Efficient Supervision for Robot Learning via Imitation, Simulation, and Adaptation

Markus Wulfmeier

New College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2018

"All theses are wrong, but some are useful."

misquoting George E. P. Box

Acknowledgements

If this thesis ended up being useful, the first person to thank would be my supervisor, Prof. Ingmar Posner. His support and knowledge have proven immeasurable during the last years and his guidance enabled me to gain footing in this fascinating field. I would also like to thank Prof. Paul Newman, who helped me recognise the importance of encouraging questions and to immerse oneself in the foundations, the first principles underlying an idea. Additionally, my sincere appreciation goes to Prof. Pieter Abbeel, whose exemplary guidance, encouragement, and academic perseverance were of immense value during my time as visiting scholar at UC Berkeley's Berkeley Artificial Intelligence Research (BAIR) lab.

I owe a great deal to my thesis examiners, Prof. Martin Riedmiller and Prof. Andrew Zisserman. With their thorough questions and suggestions, they provided a different angle and enabled me to greatly improve this work. Furthermore, I must express my very profound gratitude to Prof. Karl Iagnemma for initial guidance at the Massachusetts Institute of Technology on my path towards academic research.

After all this time at ORI, I would like to thank my ORI friends and coauthors, Alex Bewley, Dushyant Rao, Dominic Zeng Wang and Peter Ondruska as well as my collaborators in Berkeley, David Held and Carlos Florensa. I count these long, exhausting nights of programming, debugging and writing under imminent deadline pressure as some of my greatest experiences in Oxford.

For their great patience and not giving up on my hidden qualities as software engineer, I have to additionally give great thanks to Geoff Hester, Akshay Morye and Tom Wilcox. Their knowledge and support has been critical for the success of many projects during these years.

My sincere appreciation goes to all friends and colleagues past and present of ORI and BAIR, who have supported and inspired me throughout the years, for proving an old proverb and showing that the journey is its own reward. I'm especially grateful to those outside robotics (in Oxford, London, Berkeley, Hannover, Herford and Zurich) who have shown support and open ears for discussions.

The author would like to gratefully acknowledge, yet again, the support of the UKs Engineering and Physical Sciences Research Council (EPSRC) through the Doctoral Training Award (DTA), the Hans-Lenze-Foundation, the Dr-Jost-Henkel-Foundation, New College Oxford and the Department of Engineering Science.

Letztendlich, mein ganz besonderer Dank aber gilt meinen Eltern, Jürgen und Siglinde, die mir diesen Schritt ermöglichten und denen diese Arbeit gewidmet ist. Chciabym równie podzikowa mojej drogiej Kasi za wsparcie, otuch (i cierpliwo).

Abstract

In order to enable more widespread application of robots, we are required to reduce the human effort for the introduction of existing robotic platforms to new environments and tasks. In this thesis, we identify three complementary strategies to address this challenge, via the use of imitation learning, domain adaptation, and transfer learning based on simulations. The overall work strives to reduce the effort of generating training data by employing inexpensively obtainable labels and by transferring information between different domains with deviating underlying properties.

Imitation learning enables a straightforward way for untrained personnel to teach robots to perform tasks by providing demonstrations, which represent a comparably inexpensive source of supervision. We develop a scalable approach to identify the preferences underlying demonstration data via the framework of inverse reinforcement learning. The method enables integration of the extracted preferences as cost maps into existing motion planning systems. We further incorporate prior domain knowledge and demonstrate that the approach outperforms the baselines including manually crafted cost functions.

In addition to employing low-cost labels from demonstration, we investigate the adaptation of models to domains without available supervisory information. Specifically, the challenge of appearance changes in outdoor robotics such as illumination and weather shifts is addressed using an adversarial domain adaptation approach. A principal advantage of the method over prior work is the straightforwardness of adapting arbitrary, state-of-the-art neural network architectures. Finally, we demonstrate performance benefits of the method for semantic segmentation of drivable terrain.

Our last contribution focuses on simulation to real world transfer learning, where the characteristic differences are not only regarding the visual appearance but the underlying system dynamics. Our work aims at parallel training in both systems and mutual guidance via auxiliary alignment rewards to accelerate training for real world systems. The approach is shown to outperform various baselines as well as a unilateral alignment variant.

Statement of Authorship

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Markus Wulfmeier, New College

Contents

List of Figures

List of Tables

List of Abbreviations

ORI . . . . . Oxford Robotics Institute
RLL . . . . . Robot Learning Lab
BAIR . . . . . Berkeley Artificial Intelligence Research
MuJoCo . . . Multi-Joint Dynamics with Contact Simulator
DART . . . . . Dynamic Animation and Robotics Toolkit
LIDAR . . . . Light based Detection and Range
GP . . . . . Gaussian Process
NN . . . . . Neural Network
CNN . . . . . Convolutional Neural Network
RNN . . . . . Recurrent Neural Network
FCN . . . . . Fully Convolutional Network
SGD . . . . . Stochastic Gradient Descent
Adam . . . . . Adaptive Moment Optimiser
RL . . . . . Reinforcement Learning
MDP . . . . . Markov Decision Process
IRL . . . . . Inverse Reinforcement Learning
MEDIRL . . . Maximum Entropy Inverse Reinforcement Learning
MaxEnt . . . . Maximum Entropy
MATL . . . . . Mutual Alignment Transfer Learning
TRPO . . . . . Trust Region Policy Optimisation
GAN . . . . . Generative Adversarial Network
ADA . . . . . Adversarial Domain Adaptation
DAN . . . . . Domain Adversarial Networks
NLL . . . . . Negative Log Likelihood
MHD . . . . . Modified Hausdorff Distance
FNR/FPR . . False Negatives/Positives Rate

List of Symbols

$\mathcal{M}$	Markov Decision Process
s	State
$\mathcal{S}$	State Space
a	Action
$\mathcal{A}$	Action Space
T	Transition Model
r	Reward
κ	Auxiliary Reward
γ	Discount Factor
ζ	Trajectory
p	Probability
π	Policy
μ	State Visiting Frequencies
ρ	Discounted State Visiting Frequencies
Q	State Action Value Function
V	State Value Function
A	Advantage Function
η	Expected Returns
$\mathcal{D}$	Demonstration Trajectories
f	Features
D	Discriminator Module
E	Encoder Module
S	Supervised Module
D_{KL}	Kullback-Leibler Divergence
$\mathcal{X}$	Active Set (Sparse Gaussian Process)
$\mathcal{K}$	Covariance (Gaussian Process)
θ, ϕ, ω	Model Parameters

1 Introduction

Contents

1.1 Motivation

Recent successes in machine learning, vision and robotics have lead to the widespread expectation that robotic platforms will increasingly succeed in applications of real value to the public domain. To enable more general use of robots in various applications, it is necessary to reduce the effort for human supervision and render the process more efficient. In particular, this includes maximising the impact of existing supervised data and decreasing the cost of obtaining new useful data.

Many subfields of machine learning research contribute to this goal, including but not restricted to unsupervised learning, self-supervised learning, imitation learning, transfer learning and model adaptation. In this thesis, we will focus on the last three.

Not all forms of human supervision require the same amount of effort to guide the learning process. Imitation learning from expert demonstrations for a particular task can straightforwardly lead to increasingly personalised services, improved human-machine interaction and endows robots with an ability to learn the execution of complex tasks from examples.

Existing approaches on imitation learning in robotics often aim at directly copying expert behaviour [Bojarski et al., 2016, Pomerleau, 1990, Muller et al., 2006], which has several shortcomings: sample inefficiency, *compounding of errors* [Ross et al., 2010], and - realised as end-to-end approach from images to steering commands - complicates the integration into existing software architectures and can lead to significant growth in the number of required training samples [Shalev-Shwartz and Shashua, 2016] (more details in Section 2.3). Our contributions in imitation learning focus on the field of inverse reinforcement learning (IRL), which addresses many of the shortcomings mentioned and can be straightforwardly integrated into the modular software structure of modern autonomy systems (Chapter 3). IRL generally provides a more efficient and robust approach for learning to imitate. The approach iteratively refines a model for the expert's preferences based on which a planning system creates trajectories similar to the expert. In this way, the preference model - formally described as reward function - is optimised to result in corrective actions in case the planner produces suboptimal trajectories. The approach is described in detail in Section 2.3.2.

An important prerequisite for enabling real world application of IRL-based approaches is to facilitate scaling with respect to the complexity of the task and the correlated required size of the training dataset. Our work addresses the limited capacity of existing approaches [Ziebart et al., 2008, Choi and Kim, 2013, Ratliff et al., 2006] as well as expand the ability to handle large datasets [Levine et al., 2011].

While imitation learning enables us to benefit from comparably low-cost supervised demonstration samples, the creation and curation of supervised data can be generally very time consuming. After optimising a model on a fixed dataset, appearance changes in outdoor environments (caused for example by different

weather and lighting conditions) can reduce model accuracy as the new input data distribution differs from the training distribution. A common and very laborious approach is to try to collect data from all possible domains in advance and annotate. While this is often possible, it requires significant effort and can be prone to misjudgement of required data.

Addressing the challenge with only unsupervised data from new domains is a challenging approach and prior work on large-scale robotics tasks has previously focused on imposing prior structure on the the type of shift between domains as described in detail in Section 2.4. We focus on extending methods from a subfield of transfer learning: adversarial domain adaptation (ADA) in Chapter 4 to obtain representations that are invariant with respect to the underlying domain and improve model performance in domains without supervised data.

Recent successes of training in games and simulation-like environments [Silver et al., 2017, Mnih et al., 2015, Campbell et al., 2002] have caused strong public confidence in current machine learning and reinforcement learning technology. Research in real robot applications has not been able to match similar progress as limitations arise based on various costs: human effort, time required, expensive hardware. Training in simulation enables us to overcome most of them.

However, simulations consistently differ from the final application not only visually (as can be addressed via domain adaptation - Chapter 4) but additionally in terms of the underlying system dynamics. Parameters such as friction, density, mass, and the continuous distribution of these properties throughout objects are only approximations. To enable the strengths of training in simulation to be ported to real world applications, we focus on an approach for parallel training of both systems in Chapter 5, where the learning process of one system is exploited to guide the training of the other.

1.2 Contributions

This thesis' contributions are threefold in the field of improving the supervision for robot learning algorithms. We focus in this context on three important sub-

fields: training through imitation learning, domain adaptation, and transfer learning. These areas represent prime opportunities for simplifying and accelerating robot training and are intended to move robotic research closer to application in real world tasks.

1.2.1 Inverse Reinforcement Learning

We develop an approach for highly-scalable inverse reinforcement learning [Wulfmeier et al., 2015]. Building on deep neural networks, we present a method which is able to learn from large datasets of human driving to extract the underlying preferences and enable the use of extracted knowledge as cost maps for motion planning.

The approach is evaluated on a large 120 km driving dataset, which was collected over the course of a year with 13 different drivers, resulting in a total of 25.000 trajectory samples. The evaluation demonstrates the improved performance of the learning based approach over manually crafted cost function with respect to the classification of traversable paths and prediction of human behaviour [Wulfmeier et al., 2016a, 2017c].

To further benefit from human intuition and domain knowledge the method is extended in [Wulfmeier et al., 2016b, 2017c] to incorporate manually designed cost function as prior initialisation into the learning process.

1.2.2 Unsupervised Domain Adaptation

Machine Learning models - even when trained to high performance in their source domain - can display significantly reduced accuracy when applied on input data distributions that have shifted from the original training data. Prior work in robotics on addressing appearance shifts often aims to address this challenge by imposing additional constraints on the type of shift between domains [Ratnasingam and Collins, 2010, Sugiyama et al., 2008].

Current state-of-the-art results on toy domains are obtained without additional constraints via adversarial domain adaptation (details in Section 2.4). However, the approach is known to be notoriously unstable during training such that application

on more complex task represents a challenge. While prior work on ADA has focused on toy-problems based on MNIST, SVHN and similar datasets [Ganin et al., 2016, Bousmalis et al., 2016], we investigate relevant aspects for scaling the approach to state-of-the-art architectures and classification as well as semantic segmentation applications in outdoor robotics.

We identify relevant hyperparameters and further training details, and enable scaling the method to adapt popular, high-performing architectures for classification. Furthermore, we introduce adversarial domain adaptation to the task of pixel-wise segmentation [Wulfmeier et al., 2017a]. The method is evaluated based on performance on real world driving data with focus on robustness to weather as well as lighting conditions.

1.2.3 Transfer Learning for Reinforcement Learning

Simulation to real world transfer has gained in relevance in recent years as reinforcement learning approaches have been successfully applied in simulations and games, partially achieving superhuman performance. However, most of these successes rely on enormous amounts of training data, which can be only created in simulation environments because of faster-than-realtime speeds and no safety requirements.

We focus in our work on transfer learning between agents in different environments (simulation-real platform), while performing the same task. The simulation to real transfer scenario commonly includes two challenges: differences in observation space as well as system dynamics [Wulfmeier et al., 2017b]. When focussing on the latter with our work we train real-world and simulation agents in parallel. By introducing auxiliary rewards into both systems based on the distributions over visited states, we can accelerate training in the real world based on knowledge gained in simulation.

While previous work on transfer between systems of different dynamics commonly focuses on the transfer of trained models such as robust policies [Rajeswaran et al., 2016] or identified inverse dynamics models [Christiano et al., 2016], the

parallel training method guides exploration continuously during optimisation and utilises information from each system to further training on the other.

1.3 Thesis Outline

This thesis is split into six chapters, following the introduction in this chapter with an extended review of the theoretical background in Chapter 2. It covers the preliminaries for imitation learning, domain adaptation as well as transfer learning required to fit the contributions in the following chapter into their academic context. Chapter 3 summarises the research contributions in the field of inverse reinforcement learning and focuses on the application and adaptation of the developed frameworks to an urban driving scenario. It covers multiple publications, including the development of Maximum Entropy Deep Inverse Reinforcement Learning [Wulfmeier et al., 2015] as well as applications in learning to drive by imitating human experts [Wulfmeier et al., 2016a,b, 2017c][1] . Chapter 4 focuses on unsupervised domain adaptation for the adaptation of machine learning models to domains without labels. The chapter introduces in particular a detailed evaluation of the customisation and optimisation of adversarial domain adaptation procedures with the application to outdoor robotics and extends the approach to adapt models for pixel-wise segmentation [Wulfmeier et al., 2017a]. Chapter 5 targets simulation to real robot transfer within the reinforcement learning paradigm. The presented method on Mutual Alignment Transfer Learning [Wulfmeier et al., 2017b] determines auxiliary rewards to accelerate the learning process of a real world robot by training in parallel in simulation for the same tasks. Finally, Chapter 6 concludes this thesis by summarising the contributions made in Chapters 3 to 5 and points towards future work to improved the presented methods and move robotic learning closer to practical applications.

[1] International Conference on Intelligent Robots and Systems 2015 - Best Student Paper Award

2

Efficient Supervision for Machine Learning in Robotics

Contents

This thesis adapts and extends various machine learning methods for increasing the efficiency of providing supervision to robotic systems. To enable insight into the tools used, we will cover the relevant background in this section.

Machine learning is most commonly divided into three categories: *supervised*

learning, *unsupervised learning* and *reinforcement learning*. This section aims to quickly introduce the reader to the ideas behind the two categories which are of relevance for understanding this thesis, supervised and reinforcement learning. The interested reader is furthermore referred to the excellent textbooks of Bishop [2006] and Murphy [2012] for more detailed introductions to machine learning.

The later sections aim to introduce further subfields of importance for efficient design of supervision systems in robot learning. In particular, we rely on learning from human demonstrations, adaptation of models between different domains and simulation-to-robot transfer learning.

2.1 Supervised Learning

Supervised learning addresses the problem of approximating the mapping between input-output pairs in a dataset. Independent of the type of model used to represent this mapping, there can be two streams of supervised learning - *regression* and *classification*.

In regression, our aim is to learn a mapping $f(x) = y$, a function from input samples $x \sim X$ to the corresponding *continuous* labels $y \sim Y$ of dimensionality n. In order to achieve this goal, we commonly optimise f to minimise the expectation of the loss function $\mathcal{L}_f$,

$$f = \underset{f}{\operatorname{argmin}}\ \mathbb{E}_{x,y\sim X,Y}[\mathcal{L}_f(x, y)] \tag{2.1}$$

One example for the loss function is the mean squared error as represented via Equation 2.2.

$$\mathcal{L}_{MSE}(x, y) = \sum_{i=1}^{n} (f_i(x) - y_i)^2 \tag{2.2}$$

Similar to regression, classification aims to learn a similar mapping. However, in the context of classification the labels are *categorical* represented as one-hot

encoding of dimensionality n. A common loss function for the problem of classification is the cross entropy (Eq. 2.3)

$$\mathcal{L}_{CE}(x, y) = \sum_{i=1}^{n} y_i \log(f_i(x)) \tag{2.3}$$

2.2 Reinforcement Learning

In reinforcement learning (RL), the algorithm learns from interaction with its environment. While a supervised learning algorithm receives its training signal based on an already existing dataset, the RL agent shapes the training dataset by affecting with its actions both rewards gained and future states. The algorithm - in the context of RL often called *agent* - observes its environment and performs an action. Subsequently, it receives a reward and the environment potentially changes its state based on the previous state and the agent's action. Generally, the agent might not be able to observe the complete state such as in systems which are only partially observable. However, for the purpose of the following chapters and for clearness of the following equations, it suffices to focus on direct and complete observation of the environment state.

The RL problem is commonly formulated using the Markov Decision Process (MDP) framework $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p, \gamma, r\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the set of possible actions, and γ is a discount factor that modulates the influence of future rewards. Let state and action at time t be defined respectively as s_t and a_t. The system dynamics describe how the environment state changes as a function of previous state and agent action as $p(s_{t+1}|s_t, a_t)$. The reward function $r(s_t)$ describes the agents objective, a preference for being in a particular state.

Let μ_0 be the initial state distribution, which describes the initial state, and μ_t be the state visiting frequencies at time t determined according to Equation 2.4. The state visiting frequencies represent a distribution over the states at time t.

$$\mu(s_{t+1}) = \sum_{s_t} \sum_{a_t} \mu(s_t)\ p(s_{t+1}|s_t, a_t)\ \pi(a_t|s_t) \quad \text{where } \mu(s_0) = \mu_0 \tag{2.4}$$

The discounted visiting frequencies ρ_π can be determined following Equation 2.5.

$$\rho_\pi(s) = \sum_{i=0}^{\infty} \gamma^i \mu(s_i) \quad \text{with } s = s_i \tag{2.5}$$

Let V_π, Q_π, A_π respectively be the state value function, state-action value function, and advantage function. All three functions can be utilised to either judge quality of states or state-action pairs. The state value function describes the expected value of being in a state s, which is being extended by the state-action value function to describe the expected value of performing a specific action a while being in that state. The advantage function represents the expected advantage we gain by performing an action a in state s over the expected value of being in this state.

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}[\sum_{i=0}^{\infty} \gamma^i r(s_{t+i})] \tag{2.6}$$

$$V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, a_{t+1}, \ldots}[\sum_{i=0}^{\infty} \gamma^i r(s_{t+i})] \tag{2.7}$$

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) \tag{2.8}$$

Policy gradient (PG) algorithms represent a common approach to the reinforcement paradigm, which directly update the parameters $\boldsymbol{\theta}$ that define the agent's policy π, to maximise the expected discounted reward $\eta(\pi)$.

$$\eta(\pi) = \eta(\theta) = \mathbb{E}_{s_0, a_0, \ldots}[\sum_{t=0}^{\infty} \gamma^t\ r(s_t)] \quad \text{where } s_0 \sim \mu_0, a_t \sim \pi(a_t|s_t), s_{t+1} \sim p(s_{t+1}|s_t, a_t) \tag{2.9}$$

PG algorithms are commonly applied in robotics for environments with continuous action spaces. For a more detailed introduction to reinforcement learning algorithms, the reader is referred to Sutton and Barto [1998]. To maximise the return we update the parameters by taking a step into the direction of the gradient $\nabla_\theta \eta(\theta)$ weighted by our stepsize α.

$$\theta = \theta + \alpha \nabla_\theta \eta(\theta) \tag{2.10}$$

with the gradient given by

$$\begin{aligned}
\nabla_\theta \eta(\theta) &= \nabla_\theta \mathbb{E}_{s_0, a_0, \ldots}[r(\tau)] && (2.11)\\
&= \nabla_\theta \int_{\mathbb{T}} p_\theta(\tau) r(\tau) d\tau && (2.12)\\
&= \int_{\mathbb{T}} \nabla_\theta p_\theta(\tau) r(\tau) d\tau && (2.13)\\
&= \int_{\mathbb{T}} p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) d\tau && (2.14)\\
&= \mathbb{E}_{s_0, a_0, \ldots}[\nabla_\theta \log p_\theta(\tau) r(\tau)] && (2.15)
\end{aligned}$$

where $r(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t)$ describes the sum over discounted rewards along a trajectory rollout $\tau = \{(s_0, a_0), (s_1, a_1), ...\}$ to simplify the notation. Let $p_\theta(\tau)$ be the probability for the rollout given the current policy parameters and $\mathbb{T}$ be the space of trajectories to integrate over for estimating the gradient step. The step between Equations 2.13 and 2.14 is known as the likelihood ratio or REINFORCE [Williams, 1992] trick and enables us to compute a Monte Carlo estimator of the gradient.

2.3 Imitation Learning

Learning from demonstration (LfD), also known as imitation learning, usually describes a sub-field of supervised learning, a set of methods for learning to solve tasks via imitation. As later discussed under inverse reinforcement learning, some approaches rely on a combination of supervised and reinforcement learning.

In order to deploy autonomous platforms in new locations and open the task to untrained personnel, learning from demonstration approaches represent a promising alternative and have led to recent advances in two principal areas, *policy learning* (or behavioural cloning) and *inverse reinforcement learning* (IRL). The following sections represent a compressed summary of relevant work regarding both areas underlying the contributions in this thesis. The interested reader is forwarded to a more comprehensive survey by Argall et al. [2009].

2.3.1 Policy Learning

Policy learning, also commonly known as *behavioural cloning* (BC), focuses on directly learning the expert's policy, a mapping from observations to the agent's actions, while IRL in contrast focuses on inferring the reward structure which is underlying the agent's demonstrations - the preferences that cause specific behaviours. Behavioural cloning is a straightforward approach which frames LfD as supervised learning problem and trains a regression model or classifier (in the categorical setting) to replicate the expert's behaviour. While BC has been applied in many fields, we will focus on applications regarding mobile platforms as this represents the field of application in Chapter 3.

Early work on modelling driving behaviour with neural networks [Pomerleau, 1990] focuses on learning a discretised steering output based on down-sampled front-facing image streams and fully connected layers in a neural network. The approach is extended in recent work via the application of convolutional neural networks (CNNs) [Sermanet et al., 2010, Chen et al., 2015] and scaled significantly with respect to dataset size, task complexity and performance.

These methods for learning from demonstration differ from regular supervised learning mostly at test time. The actions performed at timestep t will influence the state and therefore the input data at time $t + 1$. Direct imitation of a policy is well suited for the design of reactive, short-horizon controllers. However long-term decision making systems require to plan further ahead which can lead to a problem known as *compounding of errors*. The described principal challenge for

policy learning from demonstration lies in generalisation. Policies are learned for states along demonstrator trajectories. After training the model on the input data distribution collected from demonstration, models have to generalise to the data encountered at test time. Small prediction errors for the agent's actions affect future encountered states which again affect future actions. As the agent diverges from its training distribution its performance reduces continuously, such that finally the recovery to stable behaviour might become impossible.

A principal contribution of Chen et al. [2015] is the placement of multiple cameras on the car to simulate the field of view for positions further away to extend training data with correcting behaviours. Another common approach to address this problem is given by Ross et al. [2010] who query an expert to improve the policy for states not encountered in the original trajectories. While presenting one method to target the problem of compounding of errors, the approach increases demonstrator effort by requiring continuous supervision and monitoring. Further work on extending DAgger [Ross et al., 2010] mostly focuses on reducing this extra supervision effort [Laskey et al., 2016, Kahn et al., 2017b, Laskey et al., 2017].

2.3.2 Inverse Reinforcement Learning

Inverse reinforcement learning, also commonly known as *inverse optimal control*, builds upon learning the reward function underlying expert behaviour and is seen as an often more succinct and robust approach for describing long horizon tasks [Abbeel and Ng, 2004, Abbeel et al., 2008]. IRL-based reward functions and the corresponding policies prove to be robust with respect to the earlier mentioned challenge of compounding errors by interacting with the environment or a model during training and learn corrective behaviour for states not visited by the demonstrator.

While behavioural cloning in general is directed towards solving a specific task - similar to imitating particular trajectories with given initial position and goal, the reward function based approach can be significantly more adaptive by switching the goal and deriving a new policy for the changed setting.

Most IRL approaches can be formulated as running reinforcement learning in the inner loop of a supervised learning problem; optimising the reward function r to minimise the differences between RL agent behaviour and expert demonstrations.

Linear Inverse Reinforcement Learning

Early work introducing IRL into the machine learning community [Russell, 1998, Ng and Russell, 2000] sets the problem into the Markov Decision Process framework. They point out that the IRL formulation suffers two main problems, reward ambiguity - where multiple reward functions are equally well suited to explain expert behaviour - and the inherently suboptimal behaviour of human experts, which has to be explicitly modelled.

The reward function $r(s) = r(f(s))$ in IRL can be defined as a function of the features $f(s)$ describing the state.

To enable the use of simple, linear reward mappings, the feature representation, f, is usually hand-crafted based on preprocessing such as segmentation and manually defined metrics, but can be learned in order to be optimised for the task - as displayed in Chapter 3.

Abbeel and Ng [2004] extend the original IRL formulation by introducing feature matching as approach for solving the IRL problem, where the reward function is optimised to match the feature visiting frequencies ν between expert demonstration and agent behaviour.

The feature frequencies $\nu = \sum_s \mu(s)\ f(s)$ describe how often different features are traversed and can be calculated as follows for a linear reward function $r(f(s)) = \theta^\top\ f(s)$ with parameters θ.

Maximum margin planning (MMP) [Ratliff et al., 2006] extends the original works on IRL [Ng and Russell, 2000, Abbeel and Ng, 2004] by addressing the problem of reward ambiguity via optimising the model to maximise the margin between the performance of the demonstration trajectories and any other other trajectories. They apply a quadratic programming formulation to solve the resulting constrained optimisation problem.

A popular, probabilistic extension is given by Maximum Entropy IRL [Ziebart et al., 2008], which represents the foundation of the approach presented in Chapter 3. This method applies the principle of maximum entropy to constrain the model for the distribution over expert trajectories as the least biased estimate for a given amount of information [Jaynes, 1957b]. The probability of any trajectory in Maximum Entropy IRL is exponentially dependent on the sum of rewards alongside it. Additionally, by using a stochastic policy for expert behaviour, we can describe suboptimal behaviour straightforward as samples of the model. The derivation of Maximum Entropy Deep IRL in Chapter 3 presents a more detailed description of the approach as well as the associated equations.

