Universitat Autònoma de Barcelona

On Building End-to-End Driving Models Through Imitation Learning

Felipe Codevilla

Universitat Autònoma
de Barcelona

On Building End-to-End Driving Models Through Imitation Learning

A dissertation submitted by **Felipe Codevilla** at Universitat Autònoma de Barcelonato fulfil the degree of **Doctor of Philosophy**.

Bellaterra, March 28, 2019

Director	**Dr. Antonio López Peña** Dept. Ciències de la computació & Centre de Visió per Computador Universitat Autònoma de Barcelona (UAB)
Thesis committee	**Dr. Francesc Moreno Noguer** Institut de Robòtica i Informàtica Industrial Universitat Politècnica de Catalunya (UPC) **Dr. Joan Serrat Gual** Dept. Ciències de la computació & Centre de Visió per Computador Universitat Autònoma de Barcelona (UAB) **Dr. Andreas Geiger** Computer Science Department University of Tübingen MPI for Intelligent Systems
International evaluators	**Dr. Adrien Gaidon** Toyota Research Institute Los Altos, USA **Dr. David Vázquez** Element AI Montreal, Canada

This document was typeset by the author using LaTeX 2$_\varepsilon$.

ISBN: 978-84-945373-1-8

Printed by Ediciones Gráficas Rey, S.L.

It is change, continuing change, inevitable change, that is the dominant factor in society today.

No sensible decision can be made any longer without taking into account not only the world as it is, but the world as it will be.

Isaac Asimov

To the CARLA team /c\...

Acknowledgements

The universe we live in is highly unpredictable. It is almost impossible to predict the exact effects that your own actions, or the actions of the ones around you, will have in our future. When looking to the past, however, you can see some things that were crucial to achieve certain objectives. If my father was not insistent for me to study the English language, I would not probably not be writing this now. If intellectual dedication were not examples at home, I also would not be here. This cycle continues.

I own a lot to people along my career that helped me to get into a PhD program in the first place. Without all the influence and great example from my masters and undergrad advisor, Silvia Botelho I maybe would not have started a PhD in the first place.

After the PhD started I own some deep thanks, first, to my advisor, Antonio Lopez, that is not only a great reference scientifically but also provided great support. He is not only great to work with but also a great human being. I have to thank a lot to German Ros for the initial ideas for the thesis, without his help, initially, and during this PhD work, for sure this thesis would be different, and probably quite worse.

I own a very special thanks to Alexey Dosovitskiy. I learned a lot from him. I could say he is the unofficial co-supervisor that really made this thesis to happen. His way of work really inspired me and I believe made me a better researcher.

However, this thesis is dedicated to the CARLA team who were not only co-workers but also friends that built this amazing simulator that allowed me to do my research. Special thanks to the amazing lead programmer Nestor Subiron and his second in command Marc Puig. They made an amazing code and also taught me how to be a much better coder by following their instructions and examples. I also own a lot to the other members, Mario, Iris, Fran Dos, Xavi and Xisco. Special thanks to Gabriel Villalonga for the amazing discussions during the PhD.

Many thanks to my fellow PhD student friends that gave me support and mutual understanding during this PhD, Bojana, Gemma, Dena, Francesco, Manu, Onur, Cesar, Carlos, Edgar and others, without your support things would be definitely harder. Special thanks to Igor Peric and our productive off-site co-workings.

Finally, I want also thank the people from TRI, Adrien Gaidon and Eder Santana that allowed and helped me to do an amazing closing project project for this thesis.

Exponho aqui, em português, agradecimentos para pessoas muito próximas e especiais em minha vida. Sem os sempre sábios e práticos conselhos da minha irmã, Tati, eu provavelmente teria cometido algum erro muito óbvio e me perdido por aí.

Obviamente o apoio, mesmo a distância, dos meus pais também foi fundamental. Cada "web-mate" tomado foi indispensável ! Amo vocês. Agradeço também ao pessoal do grupo familia, Tia Lu, Stela e Julia, com suas maravilhosas visitas para passar muito trabalho na Espanha e na Itália!

Abstract

Autonomous vehicles are now considered as an assured asset in the future. Literally, all the relevant car-markers are now in a race to produce fully autonomous vehicles. These car-makers usually make use of modular pipelines for designing autonomous vehicles. This strategy decomposes the problem in a variety of tasks such as object detection and recognition, semantic and instance segmentation, depth estimation, SLAM and place recognition, as well as planning and control. Each module requires a separate set of expert algorithms, which are costly specially in the amount of human labor and necessity of data labelling. An alternative that recently has driven considerable interest is the end-to-end driving. In the end-to-end driving paradigm, perception and control are learned simultaneously using a deep network. These sensorimotor models are typically obtained by imitation learning from human demonstrations. The main advantage is that this approach can directly learn from large fleets of human-driven vehicles without requiring a fixed ontology and extensive amounts of labeling. However, scaling end-to-end driving methods to behaviors more complex than simple lane keeping or lead vehicle following remains an open problem. On this thesis, in order to achieve more complex behaviours, we address some issues when creating end-to-end driving system through imitation learning. The first of them is a necessity of an environment for algorithm evaluation and collection of driving demonstrations. On this matter, we participated on the creation of the CARLA simulator, an open source platform built from ground up for autonomous driving validation and prototyping. Since the end-to-end approach is purely reactive, there is also the necessity to provide an interface with a global planning system. With this, we propose the conditional imitation learning that conditions the actions produced into some high level command. Evaluation is also a concern and is commonly performed by comparing the end-to-end network output to some pre-collected driving dataset. We show that this is surprisingly weakly correlated to the actual driving and propose strategies on how to better acquire data and a better comparison strategy. Finally, we confirm well-known generalization issues (due to dataset bias and overfitting), new ones (due to dynamic objects and the lack of a causal model), and training instability; problems requiring further research before end-to-end driving through imitation can scale to real-world driving.

Key words: *autonomous driving, computer vision, machine learning, deep learning, imitation learning, simulation*

Resumen

Los vehículos autónomos ahora se consideran como un activo asegurado en el futuro. Literalmente, todos los marcadores de automóviles relevantes se encuentran ahora en una carrera para producir vehículos totalmente autónomos. Estos fabricantes de automóviles generalmente utilizan tuberías modulares para diseñar vehículos autónomos. Esta estrategia descompone el problema en una variedad de tareas tales como detección y reconocimiento de objetos, segmentación semántica y de instancias, estimación de profundidad, SLAM y reconocimiento de lugares, así como planificación y control. Cada módulo requiere un conjunto separado de algoritmos expertos, que son costosos, especialmente por la cantidad de mano de obra humana y la necesidad de etiquetar los datos. Una alternativa que recientemente ha despertado un interés considerable es la conducción *end-to-end*. En el paradigma de conducción de *end-to-end*, la percepción y el control se aprenden simultáneamente utilizando una red profunda. Estos modelos sensoriomotores se obtienen típicamente mediante imitación de aprendizaje a partir de demostraciones de humanos. La principal ventaja es que este enfoque puede aprender directamente de grandes flotas de vehículos conducidos por el hombre sin requerir una ontología fija y una gran cantidad de datos etiquetados. Sin embargo, los métodos *end-to-end* se usaban comúnmente para aprender comportamientos simples como el mantenimiento de carriles y el seguimiento del vehículo. En esta tesis, para lograr comportamientos más complejos, abordamos algunos problemas al crear un sistema de conducción de extremo a extremo a través del aprendizaje por imitación. El primero de ellos es la necesidad de un entorno para la evaluación de algoritmos y la recopilación de demostraciones de conducción. En este asunto, participamos en la creación del simulador de CARLA, una plataforma de código abierto creada desde cero para la validación y creación de prototipos de conducción autónoma. Dado que el enfoque de *end-to-end* es puramente reactivo, también existe la necesidad de proporcionar una interfaz con un sistema de planificación global. Con esto, proponemos el aprendizaje de imitación condicional que condiciona las acciones producidas en un comando de alto nivel. La evaluación también es una preocupación y se realiza comúnmente al comparar la salida de la red de extremo a extremo con un conjunto de datos de conducción recolectados de forma previa. Demostramos que esto está sorprendentemente débilmente relacionado con la conducción real y proponemos estrategias sobre cómo adquirir mejor los datos y una mejor estrategia de comparación. Finalmente, confirmamos los problemas de generalización conocidos (debido al sesgo de sobreajuste del conjunto de datos y el sobreajuste), los nuevos (debido a los objetos dinámicos y la falta de modelo

acausal) y la inestabilidad de la capacitación; Los problemas que requieren investigación adicional antes de la finalización de la conducción a través de la imitación pueden escalar a la conducción en el mundo real.

Palabras clave: *vehículos autónomos, visión artificial, aprendizaje automático*

Resum

Els vehicles autònoms es consideren ara com a actius assegurats en el futur. Literalment, tots els marcadors d'automòbils rellevants es troben en una cursa per produir vehicles totalment autònoms. Aquests fabricants de cotxes solen fer ús de canonades modulars per al disseny de vehicles autònoms. Aquesta estratègia descompon el problema en diverses tasques com la detecció i el reconeixement d'objectes, la segmentació semàntica i la instància, l'estimació de profunditat, el reconeixement de llocs i SLAM, així com la planificació i el control. Cada mòdul requereix un conjunt separat d'algoritmes experts, que són costosos especialment quant al treball humà i la necessitat d'etiquetatge de dades. Una alternativa que recentment té un interès significatiu és la conducció integral. En el paradigma de conducció de extrem a extrem, la percepció i el control s'obtenen simultàniament mitjançant una xarxa profunda. Els models de tesisensorotor s'obtenen normalment mitjançant l'aprenentatge de imitacions de les demostracions de humà. L'avantatge principal és que aquest enfocament pot aprendre directament de les grans flotes de vehicles dirigits per humans sense necessitat d'un ontologia fixa i d'una àmplia quantitat d'etiquetatge. No obstant això, els mètodes de extrem a extrem es van utilitzar habitualment per aprendre conductes simples com ara manteniment de carrils i el vehicle principal. En aquesta tesi, per tal d'aconseguir comportaments més complexos, abordem alguns problemes quan es crea un sistema de conducció de extrem a extrem mitjançant l'aprenentatge de la imitació. El primer d'aquests és la necessitat d'un entorn per a l'avaluació d'algorismes i la recopilació de demostracions d'administració. En aquest sentit, hem participat en la creació del simulador de Carla, una plataforma de codi obert construïda des de la base per a la validació i el prototipatge d'autònoms. Atès que l'enfocament de extrem a extrem és purament re-actiu, també hi ha la necessitat de proporcionar una interfície amb un sistema de planificació global. Amb això, proposem l'aprenentatge d'imitació condicional que condiciona les accions produïdes en algun comandament d'alt nivell. L'avaluació és també una qüestió i normalment es fa mitjançant la comparació de la sortida de la xarxa de cap a cap a un conjunt de dades de conducció que es recull. Demostrem que això és correlacionat sorprenentment debilitat amb la conducció real i proposa estratègies sobre com adquirir millor les dades i una estratègia de comparació millor. Finalment, confirmem problemes de generalització ben coneguts (deguts a biaixos i sobraccessos actuals), de nous (a causa d'objectes dinàmics i la manca de model acausal) i la inestabilitat de la formació; Els problemes que requereixen més investigacions abans de finalitzar la conducció a través de la imitació poden escalar a la conducció del món real.

