rep., University of Oxford, 2018.

[134] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems* (2014), pp. 3320–3328.

[135] Yosinski, J., Clune, J., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. In *In ICML Workshop on Deep Learning* (2015).

[136] Zhang, J., and Cho, K. Query-Efficient Imitation Learning for End-to-End Simulated Driving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)* (2017), pp. 2891–2897.

[137] Zhang, S., Jiang, Y., Guni, S., and Stone, P. Multirobot Symbolic Planning under Temporal Uncertainty.