Nonlinear Inverse Reinforcement Learning

A linear reward model, while appropriate in some scenarios, is suboptimal if the true reward is a more complex, nonlinear function.

Learning to Search (LEARCH) [Ratliff et al., 2009] presents a natural extension of MMP [Ratliff et al., 2006] and combines the maximum margin approach with boosting [Freund and Schapire, 1995] to learn nonlinear reward functions.

A further approach for nonlinear IRL, is given by Choi and Kim [2013] who, in order to alleviate the modelling limitation, extend the linear model by introducing a mapping $\Phi : \mathbb{R}^N \rightarrow \{0,1\}^N$ such that

$$r(f, \Phi) = \theta^\top \Phi(f). \tag{2.16}$$

Here Φ denotes a set of composite features which are jointly learned as part of the objective function. These composites are assumed to be the logical conjunctions of the predefined, atomic features f. Due to the nature of the features used the representational power of this approach is limited to the family of piecewise constant functions.

More flexible non-linear function approximators such as Gaussian processes (GPs) further extend the modelling capacity of IRL models [Levine et al., 2011]. The expert demonstrations $\mathcal{D}$ are used in this context to identify an active set of

GP support points, $\mathcal{X}_u$, and associated rewards u. The mean function is then used to represent the individual reward at a state described by f.

$$r(f, \mathcal{X}_u, u) = K_{f,u}^{\top} K_{u,u}^{-1} u. \tag{2.17}$$

Here $K_{f,u}$ denotes the covariance of the reward at f with the active set reward values u located at $\mathcal{X}_u$ and $K_{u,u}$ denotes the covariance matrix of the rewards in the active set computed via a covariance function $k_\theta(f_i, f_j)$ with hyperparameters θ.

However, the application of a non-parametric approach causes the approach to scale poorly with the numbers of demonstration samples, quickly rendering the GP framework impractical due to the unfavourable computational complexity. Even sparse GP approximations lead to training computation complexity that is dependent on the size of the active set and the number of encountered samples ($\mathcal{O}(n \times m^2)$ where n denotes the number of experienced state-reward pairs and m the number of inducing points). Therefore, situations with increasingly complex, highly varying reward functions, that lead to higher requirements regarding the number of inducing points will render a non-parametric approach impracticable [Bengio et al., 2007].

In parallel to the work presented in Chapter 3, Finn et al. [2016] developed a sampling based approach to IRL for training neural networks with only fully connected layers in a MaxEnt IRL based framework for robotic manipulation and navigation tasks. The reward function is determined from handcrafted features that are parametrised based on the trajectory representation. The approach requires sampling of real-world training data during the training process to approximate a complex dynamic model.

2.3.3 Inverse Reinforcement Learning in Robotics

Inverse reinforcement learning is playing a pivotal role to address LfD tasks and finds applications in robotics [Argall et al., 2009, Finn et al., 2016, Pfeiffer et al.,

2016] where it allows robots to learn complex behaviour from human demonstrations. Due to its strengths in long-horizon planning tasks and the capability for integration into existing systems, recent work successfully applies IRL to planning [Shiarlis et al., 2017, Ratliff et al., 2009]. Further applications include improving driving and robotic navigation with focus on interaction of robots and humans [Kuderer et al., 2015, Kretzschmar et al., 2016, Pfeiffer et al., 2016, Hadfield-Menell et al., 2016], which target the modelling of the behaviour of other interacting agents. Finally, it is relevant to the fields of cognition [Baker et al., 2009] and preference learning [Ziebart et al., 2008] where it serves as a tool to better understand human decisions. A detailed review of both methods and applications of IRL can be found in [Zhifei and Joo, 2012].

2.3.4 Contributions to Inverse Reinforcement Learning

Our work focuses on employing fully convolutional networks in large-scale motion planning problems such as autonomous driving. In order to train neural networks end-to-end we adapt the Maximum Entropy IRL algorithm [Ziebart et al., 2008] and apply backpropagation to all network parameters to enable feature learning for the IRL objective. The method extends existing approaches and enables scalability with respect to complexity of environment, behaviour, and size of training datasets. As model-based IRL approach, our proposed method removes the requirement of sampling on the actual platform [Finn et al., 2016] resulting in increased safety and enabling the application of our approach to autonomous driving.

The approach is applied to learn from an extensive dataset of over 120 km capturing the common driving behaviour of 13 different human drivers in over 25,000 training samples. We furthermore demonstrate robustness towards systematic noise in form of miscalibration of the sensor setup. We extend the method to incorporate expert domain knowledge in a prior training step to address complexities of real world sensor data. The extension enables us to combine benefits of human expert knowledge with large-scale demonstration datasets.

2.4 Domain Adaptation

Machine learning models such as deep neural networks demonstrate proficiency in the domain of their training data. When required to generalise though they will deliver degraded performance in application domains that underlie distributional shifts. Traditionally, this problem has been addressed if possible via the collection of labelled data in all relevant domains.

Prior research in robotics focuses on appearance changes which can cause shifts between domains. Commonly, work in the field addresses the challenge by imposing additional constraints on the type of transformations between domains [Ratnasingam and Collins, 2010].

Unsupervised methods for domain adaptation have been applied with great success in computer vision [Daume III and Marcu, 2006, Sugiyama et al., 2008, Shrivastava et al., 2017] with little to no assumption on the types of shifts. These approaches are commonly applied if the collection of supervised data in all relevant domains is too expensive, time-consuming or generally not possible. The principal objective is to generate a mapping from source and target domains into a space that is invariant with respect to the domain of data origin. This can be achieved by finding a mapping from both domains into a third invariant domain, or by learning the mapping from one domain to the other.

2.4.1 Unsupervised Domain Adaptation

The field of unsupervised domain adaptation is of particular interest as it represents a common scheme behind many applications where labelled datasets are created at once and subsequently only unlabelled data becomes available. Traditional approaches for addressing the problem have focused on separately modelling the densities of source and target domain [Daume III and Marcu, 2006] and imposing additional structure [Sugiyama et al., 2008]. Density estimation itself presents a challenging task without the prior assumption of specific structure [Sugiyama et al., 2008].

In the context of domain shift between training and test datasets, Ben-David et al. [2007] derived theoretical upper bounds on source-optimised classifiers' performance on the test data. In the following section, we will focus on literature on unsupervised methods for domain adaptation without prior constraints on the type of transformation other than representability in the function space of arbitrary neural networks.

2.4.2 Domain Adaptation with Neural Networks

As in many related fields, a majority of recent achievements in domain adaption [Shrivastava et al., 2017, Hong et al., 2017, Hoffman et al., 2016] has built on the steady trend towards applying deep neural networks, enabling learning of hierarchical representations for arbitrary combinations of differential objectives.

In the context of domain adaptation, we combine objectives for the minimisation of mutual information between feature representations and the underlying domain while maximising relevance towards a supervised objective in the source domain. This possibility to directly optimise feature representations via backpropagation for domain invariance [Ganin et al., 2016] or target-source mappings [Long et al., 2017] has lead to the significant success of deep architectures in this field.

Long et al. [2017] focus on minimising the Maximum Mean Discrepancy for the feature distributions of multiple layers of the network architecture. Rozantsev et al. [2016] extend the idea and impose a penalty for deviations between the network parameters across domains. Sun et al. [2017] align second order statistics of layer activations for source and target domains. Hoffman et al. [2016] match the label statistics between the true source and predicted target labels for semantic classification.

Recently, the field has been extended with adversarial approaches to domain adaptation [Ganin et al., 2016, Ajakan et al., 2014] which have resulted in strong performance improvements [Bousmalis et al., 2016]. Adversarial domain adaptation (ADA), also known as *domain adversarial training*, relies on training a discriminator to differentiate the domains underlying feature distributions while

competing against an encoder which attempts to conceal the origin domain of input samples. The method presents a fully differentiable mechanism for density ratio estimation and subsequent density adaptation between the feature space distributions of different domains.

While adversarial training techniques have been shown to be notoriously unstable and difficult to optimise, there has been a pronounced body of work towards improving their stability, including more dominant use of the confusion loss [Goodfellow et al., 2014] and more recently the Wasserstein GAN framework [Arjovsky and Bottou, 2017].

Practical successes of these methods has been demonstrated in tasks such as medical image segmentation [Kamnitsas et al., 2017], gaze estimation [Shrivastava et al., 2017] and transfer for reinforcement learning [Tzeng et al., 2015a].

2.4.3 Domain Adaptation in Robotics

In outdoor robotics, the problem of appearance change has been a long-standing challenge for both laser and in particular image data. During the continuing deployment of a platform, we will unavoidably perceive data from domains that have shifted from the original source domain of training data.

A great deal of effort has focused on designing and comparing various feature transformations with the goal of creating representations invariant to environmental change [Lowry et al., 2016]. For visual appearance, the colour-constant transform [Ratnasingam and Collins, 2010] is commonly applied directly on the image to achieve lighting invariance for localisation [Corke et al., 2013], visual teach and repeat [Paton et al., 2015] and segmentation [Maddern et al., 2014]. Lai and Fox [2009] focus on LIDAR and present an exemplar based approach to align spin-image features from web based 3D models to dense laser scan.

Other approaches address the problem through the retention of multiple experiences [Churchill and Newman, 2013] or synthesising images between discrete domains [Neubert et al., 2013]. Recent work aims at extending the applications of unsupervised domain adaptation improving adversarial approaches (Section 2.4.2)

which can be applied independently of prior assumptions on the structure of distributional shifts [Hoffman et al., 2016].

2.4.4 Contributions to Domain Adaptation

In extension of the reviewed work, we develop a framework for applying adversarial techniques to adapt popular, state-of-the-art network architectures with the new objective to align feature distribution across domains, streamlining domain adaptation for these models. Adversarial training is known to be notoriously unstable. Therefore, we begin our work by performing an extensive ablation study, adapting many techniques known to stabilise generative adversarial networks, and evaluate on a surrogate task with the same underlying appearance changes. The final application of our work lies in the domain of semantic segmentation of drivable terrain in an urban scenario.

2.5 Transfer Learning

Transfer learning has spread into many branches of machine learning including transferring knowledge between different environments to the performance of the different tasks and arbitrary variations of these scenarios.

While domain adaptation specifically addresses differences in the distributions underlying observations between domains, transfer learning more broadly considers the transfer of either model or information between different environments. While it is a topic for many fields, here we focus on the reinforcement learning paradigm as a basis for simulation to real robot transfer learning in Chapter 5.

2.5.1 Transfer for Reinforcement Learning

Given significant advances in training reinforcement learning agents in simulations, the field of transfer learning has become more important in recent years. In some cases the simulator-given environment represents the final application environment, as for example when solving computer games, such as various Atari games [Mnih et al., 2015] and Ms. Pacman [Van Seijen et al., 2017], as well as board games,

such as Chess [Campbell et al., 2002] and Go [Silver et al., 2016]. In most cases however, the direct application of models trained in simulation to real world tasks results in significantly decreased performance and correlated safety concerns. We constrain our literature review here to transfer between different environments while solving equivalent tasks. In these situations, direct transfer of fixed policies can underperform due to two possible types of discrepancies: different observation model or different system dynamics between the scenarios.

In general, the problem of different observation spaces has been addressed in the framework of unsupervised domain adaptation [Ganin et al., 2016, Long et al., 2017] as well as with particular application to robotics [Tzeng et al., 2015a, Stadie et al., 2017]. Rusu et al. tackles the transfer task by reusing features learned for simulation and focusing on learning residual representations required to adapt to the real world task. Furthermore, recent work addresses the task via visual domain randomisation, showing that the requirement for exact simulation can be minimised when instead training for various variations of textures and colours [Tobin et al., 2017, Sadeghi and Levine, 2016] without any training data required from the final application system.

The second challenge is based on differences between the system dynamics of simulator and real task, which results in different optimal policies even when given access to the same observation model, is the focus of our work. Christiano et al. [2016] address the challenge by learning an inverse dynamics model for the real platform to determine the correct real world action based on the trajectory in simulation. Gupta et al. [2017] learn invariant feature spaces which are employed to transfer task knowledge even between systematically different robot architectures. However, the approach relies on prior information through proxy tasks to determine the invariant spaces. Another approach is given by Rajeswaran et al. [2016], who train robust policies based on sampling from an ensemble of simulated source domains while focusing on simulations with reduced performance. Additional methods [Abbeel et al., 2006, Marco et al., 2017] focus on exploiting inaccurate

simulations to improve real world performance with the latter employing Bayesian optimisation to improve the split between simulator and real world experiments.

2.5.2 Contributions to Transfer Learning

Our method for transfer learning in RL builds on recent advances in imitation learning. By training both agents in parallel and utilising an adversarial discriminator as density ratio estimator between the agent's state visitation distributions, we can create auxiliary rewards to guide training similar to the notion of imitation. These rewards generally support the alignment of the distributions over visited states between two agents. The approach is conceptually similar to adversarial imitation learning methods [Ho and Ermon, 2016, Stadie et al., 2017, Hausman et al., 2017] which train for one-sided alignment between the agent's policy and a set of demonstration trajectories.

We extend existing approaches by incorporating a density estimator for both distributions over visited states and we build on Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] and train a discriminator to classify the origin for visited states. The alignment rewards are generated based on the confusion (negated performance) of this discriminator. Naturally, the robot agent benefits from being guided by the simulator agent based rewards, as training in simulation happens at a faster rate and the agent is commonly more proficient at solving the task. Additionally, by training the simulator agent with the auxiliary alignment rewards, we guide the agent in simulation to visit similar states as the robot agent. In this way, it learns not only general behaviour for solving the task but focuses in particular on the state distribution visited by the robot agent.

3

Inverse Reinforcement Learning for Large-Scale Problems

Contents

Learning from demonstration (LfD), also known as *programming by demonstration*, is intended to render the task of specifying robot behaviour independent of knowledge in robotics and applied algorithms. The approach can be employed to enable untrained personnel to train and adapt machines to solve tasks via imitation. While other approaches to providing supervision result in potentially

large additional amounts of manual annotation (e.g. in pixel-wise segmentation), LfD enables us to easily generate large amounts of supervised data or even rely on existing datasets. In the following chapter, we employ methods from LfD to generate cost functions for motion planning in the context of navigation tasks.

A mission-critical capability for any mobile robotic platform is determining the traversability of its surroundings and computing a safe trajectory to its goal. Typically, this is performed by exploiting manually designed cost functions, which specify the cost of visiting each location or state as a function of sensory input data. However, the mapping between sensor input and cost is likely to be very complex. In particular, in environments containing a wide variety of obstacles, such as stairs, bollards, underpasses, or slopes, the cost of traversing a particular location is a complex function of the sensor data. In addition, the cost can be dependent on features in the spatial neighbourhood, rather than just the location itself. Consequently, the manual handcrafting of cost functions for motion planning is a difficult and laborious task, requiring expertise in robotics, sensing, and motion planning.

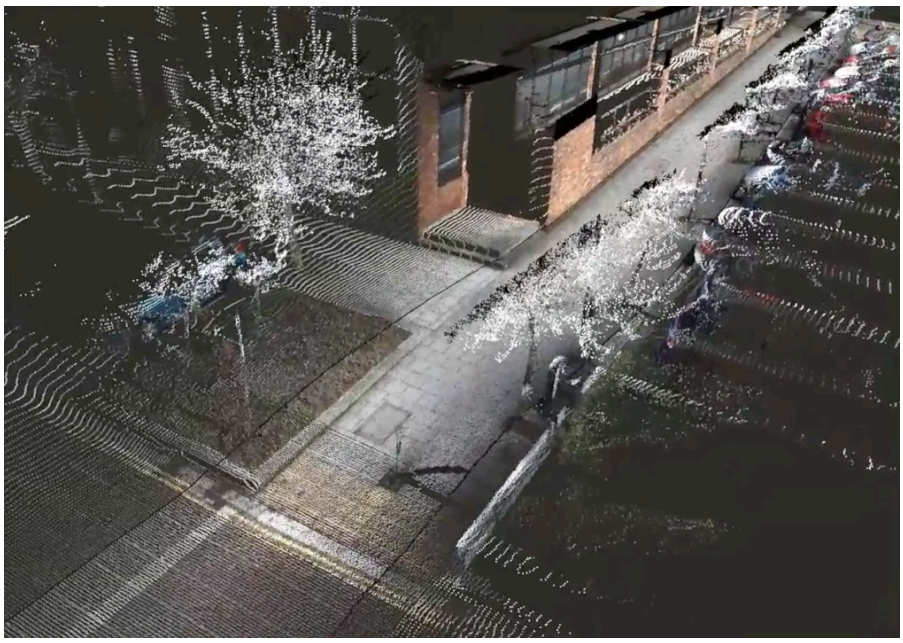


Figure 3.1: Coloured 3D LIDAR scan from the pedestrian areas in Milton Keynes. Colour for the points is provided via side-facing cameras with known calibration to the LIDAR system.

The inverse reinforcement learning (IRL) framework centres on learning the underlying reward or cost structure from demonstrations of human behaviour [Ziebart et al., 2008]. It has been applied to a wide range of robotic applications, including autonomous driving, robotic manipulation [Finn et al., 2016], and grid-world planning [Levine et al., 2011]. In principle, IRL eliminates the expertise and effort required to manually design cost functions, and can produce cost maps that are more consistent with human behaviour. However, the seminal IRL frameworks utilise a linear mapping between input features and output reward, which severely restricts the complexity of cost structures that can be accurately modelled. More recent state-of-the-art methods that have the required capacity are built upon non-parametric models such as Gaussian processes, which have a runtime dependent on the size of the dataset and are quickly rendered intractable for large amounts of demonstration data [Ramachandran and Amir, 2007, Levine et al., 2011].

The expressive capacity of deep neural networks has led to state-of-the-art performance in numerous applications and tasks [Mnih et al., 2015, Silver et al., 2016, Long et al., 2015]. We posit that as flexible, parametric function approximators they are well-suited to an IRL framework. In particular, they have the capacity to model the complex relationship between sensory input and reward structure, are able to capture spatial correlations in the data using convolutional layers, and exhibit constant time operation, allowing them to scale to very large datasets.

In common with most approaches to IRL, the presented method requires a transition model of the environment to simulate behaviour under the approximated reward function. This model can be either handcrafted, as in this work, or learned from sampling trajectories on the actual platform, as investigated by Finn et al. [2016].

The work presented in this chapter explores the use of fully convolutional neural networks (FCNs) to model the reward structure for a synthetic dataset [Wulfmeier et al., 2015], and for a real-world application to LIDAR data from a mobile vehicle [Wulfmeier et al., 2016a, 2017a,c]. We will evaluate the ability of FCNs to

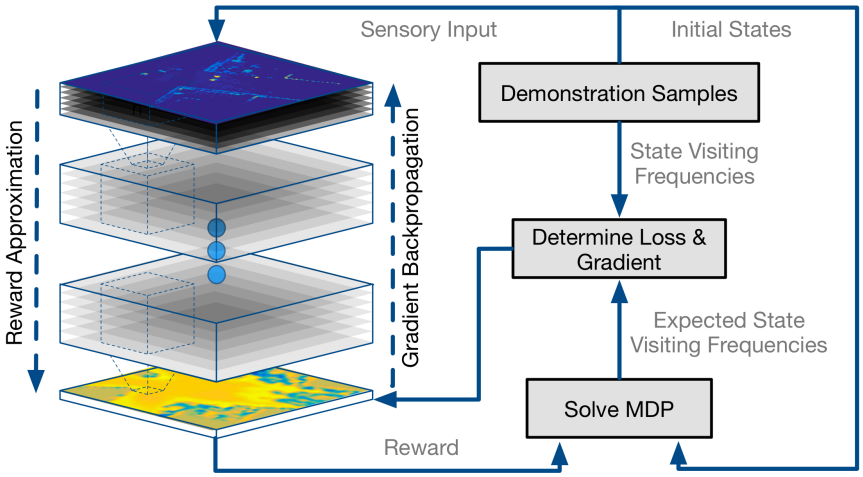


Figure 3.2: Schema for training neural networks in the Maximum Entropy paradigm for IRL. A randomly initialised network is iteratively refined to align the MDP solutions for given scenarios to the respective demonstration trajectories. The complete procedure is described in Section 3.1.3

.

accurately infer the underlying reward structure from human driving demonstrations in the following sections. The approach leads to control policies that more closely mimicked the behaviour of a human driver.

The principal contributions of our work are:

- Development of Maximum Entropy Deep Inverse Reinforcement Learning, a framework for applying high-capacity neural network architectures to solve the inverse reinforcement learning problem. This extends scalability with respect to complexity of environment, behaviour, and size of training datasets. This was first proposed in Wulfmeier et al. [2015].

- Qualitative and quantitative evaluation of the approach on small scale scenarios against state-of-the-art approaches in IRL. We experimentally validate the convergence of task-optimal spatial features through IRL based training given enough demonstration samples on common benchmark scenarios. This was also presented in Wulfmeier et al. [2015].

- Evaluation of scalability and efficacy in real world tasks on an extensive dataset of over 120 km capturing the common driving behaviour of 13 different

human drivers, resulting in over 25,000 training samples. This was first demonstrated in Wulfmeier et al. [2016a].

- Demonstration of robustness towards systematic noise via miscalibration of the sensor setup and demonstrating significantly stronger performance for prediction of human trajectories and classification of traversable terrain. This was also included in Wulfmeier et al. [2016a].

- Refining human prior cost maps to address additional complexities of real world sensor data and sparse feedback from IRL to combine benefits of human intuition with large scale demonstration datasets. This was introduced in Wulfmeier et al. [2016b].

3.1 Cost Function Learning

Manual cost function design builds a foundation for the majority of state-of-the-art motion planning systems for robotic platforms (e.g. Choset [2005]), with recent successful examples for the application in autonomous driving given by the competing teams in the DARPA Grand [Thrun et al., 2006] and Urban Challenges [Urmson et al., 2009, Montemerlo et al., 2008]. Obstacles have to be explicitly identified and are typically inflated as a function of the vehicle size. The necessary weighting of costs from different sensing modalities and the adaptation for different driving behaviours relies on detailed domain knowledge. In addition, the task of extracting good features from raw input data for computing the cost maps is often non-trivial and relies heavily on a well-calibrated hardware setup.

When employing a mobile robot in a new environment, it is pivotal to incorporate knowledge about terrain traversability into the system. While it is possible to describe generally reasonable cost functions to capture the terrain constraints, it is very difficult and time consuming to consider all possible corner cases. Manually designing these cost functions can prove challenging and time intensive, and limits the task to highly specialised personnel. In the following section, we present an approach to automatically learn these cost functions based

on inverse reinforcement learning for large datasets of demonstration trajectories within the example application of autonomous driving.

3.1.1 Inverse Reinforcement Learning

To overcome the limitations of manual handcrafting of cost functions, we frame the aforementioned task in the context of inverse reinforcement learning, which has the principal goal of inferring the preferences or the reward structure that underlies specific behaviours. “Reward" and “cost" will be used interchangeably in the following sections as one is the straightforward negation of the other.

The approach is commonly formulated using the Markov Decision Process framework $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, \gamma, r\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the set of possible actions, T is the state transition model, γ is a discount factor that modulates the influence of future rewards, and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a function specifying the reward structure. As r is not provided, it must be inferred from a set of demonstrations $\mathcal{D} = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_N\}$, each of which is a sequence of state-action pairs $\varsigma_i = \{(s_1, a_1), (s_2, a_2), \ldots, (s_K, a_K)\}$ representing a sample trajectory.

3.1.2 Maximum Entropy IRL

When solving the IRL problem, one inherently needs to address two main challenges: suboptimality of demonstration trajectories given the underlying reward - as no human expert will ever act completely optimally; and reward ambiguity - as multiple rewards can explain the same behaviour.

The Maximum Entropy approach addresses these challenges by modelling expert behaviour as a distribution over trajectories and constraining this distribution to the one of highest entropy [Ziebart et al., 2008]. It defines the experts' underlying policy $\pi_E(a|s)$ such that the probability $p(\varsigma \mid \theta)$ for user preference of any given trajectory ς between specified start and goal states is proportional to the exponential of the sum of rewards $r(\varsigma, \theta)$ along a trajectory of length K:

$$r(\varsigma, \theta) = \sum_{i=1}^{K} r(s_i, \theta) = \mu_\varsigma^T \; r_S \tag{3.1}$$

where μ_ς denotes the vector of state visiting frequencies of a trajectory ς and r_S the vector of rewards for all states. The state visiting frequencies intuitively describe how often a particular state is visited during the trajectory ς. Finally, the probability of a trajectory is given in Equation 3.2.

$$p(\varsigma \mid \theta) \;\; = \;\; \frac{e^{r(\varsigma,\theta)}}{\sum_{\varsigma_p \in P} e^{r(\varsigma_p,\theta)}} \tag{3.2}$$

where P is the set of all possible trajectories. By employing the principle of Maximum Entropy Modelling, the approach results in the least biased estimate given the demonstration trajectories [Jaynes, 1957a].

3.1.3 Maximum Entropy Deep IRL

In the original MaxEnt IRL formulation [Ziebart et al., 2008] the reward $r(s, \theta)$ is defined as a linear function of input features $f(s)$, i.e. $r(s, \theta) = \theta^T f(s)$. The features in this context are the observations at a given state $f(s)$ such that the reward is parametrised on sensor observations rather than absolute state, which supports generalisation and enables application in large state spaces.

The application of a linear model relies on the ability to extract meaningful features from the data, and restricts the complexity of the mapping from input to reward that can be learned. This approach moves all complexity into the manual design of well-suited features.

We propose Maximum Entropy Deep Inverse Reinforcement Learning (MEDIRL) which employs fully convolutional neural networks (FCNs) to model the reward structure directly as a nonlinear function of the inputs [Wulfmeier et al., 2015]. Neural networks are flexible, powerful, function approximators, which makes them ideal for this task of predicting the reward structure for a given sensor input. By utilising convolutional layers for this task, we eliminate the need to handcraft

spatial feature mappings from raw data input, which instead can be learned to be optimal for the respective task. Furthermore, the approach can be scaled to large datasets and applied to learn non-linear mappings.

The network parameters θ is trained to maximise the probability $p(\mathcal{D} \mid \theta)$ of the demonstration dataset $\mathcal{D}$ and model parameters under their prior $p(\theta)$:

$$\mathcal{L}(\theta) = \underbrace{\log p(\mathcal{D} \mid \theta)}_{\mathcal{L}_\mathcal{D}} + \underbrace{\log p(\theta)}_{\mathcal{L}_\theta} \tag{3.3}$$

where the loss function $\mathcal{L}$ can be split into the log-likelihood of the demonstration data $\mathcal{L}_\mathcal{D}$, and a regularisation term denoting the log probability of the network parameters $\mathcal{L}_\theta$. Here, we will focus on the data likelihood since parameter regularisation terms can trivially build on various, common approaches such as L1-norm and L2-norm or Dropout [Hinton et al., 2012].