Paraules clau: *vehicles autònoms, visió artificial, aprenentatge automàtic*

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Robotic cars that are able to drive themselves are an asset imagined in the literature since cars have became a central good in our society. From just imagination, this idea of having Autonomous Vehicles (AVs) is now considered as inevitable to happen at some point in the future. There are several reasons to build an autonomous vehicle. One of them is safety. The world health organization estimates that around 1.35 million fatalities are caused by road vehicle crashes [81]. From all these fatalities at least 94% [4] of them are caused by human errors, such as distraction or inaccurate decision making. As is currently estimated, autonomous vehicles could reduce the number of fatalities by 80% [49][68] . However, the motivation that attracts now, literally, every relevant automaker to the AVs development market is economical. Autonomous Vehicles are estimated to inject around 7 trillion US dollars into the world economy by 2050 [59]. By removing the driver from the equation, companies can further profit by proving mobility as a service, in a cheaper way, as Uber or Lift does today.

What is the precise definition of an autonomous vehicle ? There is no standard single definition of an AV since it depends on what can the vehicle perform and on which context. The NHTSA, since 2016, gives an updated definition that divides an AV based on five levels of autonomy:

- Level 0: There is not automation involved, so the vehicle is totally controlled by a human driver.
- Level 1: The human driver literally is still controlling the vehicle but some assist features can provide help to the driver.
- Level 2: The vehicle has some automated functions and is able to control steering and acceleration eventually. However, the driver must remain engaged in full attention.

- Level 3: The driver is necessary but does not need full attention. The vehicle is usually on control and warns the driver when it needs assistance.
- Level 4: Under certain conditions the vehicle can take full control without need of the human driver.
- Level 5: The vehicle does not need a steering wheel. This level represents the full autonomy.

At the time this thesis is being written some autonomous vehicles, that could arguably be placed on different levels of autonomy, are being commercialized. We can already have samples of commercial cars that incorporate features from levels 2 or 3. However, the holly grail of autonomy that could, in theory, fully provide both the economical and safety benefits are the levels 4 and 5. For that, several strategies are being developed, and these, including the one of interest of this thesis, are going to be detailed on the next sections.

1.2 The Modular Autonomous Vehicle

Building a level 5 autonomous driving agent is really a multi-disciplinary and complex task. It encapsulates many areas of knowledge such as sensor development, computer vision, machine learning, control theory, planning and even ethics. A fully autonomous vehicle needs to pick up a passenger, anywhere, and drive this person to a desired destination by travelling over any city and respecting all the traffic rules. To do that safely, it also needs to react accordingly to all troublesome situations that can eventually happen, such as careless drivers or sudden crossings of pedestrians. Further, everything needs to be performed under any weather or road conditions.

Traditionally, as shown on Figure 1.1, several modules are used for the purpose of driving. The driving agent needs to be able to *understand the scene* as it drives through the environment. For that, it needs to detect all the moving objects [69, 86, 87] and recognize them [126], as well as segment the road and other parts of the scene [70, 80, 117, 122]. The AV also needs to *localize itself* on the road (GPS is not precise enough) and create an adequate *plan* to reach its destination [65, 109, 121, 125]. Finally it needs to translate the plan and what it recognized into *control* [82], which uniquely determine the actions performed in the world. Each module requires a separate set of expert algorithms which are costly in terms of development effort, integration, testing, and adaptability in the constantly changing open world.

Recently, there has been huge advances in each of these specific modules, due to the development of deep learning allied with huge amounts of available data.

Cheaper sensors and huge investment allows companies to collect thousands miles of footage. This advance is specially significant on scene understanding, since in Computer Vision, for instance, the use of convolutional neural networks (CNNs) allied with a huge amount of data and less manual design of features has lead to super-human performance on image classification [116].

Hence, developing AV modules is changing from an engineering centered approach to a data centered one. On the engineered centered approach, each specific problem is carefully engineered. For example, to detect pedestrians, the traditional way would be to design features to represent a pedestrian and try to match those in an image by training a classifier and requiring little amount of training data [34]. The data centered approach, on the other hand, solves that by using hundreds of thousands of labelled examples of pedestrians. These labelled examples are a ground truth (GT) of some raw data (e.g. images) that is used as a supervision for the CNNs to learn.

However, not all the data that is collected can readily be used for improving the driving agent. One of the main drawbacks of the data centered approaches is the necessity for labelling. This usually can be done manually (e.g., assigning a semantic class to each pixel of an image) by using crowdsourcing tools such as Amazon's Mechanical Turk [15] and LabelMe [95]. However, this cumbersome task does not scale to the full length of the available data, since the annotators must have certain degree of expertise annotating data, and their annotations come with errors and inaccuracies.

Even though learning has been set as the main paradigm especially for perception, the decision making, planning and control are still mostly being made on an engineered based approach which does not allow to fully leverage the available data. Different techniques that could use all the available data without the burden of high dense quantity of labelling would be more scalable. These techniques could also be able to introduce the learning approach not only on perception and mapping but also in the decision making. The exploration of these alternatives is what we address on this thesis.

1.3 End-to-end Driving

Instead of learning each single component, we could think of a driving agent as a single CNN that could map directly what the vehicle perceives (e.g., from a camera) in terms of vehicle control parameters producing a target maneuver (Fig. 1.2). With this, all the sub-tasks are not defined, but may be implicitly learned from data or interaction. This approach, called end-to-end driving, was pioneered in 1988 by Pomerleau [61, 84] and has recently attracted renewed interest [11, 17, 20, 77, 115].

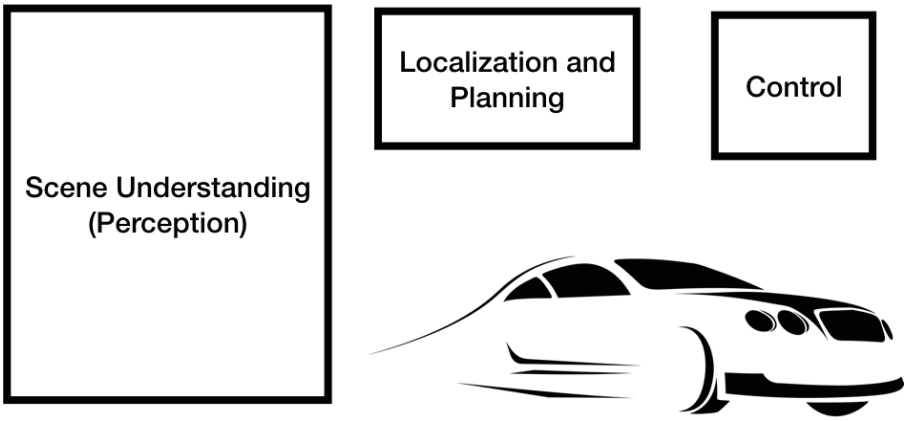


Figure 1.1 – Some key modules of an autonomous vehicle. The scene understanding provides information for the localization and planning algorithms to work. Finally, based on the plan the control outputs can be produced.

This interest was mainly due to the advances in deep learning and convolutional neural networks allied with a higher availability of annotated data and computing power.

There are several ways to train an end-to-end driving agent (i.e. a CNN). One way is to mimic the actions from expert human demonstrators [2, 3, 85, 104]. Given some sensor inputs (i.e. observations of the current state of the world) and desired control actions made by the human, the neural network is taught to produce the same outputs as the human. The process happen by the continuous backpropagation of the error between the controller's action and the desired human action. The idea is to minimize this propagated error through some optimization technique such as gradient descent [36]. This approach is commonly called imitation learning (IL).

Instead of directly imitating a human reference that could be not perfect, an alternative way with great success in many fields is reinforcement learning (RL) [105] [76]. In this approach, the driving agent would interact with the environment and receive rewards when it drives according to all traffic and safety rules and is penalized for any mistake. This way, the CNN-based agent would made a minimum number of mistakes and try to perform in a way that would maximize its rewards.

However, RL poses some complications when learning within physical environments since it can generate unsafe situations before convergence. Also, it typically requires millions of trial and error runs in the target environment [64] or a faithful simulation. End-to-end imitative systems, on the contrary, allows for safe off-line learning and can directly rely on data collected from large fleets of human-driven

vehicles without requiring a fixed ontology and extensive amounts of sensor data labeling. It is also a simple and efficient system, able to run in real-time on embedded hardware [11], a key requirement for online safety-critical decision making in robotic platforms. Yet, RL can be indeed complementary through imitation learning as RL can be used to improve an initial imitative policy [105].