The data likelihood term relates to differences in feature counts $(f_E - \mathbb{E}[f])$ [Abbeel and Ng, 2004] as we will show in the following paragraphs. In the original MaxEnt formulation [Ziebart et al., 2008], the feature counts are computed as a linear model, with the differences in state visitation frequencies $(\mu_E - \mathbb{E}[\mu])$ multiplied by a matrix F_s, as in Equation 3.10. Let F_s denote in this context the feature representations for all possible states. In the following, μ_E refers to the state visitation frequencies according to the expert trajectories, and $\mathbb{E}[\mu]$ is the expected state visitation frequencies under the policy. Thus, both feature and state visitation difference terms express how much the behaviour of the learned model differs from the behaviour underlying the demonstration trajectories.

Our final goal is the optimisation of the network parameters θ defining the reward function r to maximise the probability of expert demonstrations described by Equation 3.3. The optimisation can be achieved via gradient descent, where the gradients are calculated based on the following equations. The product over all trajectories (sum in log-space) in the dataset D is omitted in the following equations for clearness and simplicity reasons, but can be trivially included since the

transformation into log-space enables independent computation for the gradients of each trajectory.

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial \theta} &= \frac{\partial \log \frac{e^{r(\varsigma,\theta)}}{\sum_{\varsigma_p \in P} e^{r(\varsigma_p,\theta)}}}{\partial \theta} && (3.4)\\
&= \frac{\partial r(\varsigma,\theta)}{\partial \theta} - \frac{1}{\sum_{\varsigma_q \in P} e^{r(\varsigma_q,\theta)}} \sum_{\varsigma_p \in P} e^{r(\varsigma_p,\theta)} \frac{\partial r(\varsigma_p,\theta)}{\partial \theta} && (3.5)\\
&= \frac{\partial r(\varsigma,\theta)}{\partial \theta} - \sum_{\varsigma_p \in P} \frac{e^{r(\varsigma_p,\theta)}}{\sum_{\varsigma_q \in P} e^{r(\varsigma_q,\theta)}} \frac{\partial r(\varsigma_p,\theta)}{\partial \theta} && (3.6)\\
&= \frac{\partial r(\varsigma,\theta)}{\partial \theta} - \sum_{\varsigma_p \in P} p(\varsigma_p \mid \theta) \frac{\partial r(\varsigma_p,\theta)}{\partial \theta} && (3.7)
\end{aligned}$$

Based on the definition of the reward of a trajectory in Equation 3.1, the final gradients per sample are given by

$$\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial \theta} = \mu_{\varsigma} \frac{\partial r_S}{\partial \theta} - \mathbb{E}[\mu_{\varsigma}] \frac{\partial r_S}{\partial \theta}. \tag{3.8}$$

In the context of linear Maximum Entropy IRL the resulting equation for the gradients over the whole dataset is displayed below. Note that in this case the state visiting frequencies per trajectory μ_{ς} are replaced with the overall frequencies of the complete dataset $\mu_{\mathcal{D}}$.

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial \theta} &= \mu_E F_s - \mathbb{E}[\mu] F_s && (3.9)\\
&= f_E - \mathbb{E}[f] && (3.10)
\end{aligned}$$

By extending the original linear formulation to neural networks, we benefit from the efficiency of gradient backpropagation - expressed in the second term in Equation 3.11 - to find the nonlinear mapping from the difference in state visitation frequencies μ to the necessary updates for all relevant parameters.

$$\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial \theta} = \underbrace{(\mu_E - \mathbb{E}[\mu])}_{\text{State Visitation Matching}} \underbrace{\frac{\partial r_S}{\partial \theta}}_{\text{Backpropagation}}. \tag{3.11}$$

Algorithm 1: MEDIRL

Input: $\mu_E^a, f, \mathcal{S}, \mathcal{A}, T, \gamma, \alpha$

Output: network parameters θ^*

1: $\theta^1 = \text{initialise\_weights}()$
2: **for** n = 1 : N **do**
3: $r^n = r(f, \theta^n)$

planning step

4: $\pi^n = \text{approx\_value\_iteration}(r^n, \mathcal{S}, \mathcal{A}, T, \gamma)$
5: $\mathbb{E}[\mu^n] = \text{propagate\_policy}(\pi^n, \mathcal{S}, \mathcal{A}, T)$

determine objective

6: $\mathcal{L}_D^n = \log(\pi^n) \times \mu_E^a$
7: $\frac{\partial \mathcal{L}_D^n}{\partial r^n} = \mu_E - \mathbb{E}[\mu^n]$

parameter update

8: $\frac{\partial \mathcal{L}_D^n}{\partial \theta^n} = \frac{\partial \mathcal{L}_D^n}{\partial r^n} \frac{\partial r^n}{\partial \theta^n}$
9: $\theta^{n+1} = \theta^n + \alpha \frac{\partial \mathcal{L}_D^n}{\partial \theta^n}$
10: **end for**

Algorithm 1 illustrates the steps for iteratively refining cost functions in the MEDIRL framework to maximise the probability of expert demonstration trajectories. In each step the current cost function is evaluated based on input observations f and the current parameters θ^n. Given the current cost function, lines 4 and 5 determine the state visiting frequencies based on a Maximum Entropy policy described in Equation 3.2. Lines 6 and 7 subsequently compute the current objective and the gradient with respect to the reward, leading to the network parameter update in lines 8 and 9. Algorithm 2 describes a common method for computing approximate value iteration and determines the policy based on the current approximation of the reward and the transition model.

Algorithm 2: Approximate Value Iteration

Input: $r^n, \mathcal{S}, \mathcal{A}, T, \gamma$

Output: π^n

1: $V(s) = -\infty$

2: **repeat**

3: $V_t(s) = V(s); \quad V(s_{goal}) = 0$

4: $Q(s,a) = r(s,a) + E_{T(s,a,s')}[V(s')]$

5: $V(s) = softmax_a \; Q_i(s,a)$

6: **until** $max_s(V(s) - V_t(s)) < \epsilon$

7: $\pi^n(a|s) = e^{Q(s,a)-V(s)}$

3.1.4 Architectures

We will investigate MEDIRL in the context of both standard toy benchmarks and large-scale application in the field of autonomous driving.

Section 3.2.1 focuses on toy scenarios and applies only simple, relatively shallow networks as they suffice for the task at hand. In the subsequent sections, we evaluate three more advanced architectures with varying properties for the application of cost function learning for motion planning on a large scale real world dataset.

The basic, serial architecture displayed in Figure 3.3 is limited regarding the size of its receptive field for every cell in the final cost map as the width is inherently function of the sum of widths of each layers kernels.

In addition to a standard, serial FCN, we introduce two variations to overcome these shortcomings. We argue that the exact position of a feature is often less relevant than its general presence, as, for example, any obstacle will influence a larger surrounding group of states in the context of motion planning cost functions.

The requirement for wider receptive fields in combination with a notion of limited translational invariance is commonly addressed via a max-pooling layer, leading to the implementation of the pooling FCN. This approach reduces the

amount of parameters needed for a fixed size receptive field and therefore helps to prevent overfitting.

Due to the distance-dependent sparsity of LIDAR pointclouds from the Velodyne scanners applied in 3.2.2, the feature representations for objects differ depending on where they are located. The max-pooling approach additionally reduces the need to learn all resulting feature patterns in this context.

While the pooling based architecture leads to increased spatial invariance with the advantages mentioned earlier in this section, this also leads to the irreversible loss of location information for low-level features. In order to combine the benefits while maintaining the networks ability to utilise exact location information when needed, we introduce the Multi-Scale (MS) FCN architecture shown in Figure 3.3 that is related to the Multi-Scale Deep-Jet architecture [Long et al., 2015] which integrates features of different scales for image segmentation. The proposed MS FCN architecture is able to treat the feature channels by concatenation instead of summation, similar to Szegedy et al. [2015]. In contrast to the MS Deep-Jet architecture, where feature channels from parallel branches share the same semantic meaning, we learn independent filter kernels representing different factors of influence for our cost maps.

Furthermore, a benefit of keeping all architectures fully convolutional is that we can design the size of the receptive field for each location in the final cost map to ensure coverage for an area encapsulating the size of the platform and all factors that might influence motion behaviour according to human intuition.

3.2 Evaluation

3.2.1 Standard Benchmarks

We benchmark the approach on commonly used small-scale toy scenarios for a proof of concept and a first quantitative evaluation against current state of the art approaches: GPIRL [Levine et al., 2011], NPB-FIRL [Choi and Kim, 2013] and the original MaxEnt [Ziebart et al., 2008]. Furthermore, we demonstrate that the

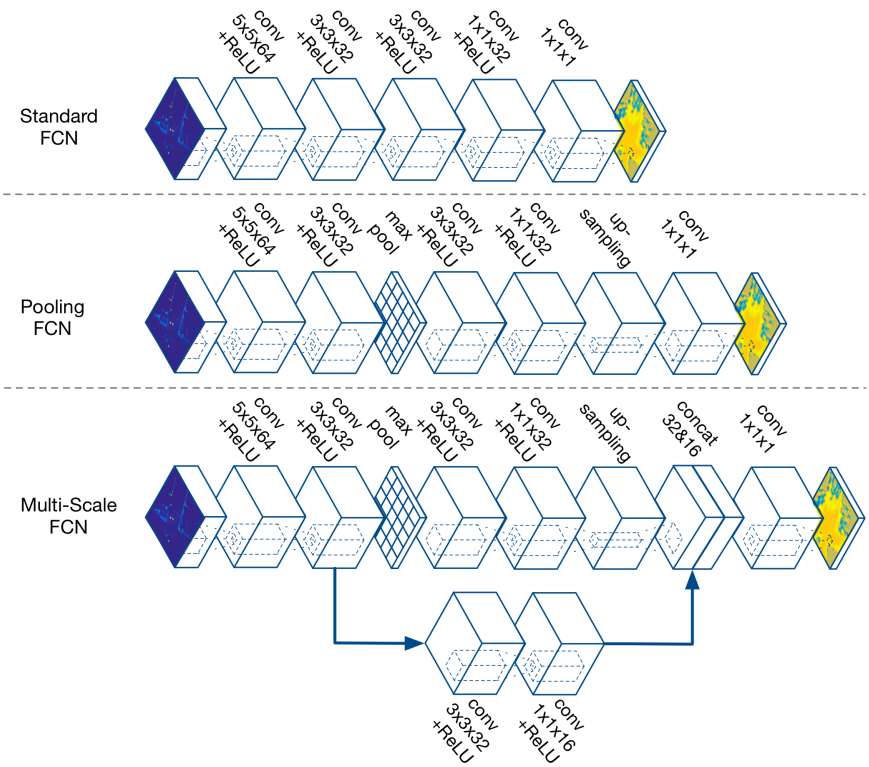


Figure 3.3: Illustration of the three proposed network architectures. The *standard FCN* represents a sequence of only convolutional and activation layers. The *pooling FCN* adds limited translational invariance to the standard architecture. To combine the capacity for translationally variant and invariant features, the *Multi-Scale FCN* combines two parallel branches with and without max-pooling layer.

application of convolutional layers can remove the dependency on separately defined spatial features. All experiments are performed in this scenario based on a small number of demonstration trajectories, typically less than 100. While the number of samples does represent a challenging application for deep architectures, it serves to show the ability of our approach to approximate arbitrary non-linear cost functions.

As these benchmarks focus on low complexity reward functions and very limited demonstration datasets, we employ comparably shallow networks based on two hidden layers and rectified linear units. To enable direct comparison against the other approaches, we limit the network to approximate the reward for a state only on the corresponding feature representation of that particular state, which corresponds to a FCN limited to filter widths of 1×1. This limitation is removed in Section 3.2.1 as the analysis targets the ability of the algorithm to learn spatial

features and eliminate the dependency on predetermining useful representations. The implementation of the experiments is based on MatConvNet [Vedaldi and Lenc, 2014] applying AdaGrad [Duchi et al., 2011] for network training.

Since the ground truth reward is given in these scenarios, we use *expected value difference* as the principal metric of evaluation. It represents a measure for the sub-optimality of the optimal policy for the learned reward function under the true reward. The analysis focuses on evaluating each specific training scenario and a number of randomly generated test environments. The *transfer* examples serve to analyse each algorithm's ability to generalise to the true reward structure and resist over-fitting.

Both evaluation scenarios, *Objectworld* and *Binaryworld*, consist of a map of $M \times M$ states for $M = 32$, where possible actions include motions in all four directions as well as staying in place. In addition, two different sets of feature representations are used to evaluate the algorithms for discrete and continuous representations. For the continuous features $x \in \mathbb{R}^C$. each feature dimension describes the minimum distance to an object of one of C colours. Building on the continuous representation the discrete ones includes $C \times M$ binary features, where each dimension indicates whether an object of a given colour is closer than the threshold $d \in \{1, ..., M\}$. The ground truth reward in *Objectworld* depends on a two-step decision tree based on the value range of the available features, while *Binaryworld* is built on a more complex model which is based on counting the number of active features in the direct environment of a state. A more detailed description of the scenarios and ground truth reward calculation can be found in Appendix A.

The algorithms are assessed with increasing number of demonstration samples on the training and test scenarios. While the original MaxEnt is inherently unable to capture the nonlinear reward structure independent of the number of demonstration samples, both MEDIRL and GPIRL converge towards exact approximation as represented in Figure 3.4. NPB-FIRL was shown to perform on a similar level as

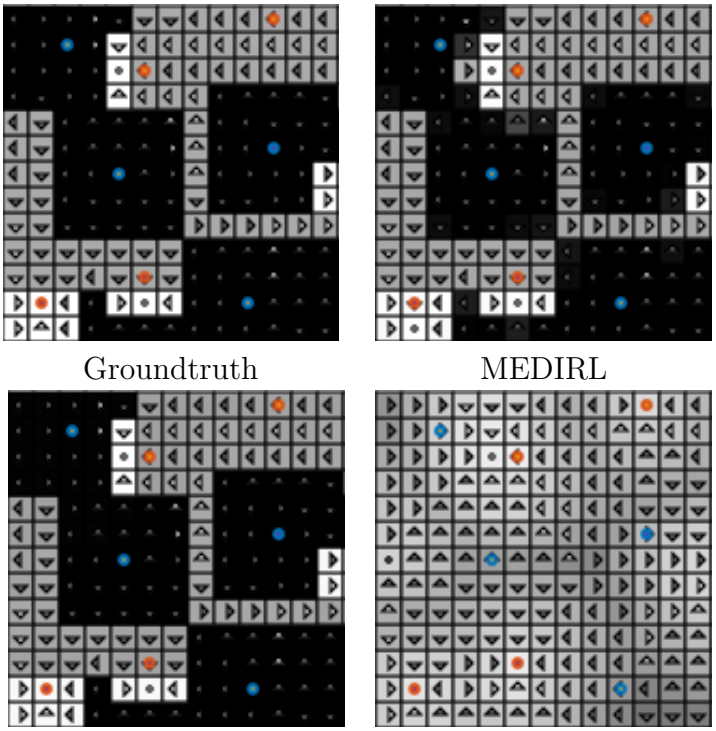


Figure 3.4: Reward reconstruction sample in the *Objectworld* benchmark provided $N = 64$ examples and $C = 2$ colours with continuous features. White - high reward; black - low reward. The arrows represent the optimal policy for the estimated reward function.

GPIRL by Choi and Kim [2013]. MEDIRL achieves performance commensurate to GPIRL with slight increase in the number of available expert demonstrations. The same behaviour is exhibited for both continuous and discrete feature representations (Fig. 3.5).

The *Binaryworld* scenario differs to *Objectworld* in its feature representation as every state is randomly assigned one of two colours and the feature vector for each state consequently consists of a binary vector of length 9, encoding the colour of each cell in its 3×3 neighbourhood. Since the reward depends on a more complex combination of the basic features - that is to say the number of specific features - this case is applied to represent a more complex nonlinear function.

The performance of MEDIRL compared to GPIRL, linear MaxEnt and NPB-

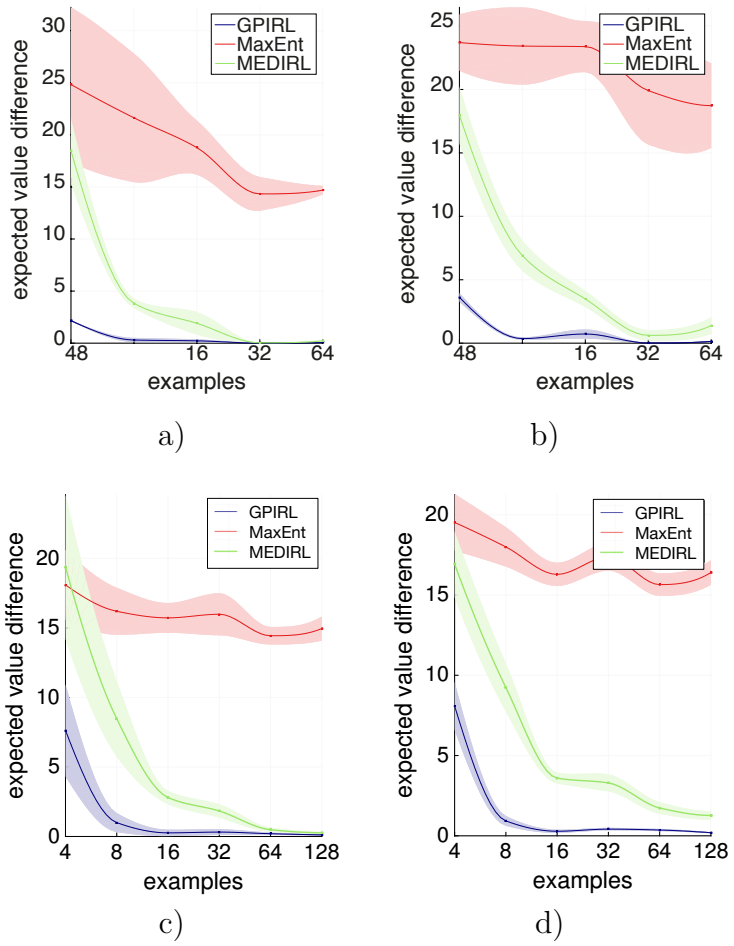


Figure 3.5: *Objectworld* benchmark. From top left to bottom right: expected value difference (EVD) with $C = 2$ colours and varying number of demonstrations N for training a) and transfer case b) with continuous and respectively with discrete features in c) & d) ; As the number of demonstrations grows MEDIRL is able to quickly match performance of GPIRL on the task.

FIRL is depicted in Fig. 3.6. In this scenario, MEDIRL is able to learn the higher-order dependencies between features more quickly, whereas GPIRL needs a higher number of samples as the inherent kernel measure is unsuited for relating the reward of different examples with similarity in their state features. The NPB-FIRL algorithm struggles significantly in this test, which can be explained by the fact that describing the true reward in this scenario with the logical conjunctions is quite inefficient. In fact, it would require 2^9 different logical conjunctions, each capturing all possible combinations of features, to accurately model the reward in this framework.

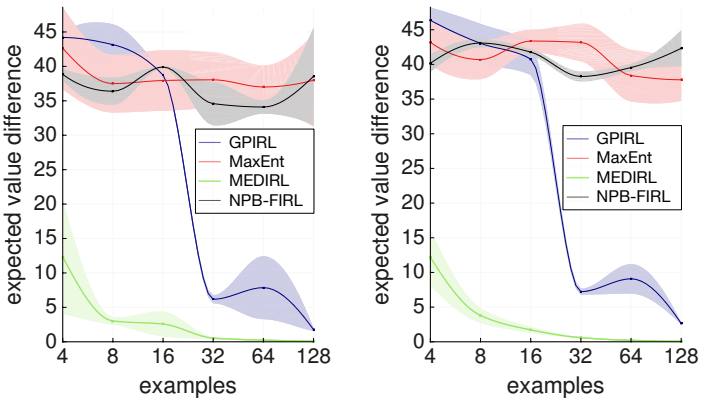


Figure 3.6: Value differences observed in the *Binaryworld* benchmark for GPIRL, MaxEnt and MEDIRL for the training scenario (left) and the transfer task (right). While MEDIRL accurately approximates the true reward function, both MaxEnt IRL and NPB-FIRL are unable to provide good results, due to missing capacity. GPIRL requires more samples in this scenario.

One of the principal advantages of MEDIRL is its scalability with respect to the amount of training data, leading to an algorithm with constant-time execution at test time. The partial requirement of MEDIRL for more training will be rendered unimportant in some robot applications based on low-cost data acquisition, rendering representation learning and the aforementioned lower algorithmic complexity as the dominant advantage of a parametric approach.

Spatial Feature Learning

One principal advantage of being able to access the full gamut of different neural network architectures is the possibility to extend the model via the use of wider convolution kernels to eliminate the requirement of manual design of input features.

We demonstrate the capability of our method to learn optimal spatial features from the raw input representation. Figure 3.7 represents the results for both toy scenarios, but instead of using the earlier described handcrafted feature representations the input representation for our models only includes the presence of any object at every state. All spatial information has to be derived based on the convolutional filters in this context. We employed a five layer approach with 3×3 convolutional kernels in the first two layers, followed by 1×1 width layers.

By including wider convolutional filters, we add the capacity to the network to learn the hierarchy of features including their spatial interaction as optimised for the task at hand.

Due to the increasing number of parameters, the approach requires additional training data to learn the actual correlations and generalise. When increasing the number of demonstration samples, it converges quickly towards the performance with predefined features and ultimately converges towards the ground truth reward function. In these simplified toy problems the used handcrafted features are optimal and the true reward is directly calculated on their basis. Therefore, automatically learned features cannot exceed their performance. However, in real-world scenarios, the compression of raw data - such as images - to feature representations inherently leads to loss of information and the learning of features that optimise the utilisation of capacity with respect to the task represents a principal gain of the approach.

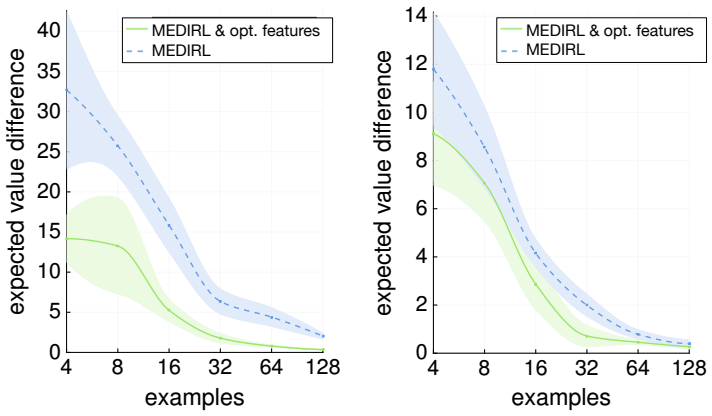


Figure 3.7: Application of convolutional layers for spatial feature learning. Spatial feature learning (MEDIRL) quickly converges to performance with optimally designed features (MEDIRL & opt. features). Additionally to only access to less informative features, the approach has to handle higher dimensional input that the baseline.

Our test scenarios demonstrate the feasibility of the approach on small scale benchmarks and spatial feature learning with convolutional neural networks but only represent a limited evaluation. The cost functions in this context are based on handcrafted decision trees with algorithmically generated demonstration behaviours. In the following section, our method is applied in a real world scenario to

learn cost functions for motion planning. We utilise the application for autonomous vehicles in an urban environment in this context as a general example for large scale deployment of robotic platforms.

3.2.2 Large Scale Application

After demonstrating the capability of our model in simple, commonly used benchmarks, this section aims at evaluating its real-world applicability: as regression model for cost functions used in motion planning.

Dataset

The driving dataset was collected over the course of one year involving 13 different drivers navigating pedestrian walkways and cycle lanes in the city of Milton Keynes, UK.

The data collection platform, a modified GEM (Global Electric Motorcars) golf cart (cf. Figure 3.8), is equipped with various sensors including 2D as well as 3D LIDARs and stereo- and mono- cameras. The relevant sensors for this work are two Velodyne HDL-32E scanners and a Bumblebee XB3 stereo-camera.

A total of over 25,000 trajectories are included for the dataset, each about 12-15m long, from more than 120km of urban driving. Each trajectory represents a single training input sample based on a ground-plane projection of statistics of the LIDAR data as represented in Figure 3.9. The trajectories were extracted with about 50-60% overlap which was empirically determined to be a good compromise between final model performance and training time. When increasing the overlap even further, we obtain more samples. However, model performance does not increase when employing these samples and we hypothesise that the model's ability to extract information does not increase with even more trajectory overlap as the overall driven distance stays the same and trajectory sections are used more often.

The network input is based on pointclouds measured by the two Velodyne 3D LIDARs. To enable the application of 2D convolutional layers, we require a 2D gridded representation. The pointclouds are first mapped into a grid-based map on

Figure 3.8: Mobile research platform – a modified GEM golf cart. For the experiments presented in this chapter, only the 2 Velodyne LIDAR scanners and the Bumblebee stereo camera are of important. The exact calibration between all parts is provided.

the ground plane of the vehicle. In addition, we extract statistics for each cell, which include mean height, height variance and a binary indicator if the cell is visible in any scan. The grid has a size of 25m x 25m and a resolution of 0.25m per cell.

An example of the gathered demonstration trajectories is depicted in Figure 3.9. The trajectories are extracted from the chain of vehicle poses estimated by applying visual odometry [Churchill and Newman, 2012] on stereo images from the Bumblebee XB3 camera. The extracted transform chain is mapped into the static map frame and discretised to fit into the grid representation. The action space is simplified to a discrete set of motions around the current position. Only spatial aspects of the trajectories are considered for this model. An expansion towards temporal aspects is discussed in Section 3.3.

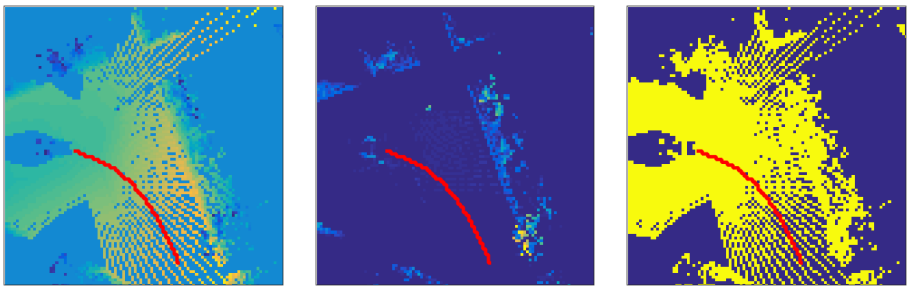


Figure 3.9: Visualisation of demonstration trajectories (in red) on all channels of a static map. From left to right: mean height, height variance, cell visibility.

In the following sections, we evaluate the different models against a handcrafted cost function (displayed in Figure 3.10). The main concept behind the design of this cost function is the detection of obstacles as areas of large height variance in each cell. Areas without LIDAR information are labelled as unknown, which in the classification task is treated as not traversable. Furthermore, obstacles are extended by the radius of the smallest circle encapsulating the GEM platform (Minkowski sum [LaValle, 2006]).

Prediction and Classification

Model quality is evaluated based on the performance in two tasks, prediction of human behaviour and classification of traversable terrain. Taking into account that one goal of this chapter lies in the use of MEDIRL to generate cost maps for motion planning, the evaluation focuses on qualitatively and quantitatively determining the accuracy of terrain assessment.

For prediction accuracy, we apply two common metrics: the Negative Log-Likelihood of the demonstration data (NLL) as well as the Modified Hausdorff Distance (MHD) [Kitani et al., 2012]. The first metric represents the probability of expert demonstrations given the current cost function and the latter is a spatial metric for how close the demonstrations are to samples generated from the stochastic policy as a function of the cost map. In other words, it evaluates the similarity between agent and demonstration trajectories - the accuracy in imitating human driving behaviour.

The standard FCN describes human driving behaviour more accurately than the manual cost function (see Table 3.1). This result is expected as the training objective in Equation 3.3 includes the optimisation the NLL of demonstration trajectories. The use of the pooling architecture without parallel information branches performs better in terms of a smooth cost function when interpreting the same areas as displayed in Figure 3.10. The MaxPooling approach is able to infer traversable terrain. However, the pooling step leads to a loss of spatial information as discussed in Section 3.1.4. Best performance is achieved when the

model is given the capability to learn spatial invariance in one branch while being able to preserve feature location in a parallel chain of the MS-FCN architecture.

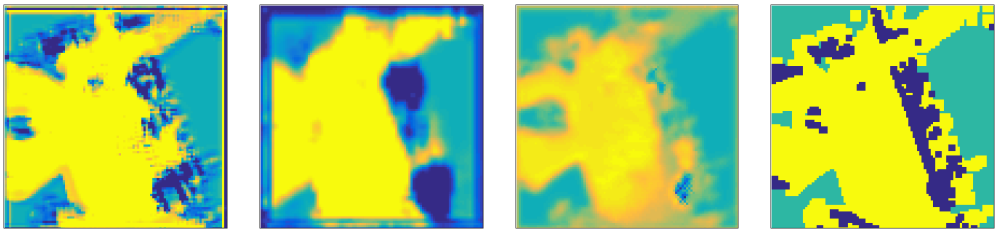


Figure 3.10: Visualisation of the learned cost maps based on (a) Standard FCN, (b) Pooling FCN, and (c) MS FCN. Colour scheme: from yellow to blue increasing cost. (d) Handcrafted Cost Function. Colour scheme: yellow - traversable; cyan - unknown; blue - obstacle

One major drawback in the evaluation of all approaches in a real world driving setup is the absence of absolute ground truth for the cost map. However, the driven trajectories represent ground truth knowledge about traversable terrain and the annotation of synthetic collision trajectories is less time consuming than full manual annotation of cost maps. For this purpose, we annotate synthetic collision trajectories of the same number as positive samples in our classification test dataset based on pointcloud and image data, as actual collision trajectories must be prevented. We can overcome the impediment of no ground truth for the cost function and evaluate our methods as classifiers for feasible trajectories. In the following evaluation, we treat collision trajectories that are classified as feasible false positives and feasible trajectories that are classified as obstacles as false negatives.

False Positives Rate (FPR) and False Negatives Rate (FNR) are therefore computed via Equations 3.12 and 3.13. Let the false positives and false negatives respectively be represented by FP and FN and the true positives and true negatives by TP and TN.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{3.12}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \tag{3.13}$$

In the classification setup, we need to set a threshold for deciding the minimal cost at which collisions occur and the trajectory can be classified as untraversable. This threshold is set to reduce FPR to 0%, meaning that no untraversable terrain in our validation set is incorrectly classified as acceptable trajectory and planning safety is maximised.

Metric	NLL	MHD	FNR	FPR
Manual CF	78.13	0.284	0.441	0.000
Standard FCN	69.35	0.221	0.471	0.000
Pooling FCN	69.73	0.230	1.000	0.000
MS FCN	**65.39**	**0.200**	**0.206**	0.000

Table 3.1: Model performance on calibrated data.

The manual cost function has no additional threshold parameter but is designed instead to be strictly conservative with respect to the traversability of terrain. This results in 0% false positive rate (FPR) in our validation set while showing a significant number of false negatives. While the system is tuned to produce no classification that could result in a collision, it often wrongfully rejects paths as not feasible.

Adapting the cost function thresholds to achieve 0% FPR requires elaborate tuning. It is straightforward to achieve rates of under 5%, though the final adjustment to erase false positives significantly increases the false negative rate. While it leads to 47% FNR for the standard architecture, it renders the application of the pooling model infeasible as FNR of 100% represents that none of the paths are classified as traversable. The MS FCN - being able to combine the benefits of spatially variant and invariant features (see Section 3.1.4) - performs best with a reduction in FNR of about 50% in comparison to the manual cost map. The results demonstrate that we are able to reduce the number of falsely as intraversable classified trajectories by a factor of two compared to the manual cost function when applying MEDIRL instead.

Robustness to Systematic Noise

A principal gain in learning the cost function from raw features is the significant increase in robustness towards systematic flaws in the configuration of the robot. An inaccurate calibration for example will lead to complete failure for a manually crafted cost function. As presented in the example in Figure 3.11, it can result in artificial obstacles due to imprecise calibration of the pitch angle between the platform and one of the LIDARs of as little as 1°.

Due to the introduced perturbation in pitch angle of the right Velodyne, the manually defined cost map creates obstacles in this instance. It is caused by the increased height variance of points in a specific cell which now directly depends on a cell's distance from the vehicle.

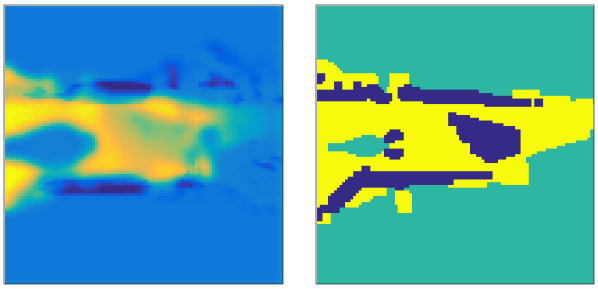


Figure 3.11: Example cost maps based on miscalibrated data with MS FCN (left), and the handcrafted cost function (right). The learned cost function is better equipped to handle miscalibration and less prone to the creation of false obstacles.

The learned cost map is able to infer realistic cost maps even in the presence of miscalibration. As long as the input representation is rich enough to distinguish between the new features - describing artificial obstacles in the manual cost map - and real ones, the system learns to approximate the decision boundary that separates traversable from untraversable terrain.

By applying the metrics introduced in Section 3.2.2, we evaluate the performance of the MS FCN architecture versus the manually defined cost function. The evaluation shown in Table 3.2 emphasises that the handcrafted cost function leads to nearly impassable cost maps in this scenario with an FNR of more than 97%. The learned model, on the other hand, results in an FNR that, though compares worse

than the case with the correct calibration (cf. Table 3.1), still enables basic mobility in most cases by correctly classifying broadly half of all feasible trajectories.

Metric	NLL	MHD	FNR	FPR
Manual CF	89.40	0.432	0.971	0.000
MS FCN	**69.35**	**0.267**	**0.525**	0.000

Table 3.2: Model performance on miscalibrated data. By applying the MS FCN architecture we gain significantly more robustness towards systematic bias in the sensor calibration in comparison to a handcrafted cost function.

3.2.3 Addressing Real World Challenges

When transferring the MEDIRL approach from toy scenarios to full-scale, urban driving scenarios, additional challenges emerge. The iterative nature of planning and refining the cost model in the training process, as described in Section 3.1.3, leads to a principal focus on learning discriminative features around highly visited states as depicted in Figure 3.12. Features describing terrains that are neither explored by demonstration samples nor the planning step will not be formed. The model has to generalise to describe these states based on similarity to more commonly traversed areas, resulting in inaccuracies, artefacts, blur, and noisy reward maps.

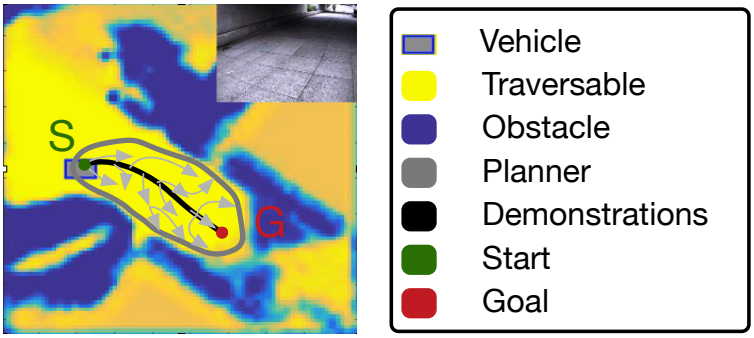


Figure 3.12: Illustration of sparse feedback, showing a demonstration trajectory on the spatial cost map around the vehicle, as well as the region explored by the planning algorithm. Error feedback is only created for the area surrounding sample trajectories.

While this does not represent any limitations for the toy scenarios, it is of concern where feature representations for similar places differ based on their position, as is the case in the dataset used in this work. The LIDAR scan points in this setup will be spread more sparsely at greater distance from the car, resulting in spatially variant representations for the same objects in our environment. This behaviour is illustrated in the cost maps in Figure 3.10 where areas with sparser LIDAR which differ more significantly from the traversed areas display more grainy and noisy cost representations.

To illustrate the problem further, we consider the following example. When training a model for image segmentation, the objective creates feedback for each individual pixel. When training with a loss based on inverse reinforcement learning on the other hand, error terms will focus on the region around the demonstration trajectories. These error terms are based on states visited by the demonstration samples and the planning algorithm, which inherently focuses around sample data, resulting in accurate cost maps only for commonly traversed features close to the vehicle. Our work addresses these shortcomings by pretraining the network towards a dense human-provided prior, as visualised in Figure 3.13, to learn richer feature representations for untraversed areas and increase the network's ability to generalise.

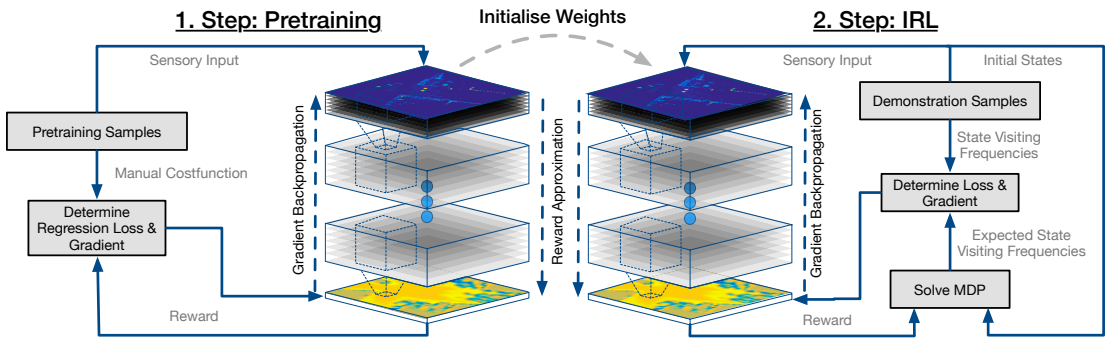


Figure 3.13: Schema for additional network pretraining, where the model learns to regress to a manual prior cost map. Subsequently the network is fine-tuned to predict the reward under the MEDIRL framework.

These cost maps can be automatically generated from the laser input data based on manually designed rules, which enables us to utilise the availability of

large amounts of data without human labelling efforts. The principal benefit of the additional, spatially dense pretraining step is the ability to explore features relevant to generating the cost function independent of position with respect to the vehicle. This leads to better generalisation in areas with greater distance from the demonstration trajectories.

To increase the accuracy of our approach for generating planning cost maps, we add more common preprocessing steps to the dataset from Section 3.2.2. This includes normalising the input data and training on shorter trajectories that are more representative for the motion primitives employed in the final motion planning framework of closer to 10 meters. For the normalisation procedure we center each feature with zero mean and unit standard deviation. When performing path planning, a threshold has to be determined to define untraversable and unsafe terrain. This step is simplified by normalising the output to range [0,1] by applying a final sigmoid activation function. Furthermore, we train with reduced learning rates and introduce early stopping based on validation performance to increase generalisation. These changes in the training procedure aim at preventing overfitting as well as the overriding of prior information from the weight initialisation. For the procedure we randomly separate 5% of our training data as a validation set.

Metric	NLL	MHD
Manual CF	56.402	0.286
w/o Pretraining	47.535	0.218
w Pretraining	**46.767**	**0.182**

Table 3.3: Evaluation of cost functions for urban driving under the negative log-likelihood (NLL) and Modified Hausdorff Distance (MHD) metrics. Lower numbers represent models that are approximating human behaviour with higher precision.

Utilising prior knowledge in this approach improves accuracy for prediction and therefore imitation of human driving behaviour as displayed in Table 3.3. However, its principal gain lies in being able to improve the robustness and spatial generalisation of the learned cost functions for motion planning, leading to safer driving.

To extend our evaluation from of one specific threshold in Section 3.2.2, we vary the traversability thresholds for our models between 0 and 1 and, aligned with the previous sections, record true and false - positives and negatives. Based on these results and Equations 3.14 to 3.16, we can determine the precision & recall as well as the true & false positive rate metrics. Precision describes the ratio of how often the classifier is correct for positive predictions and recall describes how often it is correct for ground truth positive labels.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.14}$$

$$\text{recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.15}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.16}$$

The Precision-Recall curves for networks with and without pretraining are depicted in Figure 3.14. The manual function is represented by a single point as it does not include an adaptable threshold parameter. The approach is manually designed to be conservative and enables us to operate at very good precision. However, it falsely rejects a significant portion of the terrain as described in Section 3.2.2. In contrast, the learned cost functions find possible paths in many situations where the manual cost function will get stuck. The additional introduction of prior knowledge into the training process achieves a significant gain in precision compared to random initialisation. Hence, utilising this knowledge is an important step towards robust application of the learned cost maps.

When explicitly handcrafting a cost function, cornercases, can represent relevant shortcomings, as displayed in Table 3.4. The manual cost function as described in Section 3.2.2 can lead to inaccuracies in the presence of slopes, which can exceed the threshold and will be shown as untraversable. Stairs on the other hand can still fit within the same threshold, but present obstacles for any wheeled vehicle as they cannot be traversed due to their indiscontinuity. Furthermore, underpasses

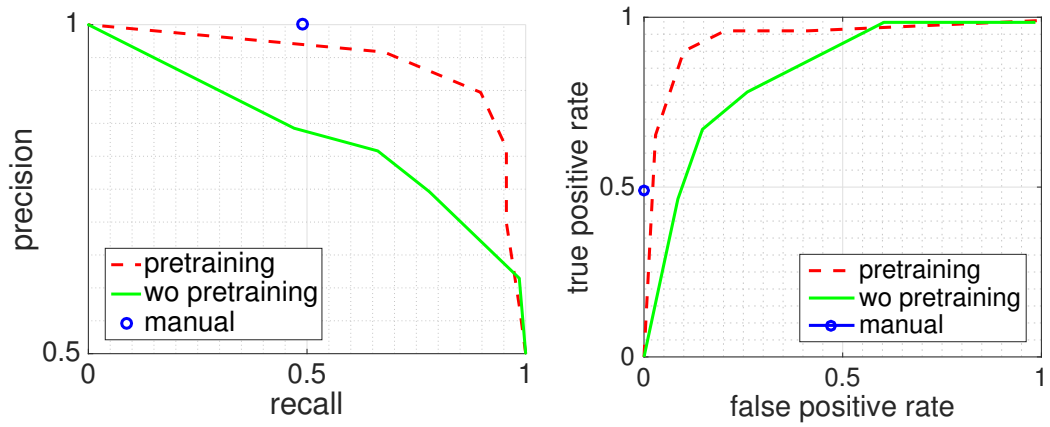


Figure 3.14: Precision Recall (PR) curves and Receiver Operating Characteristic (ROC) curves for trajectory classification. The manual cost function has high precision but low recall, meaning that it is safe but conservative and will falsely classify a significant number of feasible trajectories as untraversable. Applying human priors in the pretraining step enables a significant gain in precision towards our baseline. This method approaches the precision of the manual cost function while strongly exceeding it in recall. The ROC curves provide a different representation to support these insights.

might incorrectly exceed the height threshold for obstacles since scans from ceiling and floor result in a high height range. Bollards that are extended slightly too far will seem untraversable and areas such as grass can look very similar in features to pathways but should not be traversed.

The randomly initialised network already learns to represent the main obstacles and traversable areas, but it results in noisy areas and artificial obstacles. When initialised based on prior domain knowledge, the network additionally learns to refine the representation and is significantly more robust. It is optimised to represent distinct obstacle boundaries and displays fewer artefacts. This approach is able to distinguish slopes from stairs and extends obstacle boundaries more precisely as necessary for safe traversal as seen in the respective cases in Table 3.4.

As additional safety benefit, we can define how far we trust our perception systems by adapting the length of demonstration trajectories. The learned cost functions show untraversable terrain starting at about 13 meters distance from the vehicle position, which is the length of the demonstration trajectories. Since features in distant areas are only traversed by the planning step and not demonstration samples in the training process, they will be classified as untraversable

with high probability.

We argue that without pretraining, the expressive power of the network is employed to learn the very specific representation focused around the demonstration trajectories as this strongly influences the training objective. When we instead pretrain the model to predict an existing cost map and learn with more dense feedback for the whole environment, the remainder of the training process is able to better capture some of the corner cases described previously. The approach results in more distinct obstacle boundaries and more robust cost maps as demonstrated in this section.

3.3 Discussion

We argue for the use of a high-capacity, parametric approach to IRL using FCNs, in order to tractably approximate the cost function in a complex real world scenario. Additionally to high accuracy, the approach computes after training at a runtime independent of the size of the demonstration dataset.

Manually handcrafted cost functions for this task would require significant expert knowledge of the application domain, sensory pipeline and decision making approach. Further, as demonstrated in Section 3.2.2, some necessary features can be easily missed in the preprocessing setup given the amount of variation in domain specific obstacles. In contrast, the spatial features as represented by the filters of the FCN on the other hand are inherently optimised for this task.

While non-parametric approaches such as GPIRL possess the capacity to learn complex nonlinear cost functions when given these hypothetically perfect features, but do not exhibit constant-time behaviour and are rendered intractable with the large amounts of required training data.

Additional benefits of the proposed approach include robustness towards unknown, systematic noise, as exemplified by miscalibration in Section 3.2.2. This capability does not exist for approaches that rely on predetermined feature representations, because this would require knowledge of the applied noise when constructing features for the task.

While our model learns the spatial preferences underlying human driving behaviour, the approach - in its current form - does not address the velocity profiles alongside driven paths as well as aspects of temporal consistency between consecutive cost-maps. This extension requires the processing of sensor inputs across the extend trajectory instead of focusing on a limited period around the start of each trajectory, since the position of dynamic obstacles will change during the traversal of a given trajectory. Both challenges can be partially addressed on the model side with extension towards recurrent network models as well as temporal convolutions or as separate post-processing step. Large-scale data collection efforts as currently predominant in the industrial sector can build the foundation for training these more complex, high-capacity models to expose the factors that determine temporal aspects of taken trajectories.

3.4 Conclusions

We develop a highly scalable approach to learning cost functions for mobile platforms from large collections of demonstration samples for human driving. Common simulation benchmarks are employed to prove the efficacy of the algorithm and confirm that it can approximate the given ground truth reward functions, with convergence commensurate to current state-of-the-art IRL approaches. We demonstrate the principal benefits of Maximum Entropy Deep Inverse Reinforcement Learning by applying it to an urban driving scenario based on over 120 km of driving with over 25k demonstration samples. The scale and complexity exceeds the reach of existing approaches, either due to computational intractability (such as for GPIRL) or limited representative capacity (such as for linear IRL).

By employing an end-to-end method that maps directly from low level features to cost maps, the approach is able to correct for systematic biases and learn task-optimal features which help our method to perform well even in presence of systematic noise as demonstrated in Section 3.2.2 by applying it to a miscalibrated dataset.

Additional challenges emerging from the application on real world sensor data in the large scale evaluation such as spatially sparse feedback are furthermore

addressed by incorporating human priors into a network pretraining step. This allows the MEDIRL framework to perform parameter fine-tuning based on an educated human prior cost map.

Additional benefits might be found in a semi-supervised training procedure, exploiting both reconstruction and IRL based cost functions for the final training procedure as well as utilising additional input representation samples in absence of driven trajectories. The approach can help to address overfitting and increase generalisation performance.

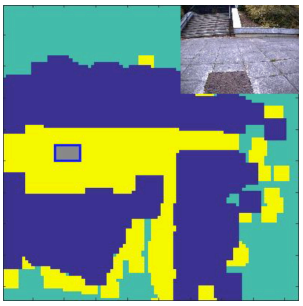
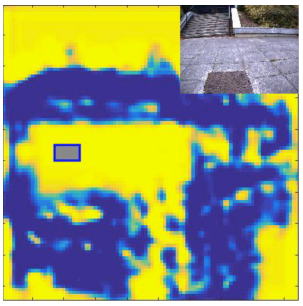
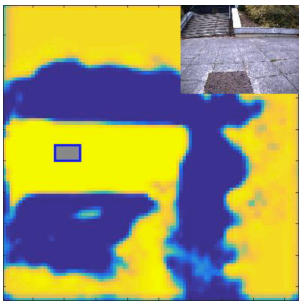
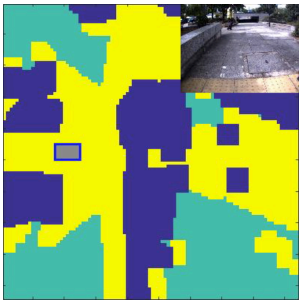
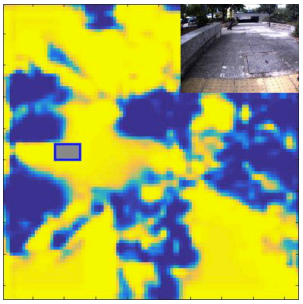
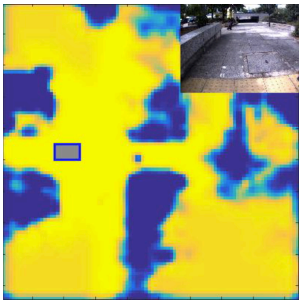
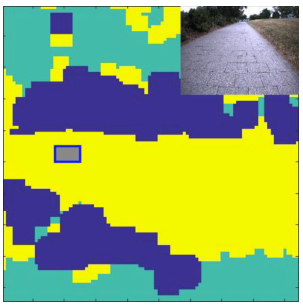
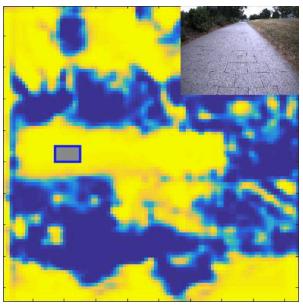
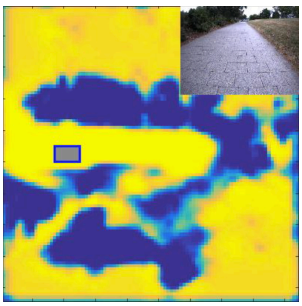
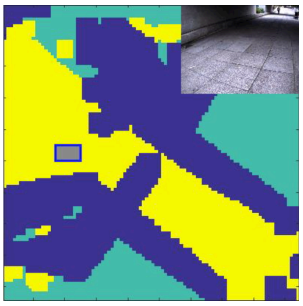
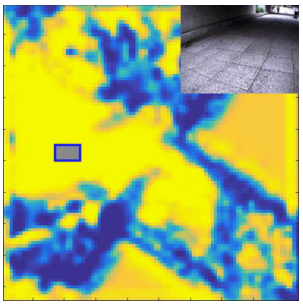
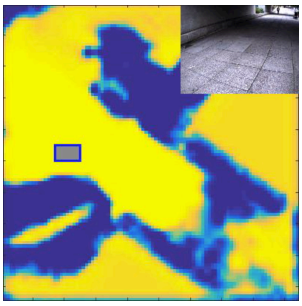
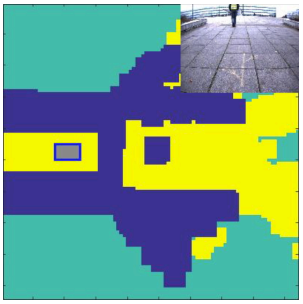
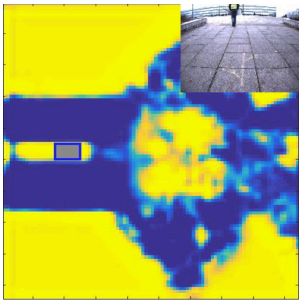
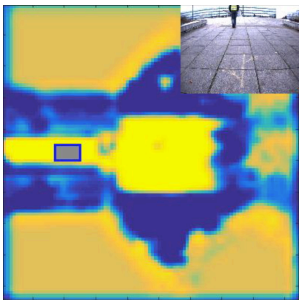
Scenario	Manual CF	w/o Pretraining	w Pretraining
Stairs			
Bollards			
Grass			
Underpass			
Slope			

Table 3.4: Corner Cases for the Cost Function. The images include views from the front facing camera module with the vehicle represented as a grey rectangle driving towards the right side of each cost map. Obstacles are represented in blue, while yellow depicts traversable terrain.

4

Unsupervised Domain Adaptation for Outdoor Appearance Changes

Contents

Some types of supervision and labels can be generated at comparably low cost, as displayed in the previous chapter in the context of learning from demonstration. However, often it is expensive as well as time-consuming to generate manual annotations for different tasks. In particular, it affects tasks which require multiple labels

for each sensor input, with an especially strong case given by image segmentation where annotation efforts are especially extensive. In the following chapter, we investigate and expand methods from the field of domain adaptation to reduce the effort of collecting labels. This work aims at optimising models to increase their performance in domains in which we only have access to unsupervised data, transferring knowledge from similar domains.

Appearance change under the influence of numerous factors including time of day, weather, and seasonal variation has been a long-standing challenge for outdoor robotics. Dealing with these changes becomes relevant in all modules of the robotic perception system including localisation [Paton et al., 2015], mapping [Churchill and Newman, 2013] and obstacle detection [Maddern et al., 2014]. Given the recent adoption of high-capacity deep neural networks for many robotic vision tasks, the effects of condition variation have been moderately alleviated with the assumption that we can collect large datasets from all environments in which we intend to deploy our models. However, due to the expense and impracticality of collecting labels across all environmental conditions, our training data is often captured only in a subset of domains.

This chapter addresses the problem of appearance change by reframing it in the context of unsupervised domain adaptation. Methods for domain adaptation have already found success in robotics for transfer from 3D models to laser data [Lai and Fox, 2009] or from simulation to real images collected indoors [Tzeng et al., 2015a]. Here we focus on developing a general and flexible framework for adapting supervised machine learning models to address appearance change for outdoor robotics problems (as depicted in Figure 4.1).