Imitation learning essentially is performed by training a regressor to replicate the driving policy (behavior) of a referential demonstrator. However, it is erroneous to assume this as a simple classification or regression problem, since every action performed does affect the future state of the world. Thus, there will be always shifts on what is seen at training and at test time [92]. When human drivers collect reference data they rarely demonstrate "recovery behavior" for potential mistakes. Thus, since the learning process is imperfect, as soon as the agent makes a mistake it will not have any reference from the demonstrator on how to recover and will continue to make further mistakes until failure. There are several ways to solve this problem. Usually they require querying an expert when the agent is driving [93][42] [63]. With this the agent could benefit from expert information to further learn how to correct mistakes. This approaches have the expensive requirement of needing an expert during the learning process. The simpler strategy, that does not need to use information from when the model is driving is commonly called *behavior cloning*, a sub-part of imitation learning, and has been shown to have good success recently [11] [20][115]. The key is to use extra subterfuges to force the human driver to provide "recovery behavior" data, either by using extra sensors [11] or by adding noise to the demonstrator. From now on when we refer to imitation learning (IL) , we actually refer to *behavior cloning* that does not need to query the expert when the agent is driving.

In this thesis we focus on end-to-end driving behavior cloning and how to make it more advantageous, either by increasing the controllability (integrate it with higher level systems), by improving the understanding of its limitations or by providing tools and mechanisms to better evaluate such systems. We choose to use behavior cloning since we aim to propose techniques that can easily benefit from the huge load of data that can be collected by human drivers without any additional human labor for labelling or the presence of humans during learning.

1.4 Objectives

Even though it was pioneered a while ago [61, 84], end-to-end driving through behavior cloning still lack development on some critical topics. In this thesis we focus on three issues that we found as the main barriers: *Simulation*, *Controllability* and *Evaluation*. By addressing these three issues we put our final efforts on exploring

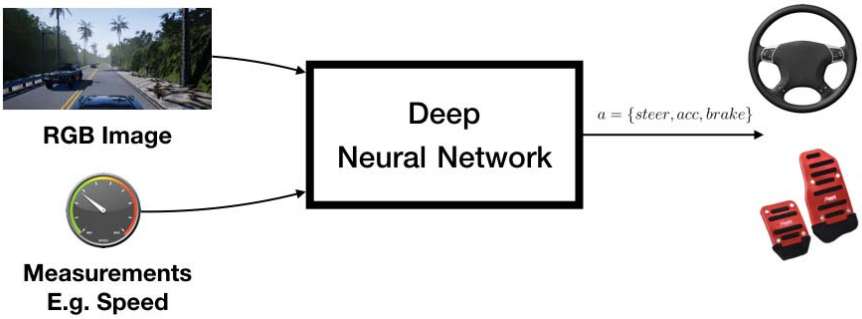


Figure 1.2 – End-to-end driving scheme. The sensor input composed by sensor data (images, speed) is directly mapped by a deep neural network into car controls (throttle, steer, brake).

the limitations of current end-to-end driving agents based on behavior cloning..

Simulation *Where can we develop end-to-end driving agents?*

End-to-end driving requires environments where the actual driving interaction can happen. Relying on physical environments during all the development phase is expensive and time consuming, making it hard for quick prototyping and testing. Further, initial versions of the driving agents can be potentially unsafe.

This could be addressed by using simulation. However, when we started this PhD work we realized the lack of simulated environments with the required conditions to study the end-to-end driving algorithms . For this reason, we helped significantly to the development of the CARLA simulator (CARs Learning to Act; carla.org). CARLA is an open source simulator made from ground up to be used for autonomous driving validation and prototyping. The simulator allows quick testing of end-to-end driving agents and to have access to privileged information such as image labels ground truth. Further, it allows the creation of repeatable benchmarks, that are fundamental to assess the driving capabilities of different agents.

Controllability *How do we control end-to-end driving agents ?*

The end-to-end agents as shown in Figure 1.2, are purely reactive, without the ability to incorporate any route plan. Thus, these systems, when approaching a decision point, such as an intersection, will have no route context in order to make a decision. For example, the CNN trained by Bojarski et al. [11] was given control over lane and road following only. When a lane change or a turn from one road to another were required, the human driver had to take control. The lack of possibility

to guide the driving agent to a certain objective limits the kind of task it can perform, specially on urban environments. For instance, when driving in a city, one cannot evaluate if a given system is able to reach a certain objective.

To make the end-to-end agents controllable, in this PhD work we focus on proposing a new formulation where the control produced is conditioned on high level commands. We call this formulation as conditional imitation learning. This allows, at test time, for the network to resolve the ambiguity in the perceptuomotor mapping and allow the trained model to be controlled by the input provided by a passenger or a topological planner.

Evaluation *How do we evaluate end-to-end driving agents ?*

In order to find the best training parameters and strategies for driving agents based on imitation learning, it is necessary to have a quick and adequate evaluation protocol. In this context, there is a tendency to evaluate end-to-end vision-based driving systems in similar way as in a generic computer vision task. This idea is appealing, since camera-based autonomous driving can be viewed as a computer vision problem. The usual evaluation methodology in computer vision consists of comparing the results of a given model or algorithm with the ground truth of some collected dataset. This evaluation strategy is very advantageous since, while the evaluation using real physical systems needs lots of resources and time, evaluating a driving algorithm on a dataset can take maximum a few hours under an appropriate hardware budget.

In this thesis, we empirically investigate the relation between (offline) driving control accuracy on a dataset and (online) the actual driving quality on a simulator. We found that the common offline computer vision evaluation approach has a weak correlation to actual driving, since a driving agent with low prediction error can be surprisingly bad at driving. We also show two general approaches for increasing this poor correlation between prediction and driving.

Limitations *How can we improve the performance of end-to-end driving agents?*

Commonly, end-to-end driving IL approaches have been used to more simple tasks such as lane following or lead vehicle following. More complex task such as avoiding obstacles are achieved by hybridizing end-to-end driving with modular approaches [18, 77, 97, 102], but the real limits of end-to-end driving are not yet clear.

The final contribution of this thesis is that, by using the simulation infrastructure and the conditional imitation, we were able to analyze some limitations of end-to-end models. This was possible since we had a repeatable benchmark in a controlled environment making it easy to understand different aspects of the IL based end-

to-end driving agent. We found that there is a considerable amount of sensitivity towards initialization and order of data sampling during training. A model with the same network parameters can have significant different performance of up to 42%. This finding is analogous to recent debates on reproducibility of RL algorithms [46]. We also discover that higher amounts of data do not necessarily scales the performance since there could be biases on the data that burdens the network generalization, specially when reacting to complex dynamic scenes.

However, by leveraging these limitations we were able to outperform other modularized approaches on the CARLA benchmarks and we also show that there is still some degree of interpretability on end-to-end models. Yet, these specific issues must be addressed in order to further improve the results obtained by IL.

1.5 Contributions

This thesis addresses the problem of end-to-end imitation learning for autonomous driving contributing to several problems:

- We contribute in the development of the CARLA simulator. This simulator allows large scale collection of demonstrations for imitation learning approaches as well as multiple benchmarks to evaluate end-to-end models, as well as modular paradigms.
- We propose the conditional imitation learning approach. This allows an imitation learning network to learn multiple policies conditioned on a topological planner or on the passengers intentions.
- We show that the commonly used average prediction of driving controls on an static driving dataset is not so correlated with actual driving. This enforces the use of simulation as a better alternative. We also propose strategies to use offline datasets which are more correlated to actual driving.
- We disclose some limitations found on end-to-end driving agents, that when leveraged to small degree can already outperform other existent approaches.

1.6 Outline

This thesis is organizing in self-contained chapters that further develop each of the contributions exposed on the previous section. Each chapter will present an introduction, the state-of-the-art within its context and the obtained experimental results and conclusions.

Chapter 2 presents the CARLA simulator as well as two benchmarks inside the simulator to better measure end-to-end imitation learning approaches performance as well as comparing to others. Chapter 3 presents the conditional imitation learning approach. Chapter 4 discusses the correlation between offline computer vision based evaluation and actual driving. Chapter 5 discusses the limitations of imitation learning as well as insights on how to obtain better driving agents for end-to-end imitation learning approaches. Finally, Chapter 6 presents the conclusions and some possible future perspectives for continuing the work presented by this thesis.

2 CARLA: An Open Urban Driving Simulator

In order to be able to effectively evaluate end-to-end learned agents it is first fundamental to have a realistic and safe environment where the agent can be evaluated. On this chapter we introduce CARLA, an open-source simulator for autonomous driving research. CARLA has been developed from the ground up to support development, training, and validation of autonomous urban driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites and environmental conditions. We use CARLA to build benchmarks to compare and develop end-to-end autonomous driving algorithms. We build controlled scenarios of increasing difficulty, to analyze methods performance via repeatable metrics provided by CARLA, illustrating the platform's utility for autonomous driving research. [a]

[a] In this chapter we describe the family of CARLA releases until version 0.8.4 that are the ones used on this thesis. CARLA is still under constant development. To find information about the current system please visit http://carla.org

2.1 Introduction

Sensorimotor control in three-dimensional environments remains a major challenge in machine learning and robotics. The development of autonomous ground vehicles is a long-studied instantiation of this problem [84, 103]. Its most difficult form is navigation in densely populated urban environments [82]. This setting is particularly challenging due to complex multi-agent dynamics at traffic intersections; the necessity to track and respond to the motion of tens or hundreds of other actors that may be in view at any given time; prescriptive traffic rules that necessitate recognizing street signs, street lights, and road markings and distinguishing between multiple types of other vehicles; the long tail of rare events – road construction, a child running onto the road, an accident ahead, a rogue driver barreling on

the wrong side; and the necessity to rapidly reconcile conflicting objectives, such as applying appropriate deceleration when an absent-minded pedestrian strays onto the road ahead but another car is rapidly approaching from behind and may rear-end if one brakes too hard.

Research in autonomous urban driving is hindered by infrastructure costs and the logistical difficulties of training and testing systems in the physical world. Instrumenting and operating even one robotic car requires significant funds and manpower. And a single vehicle is far from sufficient for collecting the requisite data that cover the multitude of corner cases that must be processed for both training and validation. This is true for classic modular pipelines [31, 82] and even more so for data-hungry deep learning techniques. Training and validation of sensorimotor control models for urban driving in the physical world is beyond the reach of most research groups.

An alternative is to train and validate driving strategies in simulation. Simulation can democratize research in autonomous urban driving. It is also necessary for system verification, since some scenarios are too dangerous to be staged in the physical world (e.g., a child running onto the road ahead of the car). Simulation has been used for training driving models since the early days of autonomous driving research [84]. More recently, racing simulators have been used to evaluate new approaches to autonomous driving [18, 118]. Custom simulation setups are commonly used to train and benchmark robotic vision systems [10, 32, 38, 40, 78, 91, 106, 124]. And commercial games have been used to acquire high-fidelity data for training and benchmarking visual perception systems [88, 89].

While ad-hoc use of simulation in autonomous driving research is widespread, existing simulation platforms are limited. Open-source racing simulators such as TORCS [118] do not present the complexity of urban driving: they lack pedestrians, intersections, cross traffic, traffic rules, and other complications that distinguish urban driving from track racing. And commercial games that simulate urban environments at high fidelity, such as Grand Theft Auto V [88, 89], do not support detailed benchmarking of driving policies: they have little customization and control over the environment, limited scripting and scenario specification, severely limited sensor suite specification, no detailed feedback upon violation of traffic rules, and other limitations due to their closed-source commercial nature and fundamentally different objectives during their development.

In this chapter, we introduce CARLA (Car Learning to Act) – an open simulator for urban driving. CARLA has been developed from the ground up to support training, prototyping, and validation of autonomous driving models, including both perception and control. CARLA is an open platform. Uniquely, the content of urban environments provided with CARLA is also free. The content was created from scratch by a dedicated team of digital artists employed for this purpose. It includes

urban layouts, a multitude of vehicle models, buildings, pedestrians, street signs, etc. The simulation platform supports flexible setup of sensor suites and provides signals that can be used to train driving strategies, such as GPS coordinates, speed, acceleration, and detailed data on collisions and other infractions. A wide range of environmental conditions can be specified, including weather and time of day. A number of such environmental conditions are illustrated in Figure 2.1.

We use CARLA to build different benchmarks to evaluate the performance of end-to-end driving via imitation learning compared to other two approaches to autonomous driving. We stage controlled goal-directed navigation scenarios of increasing difficulty. We manipulate the complexity of the route that must be traversed, the presence of traffic, and the environmental conditions. With this into account, we propose two benchmarks that concerns to evaluate different features on autonomous driving. The first one correspond to the CoRL 2017 benchmark that mainly focuses on testing navigation and lane following abilities of the driving agents. Finally, the *NoCrash* benchmark is proposed in order to evaluate more the longitudinal control and ability of the driving agent on how to react to more complex situations such as "jaywalking" pedestrians or suddenly stopping vehicles.

2.2 Related Work

Raw sensor data together with ground truth and/or privileged information is critical for designing, training and validating autonomous agents. Simulators allow to produce all this information in an efficient manner and can lead to the generation of groundbreaking new knowledge.

Pursuing this idea, different works have focused on the use and creation of simulated environments to produce synthetic data for developing algorithms to obtain visual cues such as optical flow [16, 75] and disparity from stereo [37]. Other works proposed the use of synthetic data to develop scene understanding modules for indoor scenes [41] and for driving scenarios [32, 38, 89, 91, 101]; even to learn unwritten common sense [113] or classifying videos according to actions [23]. Since the ultimate goal of autonomous agents is, indeed, operating within a desired environment, other works focused on training the agents by in simulation rather than only producing datasets. Examples of these are used to learn physical intuition in a simple blocks world [62], or to perform interaction with static objects in photo-realistic indoor scenarios[127]. For the specific domain of autonomous driving, TORCS [118] has been used for learning mediated driving representations for autonomous navigation[18]. However, in most of the cases the available open frameworks are very constrained in terms of scenario complexity and assets variability, limiting the scope of the study. The research community found a more

challenging and complex alternative in AAA video games, such as GTA-V, which can be used for urban driving simulation, not only to produce datasets such as in [89], but also to perform in-game training [54]. However, these engines are not open source and, therefore, their usability for simulation purposes is restricted and recently it has even claimed to be illegal.

CARLA brings the best qualities of the previous tools. Firstly, it is totally open source, both the code and also all the 3D assets, including towns, vehicles, pedestrians, etc. It can generate synthetic datasets with ground truth including images with pixel-wise semantic classes, depth, etc. It also provides privileged information such as GPS-like information of the vehicle, speed, 3D acceleration vector, timestamps (in game and OS), degree of invasion in both opposite lanes and sidewalks, collision strength segregated per vehicles, pedestrians and others (infrastructure, houses, trees, rocks), etc. But most importantly, CARLA is an urban driving simulator that allows autonomous agents to drive within urban scenarios for learning to act and assessing their capabilities. In this sense, CARLA is a clear step forward in simulation, addressing urban scenarios in a highly realistic way, both in terms of graphics and fidelity. CARLA's focus is set on challenging situations, where the layout of the scene is complex (e.g., intersections with traffic signs, sidewalks, buildings, etc.) and strong interaction and negotiation with other vehicles and pedestrians is required.

2.3 Simulation Engine

CARLA has been built for flexibility and realism in the rendering and physics simulation. It is implemented as an open-source layer over Unreal Engine 4 (UE4) [29], enabling future extensions by the community. The engine provides state-of-the-art rendering quality, realistic physics, basic NPC logic, and an ecosystem of interoperable plugins. The engine itself is free for non-commercial use.

CARLA simulates a dynamic world and provides a simple interface between the world and an agent that interacts with the world. To support this functionality, CARLA is designed as a server-client system, where the server runs the simulation and renders the scene. The client API is implemented in Python and is responsible for the interaction between the autonomous agent and the server via sockets. The client sends commands and meta-commands to the server and receives sensor readings in return. Commands control the vehicle and include steering, accelerating, and braking. Meta-commands control the behavior of the server and are used for resetting the simulation, changing the properties of the environment, and modifying the sensor suite. Environmental properties include weather conditions, illumination, and density of cars and pedestrians. When the server is reset, the agent is re-initialized at a new location specified by the client.

Figure 2.1 – A street in Town 2, shown from a third-person view in four weather conditions. Clockwise from top left: clear day, daytime rain, daytime shortly after rain, and clear sunset. See the supplementary video for recordings from the simulator.

Environment. The environment is composed of 3D models of static objects such as buildings, vegetation, traffic signs, and infrastructure, as well as dynamic objects such as vehicles and pedestrians. All models are carefully designed to reconcile visual quality and rendering speed: we use low-weight geometric models and textures, but maintain visual realism by carefully crafting the materials and making use of variable level of detail. All 3D models share a common scale, and their sizes reflect those of real objects. At the time of writing, our asset library includes 40 different buildings, 16 animated vehicle models, and 50 animated pedestrian models.

We used these assets to build urban environments via the following steps: (a) laying out roads and sidewalks; (b) manually placing houses, vegetation, terrain, and traffic infrastructure; and (c) specifying locations where dynamic objects can appear (spawn). This way we have designed two towns: Town 1 with a total of 2.9 km of drivable roads, used for training, and Town 2 with 1.4 km of drivable roads, used for testing. Images from both of the two towns are shown on Fig. 2.2

One of the challenges in the development of CARLA was the configuration of the behavior of non-player characters, which is important for realism. We based the non-player vehicles on the standard UE4 vehicle model (PhysXVehicles). Kinematic

parameters were adjusted for realism. We also implemented a basic controller that governs non-player vehicle behavior: lane following, respecting traffic lights, speed limits, and decision making at intersections. Vehicles and pedestrians can detect and avoid each other. More advanced non-player vehicle controllers can are integrated on the newer versions of CARLA.

Pedestrians navigate the streets according to a town-specific navigation map, which conveys a location-based cost. This cost is designed to encourage pedestrians to walk along sidewalks and marked road crossings, but allows them to cross roads at any point. Pedestrians wander around town in accordance with this map, avoiding each other and trying to avoid vehicles. If a car collides with a pedestrian, the pedestrian is deleted from the simulation and a new pedestrian is spawned at a different location after a brief time interval.

To increase visual diversity, we randomize the appearance of non-player characters when they are added to the simulation. Each pedestrian is clothed in a random outfit sampled from a pre-specified wardrobe and is optionally equipped with one or more of the following: a smartphone, shopping bags, a guitar case, a suitcase, a rolling bag, or an umbrella. Each vehicle is painted at random according to a model-specific set of materials. This diversity is shown on Fig. 2.3.

We have also implemented a variety of atmospheric conditions and illumination regimes. These differ in the position and color of the sun, the intensity and color of diffuse sky radiation, as well as ambient occlusion, atmospheric fog, cloudiness, and precipitation. Currently, the simulator supports two lighting conditions – midday and sunset – as well as nine weather conditions, differing in cloud cover, level of precipitation, and the presence of puddles in the streets. This results in a total of 18 illumination-weather combinations. (In what follows we refer to these as weather, for brevity.) Four of these are illustrated in Figure 2.1.

Sensors. CARLA allows for flexible configuration of the agent's sensor suite. On version 0.8.4, sensors are limited to RGB cameras and to pseudo-sensors that provide ground-truth depth, semantic segmentation and LIDAR. Some of these are illustrated in Figure 2.4. The number of cameras and their type and position can be specified by the client. Camera parameters include 3D location, 3D orientation with respect to the car's coordinate system, field of view, and depth of field. Our semantic segmentation pseudo-sensor provides 12 semantic classes: road, lane-marking, traffic sign, sidewalk, fence, pole, wall, building, vegetation, vehicle, pedestrian, and other.