Traditional approaches for addressing the general domain adaptation problem have focused on modelling the density of the source and target distributions separately [Daume III and Marcu, 2006] or with imposed prior structure [Sugiyama et al., 2008]. However, density estimation itself is a challenging issue, particularly for appearance, as it is difficult to impose a prior without making assumptions on the distribution. Recently, generative adversarial networks (GAN) [Goodfellow

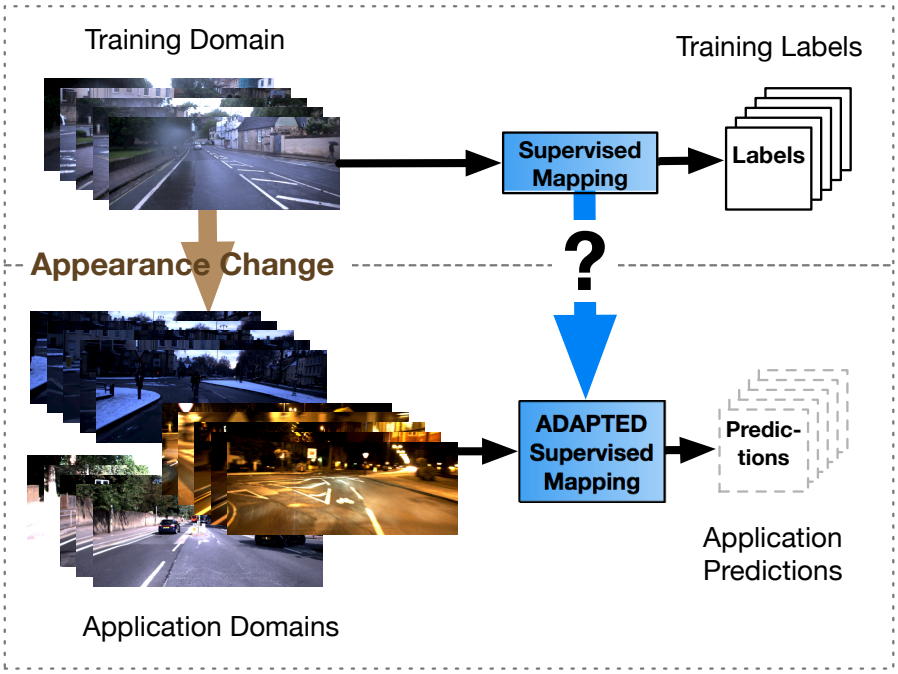


Figure 4.1: Appearance change between training and test time of machine learning systems. Adversarial domain adaptation (ADA) provides a systematic way to adapt the supervised task to perform in the application domains without supervised label information by utilising additional objectives to align feature distributions between source and target domain.

et al., 2014] were proposed as a framework to model any arbitrary distribution where a generating network is optimised to produce data indistinguishable from real data as considered by a discriminating network. Building on the flexibility of GANs, adversarial domain adaptation (ADA) has demonstrated astonishing performance for unsupervised domain adaption [Ganin et al., 2016, Shrivastava et al., 2017, Bousmalis et al., 2016].

Extending existing work [Ganin et al., 2016, Bousmalis et al., 2016], we demonstrate a straightforward framework for the adaptation of existing, commonly-used network architectures that allows to benefit from recent progress in deep learning and crystallise the principal factors of influence on target domain performance. To address real-world applicability of ADA with this schema, we modify two popular network architectures, AlexNet [Krizhevsky et al., 2012] and FCN-VGG16 [Long et al., 2015] for the tasks of classification and pixel-wise image segmentation respec-

tively. Both of these architectures have demonstrated high performance on their respective task. When considered in an outdoor context, both tasks are affected by change in appearance caused by factors such as time of day or weather conditions.

Due to the documented evidence of training instability of adversarial training procedures [Goodfellow et al., 2014, Goodfellow, 2017, Arjovsky and Bottou, 2017], we first use place classification as a surrogate task to perform an extensive evaluation of approaches for balancing our objectives and stabilising the training process. The reduction in complexity enables us to enumerate and explore various potential configurations of the network structure and training procedures to identify the key factors of influence before scaling up to the more challenging segmentation task. Particularly, we seek to answer the following questions:

1. Trade-offs: How do we stabilise training and balance supervised and adversarial objectives?

2. Architecture: How can we extend existing high-performing network architectures?

3. Performance: Where does ADA work best and how do larger appearance changes influence performance?

A series of experiments is presented in Section 4.2 to provide insights around the above questions, empirically demonstrating the relevance of robust loss functions, pretraining, supervised warmup, and details about depths of different modules. In the subsequent chapter, we extend the method in order to address full image pixel-wise traversability segmentation commonly used as an input to motion planning systems. In the final segmentation task we see a significant performance increase in the target domain by applying ADA over the FCN-VGG16 [Long et al., 2015] baseline. Finally, we also demonstrate the method's suitability as a regulariser by observing performance gains within the original domain as more unlabelled data becomes available from similar domains.

4.1 Adversarial Domain Adaptation

The principal goal of ADA is to maximise the performance of a supervised task not only in the source domain - where labels are available - but also in the unlabelled target domain. Given domain shift between training and application domains, the approach tries to align the marginal feature distributions independent of labels for both domains.

ADA can build upon arbitrary architectures known to perform well for the supervised task at hand. The existing network architecture is divided into two functional blocks referred to as *encoder* and *supervised module* as represented in Figure 4.2. Let $E : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be the encoder that transforms an input image i of dimensionality $n = width * height$ into the feature representation f. The output of the encoder subsequently serves as input for both the supervised task $S : \mathbb{R}^m \rightarrow \mathbb{R}^c$ producing label l and the adversarial domain discriminator $D : \mathbb{R}^m \rightarrow \mathbb{R}$ computing domain label d. The dimensionality c of the label l depends on the supervised task. To reduce the memory footprint of the model, we can remove the discriminator at test time as it is only an auxiliary module to determine the encoder objective.

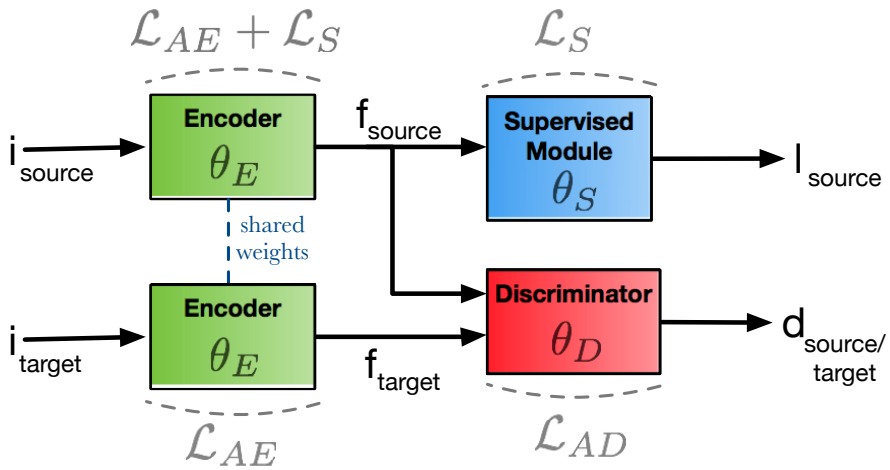


Figure 4.2: Schema for adversarial domain adaptation. Given a well-established network architecture, we determine a split layer and pass the activations of that layer additionally to a discriminator module. While the supervised module is only applied to source data, both domains share the same encoder. By training the encoder to confuse the domain discriminator, the training procedure aligns feature distributions between both domains.

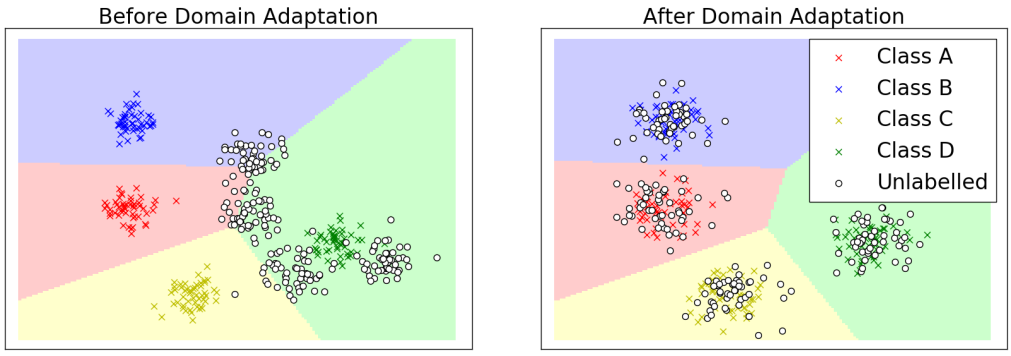


Figure 4.3: An illustration of domain shift where the distribution of the unlabelled target domain is misaligned with the labelled source domain but both distributions share structural similarities. In the context of appearance change, common structure can be provided by the similarity of places and objects underlying our observations. By aligning the two domains the decision boundary of the supervised task (denoted in colour) better aligns across both domains.

The encoder and supervised modules are optimised for the main task. Additionally, to adapt the model towards performing well in the target domain, we train the encoder to confuse the discriminator which tries to estimate the underlying domain of the feature representation for input data samples. The discriminator serves as a density ratio estimator between source and target domain representations. By training the encoder to confuse the discriminator, the module is trained to align the densities for both domains.

In other words, the adversarial training process aligns the marginal feature distributions of source and target domains. With both domains sharing partial structural similarity, this also implicitly aligns the conditional distributions given the labels. Consequently, by aligning the distributions, ADA increases the performance of a supervised module with decision boundaries optimised for the source domains as displayed in Figure 4.3.

Let $\mathcal{L}_S$ and $\mathcal{L}_A$ respectively be the supervised objective and adversarial objectives. The adversarial objectives consist of the losses applying to encoder $\mathcal{L}_{AE}$ and discriminator $\mathcal{L}_{AD}$ (see Equations 4.1 and 4.3). The supervised objective for classification is minimising the cross-entropy loss from Equation 4.2. Let encoder parameters and discriminator parameters be described by θ_E and θ_D

respectively. While the discriminator is only trained with respect to the adversarial objective in Equation 4.4, the encoder is optimised for adversarial and supervised tasks. The parameters of the supervised module θ_S are optimised with respect to the supervised loss on data only from the source domain. Let λ denote the hyperparameter for describing the relative strength of supervised and adversarial objectives. The following hyperparameter search in Section 4.2.1 aims at optimising λ to encourage the encoder to extract features that balance the relevance of the supervised task and the maximisation of domain invariance.

$$\begin{aligned}
\mathcal{L}(\theta_S, \theta_D, \theta_E) &= \mathcal{L}_S(\theta_S, \theta_E) + \lambda \mathcal{L}_A(\theta_D, \theta_E) && (4.1)\\
\mathcal{L}_S(\theta_S, \theta_E) &= \mathbb{E}_{i\sim S}[-\log(S(E(i, \theta_E), \theta_S))] && (4.2)\\
\mathcal{L}_A(\theta_D, \theta_E) &= \mathcal{L}_{AD}(\theta_D) + \mathcal{L}_{AE}(\theta_E) && (4.3)\\
\mathcal{L}_{AD}(\theta_D) &= \mathbb{E}_{i\sim S}[-\log(D(E(i, \theta_E), \theta_D))] + && (4.4)\\
&\quad \mathbb{E}_{i\sim T}[-\log(1 - D(E(i, \theta_E), \theta_D))]
\end{aligned}$$

As with the original generator loss in the GAN framework [Goodfellow et al., 2014], the encoder in ADA can be trained to maximise the discriminators domain confusion [Tzeng et al., 2015b]:

$$\begin{aligned}
\mathcal{L}_{AE}(\theta_E) &= -\mathbb{E}_{i\sim S}[\log(1 - D(E(i, \theta_E), \theta_D))] && (4.5)\\
&\quad -\mathbb{E}_{i\sim T}[\log(D(E(i, \theta_E), \theta_D))].
\end{aligned}$$

An alternative is the minimax formulation which simply negates the discriminator loss [Bousmalis et al., 2017, Shrivastava et al., 2017]. When applied with gradient reversal [Ganin et al., 2016], the minimax loss can be expressed as:

$$\begin{aligned}
\mathcal{L}_{AE}(\theta_E) &= -\mathcal{L}_{AD} && (4.6)\\
&= \mathbb{E}_{i\sim S}[\log(D(E(i, \theta_E), \theta_D))]\\
&\quad +\mathbb{E}_{i\sim T}[\log(1 - D(E(i, \theta_E), \theta_D))].
\end{aligned}$$

Contrary to the GAN framework, these losses apply to samples from both domains instead of only applying to synthetic samples as the encoder transforms images from both domains. Both the confusion and minimax losses are evaluated in Section 4.2.1.

4.2 Evaluation: Ablation Study

The lack of labelled information in the target domain complicates the alignment between feature distributions from source and target domain by restricting access to the marginal distributions. Furthermore this objective potentially conflicts with the supervised task, leading to the potential loss of relevant information and correlated reduction in performance. Various design choices and hyperparameters enable us to optimise the approach for minimum loss of relevant information and high target performance.

The field of autonomous driving and in particular repeated deployment in the same environment presents an opportunity here as the use of overlapping routes can provide similar feature distributions for both domains simplifying the overall alignment process.

The main questions we aim to address regard the :

- Stabilisation and balancing of objectives
- Adaptation of existing network architectures
- Influence of the type of appearance change

In this context, we focus for our main evaluation on the surrogate task of classification and extend the evaluation subsequently to pixel-wise semantic segmentation with focus on traversable paths for autonomous mobility [Barnes et al., 2017]. The hyperparameter study is built on a subset of labelled data from the publicly available *Oxford RobotCar Dataset* consisting of over 1000 km of driving data with corresponding images, LIDAR, GPS and INS data [Maddern et al., 2017]. Both experiments rely subsets of the same dataset with similar appearance shifts.

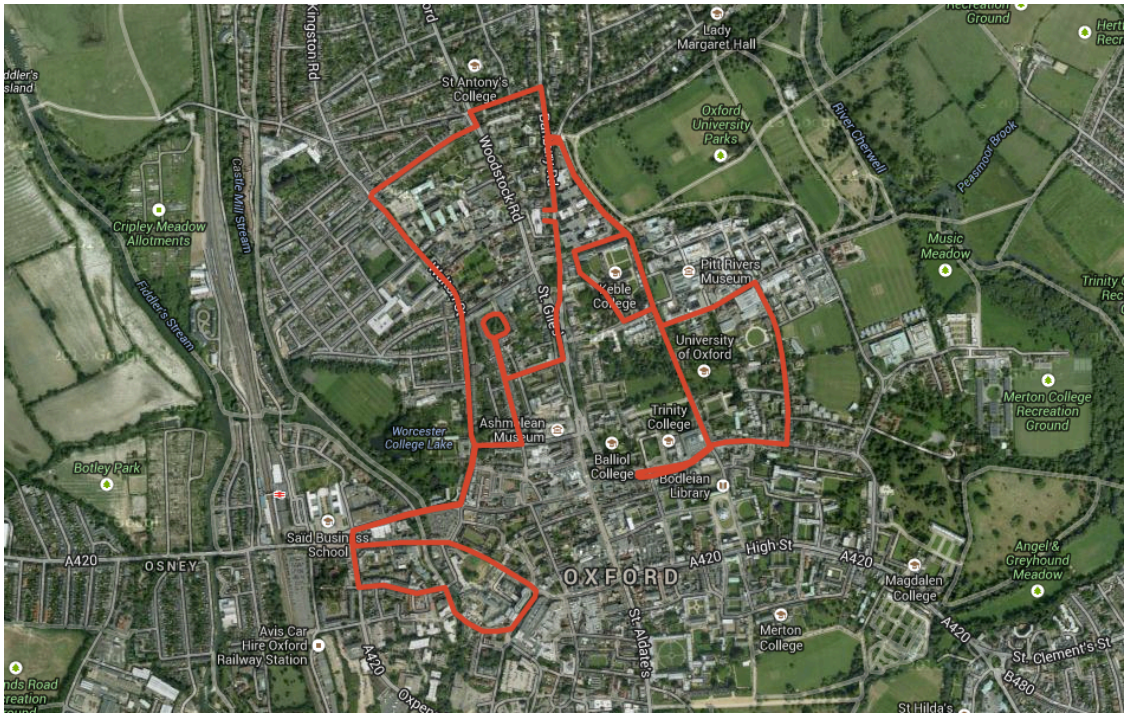


Figure 4.4: Satellite image with Oxford RobotCar Dataset [Maddern et al., 2017] route overlay. The route includes the data utilised in both experimental tasks in this section.

The experiments were performed based on 9,000 training and 1,000 test images for each domain and 20 distinct location classes. We focus on the adaptation between overcast to sunny weather and day to night as displayed in Figures 4.5 and 4.7.

Figure 4.4 presents a satellite image of the Oxford RobotCar Dataset with overlay of driven routes. Both classification and segmentation data for our experiments are homogeneously spread alongside these routes.

The network architecture builds on the popular AlexNet [Krizhevsky et al., 2012] architecture and while we adapt the split between encoder and classifier / discriminator, the overall pipeline from image to location label is standard AlexNet for all experiments, making the approach straightforward to apply to common architectures such as FCN-VGG [Long et al., 2015], which is the focus of our segmentation experiments.

We determine the mean classification accuracy P_T and standard deviation σ for all tested hyperparameter configurations over 5 runs to investigate for GAN-typical

Figure 4.5: Example images for source domain: overcast (left), and target domain: sunny (right). We do not provide aligned data between both domains and the images present different sections along the driven route for both domains.

training instability and random seed sensitivity. All experiments are performed on an NVIDIA GTX TITAN GPU. The final results are reported based on the best found set of hyperparameters for each experiment.

4.2.1 Trade-offs: How do we stabilise training and balance supervised and adversarial objectives?

One of the common challenges for ADA is a potential inability to learn task-relevant and informative features as the domain confusion loss tries to reduce discriminator performance and can result in degenerate feature representations. The following sections evaluate influence of various tools for optimising this trade-off.

Pretraining and Supervised Warm-up

Initialising network architectures via pretraining of convolutional layers on large and diverse datasets is generally known to speed up the learning process as well as leading to better generalisation [Huh et al., 2016]. Furthermore, in the context of ADA, it is helpful to include a warmup phase building only on the supervised task loss before switching on the adversarial loss.

We evaluate the influence of pretrained convolutional layers for the AlexNet architecture based on ImageNet classification task [Russakovsky et al., 2015] (marked as 'Pretraining' in Table Table 4.1). Furthermore, the 'Supervised Warmup' phase

builds on training the encoder and the supervised module over the first 15 epochs only for the supervised objective. The number of epochs was found optimal during earlier hyperparameter search evaluations and has shown to work well in a wider range of evaluation settings.

Table 4.1 displays the classification performance in the target domain for the place recognition task. The results demonstrate that for the given small number of samples, supervised warmup on its own does not suffice for a strong, classifier-relevant feature representation. Therefore we build our following evaluation on pretrained convolutional layers as well as a 15 epochs supervised warmup phase. Both tools enable the further reduction of variance in the results.

	Supervised Warmup	Pretraining	Both
$P_T[\%]$	52.68	72.03	**82.03**
$\sigma_T[\%]$	6.57	4.88	2.09

Table 4.1: Evaluation of performance for network initialisation and warmup phase. The combination of both approaches leads to significant improvement

Balancing Adversarial Objectives

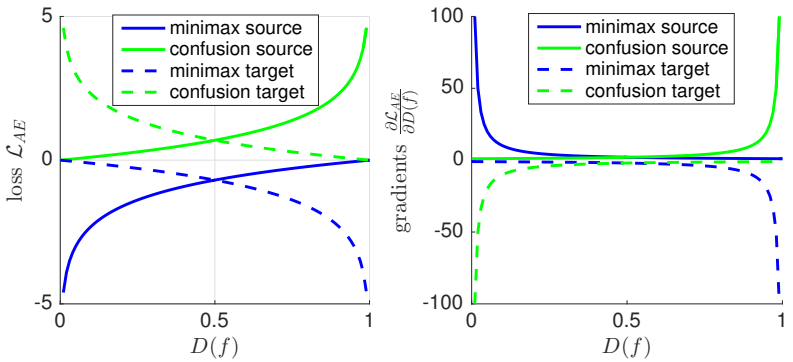


Figure 4.6: Effect of different encoder loss formulations on the gradient behaviour. The confusion loss results in significantly stronger gradients against a well performing discriminator (close to 1 for source data and 0 for target data), therefore stabilising training.

The choice for the encoder loss varies between different works and has been shown to have significant influence on convergence properties and stability of

the training process for GANs [Arjovsky and Bottou, 2017] and ADA [Ganin et al., 2016, Tzeng et al., 2015b, Bousmalis et al., 2016]. The two main generator objectives to consider are the minimax formulation (negation of the discriminator loss) which is equal to the gradient reversal layer [Ganin et al., 2016] and the confusion loss [Tzeng et al., 2015b], which was found to prevent vanishing gradients with saturated discriminator but displays higher variance in the gradients for GANs [Arjovsky and Bottou, 2017]. Additionally, the factor λ is used to vary the relative strength of the adversarial loss in the overall training process to balance both objectives.

Figure 4.6 shows minimax and confusion loss and their respective gradients in dependence of the discriminator performance on source and target domain data. When the discriminator achieves high performance (close to 1 for source data and 0 for target data), the confusion loss results in significantly stronger gradients therefore stabilising the adversarial training process.

Loss	Minimax Loss						
λ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
P_T[%]	59.68	55.22	64.35	73.28	73.66	76.32	68.76
σ_T[%]	2.05	3.40	4.73	3.76	1.89	2.13	3.62
Loss	Confusion Loss						
λ	10^{1}	10^{0}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
P_T[%]	76.04	**82.03**	71.89	71.95	71.28	53.93	37.77
σ_T[%]	1.14	2.09	2.71	2.00	1.64	29.06	29.71

Table 4.2: Evaluation of performance for minimax and confusion loss with variation of loss weighting. Optimal performance is achieved with the confusion loss formulation which does not suffer the problem of vanishing gradients for a saturated discriminator.

Table 4.2 displays that the best performance with the minimax encoder loss builds on low values for the trade-off parameter λ, higher values quickly reduce performance and increasing λ to over 10^-2 resulted in divergence of training with the performance often reduced to chance. We have found the confusion loss more straightforward to apply as it stabilises the game between the adversarial modules

as displayed in Figure 4.6. The original minimax-loss formulation was successfully employed in a number of recent works [Ganin et al., 2016, Bousmalis et al., 2016], however, we have found the tuning process more complex and obtained significantly stronger results based on the confusion loss. The minimax loss additionally introduces higher variance through our our evaluation progress.

4.2.2 Architecture: How can we extend existing high-performing network architectures?

When applying ADA to a new task, we greatly benefit by building on existing network architectures which have been optimised and proven to perform well on similar tasks. This section addresses how to adapt existing architectures easily to incorporate the additional discriminator module.

The expressiveness and flexibility of encoder and discriminator directly influence the performance of our domain adaptation task. While the flexibility of the discriminator limits the types of domain discrepancies that can be detected, the encoder structure affects the efficiency of concealing the originating domain from the discriminator while generating relevant feature distributions for classification.

Choice of Split Layer

The following evaluation focuses in particular on adjustments on the AlexNet [Krizhevsky et al., 2012] architecture, for which we analyse different feature layers as input for the discriminator as well as capacity variations on the discriminator architecture in the following sections.

Following Figure 4.2, the encoder module includes all layers before the split layer and is shared for source and target domain. We duplicate the layers following the split layer for supervised module and discriminator and adapt the final discriminator layer to output a single value per input as domain label. By varying the split layer, we directly influence the capacity ratio between encoder and discriminator while keeping the overall number of layers for the supervised task fixed.

Layer	1	2	3	4	5	6	7
P_T[%]	72.25	74.14	78.93	**82.03**	79.37	79.72	78.63
σ_T[%]	1.75	1.71	2.90	2.09	1.07	0.98	0.84

Table 4.3: Evaluation of performance for the variation of the split layer between encoder and classifier/discriminator. The optimal split lies in the middle of the network. While a split to early in the architecture can lead to low performance, moving the split further towards the last layers only slowly affects performance.

The best possible split results in the middle of the network as both - encoder and discriminator. We hypothesise that balanced capacities of both modules are beneficial for successful adversarial training (see Table 4.3). Earlier split layers reduce the encoder's capacity to minimise mutual information with respect to the domain while encoding classification-relevant features. A later split reduces the expressiveness of the discriminator as density model such that less variations can be detected.

Capacity of Discriminator

While the choice of split layer as displayed above has strong influence on the effectiveness of domain adaptation, the approach might benefit from separately adapting the discriminator capacity to improve domain discrimination while retaining the rest of the architecture as fixed. In this context, we evaluate target performance with respect to varying capacity of the discriminator by either adding or removing 2 fully connected layers before the final layer.

Experiments displayed in Table 4.4 displays that optimal performance is often achieved by maintaining discriminator capacity. For all split layer configurations from 3 to 5, trimming layers from the discriminator reduces target performance when diminishing the discriminator expressiveness. Furthermore, extending the discriminator lead to reduced performance in all evaluated scenarios. Especially for an earlier split layer, the approach results in high variance with different model initialisation.

Layer	3			4			5		
Adapted Capacity	-2	0	2	-2	0	2	-2	0	2
P_T[%]	73.79	76.30	76.52	74.47	**82.03**	80.55	79.00	81.06	80.37
σ_T[%]	6.08	3.94	3.17	2.62	2.09	1.94	0.67	0.53	1.66

Table 4.4: Evaluation of performance for adapting the discriminator capacity. For the optimal split layer, the best performance is achieved by not separately changing the discriminator architecture. In particular lowered discriminator capacity hurts performance, resulting in a less flexible model.

Figure 4.7: Example images for source domain: day (left), and target domain: night (right). We do not provide aligned data between both domains and the images present different sections along the driven route for both domains.

4.2.3 Performance: Where does ADA work best and how do larger appearance changes influence performance?

In consistency with existing work [Ganin et al., 2016, Tzeng et al., 2015b, Bousmalis et al., 2016], we calculate practical bounds for the module performance in the target domain. For the lower bound, we evaluate the classification accuracy of a classifier only trained on training data from the source domain and the upper bound is based on training only with data from the target domain. Note, all models are initialised with pretrained convolutional layers pretrained on ImageNet [Russakovsky et al., 2015].

As the task alignment becomes more challenging with increasing difference between the data distributions, we evaluate benefits of the approach on the following source-target pairs *Sunny* - *Overcast* and *Day* - *Night* with the latter representing

a more complex transfer scenario (Figure 4.7).