In addition to sensor and pseudo-sensor readings, CARLA provides a range of measurements associated with the state of the agent and compliance with traffic rules. Measurements of the agent's state include vehicle location and orientation with respect to the world coordinate system (akin to GPS and compass), speed, acceleration vector, and accumulated impact from collisions. Measurements con-

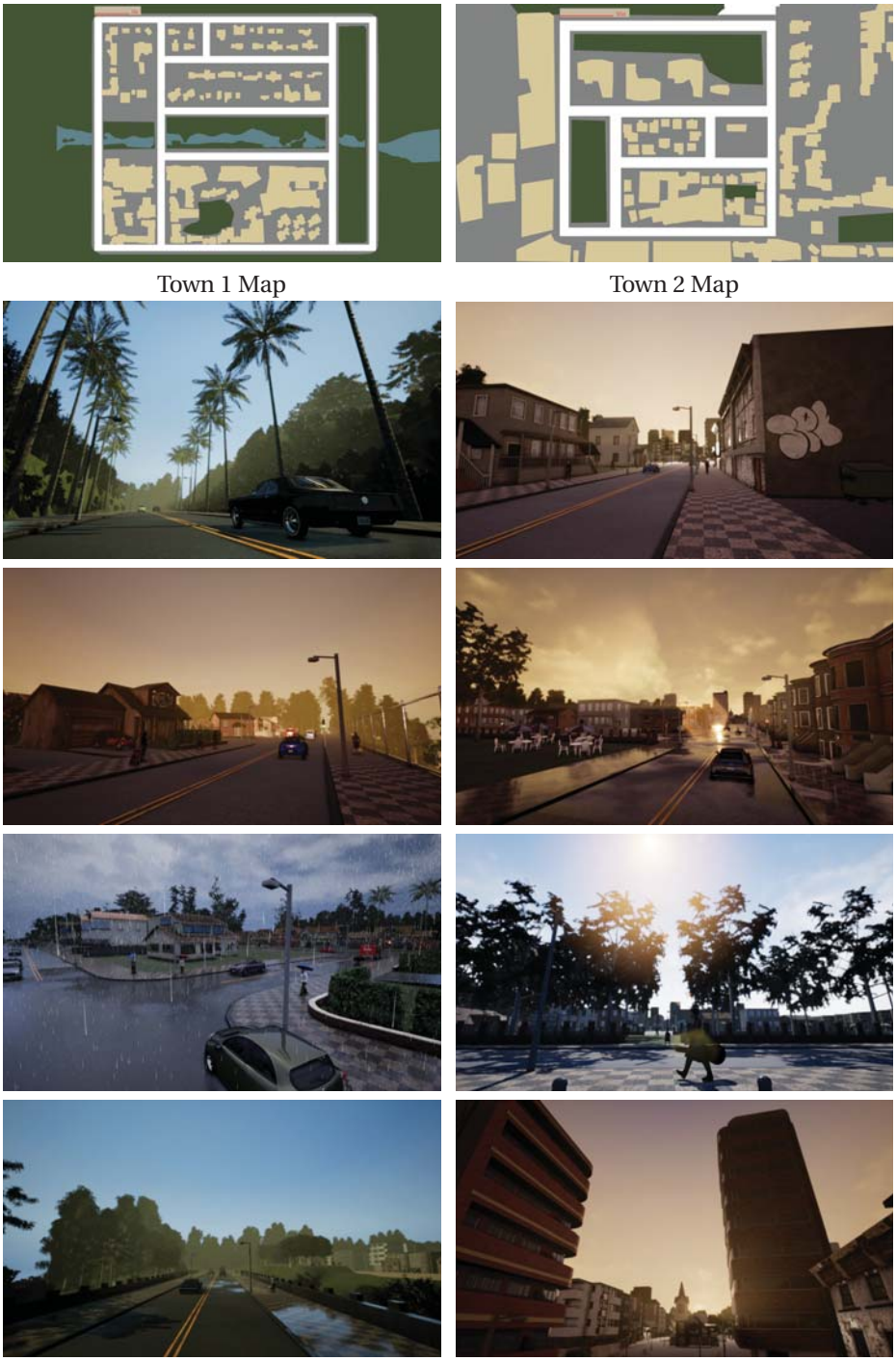


Figure 2.2 – The two CARLA towns. **Left:** views and a map of CARLA Town 1. **Right:** views and a map of CARLA Town 2.

Figure 2.3 – Diversity of cars and pedestrians currently available in CARLA.

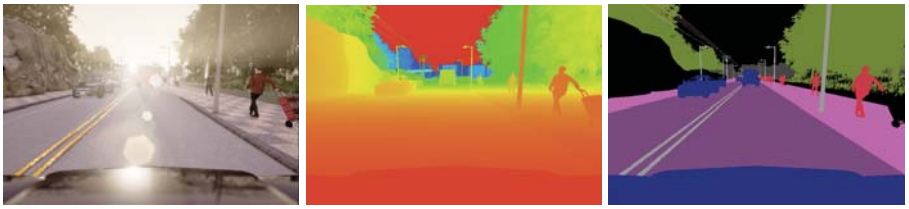


Figure 2.4 – Three of the sensing modalities provided by CARLA. From left to right: normal vision camera, ground-truth depth, and ground-truth semantic segmentation. Depth and semantic segmentation are pseudo-sensors that support experiments that control for the role of perception. Additional sensor models can be plugged in via the API.

cerning traffic rules include the percentage of the vehicle's footprint that impinges on wrong-way lanes or sidewalks, as well as states of the traffic lights and the speed limit at the current location of the vehicle. Finally, CARLA provides access to exact locations and bounding boxes of all dynamic objects in the environment. These signals play an important role in training and evaluating driving policies.

2.4 Benchmarks

CARLA supports development, training, and detailed performance analysis of autonomous driving systems. For that we create some sets of tasks, refereed as benchmarks, in order to evaluate and compare different kinds of autonomous driving algorithms. In this section we present two benchmarks built in CARLA in order to validate different capabilities of the autonomous driving models.

2.4.1 CoRL2017 Benchmark

The CoRL2017 benchmark was created in order to test some basic capabilities of autonomous driving system by using the increasingly difficult tasks within the CARLA Environment. The tasks are set up as goal-directed navigation: an agent is initialized somewhere in town and has to reach a destination point. In these experiments, the agent is allowed to ignore speed limits and traffic lights. We organize the tasks in order of increasing difficulty as follows:

- Straight: Destination is straight ahead of the starting point, and there are no dynamic objects in the environment. Average driving distance to the goal is 200 m in Town 1 and 100 m in Town 2.
- One turn: Destination is one turn away from the starting point; no dynamic

objects. Average driving distance to the goal is 400 m in Town 1 and 170 m in Town 2.
- Navigation: No restriction on the location of the destination point relative to the starting point, no dynamic objects. Average driving distance to the goal is 770 m in Town 1 and 360 m in Town 2.
- Navigation with dynamic obstacles: Same as the previous task, but with dynamic objects (cars and pedestrians).

Experiments are conducted in two towns. Town 1 is used for training, Town 2 for testing. We consider six weather conditions for the experiments, organized in two groups. Training Weather Set was used for training and includes clear day, clear sunset, daytime rain, and daytime after rain. Test Weather Set was never used during training and includes cloudy daytime and soft rain at sunset.

For each combination of a task, a town, and a weather set, testing is carried out over 25 episodes. In each episode, the objective is to reach a given goal location. An episode is considered successful if the agent reaches the goal within a time budget. The time budget is set to the time needed to reach the goal along the optimal path at a speed of 10 km/h. Infractions, such as driving on the sidewalk or collisions, do not lead to termination of an episode, but are logged and reported.

Results

Table 2.1 reports the percentage of successfully completed episodes under four different conditions. The first is the training condition: Town 1, Training Weather Set. Note that start and goal locations are different from those used during training: only the general environment and ambient conditions are the same. The other three experimental conditions test more aggressive generalization: to the previously unseen Town 2 and to previously unencountered weather from the Test Weather Set. We use the benchmark to compare a conditional imitation learning approach, explained on Chapter 3 with a modular pipeline and a reinforcement learning approach, described on [26].

Results presented in Table 2.1 suggest several general conclusions. Overall, the performance of all methods is not perfect even on the simplest task of driving in a straight line, and the success rate further declines for more difficult tasks. Generalization to new weather is easier than generalization to a new town. The modular pipeline and the agent trained with imitation learning perform on par on most tasks and conditions. Reinforcement learning underperforms relative to the other approaches.

Surprisingly, none of the methods performs perfectly even on the simplest task of driving straight on an empty street in the training conditions. Training conditions include four different weather conditions. The exact trajectories driven during

	Training conditions			New town			New weather			New town & weather		
Task	MP	IL	RL	MP	IL	RL	MP	IL	RL	MP	IL	RL
Straight	**98**	95	89	92	**97**	74	**100**	98	86	50	**80**	68
One turn	82	**89**	34	**61**	59	12	**95**	90	16	**50**	48	20
Navigation	80	**86**	14	24	**40**	3	**94**	84	2	**47**	44	6
Nav. dynamic	77	**83**	7	24	**38**	2	**89**	82	2	**44**	42	4

Table 2.1 – Quantitative evaluation of three autonomous driving systems on goal-directed navigation tasks. The table reports the percentage of successfully completed episodes in each condition. Higher is better.

training are not repeated during testing. Therefore performing perfectly on this task requires robust generalization, which is challenging for existing deep learning methods. On more advanced tasks the performance of all methods declines. On the most difficult task of navigation in a populated urban environment, the two best methods – modular pipeline and imitation learning – are below 90% success in all conditions and are below 45% in the test town. These results clearly indicate that performance is far from saturated even in the training conditions, and that generalization to new environments poses a serious challenge.

Infraction analysis. CARLA supports fine-grained analysis of driving policies. We use to benchmark to analyze the behavior of the three systems on the hardest task: navigation in the presence of dynamic objects. We characterize the approaches by average distance traveled between infractions of the following five types: driving on the opposite lane, driving on the sidewalk, colliding with other vehicles, colliding with pedestrians, and hitting static objects.

Table 2.2 reports the average distance (in kilometers) driven between two infractions. All approaches perform better in the training town. For all conditions, IL strays onto the opposite lane least frequently, and RL is the worst in this metric. A similar pattern is observed with regards to veering onto the sidewalk.

These results highlight the susceptibility of end-to-end approaches to rare events: breaking or swerving to avoid a pedestrian is a rare occurrence during training. While CARLA can be used to increase the frequency of such events during training to support end-to-end approaches, deeper advances in learning algorithms and model architectures may be necessary for significant improvements in robustness [18].

	Training conditions			New town			New weather			New town & weather		
Task	MP	IL	RL	MP	IL	RL	MP	IL	RL	MP	IL	RL
Opposite lane	10.2	**33.4**	0.18	0.45	**1.12**	0.23	16.1	**57.3**	0.09	0.40	**0.78**	0.21
Sidewalk	**18.3**	12.9	0.75	0.46	**0.76**	0.43	24.2	**>57**	0.72	0.43	**0.81**	0.48
Collision-static	**10.0**	5.38	0.42	**0.44**	0.40	0.23	**16.1**	4.05	0.24	**0.45**	0.28	0.25
Collision-car	**16.4**	3.26	0.58	0.51	**0.59**	0.41	**20.2**	1.86	0.85	**0.47**	0.44	0.37
Collision-pedestrian	**18.9**	6.35	17.8	1.40	1.88	**2.55**	20.4	11.2	**20.6**	1.46	1.41	**2.99**

Table 2.2 – Average distance (in kilometers) traveled between two infractions. Higher is better.

2.4.2 NoCrash benchmark

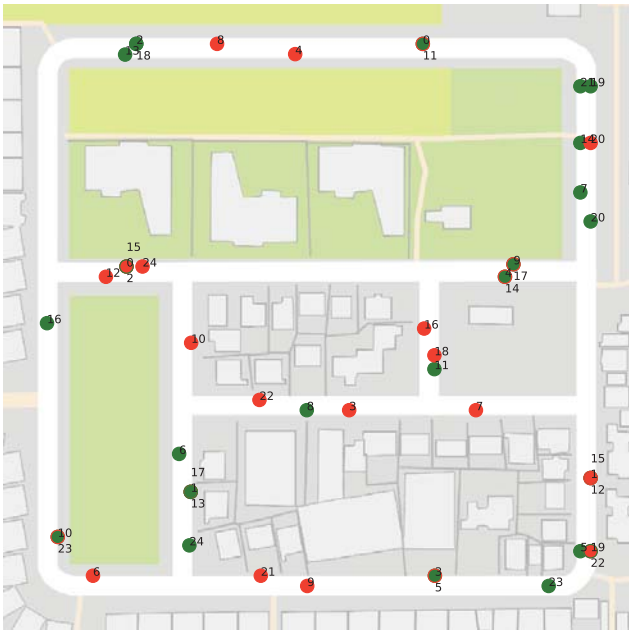
We propose an on-line driving benchmark focused on testing the ability of the ego vehicles to handle complex events caused by changing traffic conditions (e.g., traffic lights) and dynamic agents present in the scene. For this benchmark, we propose different tasks and also different metrics than the CoRL2017 in order to precisely measure specific reaction patterns that we know good drivers must master in urban conditions. For the *NoCrash* benchmark, we propose three different tasks, each task corresponding to completing 25 goal directed episodes. As in the CoRL 2017, on each episode, the agent starts in some start position and is directed by a high-level planner into reaching some goal position. The three tasks have the same set of start and end positions, as well as an increasing level of difficulty as follows:

1. Empty Town: no dynamic objects.
2. Regular Traffic: mild number of cars and pedestrians.
3. Dense Traffic: large number of pedestrians and heavy traffic (dense urban scenario).

Similar to CoRL2017 benchmark, *NoCrash* has six different weather conditions, where four were seen in training and two reserved for testing. It also has two different towns, one was seen in training and the other is reserved for testing. The tuples of start/goal positions are based on the ones used in CARLA CoRL2017 benchmark for the tasks “Navigation” and “Nav. Dynamic”. However, we increased the difficulty of some start/goal positions that had a too small distance between them. Figure 2.5 shows the start end positions for both Towns.

Evaluation protocol. As mentioned, in our evaluation the measure of success of an episode should be more representative of the agent capabilities to react to dynamic objects. The original CoRL 2017 has a goal conditioned success rate metric that is computed separately from a kilometers between infractions metric.

(a) Town01



(b) Town02

Figure 2.5 – The start and goal positions for the *NoCrash* Benchmark. The start positions are in red and the goal positions are in green. Same number correspond to matching start-end positions.

Task	Training conditions	New Weather	New Town	New town & weather
Empty Town	79	59	41	24
Regular Traffic	61	40	22	18
Dense Traffic	22	6	7.33	2.67

Table 2.3 – Results from the Conditional Imitation Learning model (Chapter 3 on the complex *NoCrash* benchmark. "Empty" does not contain other agents (cars, pedestrians), "Regular" contains a moderate amount of other agents, "Cluttered" corresponds to dense urban scenarios. The table reports the percentage of successfully completed episodes in each condition.

The latter metric was proposed to be analogous to the one commonly used by driving evaluations on real world where the number of human interventions per kilometer is counted [52]. These interventions usually happen when the safety driver notices some inconsistent behaviour that would lead the vehicle to a possibly dangerous state. On a potentially inconsistent behavior, the human intervention will put the vehicle back to a safe state of the system. However, on the CoRL2017 benchmark analysis, when a infraction is made, the episode continues after the infraction, leading to some inconsistencies on infraction counting. Some examples of inconsistencies include whether a crash after leaving the road be counted as one or two infractions.

In *NoCrash*, instead of counting the number of infractions per kilometer, we end the episode as failing when any collision, bigger than a fixed magnitude, happens. With this, we are setting a lower bound and have a guarantee of acceptable behaviors based on the measured percentage of success. Further, this makes the evaluation also similar to the km/interventions evaluation, since the episode goes back to a safe state, in this case the next episode. In summary, we consider a task as a success if the agent reaches a certain goal under a time limit without colliding with any object. We also care about the ability of the agent to obey traffic rules. In particular, we measure and report the percentage of traffic light violations. Note that an episode is not terminated when a traffic light violation occurs.

Results

On Table 2.3 we present the results of the conditional imitation learning described on Chapter 3 for the new tasks and evaluation protocol proposed at *NoCrash* benchmark.

We notice some clear significant drop specially for the Cluttered scenario, enforcing the conclusions obtained with the infraction analysis from section 2.4.1.

This benchmark is going to be further explored on Chapter 5 where we will use to make a deeper analysis of the limitations of imitation learning through behavior cloning approach.

2.5 Conclusion

We have presented CARLA, an open simulator for autonomous driving. In addition to open-source code and protocols, CARLA provides digital assets that were created specifically for this purpose and can be reused freely. We leverage CARLA's simulation engine to propose different kind of benchmarking tools to be able to compare different algorithmic approaches. CARLA provided us with the tools to develop and train the systems and then evaluate them in controlled scenarios. The feedback provided by the simulator enables detailed analyses that highlight particular failure modes and opportunities for future work. We hope that CARLA will enable a broad community to actively engage in autonomous driving research. The simulator and accompanying assets are released open-source at http://carla.org.

3 Conditional Imitation Learning

(a) Vehicle following (b) Stopping for traffic lights (c) Stopping for "jaywalkers"

Figure 3.1 – Conditional imitation learning allows an autonomous vehicle trained end-to-end to be directed by high-level commands. (a) We train and evaluate robotic vehicles in the physical world (top) and in simulated urban environments (bottom). (b) The vehicles drive based on video from a forward-facing onboard camera. At the time these images were taken, the vehicle was given the command "turn right at the next intersection". (c) The trained controller handles sensorimotor coordination and follows the provided commands.

Deep networks trained on demonstrations of human driving have learned to follow roads and avoid obstacles. However, driving policies trained via imitation learning cannot be controlled at test time. A vehicle trained end-to-end to imitate an expert cannot be guided to take a specific turn at an upcoming intersection. This limits the utility of such systems. We propose to condition imitation learning on high-level command input. At test time, the learned driving policy functions as a chauffeur that handles sensorimotor coordination but continues to respond to navigational commands. We evaluate different architectures for conditional imitation learning in vision-based driving. We conduct experiments in realistic three-dimensional simulations of urban driving and on a 1/5 scale robotic truck that is trained to drive in a residential area. Both systems drive based on visual input yet remain responsive to high-level navigational commands.

3.1 Introduction

Why has imitation learning not scaled up to fully autonomous urban driving? One limitation is in the assumption that the optimal action can be inferred from the perceptual input alone. This assumption often does not hold in practice: for instance, when a car approaches an intersection, the camera input is not sufficient to predict whether the car should turn left, right, or go straight. Mathematically, the mapping from the image to the control command is no longer a function. Fitting a function approximator is thus bound to run into difficulties. This had already been observed in the work of Pomerleau: "Currently upon reaching a fork, the network may output two widely discrepant travel directions, one for each choice. The result is often an oscillation in the dictated travel direction" [84]. Even if the network can resolve the ambiguity in favor of some course of action, it may not be the one desired by the passenger, who lacks a communication channel for controlling the network itself.

In this chapter, we address this challenge with conditional imitation learning. At training time, the model is given not only the perceptual input and the control signal, but also a representation of the expert's intention. At test time, the network can be given corresponding commands, which resolve the ambiguity in the perceptuomotor mapping and allow the trained model to be controlled by a passenger or a topological planner, just as mapping applications and passengers provide turn-by-turn directions to human drivers. The trained network is thus freed from the task of planning and can devote its representational capacity to driving. This enables scaling imitation learning to vision-based driving in complex urban environments.

We evaluate the presented approach in realistic simulations of urban driving and on a 1/5 scale robotic truck. Both systems are shown in Figure 3.1. Simulation allows us to thoroughly analyze the importance of different modeling decisions, carefully compare the approach to relevant baselines, and conduct detailed ablation studies. Experiments with the physical system demonstrate that the approach can be successfully deployed in the physical world. Recordings of both systems are provided in the supplementary video.