Classification

	domains	AlexNet	AlexNet w ADA	AlexNet w target labels [1]
P_T[%]	overcast-sunny	67.95±1.02	**82.03**±2.09	(87.96±1.40)
P_T[%]	day-night	26.79±4.90	**30.21**±4.94	(90.42±1.04)

Table 4.5: Evaluation of performance improvement in the target domain based on ADA for the place recognition task. ADA leads to significantly higher accuracy in all transfer domains and approaches in scenarios with minor shift towards the performance with known target labels. In the day-night scenario based on a strong difference in appearance, the approach only leads to minor benefits and higher variance in the results.

By investigating more significant appearance changes between source an target data, we can evaluate limitations and strengths of ADA. While the approach leads to performance gains in all scenarios, it is better suited to address domains with limited shift, such as the overcast-sunny scenario, and performs significantly closer to the upper bound obtained from training on labelled target data as displayed in Table 4.5. Furthermore, the standard deviations are significantly higher for experiments based on the larger shift from day to night. The target performance for semantic segmentation in Table 4.6 displays significant improvement over the direct use of only source training data, partially bridging the gap towards upper bound performance in the case of available target domain supervised data. The adversarial training framework as well as the balancing of supervised and adversarial objective, become more instable with higher variance for larger appearance shifts as further addressed under Section 4.4.

Improving Performance in Source Domain

ADA adapts the encoder module for source and target domains. As the encoder is trained towards an additional adversarial objective, the ADA training process

[1] Performance with available training labels on the target domain. This serves as upper bound for performance.

Free-Space Segmentation			
	FCN-VGG16	FCN-VGG16 w ADA	FCN-VGG16 w target labels [1]
P_T[%]	75.12±0.76	**85.27**±1.03	(93.94±0.84)

Table 4.6: Evaluation of performance improvement in the target domain based on ADA for the free-space segmentation in the context of autonomous driving. ADA significantly exceeds the performs of naive training with only source domain data and bridges a significant part of the performance gap to experiments with available target domain labels. The segmentation task measures performance P_T as mean average precision.

affects the performance not only in the target but also the source domain. The following experimental analysis aims at providing insights into the potentials and challenges of applying ADA in situations where the performance in the source domain P_S is of importance.

In addition to improving target performance, ADA can acts as a regulariser to enhance generalisation within the source domain as represented in Table 4.7. The gain in performance is enabled by both domains having significant structural similarities. With increasing discrepancies between the domains, it can reduce performance on the source domain as it diminishes relevant information for the supervised based on the adversarial objective, as can be seen with respect to day-night adaptation in Table 4.7.

	domains	trained on source	ADA
P_S[%]	overcast-sunny	87.83±1.56	**90.82**±1.45
P_S[%]	day-night	**90.04**±3.01	89.16±1.78

Table 4.7: Evaluation of the regularising effect of ADA on source performance. For minor appearance difference between the domains, the approach can improve source performance by acting as a regulariser. However, under stronger shifts such as the day-night transfer it can lead to reduced source performance.

4.3 Evaluation: Semantic Segmentation

The experiments on semantic segmentation utilise data with the same underlying appearance shifts given in the ablation study in the previous sections. While for these earlier tests, multiple experiments can be run in parallel on NVIDIA Titan 1080 GPUs in under 2 days, the segmentation tests (with discriminator) required the full card memory even for smaller batch sizes (4/8 images per batch).

In the following section, we apply the distilled insights for ADA to the task of free space segmentation as possible input data for motion planning systems. We use the fully convolutional FCN-VGG16 [Long et al., 2015] architecture, which is split into encoder and classifier/discriminator. We set the split layer towards the middle of the architecture after the 4th maxpool operation (see [Long et al., 2015] for the exact architecture) with fixed capacity of the discriminator architecture and apply the confusion loss following the evaluation in Section 4.2.1.

Both source and target datasets include 1000 training and 100 test images based on a midday-to-evening adaptation scenario, limiting the domain shift compared to the full day-to-night transfer from Section 4.2.3. The segmentation labels are generated for free-space/obstacles following the approach of Barnes et al. [2017].

When trained with domain adaptation, the segmentation network performs outperforms the basic supervised approach and is able to bridge the gap towards performance with available target labels as displayed in Figure 4.8.

4.4 Discussion

Adversarial domain adaptation has empirically demonstrated the capability to improve performance in different task with underlying appearance changes, as shown in Sections 4.2 and 4.3.

In general, adversarial training frameworks such as ADA tend to be notoriously complicated to optimise. However, a limited number hyperparameters has strong influence for given problems and can be adapted to stabilise and optimise the training process.

FCN-VGG16

FCN-VGG16
w ADA

FCN-VGG16
w target labels[1]

Figure 4.8: Examples of segmentation results in the target domain. Green and red represent free-space and obstacles respectively. ADA leads to significant performance gains for the unsupervised target domain. The scenario with available target labels serves as upper bound for the performance when supervised information would be available. Qualitatively, the differences focus commonly on reflections of the street lights as seen for example in the upmost segmentation map for FCN-VGG16 with and without ADA. While ADA improves the segmentation results the comparison to the experiments with target labels exemplify the remaining challenges as the segmentation is less accurate with respect to the car as well as inaccuracies around obstacles in all 3 example segmentations.

We found the principal factors for optimising performance to be:

- Using relevant initialisation including supervised warmup helps to determine relevant feature representations.

- Applying the confusion loss for the encoder enables better stabilisation of the training procedure.

- The optimum position for the split layer is mid network. Particularly the application in earlier layers can significantly reduce the benefits.

The current approach finds its limitations when the differences between source and target domains are too severe. As exemplified by the day to night transfer scenario, the methods still leads to improvement but with stronger variation in the underlying domains, the adversarial encoder loss might lead to reduced performance and training instability if weighted improperly against the supervised loss. If the adversarial loss dominates in such situations the encoder features lose more information relevant for the supervised task.

To overcome the limitation of significant domain shifts, future directions are given by curriculum approaches, dividing the alignment optimisation over multiple steps. The required data for these methods is already available in robotics applications with continuous deployment of sensing platforms. Furthermore, semi-supervised approaches can be employed to incorporate further structure and align the conditional distributions over feature representations given labels comparable to [Tzeng et al., 2015b].

4.5 Conclusions

This chapter provides another angle to increase the efficiency of providing supervision to our models. After the last chapter addressed the low-cost generation of supervised demonstration data for learning cost functions, the presented work aims at the adaptation of models to domains without access to supervised labels. The approach reduces the requirements for the supervised annotation of data on all types of appearance shifts we expect to encounter. By enabling unsupervised model adaptation between similar domains, we enable the system to transfer information between domains.

We cast the common challenge of appearance change in outdoor robotics as an unsupervised domain adaptation problem and extend recent adversarial paradigms for the adaptation of popular, existing architectures leading to performance benefits in unlabelled target domains. While instabilities of adversarial training can inhibit the extension to large scale problems, our extensive tests on a surrogate task with moderate complexity expose significant factors of influence and enable application on the full-scale path segmentation task for autonomous driving. With this framework we hope to pave the way for further application on real world tasks, in particular in the context of autonomous mobility where strong structural similarities of feature distribution can exist between different source and target domains, e.g. based on spatial overlay of driven routes and terrains.

Beyond dealing with appearance change we see many potential applications of ADA in robotics, whether it be changes in sensor modalities, or transferring models from a simulated virtual environment to improve their performance in the real world.

Lastly, as the alignment of marginal distributions was observed to be significantly more successful between similar domains, there is a strong opportunity to explore curriculum based variations of ADA with a focus on gradual alignment in continually changing environments. The approach is of relevance in particular for scenarios with continuous deployment of mobile platforms as well as in lifelong learning tasks.

5

Parallel Training for Transfer Learning

Contents

This chapter addresses transfer learning between systems with different underlying dynamics. While the previous chapter covered domain adaptation for different observation spaces caused by appearance shift, the following sections focus on inaccurate parameterisations in terms of physical properties such as friction or density as well as procedural approximations of the full equations for the system dynamics. These aspects become relevant in the context of using simulations for faster, cheaper and safer training of algorithms, which is most common for the reinforcement learning paradigm.

Recent work in reinforcement learning has led to significant successes such as

agents outperforming humans on a multitude of computer games [Mnih et al., 2015, Van Seijen et al., 2017] and surpassing the best human players in the games of Chess [Campbell et al., 2002] and Go [Silver et al., 2016]. The principal commonality between these settings is the availability of virtually unlimited training data as these systems can be trained in parallel and significantly faster than real-time, real-world executions.

However, training agents for operation in the real world presents a significant challenge to the reinforcement learning paradigm as it is constrained to learn from comparatively expensive and slow task executions. In addition, limits in the perception system and the complexity of manually providing informative real world rewards for high complexity tasks often result in only sparse and uninformative feedback being available, further increasing sampling requirements. As a result, many tasks which involve physical interaction with the real world and are simple for humans, present insurmountable challenges for robots [Moravec, 1988].

While there has been significant progress towards fast and reliable simulators [Liu et al., 2017, Todorov et al., 2012, Coumans, 2015], they do not represent exact replications of the platforms and environments we intend to emulate. Systematic model discrepancies commonly prevent us from directly porting policies from simulation to the real platform.

The principal differences between simulation and real world are based on the type of system observations as well as discrepancies between system dynamics. Recent developments aim at designing visually more similar environments [NVIDIA, 2017, Shah et al., 2017] and current research targets adapting policies to be invariant with respect to differences between the observation spaces of simulator and real platform [Stadie et al., 2017, Tobin et al., 2017, Long et al., 2017, Ganin et al., 2016].

Another approach to address invariance with respect to domain was presented in Chapter 4, where input images from different underlying domains are encoded into invariant feature representations. Both simulation and real world can be treated as different domains including the same semantic objects, such as robot and correlated environment. Furthermore, Stadie et al. [2017] address the related

challenge of imitation learning by combining a similar approach to adversarial domain adaptation and replace environment rewards in their example scenarios with rewards based on density estimates for demonstrator trajectories.

To address discrepancies between the two systems' dynamics, fine tuning pre-trained policies from simulation on the real platform is a straightforward approach. However, as policies trained via reinforcement learning will learn to exploit the specific characteristics of a system – optimising for mastery instead of generality – a policy can overfit to the simulation. The resulting initialisation can prove unfavourable to random initialisations for further training on the real platform as it might inhibit exploration and lead to the optimisation process getting stuck in local optima. A phenomenon which we demonstrate in the experiments in Section 5.2.4. On the other hand, in cases where fine tuning improves performance it can be straightforwardly combined with the presented approach as described in Section 5.2.

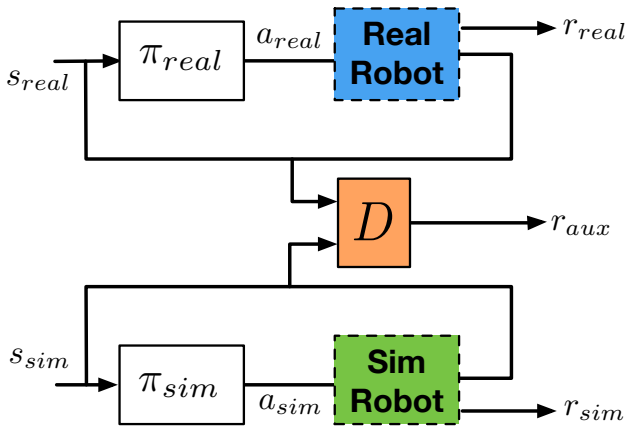


Figure 5.1: Simplified schema for Mutual Alignment Transfer Learning. Both systems are trained to not only maximise their respective environment rewards but also auxiliary alignment rewards that encourages both systems to occupy similar distributions over visited states. Furthermore, the simulation policy can be trained orders of magnitude faster than the real platform solely based on its environment reward.

While the actions performed by the simulator policy can fail to accomplish the task on the robot, the sequence of states visited by the agent in simulation represents its task under limited variation in the system dynamics. We propose

Mutual Alignment Transfer Learning (MATL), which instead of directly adapting the simulation policy, guides the exploration for both systems towards mutually aligned state distributions via auxiliary rewards. The method is displayed in Figure 5.1 and employs an adversarial approach to train policies with additional rewards based on confusing a discriminator with respect to the originating system for state sequences visited by the agents. By guiding the target agent on the robot towards states that the potentially more proficient source agent visits in simulation, we can accelerate training. In addition to aligning the robot policy to progress in simulation, we extend the approach to mutually align both systems, which can be beneficial as the agent in simulation will be driven to explore better trajectories from states visited by the real-world policy.

While the method is ultimately designed to enable transfer between simulated and real platforms, the evaluation in this chapter addresses transfer learning between different simulations, including differences in physical parameters as well as completely different implementations of dynamics simulators (such as transfer between MuJoCo [Todorov et al., 2012] and DART [Liu et al., 2017]). We evaluate the developed method on a set of common reinforcement learning benchmark tasks [Duan et al., 2016] to transfer between simulations with differences in system parameters such as density, dampening, and friction. Furthermore, we extend the experiments to address additional challenges relevant in the context of real platforms such as sparse and uninformative rewards. The final experiments investigate transfer between different simulation engines with unknown discrepancies as a stronger proxy for real robot experiments.

We demonstrate that auxiliary rewards, which guide the exploration on the target platform, improve performance in environments with sparse rewards and can even guide the agent if only uninformative or no environment rewards at all are given for the target agent. Furthermore, the approach proves to be capable of guiding training given scenarios with significant discrepancies between the system dynamics when direct transfer or fine tuning approaches fail, as investigated in Section 5.2.4.

5.1 Mutual Alignment Transfer Learning

We consider a simulation to real robot transfer learning scenario for reinforcement learning with the setup of two agents acting in the respective source and target environments, each represented as Markov Decision Process (MDP). State space $s \in \mathcal{S}$ and action space $a \in \mathcal{A}$ are equal in both environments. However, the difference lies in the underlying system dynamics represented as $p_S(s_{t+1}|s_t, a_t)$ for the simulator and $p_R(s_{t+1}|s_t, a_t)$ for the real platform. The reward functions $r_S(s_t, a_t)$ and $r_R(s_t, a_t)$ which both agents optimise, can be the same for both environments, but can also differ as it is possible that we do not have access to the full reward function in the real world. However, both agents are intended to solve the same task. Both agents, simulation/source and robot/target, act according to their respective stochastic policies $\pi_\theta(a_t|s_t)$ and $\pi_\phi(a_t|s_t)$, which are parameterised as neural networks by θ and ϕ. In the following sections the terms simulator and robot, or respectively source and target are used interchangeably. Hereinafter, the subscripts R and S refere to robot and simulator systems respectively.

Mutual Alignment Transfer Learning (MATL) leverages information gained while training in simulation to improve performance on a target robot with potentially different system dynamics via the introduction of auxiliary reward functions to guide both systems to visit similar states or state sequences. The approach is visualised in Figure 5.1. When the simulator policy cannot be directly applied due to systematic differences, the simulation trajectories still contain information about how to solve the task on a different system given limited variation in the underlying system dynamics.

5.1.1 Policy Optimisation

The method trains policies with auxiliary alignment rewards by running reinforcement learning algorithms simultaneously on both systems, here Trust Region Policy Optimisation (TRPO) [Schulman et al., 2015].

TRPO builds on Natural Policy Gradients (NPG) [Kakade, 2002], an approach that aims to provide additional guarantees for the policy update step. NPG itself is

an extension of Vanilla Policy Gradient methods [Sutton and Barto, 1998], which are described in Section 2.2.

Equation 5.1 presents a reformulation of the expected performance of a policy (Equation 2.9) based on the advantage function (Equation 2.8) - the objective we aim to maximise by updating our parameters θ.

$$\begin{aligned}
\eta(\pi) &= \eta(\pi_0) + \mathbb{E}_{s_0,a_0,\dots}[\sum_{t=0}^{\infty} A_{\pi_0}(s_t, a_t)] && (5.1)\\
&= \eta(\pi_0) + \sum_{t=0}^{\infty} \sum_{s} \mu(s_t) \sum_{a} \pi(a_t|s_t)\gamma^t A_{\pi_0}(s_t, a_t) && (5.2)\\
&= \eta(\pi_0) + \sum_{s} \sum_{t=0}^{\infty} \mu(s_t) \sum_{a} \pi(a_t|s_t)\gamma^t A_{\pi_0}(s_t, a_t) && (5.3)\\
&= \eta(\pi_0) + \sum_{s} \rho_{\pi}(s) \sum_{a} \pi(a|s)\gamma^t A_{\pi_0}(s, a) && (5.4)\\
&= \eta(\pi_0) + \mathbb{E}_{s\sim\rho_{\pi}} \mathbb{E}_{a\sim\pi(a|s)}[A_{\pi_0}(s, a)] && (5.5)
\end{aligned}$$

The identity between Equations 5.1 and 5.5 is proven in Kakade [2002]. However, to evaluate the performance gain over the previous policy π_0 we would be required to determine the expectation over the discounted state visiting frequencies ρ_{π} of the new policy π after the update. Kakade [2002] instead use ρ_{π_0} as an approximation, leading to the new objective $\mathcal{L}_{\pi_0}(\pi)$ in Equation 5.6. They furthermore prove that given a parametric policy, this objective matches $\eta(\pi)$ first order. Therefore any sufficiently small step that improves our new objective will improve the overall performance of the policy.

$$\eta(\pi) \cong \mathcal{L}_{\pi_0}(\pi) = \eta(\pi_0) + \mathbb{E}_{s\sim\rho_{\pi_0}} \mathbb{E}_{a\sim\pi(a|s)}[A_{\pi_0}(s, a)] \tag{5.6}$$

In order to enable the application of the objective with ensured performance bounds, they introduce conservative policy iteration which determines the new policy as a mixture of old and current steps. Let $\alpha \in [0, 1]$ be the mixture parameter for the update. Let $\pi' = \arg\min_{\pi} \mathcal{L}_{\pi_0}(\pi)$ be the current optimal policy for the surrogate objective $\mathcal{L}_{\pi_0}$.

$$\pi(s) = (1-\alpha)\pi_0(s) + \alpha\pi'(s) \tag{5.7}$$

A trust region [Yuan, 1999] method is used to guarantee bounds on improvement, where ϵ defines the maximum expected advantage with respect to the surrogate objective's optimal policy.

$$\eta(\pi) \geq \mathcal{L}_{\pi_0}(\pi) - \frac{2\epsilon\gamma}{(1-\gamma(1-\alpha))(1-\gamma)}\alpha^2 \tag{5.8}$$
$$\text{where} \quad \epsilon = \max_s \mathbb{E}_{a\sim\pi'(s)}[A_{\pi_0}(s,a)]$$

Schulman et al. [2015] simplify this bound further for $\alpha \ll 1$ and show that the approach can be extended from mixture policies to general stochastic policies by finding a replacement for the mixture parameter α^2 based on the the Kullback-Leibler divergence between previous and new policy (for the exact derivation the reader is referred to the appendix of [Schulman et al., 2015]).

$$\eta(\pi) \geq \mathcal{L}_{\pi_0}(\pi) - C_{TRPO} \max_s D_{KL}(\pi_0(s)||\pi(s)) \tag{5.9}$$
$$\text{where} \quad C_{TRPO} = \frac{2\epsilon\gamma}{(1-\gamma)^2}$$

While it is often still intractable to compute the maximum KL divergence $\max_s D_{KL}$ over the whole state space, we can derive a heuristic approximation by building on the mean KL divergence 5.10.

$$\bar{D}_{KL}(\pi_0(s)||\pi(s)) = \mathbb{E}_{s\sim\rho_{\pi_0}}[D_{KL}(\pi_0(s)||\pi(s))] \tag{5.10}$$

$$\mathcal{L}_{new} = \mathcal{L}_{\pi_0}(\pi) - C_{TRPO}\bar{D}_{KL}(\pi_0(s)||\pi(s)) \tag{5.11}$$

Schulman et al. [2015] suggest three variants for applying these insights. We focus on placing a hard constraint δ on the KL divergence $\bar{D}_{KL}(\pi_0(s)||\pi(s)) \leq \delta$ for the realisation of Mutual Alignment Transfer Learning, enabling us to maximise $\mathcal{L}_{\pi_0}$ via conjugate gradient.

5.1.2 Unilateral Alignment

Following the derivation in Section 5.1.1, both policies, $\pi_\theta(a|s)$ and $\pi_\phi(a|s)$ respectively with parameters θ and ϕ, are trained via TRPO with the method's gradient computed following the objective in Equation 5.12, which represents the gradient of the (non-discounted) expected reward $\eta(\pi)$ from Equation 2.9.

$$\nabla\eta(\pi) = \mathbb{E}_{a_t\sim\pi, s_{t+1}\sim p}[\nabla \ \log\pi(a_t|s_t)\, r(s_t)] \tag{5.12}$$

Auxiliary rewards for the alignment process are generated in an adversarial setting with the objective to confuse the discriminator D_ω which is trained to classify the system of origin for state sequences ζ_t from simulation and robot system. As displayed in Equation 5.13, state sequences can be subsampled to ensure significant change between successive states.

The discriminator D_ω is parameterised as neural network with weights ω and trained according to Equation 5.13 to classify the originating system (simulation-robot) of fixed length state sequences ζ_t.

$$\begin{aligned} \mathcal{L}_\mathcal{D} &= \mathbb{E}_{\zeta_t\sim\pi_\theta, p_R}[\log(D_\omega(\zeta_t))] + \mathbb{E}_{\zeta_t\sim\pi_\phi, p_S}[\log(1 - D_\omega(\zeta_t))] \\ \zeta_t &= s_t, s_{t+k}, s_{t+2k}, ..., s_{t+nk} \quad \text{with } n \in \mathbb{N}_0; k \in \mathbb{N} \end{aligned} \tag{5.13}$$

While training both agents in parallel, training in simulation can potentially be performed orders of magnitude faster. Therefore, we run the updates for the simulator policy at M times higher rate with only the environment reward being used for these updates. As the agent in simulation trains at a much higher rate, it will perform significantly better in most situations. The robot agent benefits from progress in simulation via auxiliary rewards κ_R in Equation 5.14, which are based on the discriminator's confusion. These dense rewards create an additional training signal towards the alignment of the distributions over visited states of both agents.

robot agent:

$$r(s_t) = r_R(s_t) + \lambda\, \kappa_R(s_t) \tag{5.14}$$

$$\kappa_R(s_t) = \log(D_\omega(\zeta_t)) \tag{5.15}$$

5.1.3 Mutual Alignment

In addition to aligning the robot policy to adapt to progress in simulation, the reciprocal alignment of the simulation policy can be beneficial as the agent in simulation will be driven to explore better behaviours from states visited by the robot agent.

Therefore, in the full MATL algorithm, both simulator and robot agent rewards, respectively $r_S(s_t)$ and $r_R(s_t)$ are extended via the auxiliary rewards κ_S and κ_R, for simulator policy and robot policy, as given in Equations 5.14 and 5.16. By training both policies towards mutual alignment, not only does the robot policy learn from the progress in simulation but the simulator policy is also pushed to explore better behaviour for states visited in the robot environment.

The specific formulation in Equations 5.17 and 5.15 builds upon the idea of maximising the confusion loss formulation of the GAN framework [Goodfellow et al., 2014] , which was found empirically to be better suited for the transfer task than the original minimax formulation. The confusion objective for adversarial training addresses a principal shortcoming of the original formulation by which the gradients for the generating module (represented here by the policies) vanish when the discriminator performance is maximised.

simulator agent:

$$r(s_t) = r_S(s_t) + \lambda\, \kappa_S(s_t) \tag{5.16}$$

$$\kappa_S(s_t) = -\log(D_\omega(\zeta_t)) \tag{5.17}$$

In conclusion, the gradient steps for TRPO in the MATL framework are obtained by combining Equations 5.12, 5.14, and 5.16. Both gradients for the simulator and robot policies are given in Equations 5.18 and 5.19 respectively.

$$\begin{aligned} \nabla_\theta \eta(\pi_\theta) &= \mathbb{E}_{a_t \sim \pi_\theta, s_{t+1} \sim p_S}[\nabla_\theta \ \log \pi_\theta(a_t|s_t) \ (r_S(s_t, a_t) \\ &\quad -\lambda \ \log(D_\omega(\zeta_t)))] \end{aligned} \tag{5.18}$$

$$\begin{aligned} \nabla_\phi \eta(\pi_\phi) &= \mathbb{E}_{a_t \sim \pi_\phi, s_{t+1} \sim p_R}[\nabla_\phi \ \log \pi_\phi(a_t|s_t) \ (r_R(s_t, a_t) \\ &\quad +\lambda \ \log(D_\omega(\zeta_t))] \end{aligned} \tag{5.19}$$

Algorithm 3 presents the full training procedure, iteratively taking rollouts in both systems based on the respective policies and update the discriminator as well as both policies. The simulator updates can be run faster than robot policy updates by a factor of M. While the discriminator could be potentially trained in the inner loop as well with samples from the simulator, this is to be prevented as it would lead to a highly imbalanced training dataset. If this type of update is preferred, it is required to weight the discriminator training objective for simulator samples lower.

5.2 Evaluation

We evaluate MATL on transfer learning scenarios based on various common reinforcement learning tasks including environments taken from rllab [Duan et al., 2016], OpenAI gym [Brockman et al., 2016] and DartEnv [dar, 2017]. For Sections 5.2.1 to 5.2.3 we use variations of the same simulation environment with strongly varied parameters for the system dynamics. To evaluate our approach for applications with significantly inaccurate knowledge of system dynamics, we severely alter the parameters for joint dampening, friction and densities between both systems. The reader is referred to the additional documentation[1] for detailed information about algorithm and environment parameterisation. To extend the

[1] Additional information to environment and algorithm parameterisation can be found under sites.google.com/view/matl

Algorithm 3: Mutual Alignment Transfer Learning

Input : environments $\text{MDP}_{R,S}$, alignment weight λ, iterations outer loop N and inner loop M, rollout horizon T
Output : target policy π_ϕ

$\pi_\theta, \pi_\phi, D_\omega \leftarrow$ initialise
for $i \leftarrow 1$ **to** N **do**
  $(s_{0..T}, a_{0..T}, r_{0..T})_S \leftarrow$ rollout π_θ on MDP_S
  $(s_{0..T}, a_{0..T}, r_{0..T})_R \leftarrow$ rollout π_ϕ on MDP_R
  $D_\omega \leftarrow$ gradient update following Eq. 5.13
  $\pi_\theta, \pi_\phi \leftarrow$TRPO updates following Eqs. 5.18 & 5.19
  for $j \leftarrow 1$ **to** M **do**
    $(s_{0..T}, a_{0..T}, r_{0..T})_S \leftarrow$ rollout π_θ on MDP_S
    $\pi_\theta \leftarrow$TRPO update following Eq. 5.18 with $\rho_S(s_t, a_t) = 0 \; \forall \; t \in T$
  end
end

evaluation, we address transfer learning between two different simulation engines, MuJoCo [Todorov et al., 2012] and DART [Liu et al., 2017], in Section 5.2.4.