3.2 Related Work

Imitation learning has been applied to a variety of tasks, including articulated motion [6, 28, 85], autonomous flight [1, 35, 94], modeling navigational behavior [128, 129], off-road driving [61, 103], and road following [11, 18, 84, 123]. Technically, these applications differ in the input representation (raw sensory input or hand-crafted features), the control signal being predicted, the learning algorithms, and the learned representations. Most relevant to our work are the systems of

Pomerleau [84], LeCun et al. [61], and Bojarski et al. [11], who used ground vehicles and trained deep networks to predict the driver's actions from camera input. These studies focused on purely reactive tasks, such as lane following or obstacle avoidance. In comparison, we develop a command-conditional formulation that enables the application of imitation learning to more complex urban driving. Another difference is that we learn to control not only steering but also acceleration and braking, enabling the model to assume full control of the car.

The decomposition of complex tasks into simpler sub-tasks has been studied from several perspectives. In robotics, movement primitives have been used as building blocks for advanced motor skills [55, 83]. Movement primitives represent a simple motion, such as a strike or a throw, by a parameterized dynamical system. In comparison, the policies we consider have much richer parameterizations and address more complex sensorimotor tasks that couple perception and control, such as finding the next opportunity to turn right and then making the turn while avoiding dynamic obstacles.

In reinforcement learning, hierarchical approaches aim to construct multiple levels of temporally extended sub-policies [8]. The options framework is a prominent example of such hierarchical decomposition [108]. Basic motor skills that are learned in this framework can be transferred across tasks [57]. Hierarchical approaches have also been combined with deep learning and applied to raw sensory input [58]. In these works, the main aim is to learn purely from experience and discover hierarchical structure automatically. This is hard and is in general an open problem, particularly for sensorimotor skills with the complexity we consider. In contrast, we focus on imitation learning, and we provide additional information on the expert's intentions during demonstration. This formulation makes the learning problem more tractable and yields a human-controllable policy.

Adjacent to hierarchical methods is the idea of learning multi-purpose and parameterized controllers. Parameterized goals have been used to train motion controllers in robotics [22, 24, 56]. Schaul et al. [99] proposed a general framework for reinforcement learning with parameterized value functions, shared across states and goals. Dosovitskiy and Koltun [25] studied families of parameterized goals in the context of navigation in three-dimensional environments. Javdani et al. [48] studied a scenario where a robot assists a human and changes its behavior depending on its estimate of the human's goal. Our work shares the idea of training a conditional controller, but differs in the model architecture, the application domain (vision-based autonomous driving), and the learning method (conditional imitation learning).

Autonomous driving is the subject of intensive research [82]. Broadly speaking, approaches differ in their level of modularity. On one side are highly tuned systems that deploy an array of computer vision algorithms to create a model of the

environment, which is then used for planning and control [31]. On the opposite side are end-to-end approaches that train function approximators to map sensory input to control commands [11, 84, 123]. Our approach is on the end-to-end side of the spectrum, but in addition to sensory input the controller is provided with commands that specify the driver's intent. This resolves some of the ambiguity in the perceptuomotor mapping and creates a communication channel that can be used to guide the autonomous car as one would guide a chauffeur.

Human guidance of robot actions has been studied extensively [14, 45, 74, 110, 114]. These works tackle the challenging problem of parsing natural language instructions. Our work does not address natural language communication; we limit commands to a predefined vocabulary such as "turn right at the next intersection", "turn left at the next intersection", and "keep straight". On the other hand, our work deals with end-to-end vision-based driving using deep networks. Systems in this domain have been limited to imitating the expert without the ability to naturally accept commands after deployment [11, 18, 84, 123]. We introduce such ability into deep networks for end-to-end vision-based driving.

3.3 Conditional Imitation Learning

We begin by describing the standard imitation learning setup and then proceed to our command-conditional formulation. Consider a driving agent that interacts with the environment over discrete time steps. At each time step t, the controller receives an observation $\mathbf{o}_t$ and takes an action $\mathbf{a}_t$. The basic idea behind imitation learning is to train a controller that mimics an expert. The training data is a set of observation-action pairs $D = \{\langle \mathbf{o}_i, \mathbf{a}_i \rangle\}_{i=1}^{N}$ generated by the expert. The assumption is that the expert is successful at performing the task of interest and that a controller trained to mimic the expert will also perform the task well. This is a supervised learning problem, in which the parameters $\boldsymbol{\theta}$ of a function approximator $F(\mathbf{o};\boldsymbol{\theta})$ must be optimized to fit the mapping of observations to actions:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_i \ell\big(F(\mathbf{o}_i;\boldsymbol{\theta}), \mathbf{a}_i\big). \tag{3.1}$$

An implicit assumption behind this formulation is that the expert's actions are fully explained by the observations; that is, there exists a function E that maps observations to the expert's actions: $\mathbf{a}_i = E(\mathbf{o}_i)$. If this assumption holds, a sufficiently expressive approximator will be able to fit the function E given enough data. This explains the success of imitation learning on tasks such as lane following. However, in more complex scenarios the assumption that the mapping of observations to actions is a function breaks down. Consider a driver approaching an intersection.

The driver's subsequent actions are not explained by the observations, but are additionally affected by the driver's internal state, such as the intended destination. The same observations could lead to different actions, depending on this latent state. This could be modeled as stochasticity, but a stochastic formulation misses the underlying causes of the behavior. Moreover, even if a controller trained to imitate demonstrations of urban driving did learn to make turns and avoid collisions, it would still not constitute a useful driving system. It would wander the streets, making arbitrary decisions at intersections. A passenger in such a vehicle would not be able to communicate the intended direction of travel to the controller, or give it commands regarding which turns to take.

To address this, we begin by explicitly modeling the expert's internal state by a vector $\mathbf{h}$, which together with the observation explains the expert's action: $\mathbf{a}_i = E(\mathbf{o}_i, \mathbf{h}_i)$. Vector $\mathbf{h}$ can include information about the expert's intentions, goals, and prior knowledge. The standard imitation learning objective can then be rewritten as

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_i \ell\big(F(\mathbf{o}_i;\boldsymbol{\theta}), E(\mathbf{o}_i, \mathbf{h}_i)\big). \tag{3.2}$$

It is now clear that the expert's action is affected by information that is not provided to the controller F.

We expose the latent state $\mathbf{h}$ to the controller by introducing an additional command input: $\mathbf{c} = \mathbf{c}(\mathbf{h})$. At training time, the command $\mathbf{c}$ is provided by the expert. It need not constitute the entire latent state $\mathbf{h}$, but should provide useful information about the expert's decision-making. For example, human drivers already use turn signals to communicate their intent when approaching intersections; these turn signals can be used as commands in our formulation. At test time, commands can be used to affect the behavior of the controller. These test-time commands can come from a human user or a planning module. In urban driving, a typical command would be "turn right at the next intersection", which can be provided by a navigation system or a passenger.

The training dataset becomes $D = \{\langle \mathbf{o}_i, \mathbf{c}_i, \mathbf{a}_i \rangle\}_{i=1}^N$. The command-conditional imitation learning objective is

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_i \ell\big(F(\mathbf{o}_i, \mathbf{c}_i;\boldsymbol{\theta}), \mathbf{a}_i\big). \tag{3.3}$$

In contrast with objective (3.2), the learner is informed about the expert's latent state and can use this additional information in predicting the action. This setting is illustrated in Figure 3.2.

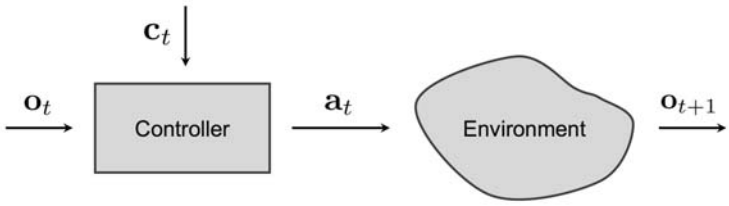


Figure 3.2 – High-level overview. The controller receives an observation $\mathbf{o}_t$ from the environment and a command $\mathbf{c}_t$. It produces an action $\mathbf{a}_t$ that affects the environment, advancing to the next time step.

3.4 Methodology

We now describe a practical implementation of command-conditional imitation learning. Code is available at https://github.com/carla-simulator/imitation-learning.

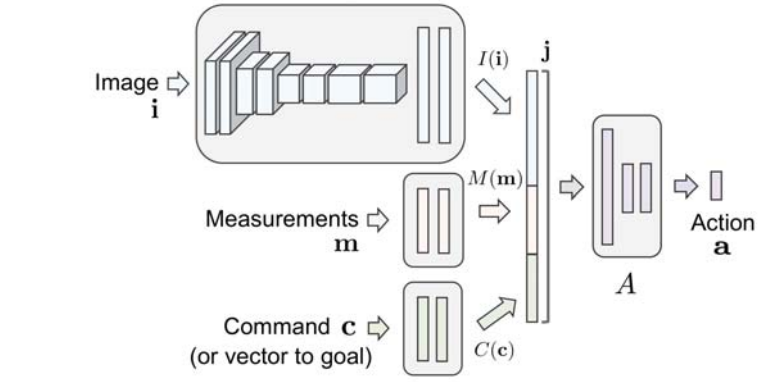
3.4.1 Network Architecture

Assume that each observation $\mathbf{o} = \langle \mathbf{i}, \mathbf{m} \rangle$ comprises an image $\mathbf{i}$ and a low-dimensional vector $\mathbf{m}$ that we refer to as measurements, following Dosovitskiy and Koltun [25]. The controller F is represented by a deep network. The network takes the image $\mathbf{i}$, the measurements $\mathbf{m}$, and the command $\mathbf{c}$ as inputs, and produces an action $\mathbf{a}$ as its output. The action space can be discrete, continuous, or a hybrid of these. In our driving experiments, the action space is continuous and two-dimensional: steering angle and acceleration. The acceleration can be negative, which corresponds to braking or driving backwards. The command $\mathbf{c}$ is a categorical variable represented by a one-hot vector.