The evaluation focuses on six different approaches in the following experiments:

- independent - Independent training of the robot policy without auxiliary rewards.
- direct_transfer - Direct application of the simulator policy on the real platform.
- fine_tuning - Transfer of the fully trained simulator policy and subsequent fine tuning based only on robot environment rewards.
- MATLu - Unilateral alignment training with environment rewards but without auxiliary reward for the simulation policy.
- MATL - Mutual alignment training with environment rewards and both auxiliary rewards.

- MATLf - Combining MATL with fine tuning. Training via MATL but starting from a transferred fully trained simulator policy.

All diagrams present only the number of real world iterations as additional iterations in simulation can be obtained at significantly higher rate and (in comparison) negligible effort. The performance for each experiment is normalised between zero and one. Additional experiment details as well as the Wasserstein GAN variant of the method, which is applied in Section 5.2.4, can be found in Appendix B.

Guiding Questions

- Are MATL's auxiliary alignment rewards suited to guide exploration in environments with sparse rewards? (Section 5.2.1)
- Do the auxiliary rewards provide enough feedback on how to solve a task in a situation where the real world agent has only access to an uninformative reward (such as a reward for not falling)? (Section 5.2.2)
- Can we succeed in training towards a task with no reward given in the target environment other than the auxiliary rewards (additionally deactivating all environment rewards in the target environment including the reward for staying alive)? (Section 5.2.3)
- How important is the mutual alignment versus a unilateral alignment of just the target policy? (Section 5.2.1 to Section 5.2.4)
- How well does the approach handle transfer between different simulation frameworks with unknown differences in system dynamics? (Section 5.2.4)

5.2.1 Sparse Rewards

The first section of our evaluation focuses on the application of MATL to environments with sparse rewards. In these experiments, both environments - simulation and real - only have access to sparse feedback which is given when the

agent is situated within a distance ϵ to the target position[1] instead of a dense, distance based reward.

The experiment scenarios utilised in this section are based on the *Cartpole*, *Cartpole Swingup*, and *Reacher2D* environments [Duan et al., 2016, Brockman et al., 2016]. The *Cartpole* environment presents a torque controlled pendulum with the goal to stabilise it in the top position. *Cartpole Swingup* presents a similar scenario but is based a downwards pointing start position of the pendulum, which first has to be swung up and then stabilised. The *Reacher2D* environment provides a 2-link reacher acting in a 2D world to reach randomly spawned in reachable distance on a 2D plane. The goal for this scenario is achieved when the reacher's endeffector comes within a small distance of the goal position. Both joints of the reacher are torque controlled and in addition to the robots configuration, the observation space includes the 2D position of the goal.

Sparse feedback renders these situations more complex than scenarios with dense rewards as the agents only seldom get feedback and learning and progress is delayed. The auxiliary rewards given by MATL however are non-sparse and can help guide the real world agent to solve the task, given it has learned to solve the task in simulation. For these simpler tasks it was found empirically sufficient to apply the discriminator to single states.

The direct transfer of the simulation policy is unsuited for application on the target system in both tasks as displayed in Figure 5.2. While independent training already learns fast and fine tuning adapts even better, the best performance is achieved by combining MATL with fine tuning.

5.2.2 Uninformative Rewards

We furthermore investigate scenarios limited to an uninformative reward for locomotion tasks. In these cases, the only remaining component is a cost for falling in the target environment. The agent in simulation is still provided with access to the full reward including a forward-guiding component. We evaluate these scenarios

based on how well the agent learns to move forward and therefore the capability of MATL to guide to forward motion. The performance is now given as metric based on the average final distance of the agent in the direction of locomotion.

The *Hopper2D* environment [Duan et al., 2016, Brockman et al., 2016] (as displayed in Figure 5.5) represents a 2D world with a 4 link hopper with 3 torque controlled joints. The environment is more complex than the previously mentioned ones and includes abruptly changing dynamics (e.g. loss of contact with the ground). The reward here is based on distance moved forward and falling (top segments close to the ground) is negatively rewarded.

Similar to Stadie et al. [2017] we exploit in this section the sequential structure of these tasks and apply the discriminator to state pairs of time-steps t and t+4, which has been shown to work well across a variety of different tasks.

The most robust policy is simply standing still as moving forward increases the risk of falling. This situation renders the task more complex as the auxiliary reward has to overcome a potentially conflicting reward signal. Figure 5.3a displays that the alignment parameter λ is highly relevant to encourage stable walking behaviour. The conflict between the reward components results in a more conservative, ankle-based running style as can be found in the videos as part of the additional material[1].

Figure 5.3 shows that independent training results in very limited motion as reward signal encourages the agent to stand still. Fine tuning as well as direct transfer of the simulation policy perform slightly better, but are significantly surpassed by all versions of MATL.

5.2.3 Without Robot Environment Reward

To evaluate if transfer is possible without any environment reward in the target environment we run a set of experiments with only auxiliary reward based updates for the robot policy. The changes in parameterisation between simulation and robot environments are equal to the settings for Sections 5.2.1 and 5.2.2.

Contrary to the other sections, the performance is represented for these experiments as the ratio of the maximum performance achieved in the same environment with available environment reward from Section 5.2.1 and 5.2.2. This metric is chosen to evaluate the relative performance reduction for all methods when access to environment rewards is prevented.

In all scenarios, MATLf – the combination of fine tuning and auxiliary rewards – outperforms the other methods. This fine-tuning of a simulator-trained policy via MATL improves from a well-suited initialisation, often exceeding other versions of MATL in its maximally achieved reward. As expected, these environments result in no significant progress for methods training only on target rewards such as fine tuning, which keeps the same performance as the pretrained policy.

5.2.4 MuJoCo to DART

While the earlier experiments are based on the MuJoCo simulator and varied parameters for different system properties, we extend the evaluation of our approach towards differences between two simulation software packages, namely from MuJoCo [Todorov et al., 2012] to DART [Liu et al., 2017]. An additional challenge of this transfer experiment is that not only parameter values vary but the underlying algorithms differ in terms of parameter types and in particular contact modelling [dar, 2017].

The confusion loss formulation for the original GAN framework did not result in improvement for these experiments and presented results are based on an adaptation of the Wasserstein GAN (WGAN) loss [Arjovsky et al., 2017], which is described in greater detail as part of the additional materials.

As the DART environments differ significantly from their MuJoCo counterparts, direct transfer as well as fine tuning result in low performance. In particular, it can be seen that pretraining in one environment results to a low quality initialisation and independently training on the target environment surpasses the fine tuning

approach. This inherently also leads to low performance for MATLf which builds on the pretrained policy, but does not affect MATLu and MATL.

5.3 Discussion

MATL has been shown to work under significant differences in system dynamics between source and target platform as demonstrated in Section 5.2, including situations when direct transfer of the simulator policy fails to show good performance. A current shortcoming is the potential instability of the adversarial training framework and connected effort in tuning the hyperparameters. The alignment weight parameter λ is of particular importance in the context of potentially conflicting rewards as is represented by the uninformative rewards in Section 5.2.2. The weight has to be increased above the value of 0.1 which is used for most other experiments as the safety based environment rewards will prevent exploration.

Different simulation engines, as given in Section 5.2.4, provide a particular challenge for the transfer learning methods. We show that in these cases, the simulation based policy can overfit and result in providing an unsuitable initialisation for fine tuning which performs worse during training than standard random initialisations. Nevertheless, MATL demonstrably accelerates training under these conditions. Mutual alignment additionally increases performance consistently across all locomotion tasks (Sections 5.2.1 and 5.2.3) while being commensurate with unilateral alignment on the tasks with sparse rewards given in Section 5.2.1.

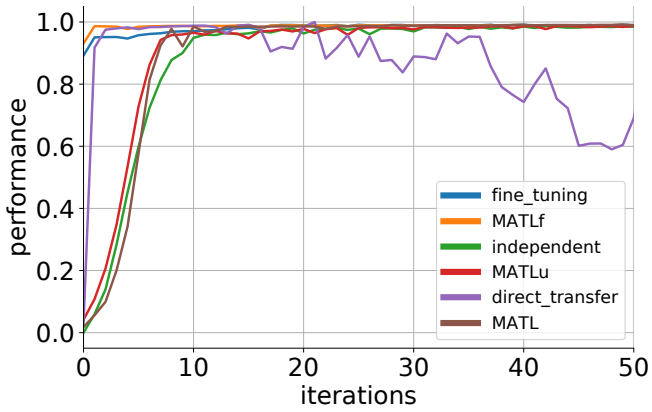
While distribution based alignment has been demonstrated to work well in the experiments in Section 5.2, evaluations based on the straightforward approach of direct alignment between the states along trajectories of each system only lead to limited performance improvements. Auxiliary rewards based on state-wise alignment of simulator and real world trajectories perform adequate mostly in low dimensional tasks with near-deterministic behaviour of the completely trained agent. The task of moving a ball towards a goal in 2D by applying forces in 2 directions serves as an example for this kind of scenario, where the optimal trajectory is - independent of initialisation - a straight line between start and target.

5.4 Conclusions

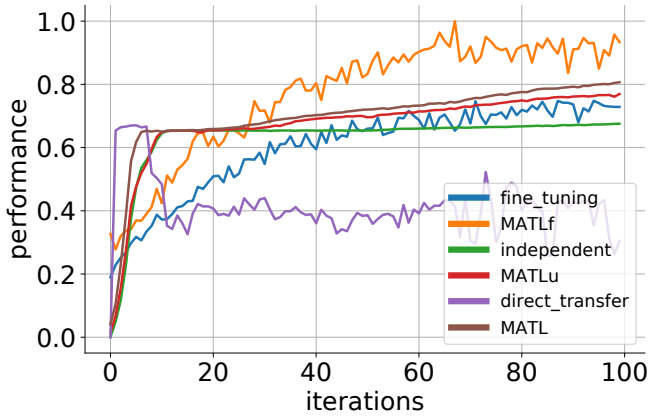
The contributions from this chapter extend the previous ones by addressing transfer learning under discrepancies in system dynamics for simulation to robot transfer. Transfer learning from simulation extends earlier scenarios with adaptation with weather/lighting based appearance changes towards different physical properties of objects. While it is reasonable to combine both approaches in future work, the performed experiments focus on systems without changes in observation spaces between both systems. The policy inputs in the experiments relies on aspects such as joint spaces which are equal between both systems.

The presented approach relies on parallel training of both agents and employs auxiliary rewards to align their respective distributions over visited states and can be straightforwardly supplemented with ideas based on fine tuning. Guiding robot exploration via alignment of state distributions between both systems has been shown to be beneficial for accelerating training and potentially lead to better final performance in scenarios with sparse or uninformative rewards for the target platform.

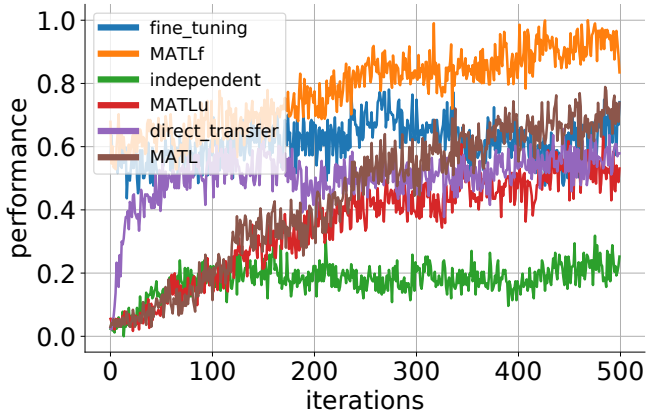
All experiments included in this chapter are concerned with transfer learning between two simulations with either different parameterisations or completely different simulation engines to create situations of misaligned and unknown system dynamics. Future work will address the full simulation to robot transfer scenario. As MATL employs partially trained policies on the real platform, further development will have to address methods for safe application of RL on real world platforms [Achiam et al., 2017, Kahn et al., 2017a]. Furthermore, one of the principal challenges going forward is the weighting of auxiliary rewards and original environment rewards in particular in situations where both can lead to conflicting behaviours.



(a) Cartpole

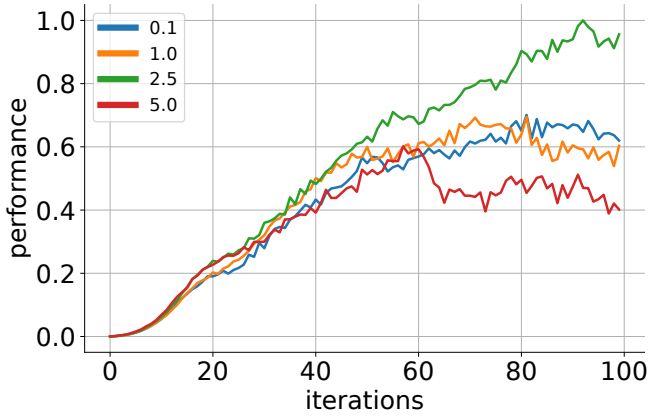


(b) Cartpole Swingup

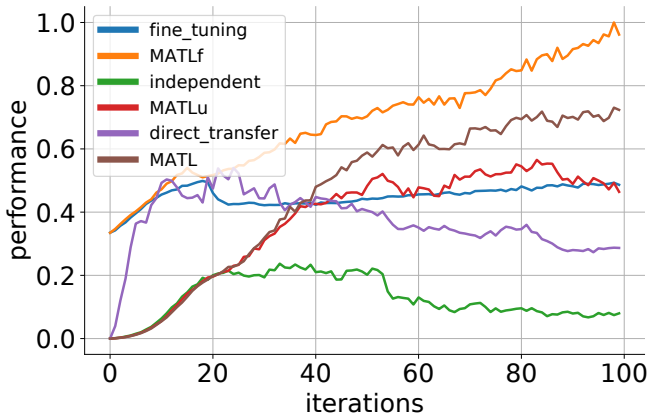


(c) Reacher2D

Figure 5.2: Transfer learning with sparse rewards in both environments. The dense auxiliary rewards from MATL help to accelerate training. While fine tuning already learns to perform optimally after a few iterations, MATLf accelerates training even further. Mutual alignment commonly slightly accelerates training progress over the unilateral approach.

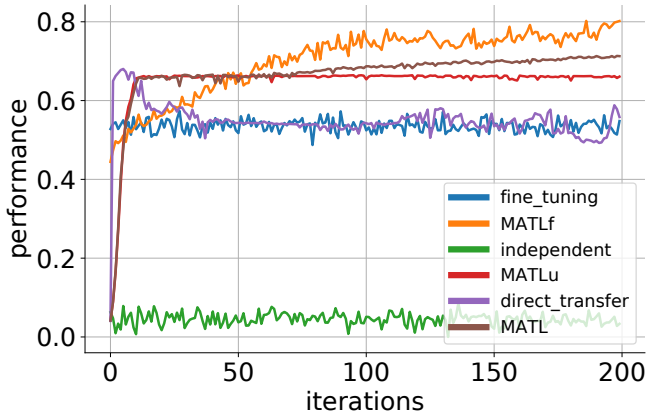


(a) Hopper2D - Alignment weight

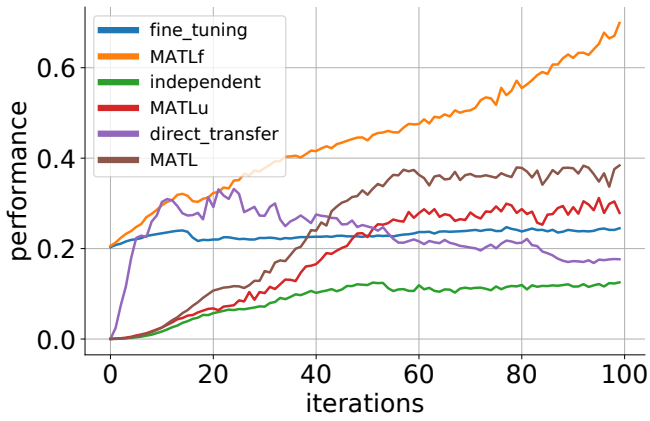


(b) Hopper2D - Methods

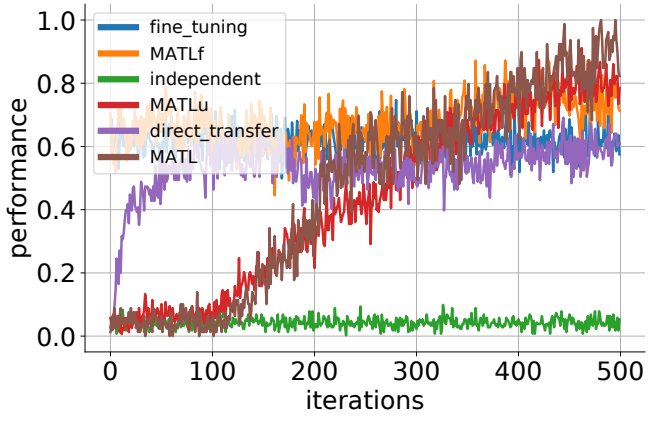
Figure 5.3: Transfer learning - Hopper2D task with uninformative rewards. Left: Transfer learning performance in dependency of alignment weight. The context of potentially conflicting rewards renders the hyperparameter choice more critical for efficient transfer. Right: Comparison against baselines with MATLf surpassing the other methods.



(a) Cartpole Swingup

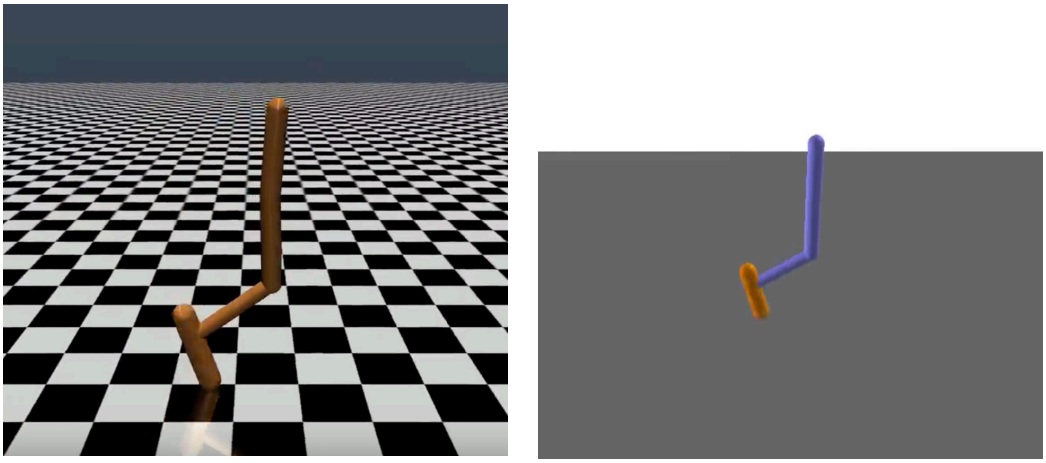


(b) Hopper2D



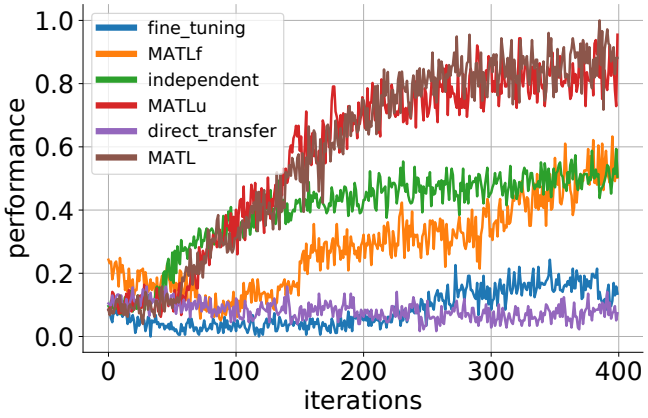
(c) Reacher2D

Figure 5.4: Transfer learning - Scenario without environment reward on the target platform. MATL consistently outperforms the other approaches with fine-tuning style approaches having an initial benefit but MATL quickly reaching higher performance. In most examples the combination of both directions is favourable.

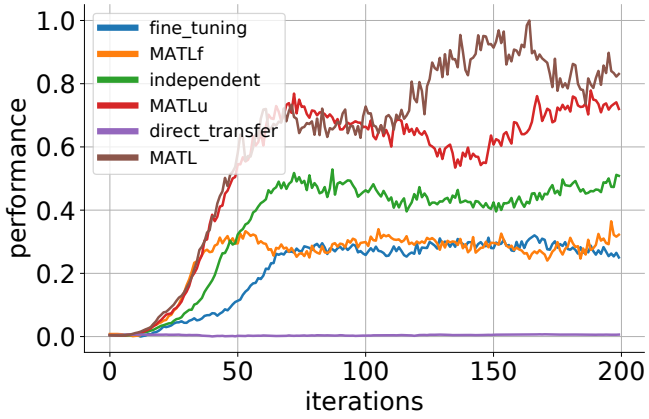


(a) Hopper2D - MuJoCo Environment **(b)** Hopper2D - DART Environment

Figure 5.5: Transfer learning - MuJoCo to DART simulator. While the images present visual differences, the focus of this work lies in the difference in implementation of the underlying system dynamics and transition functions.



(a) Reacher2D



(b) Hopper2D

Figure 5.6: Transfer learning - MuJoCo to DART simulator. While in all previous experiments, fine-tuning represented an advantageous supplement to MATL, in this case, fine-tuning results in a less suited initial parameterisation and therefore reduced performance in comparison with pure application of MATL.

6

Conclusions and Future Directions

Contents

6.1 Conclusions

To enable the integration of autonomous platforms into real world scenarios, multiple challenges have to be addressed. The complexity of perception caused by the immense variability of real environments as well as the modelling of intricate behaviours of interacting agents often render the design of manual rules as infeasible. While machine learning and particularly deep learning helped to overcome this challenge given the availability of large amounts of labeled training data, many applications are still out of reach given the high cost and time requirements of data collection and annotation.

In order to reduce the effort connected to introducing robots into new environments and tasks, the contributions of this thesis focus on three areas of machine learning to simplify the creation of supervised training datasets as well as include the use of unlabelled training data from different domains. Regarding the former, we address the use of large sets of demonstration data for learning driving behaviour in urban environments. Supervision is given in these cases by treating demonstrator behaviour as a type of annotations. In this way, we train our models to learn preferences that make them behave similarly to the exhibited human driving patterns. The latter idea extends the paradigm of data efficiency further by focusing on transferring knowledge between different domains. When trying to directly port models from domains with different underlying data distributions we experience reduced performance and can benefit strongly from unsupervised techniques to adapt to the new domain without the need for additional data annotation phases.

Additionally to the challenge of variations in the observation space, systems can exhibit differences with respect to successful control policies for dissimilar domains. Simulation-to-robot transfer learning scenarios present a prime example where given the same perceived state, optimal actions might differ as the dynamics simulation will be only approximately correct. Parameters for friction, density, mass, and the discrete approximation of continuous systems create inaccuracies which require adaptations of simulator-trained models to enable efficient and safe real world application.

6.1.1 Imitation Learning

The field of learning from demonstration provides a straightforward way to design robot behaviour for new tasks without excessive knowledge about programming or robotics. Learning the preferences underlying expert behaviour instead of directly imitating demonstrations can easier generalise to new tasks and can be incorporated into existing decision making systems as cost functions.

The contributions of this thesis focus on inverse reinforcement learning in order to learn cost functions for path planning of autonomous mobile platforms.

The contributions of this thesis in imitation learning include:

- **Maximum Entropy Deep Inverse Reinforcement Learning [Wulfmeier et al., 2015].** The development of a framework for utilising high-capacity neural network architectures to address the inverse reinforcement learning problem. The method enables straightforward scaling with respect to complexity of the environment and size of training datasets. Deep neural networks enable us in this context to be runtime independent of training dataset size and to approximate arbitrarily complex cost functions. We qualitatively and quantitatively evaluate MEDIRL on common, small-scale benchmarks against state-of-the-art approaches in IRL in Chapter 3. While the approach is shown to require slightly more training data than Gaussian process based approaches, it reaches state-of-the-art accuracy across all tasks. Further advantages of MEDIRL include the possibility to exploit raw data and learn spatial features instead of manually handcrafting the model input.

- **Large scale application for autonomous driving [Wulfmeier et al., 2016a].** We evaluate MEDIRL in terms of scalability and efficacy in real world tasks on an extensive dataset of over 120 km capturing the common driving behaviour of 13 different human drivers, resulting in over 25,000 training samples. The approach is able to outperform manually crafted cost functions for prediction of human driving behaviour. Furthermore, when classifying free and obstructed trajectories, the approach is able to significantly reduce the number of times the motion planning system would be stuck.

- **Demonstration of robustness towards systematic noise [Wulfmeier et al., 2016a].** We evaluate the method's robustness towards miscalibration of the sensor setup by synthetically changing the system calibration. We subsequently train MEDIRL and evaluate the model learned from miscalibrated data. MEDIRL demonstrates significantly better performance for prediction

of human trajectories and classification of traversable terrain than manually crafted cost functions given a miscalibrated setup.

- **Incorporation of prior domain knowledge [Wulfmeier et al., 2016b, 2017c].** To fully benefit from expert domain knowledge, MEDIRL is extended to include manually crafted cost functions into the training process. By pretraining the model and subsequently refining the cost maps to address additional complexities of real world sensor data and driver behaviour, MEDIRL's performance improves significantly over both, its baseline and a manual cost function.

Maximum Entropy Deep Inverse Reinforcement Learning as described in Chapter 3 has the following drawbacks, which can provide paths for future directions outlined in Section 6.2.1:

- **Dynamic environments** are not fully considered in the current implementation of MEDIRL as the planning step only considers the sensor input of the first step along the trajectory.

- **Catastrophic forgetting**. The current approach relies on accurately crafted learning rate schedules to prevent overwriting all prior information learned in the parameters from the pretraining step.

6.1.2 Domain Adaptation

The long-standing challenge of reduced model performance based on appearance change in a platform's environment is framed in this thesis as the domain adaptation problem. To fully benefit from models trained in other domains without the need for additional annotated data in a new domain, we focus on unsupervised domain adaptation for an outdoor robotics scenario.

The contributions of this thesis in unsupervised domain adaptation include:

- **Framework for straightforward domain adaptation training of existing architectures [Wulfmeier et al., 2017a]**. We demonstrate a method to adapt high-performing, commonly used neural network architectures and optimise them via adversarial domain adaptation. We determine the principal factors of influence on target domain performance to be the type of adversarial loss, the depth of the split-layer and stable feature learning. The latter is supported through pretrained architectures and a supervised warmup phase as describe in detail in Chapter 4.

- **Adversarial domain adaptation for semantic segmentation [Wulfmeier et al., 2017a].** While our earlier ablation study relies on place classification as a surrogate task with equal underlying appearance change, the main application of the method lies in the task of pixel-wise semantic segmentation of drivable paths for autonomous mobility. In the final segmentation task we see a significant performance increase in the target domain by applying ADA over a FCN-VGG16 baseline.

Adversarial Domain Adaptation as described in Chapter 4 has the following drawbacks, which can provide paths for future directions outlined in Section 6.2.2:

- **Instability of adversarial training**. The approach relies on the adversarial game between discriminator and encoder. While we aim to identify important factors for the stabilisation of training in Chapter 4 and recent work by other groups targets more stable GAN objectives [Arjovsky et al., 2017], the training process continues to be quite sensitive to hyperparameter selection.

6.1.3 Simulation-to-Robot Transfer Learning

Training robots for operation in the real world is a time consuming and expensive task. While sample complexity can be reduced by training policies in simulation, such policies can perform sub-optimally on the real platform given imperfect calibration of model dynamics and different observation space. The contributions in this thesis focus on the differences in system dynamics. Chapter 5 presents a method –

supplemental to fine tuning on the real robot – to benefit from parallel access to a simulator during training and reduce sample requirements on the real robot.

The contributions of this thesis in simulation-to-robot transfer learning include:

- **Mutual Alignment Transfer Learning [Wulfmeier et al., 2017b].** We introduce an approach for transfer learning that harnesses auxiliary rewards to guide the exploration for the robot agent based on the proficiency of an agent in simulation and vice versa. The approach relies on an adversarial discriminator that functions as density ratio estimator between the agents' state visitation distributions. By calculating rewards which are based on the confusion of the discriminator, it is possible to train both agents towards alignment of their respective visited states. We demonstrate that mutual alignment improves over simple unilateral alignment of the robot towards the simulator agent's distribution over visited states. We hypothesise that it is beneficial as the agent in simulation will be driven to explore better behaviour for states visited by the robot agent, thus helping to accelerate learning on the robotic platform.

- **Benchmarking of MATL [Wulfmeier et al., 2017b].** We evaluate the method on a set of common reinforcement learning benchmark tasks [Duan et al., 2016] to transfer between simulations with different system dynamics. The experiments are extended to address additional relevant challenges for robotic platforms such as sparse and uninformative rewards. The final experiments investigate transfer between different simulation engines with unknown discrepancies as a stronger proxy for real robot experiments, extending the parametric differences from earlier experiments via different procedural implementations.

Mutual Alignment Transfer Learning as described in Chapter 5 has the following drawbacks, which can provide paths for future directions outlined in Section 6.2.3 :

- **Safety requirements**. MATL potentially applies partially trained policies on the real robot. In order to ensure safe rollouts, additional safety constraints are required.

- **Real world sample efficiency**. Even though increasing the real robot sample efficiency by utilising information from process in simulation, the approach in its currently implementation relies on model-free reinforcement learning (in particular TRPO) and therefore is constrained to simple tasks if applied on real robots.

6.2 Future Directions

Based on the presented body of work, promising directions for future work emerge for the goal of providing efficient supervision for robotic platforms. In addition to extensions building on imitation, adaptation, or transfer learning separately, there exists significant potential at the intersection of these fields.

6.2.1 Imitation Learning

- **Velocity and acceleration preferences.** While our existing work focuses on the spatial preferences underlying human driving behaviour, learning preferred velocity and acceleration profiles can result in a driving behaviour that is additionally optimised for passenger comfort. This line of research is related to the work of Kuderer et al. [2015], who learn spatial and temporal cost functions for driving manoeuvres including lane changes. While their work relies on a smaller number of hand-crafted features, the combination with MEDIRL can enable to exploit more raw input data formats. This extension will require the processing of sensor inputs across the full trajectory instead of focusing on a limited period around the start of each trajectory. The current convolutional neural network architecture can be adapted with recurrent modules as well as temporal convolutions to handle the additional temporal extension of input data.

- **Addressing dynamic obstacles.** MEDIRL focuses on generating cost functions for incorporation into the spatial module of path planning systems. In this context, cost functions are generated with respect to static environment aspects. To furthermore address dynamic obstacles in the environment, one future project is represented by additionally learning the driver's preference for specific trajectories and speed profiles in presence of dynamic obstacles. Similar to research on learning velocity profiles, this step requires the processing of multiple sensor inputs from different times which can lead to introducing recurrent modules as well as temporal convolutions. Related work can be found by Pfeiffer et al. [2017] who employ convolutional and recurrent architectures to predict pedestrian motion in dynamic environments.

- **Bayesian incorporation of priors from domain knowledge.** While the extension of MEDIRL incorporates expert domain knowledge in a pretraining step, future work can focus on treating manually crafted cost functions as Bayesian priors. Existing work on Bayesian IRL [Choi and Kim, 2013] limits the determination of higher level features to logical conjunctions of atomic features and is unable to efficiently employ larger datasets. However, existing work on Bayesian neural networks [Blundell et al., 2015, Gal and Ghahramani, 2016] paves the way for more efficient combinations of Bayesian inference and deep learning.

6.2.2 Domain Adaptation

- **Semi-supervised training.** Different approaches for incorporating small amounts of labelled data from the target domain can significantly improve performance [Tzeng et al., 2015a]. Further work is needed in order to minimise additional annotation efforts while maximising the efficiency of these labels. Early work in this field can build on the approach of Tzeng et al. [2015a], who simply integrate small amounts of labelled data for training based on a task's supervised objective.

- **Different density estimators.** Unsupervised domain adaptation via forms of density estimation and subsequent alignment will not only benefit from advances regarding adversarial framework [Arjovsky et al., 2017, Arjovsky and Bottou, 2017], but also different density estimation techniques such as the Maximum Mean Discrepancy criterion as applied by Long et al. [2017]. While these techniques are currently being outperformed by adversarial approaches, they provide a promising avenue for further research.

6.2.3 Simulation-to-Robot Transfer Learning

- **Learning accurate simulation parameters.** Simulations present a great opportunity for robot learning based on execution speed and safety. Much of existing work for porting behaviour learned in simulation aims at adapting the behaviour or rendering policies invariant with respect to the application domain. An exciting new direction is given by automatic adaptation of simulation parameters. Instead of building on large datasets for learning flexible dynamics models based on neural networks [Christiano et al., 2016], Louppe and Cranmer [2017] develop an approach building on variational optimisation of adversarial objectives to adapt a small set of parameters for non-differential simulations. New directions can build on further exploiting the reparametrisation trick [Sutton and Barto, 1998] to optimise distributions over simulator parameters.

- **Randomisation of environments: relevant parameters for dynamics and appearance.** Randomisation of training environments [Tobin et al., 2017, Todorov et al., 2012] is an approach for optimising domain invariant policies, aiming for deployment on the real platform without performance loss. The approach removes the effort of training on the real world by randomising all factors regarding visual and dynamic differences between real world and simulation. However, the complexity of real world applications can result in the randomisation of large numbers of parameters and therefore render the approach practically infeasible. To address these constraints future work

may aim at identifying only the most relevant aspects for a task and focus computational resources on randomising a small set of influential parameters.

- **Addressing real world uncertainty and safety requirements.** Approaches similar to MATL will employ partially trained policies on the real platform. Further development will have to address methods for safe application of RL on real world platforms [Achiam et al., 2017, Kahn et al., 2017a].

6.2.4 Combined Approaches

- **Learning realistic behaviour for multi agent simulations.** Chapters 4 and 5 focus on differences between environments based on visual aspects and system dynamics respectively. Multi agent simulations additionally have to approximate the behaviour and interaction with simulated participants in our scenarios. Work on policy imitation as well as inverse reinforcement learning can be applied to create more exact models to generate better simulations for interactive situations.

Appendices

A

Appendix A

Contents

A.1 Maximum Entropy Deep Inverse Reinforcement Learning

The following sections include additional details about the *Objectworld* and *Binaryworld* benchmarks applied in Chapter 3 for the evaluation of inverse reinforcement learning methods. We furthermore include qualitative visualisations of the learned reward functions.

A.1.1 Objectworld Benchmark

The *Objectworld* scenario [Levine et al., 2011] consists of a map of $M \times M$ states for $M = 32$ where possible actions include motions in all four directions as well as staying in place. Two different sets of state features are implemented based on randomly placed colours to evaluate the algorithms. For the continuous set

$x \in \mathbb{R}^C$ each feature dimension describes the minimum distance to an object of one of C colours. Building on the continuous representation, the discrete set includes $C \times M$ binary features, where each dimension indicates whether an object of a given colour is closer than a threshold of $d \in \{1, ..., M\}$.

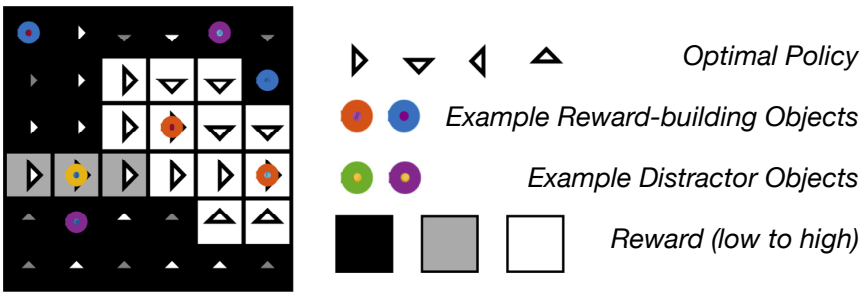


Figure A.1: Objectworld benchmark. The true reward is displayed by the brightness of each cell and based on the surrounding object configuration. Only a subset of colours influences the reward, while the others serve as distracting features.

The reward is positive for cells which are both within the distance 3 of colour 1 and distance 2 of colour 2, negative if only within distance 3 of colour 1 and zero otherwise. This is illustrated for a small subset of the state space in Figure A.1. The scenario is kept in line with common benchmarking procedures in the related literature [Choi and Kim, 2013, Levine et al., 2011].

While the original MaxEnt is unable to capture the nonlinear reward structure well, both MEDIRL and GPIRL provide significantly better approximations as represented in Figure A.2.

A.1.2 Binaryworld Benchmark

In order to test the ability of all methods to successfully approximate more complex, highly-varying reward structures, the *Binaryworld* benchmark is introduced in Wulfmeier et al. [2015]. This test scenario is similar to *Objectworld*, but in this problem every state is randomly assigned one of two colours (blue or red). The feature vector for each state consequently consists of a binary vector of length 9, encoding the colour of each cell in its 3x3 neighbourhood. The true reward

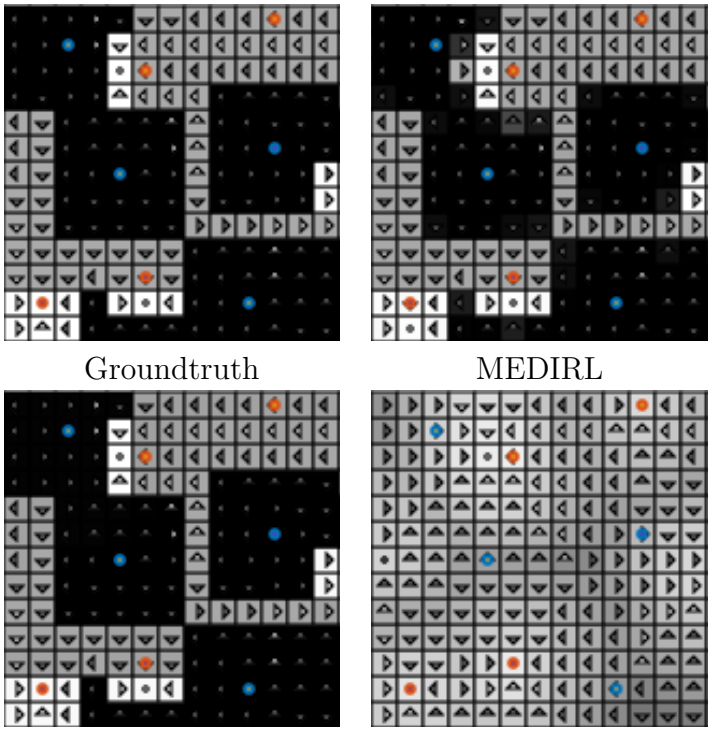


Figure A.2: Reward reconstruction sample in Objectworld benchmark provided $N = 64$ examples and $C = 2$ colours with continuous features. White - high reward; black - low reward. The arrows represent the optimal policy for the estimated reward function.

structure for a particular state is fully determined by the *number* of blue states in its local neighbourhood. It is positive if exactly four out of nine neighbouring states are blue, negative if exactly five are blue and zero otherwise. The main difference compared to the *Objectworld* scenario is that a single feature value does not carry much weight, but rather that higher-order relationships amongst the features determine the reward.

Figure A.3 shows the reconstruction of the reward structures estimated by MEDIRL, MaxEnt and GPIRL. While GPIRL was able to reconstruct the correct reward for some of the states having features it has encountered before, it provides inaccurate rewards for states which were never encountered. It produces an overall too smooth reward function due to assumptions and priors in the GP approximation. On the other hand, MEDIRL is able to reconstruct it with high accuracy

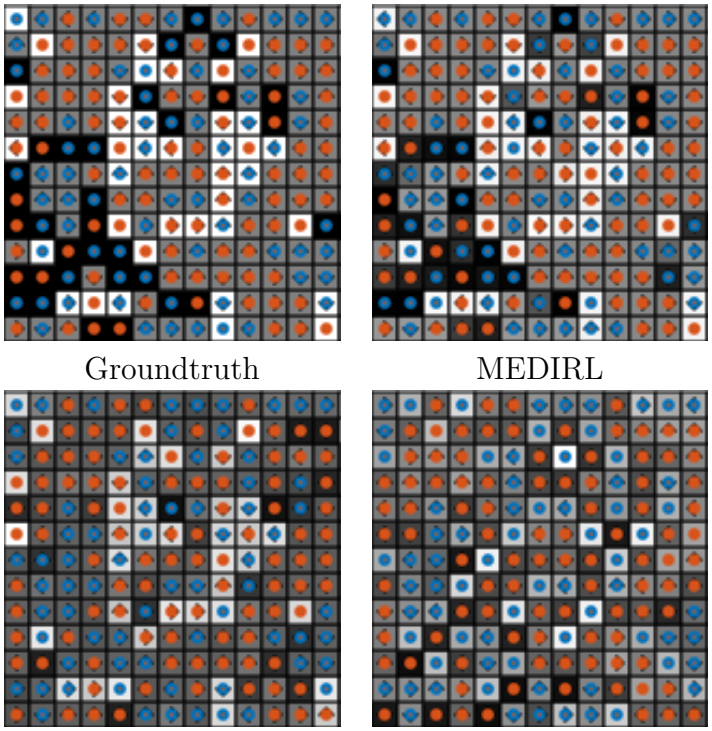


Figure A.3: Reward reconstruction sample for the *Binaryworld* benchmark provided $N = 128$ demonstrations. White - high reward; black - low reward. The arrows represent the optimal policy for the estimated reward function.

demonstrating the ability to effectively learn the highly-varying structure of the underlying function.

For more details on the performance of all algorithms please consult the experiments in Section 3.2.

B

Appendix B

Contents

B.1 Mutual Alignment Transfer Learning

The following sections provide additional details regarding the implementation and experiments for Chapter 5. In particular, the next section focuses on a variation of the MATL algorithm based on the Wasserstein GAN (WGAN) [Arjovsky et al., 2017].

B.1.1 Wasserstein Distance for Auxiliary Rewards

The Wasserstein distance adaptation of the Mutual Transfer Alignment framework builds on Equations 5.13, 5.17 and 5.15 in Chapter 5. The discriminator is replaced by the discriminator formulation from WGAN leading to the following adaptation of Equation 2.

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}_{\pi_\theta}[D_\omega(\zeta_t)] + \mathbb{E}_{\pi_\phi}[-D_\omega(\zeta_t)] \tag{B.1}$$

$$\text{with } \zeta_t = s_t, s_{t+k}, s_{t+2k}, ..., s_{t+nk} \quad \text{with } n \in \mathbb{N}_0; k \in \mathbb{N}$$

The discriminator training step is followed by weight clipping with magnitude 0.01 as described in Arjovsky et al. [2017]. Furthermore, the equations for the auxiliary alignment rewards are adapted similarly to confuse the WGAN formulation of the discriminator and optimise both distributions towards alignment.

$$\kappa_S(s_t) = -D_\omega(\zeta_t) \tag{B.2}$$

$$\kappa_R(s_t) = D_\omega(\zeta_t) \tag{B.3}$$

The WGAN formulation was found to perform more stable than the confusion loss formulation of MATL in Equations 5.13, 5.17 and 5.15 for the experiments in Section 5.2.4. All other experiments in Chapter 5 rely on the equations within the chapter as the Wasserstein variant did result in reduced performance.

B.1.2 Experiment Parameters

All parameters underlying the experiments on Mutual Alignment Transfer Learning can be found in the following table with videos available online under `https://sites.google.com/view/matl/` for a subset of the experiments including the Cartpole Swingup and Hopper tasks respectively described in Section 5.2.1 and 5.2.2.

Experiment	Alignment Weight	Discriminator	Batch Size	Discriminator Learningrate	TRPO Stepsize	Discriminator Runs	Simulator Runs	Max. Trajectory Length	Number of Iterations	Change in system dynamics
Sparse Rewards										
Cartpole	0,1	Confusion	10k	0,01	0,01	5	10	300	50	Double Density + Double Friction
Cartpole Swingup	0,1	Confusion	10k	0,01	0,01	5	10	300	200	Double Density + Double Friction
Reacher2D	0,1	Confusion	10k	0,01	0,01	10	10	200	500	Increased Armature + Dampening
Uninformative Rewards										
Hopper2D	2,5	Confusion	50k	0,01	0,01	10	10	1000	100	Double Density
Hopper2D - Alignment Weight	0,1 to 10	Confusion	50k	0,01	0,01	10	10	1000	100	Double Density
No Robot Env Rewards										
Cartpole Swingup	0,1	Confusion	10k	0,01	0,01	10	10	300	200	Double Density + Double Friction
Hopper2D	0,1	Confusion	50k	0,01	0,01	10	10	1000	100	Double Density
Reacher2D	0,1	Confusion	10k	0,01	0,01	10	10	200	500	Increased Armature + Dampening
MuJoCo to DART										
Reacher2D	0,1	Wasserstein	10k	0,001	0,01	20	10	200	500	MuJoCo-DART
Hopper2D	1	Wasserstein	50k	0,001	0,01	20	10	1000	200	MuJoCo-DART
Additional parameters across all tasks			**Additional information**							
- Discriminator Network (64,64)			- The variations of friction, density and armature differences were chosen to limit direct transfer performance.							
- Policy Network (32,32)			- Side note: Parameters have to be significantly different for simple tasks to have an effect on direct transfer performance							
- Discriminator Minibatches 1 (full batch/subbatches tested w/o significant improvement)										

Bibliography

Dartenv: Openai gym environments transferred to the dart simulator. `https://github.com/DartEnv/dart-env/wiki/OpenAI-Gym-Environments`, 2017.

P. Abbeel, Dmitri Dolgov, A.Y. Ng, and S. Thrun. Apprenticeship learning for motion planning with application to parking lot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pages 1083–1090, September 2008. doi: 10.1109/IROS.2008.4651222.

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

Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8. ACM, 2006.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. 2017.

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

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

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, volume 2016, 2017.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. 2017.

Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

Dan Barnes, Will Maddern, and Ingmar Posner. Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 203–210, 2017.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards AI. *Large-scale kernel machines*, 34(5), 2007.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 95–104, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

Jaedeug Choi and Kee-Eung Kim. Bayesian nonparametric feature construction for inverse reinforcement learning. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1287–1293. AAAI Press, 2013.

Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Winston Churchill and Paul Newman. Continually improving large scale long term visual navigation of a vehicle in dynamic urban environments. In *15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1371–1376. IEEE, 2012.

Winston Churchill and Paul Newman. Experience-based Navigation for Long-term Localisation. *The International Journal of Robotics Research (IJRR)*, 2013.

Peter Corke, Rohan Paul, Winston Churchill, and Paul Newman. Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *Intelligent Robots and Systems*, pages 2085–2092. IEEE, 2013.

Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 7. ACM, 2015.

Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12: 2121–2159, 2011.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. 2016.

Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, Urun Dogan, Marius Kloft, Francesco Orabona, and Tatiana Tommasi. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL `http://arxiv.org/abs/1701.00160`.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, pages 3909–3917, 2016.

Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *arXiv preprint arXiv:1705.10479*, 2017.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL `http://arxiv.org/abs/1207.0580`.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

Sungeun Hong, Woobin Im, Jongbin Ryu, and Hyun S Yang. Sspp-dan: Deep domain adaptation network for face recognition with single sample per person. *arXiv preprint arXiv:1702.04069*, 2017.

Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning? *CoRR*, abs/1608.08614, 2016. URL `http://arxiv.org/abs/1608.08614`.

E. T. Jaynes. Information theory and statistical mechanics. ii. *Phys. Rev.*, 108:171–190, Oct 1957a. doi: 10.1103/PhysRev.108.171. URL `https://link.aps.org/doi/10.1103/PhysRev.108.171`.

Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4): 620, 1957b.

Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017a.

Gregory Kahn, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Plato: Policy learning using adaptive trajectory optimization. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3342–3349. IEEE, 2017b.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

Konstantinos Kamnitsas, Christian F. Baumgartner, Christian Ledig, Virginia F. J. Newcombe, Joanna P. Simpson, Andrew D. Kane, David K. Menon, Aditya V. Nori, Antonio Criminisi, Daniel Rueckert, and Ben Glocker. Unsupervised domain adaptation in brain lesion segmentation with adversarial networks. In *IPMI*, 2017.

Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. Activity Forecasting. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision ECCV 2012*, number 7575 in Lecture Notes in Computer Science, pages 201–214. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33764-2 978-3-642-33765-9. URL `http://link.springer.com/chapter/10.1007/978-3-642-33765-9_15`.

Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, page 0278364915619772, 2016.

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

Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646. IEEE, 2015.

Kevin Lai and Dieter Fox. 3d laser scan classification using web data and domain adaptation. In *Robotics: Science and Systems*, volume 2, 2009.

Michael Laskey, Sam Staszak, Wesley Yu-Shu Hsieh, Jeffrey Mahler, Florian T Pokorny, Anca D Dragan, and Ken Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 462–469. IEEE, 2016.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, pages 143–156, 2017.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

C. Karen Liu, Mike Stilman, Siddhartha Srinivasa, Lee Jeongseok, and Michael Grey. Dart: Dynamic animation and robotics toolkit. `https://github.com/dartsim/dart/`, 2017.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

G. Louppe and K. Cranmer. Adversarial Variational Optimization of Non-Differentiable Simulators. *ArXiv e-prints*, 2017.

Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China*, volume 2, page 3, 2014.

Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. doi: 10.1177/0278364916679498. URL http://dx.doi.org/10.1177/0278364916679498.

Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P. Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1563, 2017.

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

Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, 25(9):569–597, September 2008. ISSN 1556-4959. doi: 10.1002/rob.v25:9. URL http://dx.doi.org/10.1002/rob.v25:9.

Hans Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746, 2006.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Peer Neubert, Niko Sunderhauf, and Peter Protzel. Appearance change prediction for long-term navigation across seasons. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 198–203. IEEE, 2013.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann Publishers Inc., 2000.

NVIDIA. Nvidia isaac: Virtual simulator for robots. https://www.nvidia.com/en-us/deep-learning-ai/industries/robotics/, 2017.

Michael Paton, Kirk MacTavish, Chris J Ostafew, and Timothy D Barfoot. It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1519–1526. IEEE, 2015.

M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena. A Data-driven Model for Interaction-aware Pedestrian Motion Prediction in Object Cluttered Environments. *ArXiv e-prints*, 2017.

Mark Pfeiffer, Ulrich Schwesinger, Hannes Sommer, Enric Galceran, and Roland Siegwart. Predicting Actions to Act Predictably: Cooperative Partial Motion Planning with Maximum Entropy Models . In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Dean A Pomerleau. Neural network based autonomous navigation. In *Vision and Navigation*, pages 83–93. Springer, 1990.

Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51:61801, 2007.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

Nathan D. Ratliff, David Silver, and J. Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009. URL `http://link.springer.com/article/10.1007/s10514-009-9121-3`.

Sivalogeswaran Ratnasingam and Steve Collins. Study of the photodetector characteristics of a camera for color constancy in natural scenes. *JOSA A*, 27(2): 286–294, 2010.

Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artifical Intelligence and Statistics*, 2010.

Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *arXiv preprint arXiv:1603.06432*, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103. ACM, 1998.

Andrei A. Rusu, Matej Vecerik, and year=2017 Thomas Rothörl and Nicolas Heess and Razvan Pascanu and Raia Hadsell, booktitle=CoRL. Sim-to-real robot learning from pixels with progressive nets.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

Pierre Sermanet, Raia Hadsell Marco Scoffier, Matt Grimes, Jan Ben, Ayse Erkan, Chris Crudele, Urs Muller, and Yann Lecun. A multi-range architecture for collision-free off-road robot navigation. *Journal of Field Robotics*, 2010.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL `https://arxiv.org/abs/1705.05065`.

S. Shalev-Shwartz and A. Shashua. On the sample complexity of end-to-end training vs. semantic abstraction training. *ArXiv e-prints*, 2016.

Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. Rapidly exploring learning trees. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1541–1548. IEEE, 2017.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251, 2017.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, oct 2017. ISSN 0028-0836. doi: 10.1038/nature24270. URL `http://www.nature.com/doifinder/10.1038/nature24270`.

Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. In *Proceedings of the 33rd International Conference on Learning Representations (ICLR)*, 2017.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440, 2008.

Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, 2017.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9): 661–692, 2006. ISSN 1556-4967. doi: 10.1002/rob.20147. URL `http://dx.doi.org/10.1002/rob.20147`.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Xingchao Peng, Sergey Levine, Kate Saenko, and Trevor Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111*, 2015a.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015b.

Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. *Autonomous Driving in Urban Environments: Boss and the Urban Challenge*, pages 1–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-03991-1. doi: 10.1007/978-3-642-03991-1_1. URL `http://dx.doi.org/10.1007/978-3-642-03991-1_1`.

Harm Van Seijen, Mehdi Fatemi, Romain Laroche, Joshua Romoff, Tavian Barnes, and Jeffrey Tsang. Hybrid reward architecture for reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5398–5408. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7123-hybrid-reward-architecture-for-reinforcement-learning.pdf`.

A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. In *Neural Information Processing Systems Conference, Deep Reinforcement Learning Workshop*, volume abs/1507.04888, 2015.

Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch this: scalable cost-function learning for path planning in urban environments . In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, oct 2016a.

Markus Wulfmeier, Dushyant Rao, and Ingmar Posner. Incorporating human domain knowledge into large scale cost function learning. In *Neural Information Processing Systems Conference, Deep Reinforcement Learning Workshop*, volume abs/1612.04318, 2016b.

Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Addressing appearance change in outdoor robotics with adversarial domain adaptation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2017a.

Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. Mutual alignment transfer learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2017b.

Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10): 1073–1087, 2017c. doi: 10.1177/0278364917722396. URL `https://doi.org/10.1177/0278364917722396`.

Ya-Xiang Yuan. A review of trust region algorithms for optimization, 1999.

Shao Zhifei and Er Meng Joo. A review of inverse reinforcement learning theory and recent advances. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.