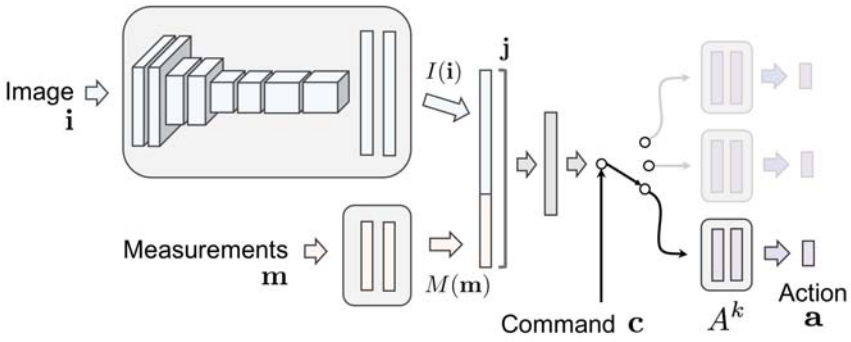
We study two approaches to incorporating the command $\mathbf{c}$ into the network. The first architecture is illustrated in Figure 3.3(a). The network takes the command as an input, alongside the image and the measurements. These three inputs are processed independently by three modules: an image module $I(\mathbf{i})$, a measurement module $M(\mathbf{m})$, and a command module $C(\mathbf{c})$. The image module is implemented as a convolutional network, the other two modules as fully-connected networks. The outputs of these modules are concatenated into a joint representation:

$$\mathbf{j} = J(\mathbf{i}, \mathbf{m}, \mathbf{c}) = \langle I(\mathbf{i}), M(\mathbf{m}), C(\mathbf{c}) \rangle . \tag{3.4}$$

The control module, implemented as a fully-connected network, takes this joint representation and outputs an action $A(\mathbf{j})$. We refer to this architecture as `command`



(a)



(b)

Figure 3.3 – Two network architectures for command-conditional imitation learning. (a) `command input`: the command is processed as input by the network, together with the image and the measurements. The same architecture can be used for goal-conditional learning (one of the baselines in our experiments), by replacing the command by a vector pointing to the goal. (b) `branched`: the command acts as a switch that selects between specialized sub-modules.

module	input dimension	channels	kernel	stride
Perception	$200 \times 88 \times 3$	32	5	2
	$98 \times 48 \times 32$	32	3	1
	$96 \times 46 \times 32$	64	3	2
	$47 \times 22 \times 64$	64	3	1
	$45 \times 20 \times 64$	128	3	2
	$22 \times 9 \times 128$	128	3	1
	$20 \times 7 \times 128$	256	3	2
	$9 \times 3 \times 256$	256	3	1
	$7 \cdot 1 \cdot 256$	512	–	–
	512	512	–	–
Measurement	1	128	–	–
	128	128	–	–
	128	128	–	–
Joint input	$512 + 128$	512	–	–
Control	512	256	–	–
	256	256	–	–
	256	1	–	–

Table 3.1 – Exact configurations of all network modules for the imitation learning approach.

`input`. It is applicable to both continuous and discrete commands of arbitrary dimensionality. However, the network is not forced to take the commands into account, which can lead to suboptimal performance in practice.

We therefore designed an alternative architecture, shown in Figure 3.3(b). The image and measurement modules are as described above, but the command module is removed. Instead, we assume a discrete set of commands $C = \{\mathbf{c}^0, \ldots, \mathbf{c}^K\}$ (including a default command $\mathbf{c}^0$ corresponding to no specific command given) and introduce a specialist branch A^i for each of the commands $\mathbf{c}^i$. The command $\mathbf{c}$ acts as a switch that selects which branch is used at any given time. The output of the network is thus

$$F(\mathbf{i}, \mathbf{m}, \mathbf{c}^i) = A^i(J(\mathbf{i}, \mathbf{m})). \tag{3.5}$$

We refer to this architecture as `branched`. The branches A^i are forced to learn sub-policies that correspond to different commands. In a driving scenario, one module might specialize in lane following, another in right turns, and a third in left turns. All modules share the perception stream.

3.4.2 Network Details

For all controllers, the observation **o** is the currently observed image at 200×88 pixel resolution. For the measurement **m**, we used the current speed of the car, if available (in the physical system the speed estimates were very noisy and we refrained from using them). All networks are composed of modules with identical architectures (e.g., the ConvNet architecture is the same in all conditions). The differences are in the configuration of modules and branches as can be seen in Figure 3.3. The image module consists of 8 convolutional and 2 fully connected layers. The convolution kernel size is 5 in the first layer and 3 in the following layers. The first, third, and fifth convolutional layers have a stride of 2. The number of channels increases from 32 in the first convolutional layer to 256 in the last. Fully-connected layers contain 512 units each. All modules with the exception of the image module are implemented as standard multilayer perceptrons. We used ReLU nonlinearities after all hidden layers, performed batch normalization after convolutional layers, applied 50% dropout after fully-connected hidden layers, and used 20% dropout after convolutional layers. The datails of the networks are detailed on Table 3.1.

Actions are two-dimensional vectors that collate steering angle and acceleration: $\mathbf{a} = \langle s, a \rangle$. Given a predicted action **a** and a ground truth action $\mathbf{a}_{\text{gt}}$, the per-sample loss function is defined as

$$\begin{aligned} \ell(\mathbf{a}, \mathbf{a}_{\text{gt}}) &= \ell\left(\langle s, a \rangle, \left\langle s_{\text{gt}}, a_{\text{gt}} \right\rangle\right) \\ &= \|s - s_{\text{gt}}\|^2 + \lambda_a \|a - a_{\text{gt}}\|^2. \end{aligned} \tag{3.6}$$

All models were trained using the Adam solver [53] with minibatches of 120 samples and an initial learning rate of 0.0002. For the command-conditional models, minibatches were constructed to contain an equal number of samples with each command.

3.4.3 Training Data Distribution

When performing imitation learning, a key decision is how to collect the training data. The simplest solution is to collect trajectories from natural demonstrations of an expert performing the task. This typically leads to unstable policies, since a model that is only trained on expert trajectories may not learn to recover from disturbance or drift [63, 93].

To overcome this problem, training data should include observations of recoveries from perturbations. In DAgger [93], the expert remains in the loop during the training of the controller: the controller is iteratively tested and samples from the obtained trajectories are re-labeled by the expert. In the system of Bojarski et

al. [11], the vehicle is instrumented to record from three cameras simultaneously: one facing forward and the other two shifted to the left and to the right. Recordings from the shifted cameras, as well as intermediate synthetically reprojected views, are added to the training set – with appropriately adjusted control signals – to simulate recovery from drift.

For our approach we adopt a three-camera setup inspired by Bojarski et al. [11]. However, we have found that the policies learned with this setup are not sufficiently robust. Therefore, to further augment the training dataset, we record some of the data while injecting noise into the expert's control signal and letting the expert recover from these perturbations. This is akin to the recent approach of Laskey et al. [60], but instead of i.i.d. noise we inject temporally correlated noise designed to simulate gradual drift away from the desired trajectory. An example is shown in Figure 3.4. For training, we use the driver's corrective response to the injected noise (not the noise itself). This provides the controller with demonstrations of recovery from drift and unexpected disturbances, but does not contaminate the training set with demonstrations of veering away from desired behavior. The perturbation is a triangular impulse: it increases linearly, reaches a maximal value, and then linearly declines. This simulates gradual drift from the desired trajectory, similar to what might happen with a poorly trained controller. The triangular impulse is parametrized by its starting time t_0, duration $\tau \in \mathbb{R}^+$, sign $\sigma \in \{-1, +1\}$, and intensity $\gamma \in \mathbb{R}^+$:

$$s_{perturb}(t) = \sigma\gamma \max\left(0, \left(1 - \left|\frac{2(t-t_0)}{\tau} - 1\right|\right)\right). \tag{3.7}$$

Every second of driving we started a perturbation with probability $p_{perturb}$. We used $p_{perturb} = 0.1$ in our experiments. The sign of each perturbation was sampled at random, the duration was sampled uniformly from 0.5 to 2 seconds, and intensity was fixed to 0.15.

3.4.4 Data Augmentation

We found data augmentation to be crucial for good generalization. We perform augmentation online during network training. For each image to be presented to the network, we apply a random subset of a set of transformations with randomly sampled magnitudes. Transformations include change in contrast, brightness, and tone, as well as addition of Gaussian blur, Gaussian noise, salt-and-pepper noise, and region dropout (masking out a random set of rectangles in the image, each rectangle taking roughly 1% of image area). No geometric augmentations such as translation or rotation were applied, since control commands are not invariant to

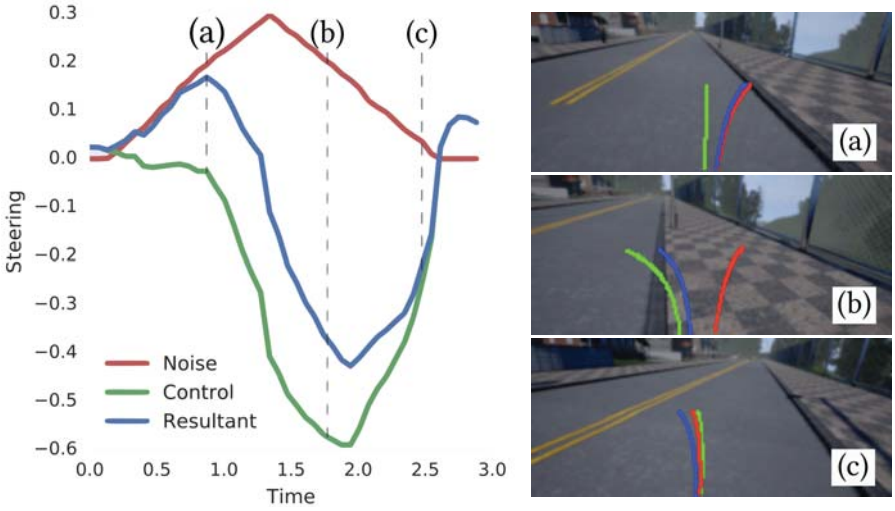


Figure 3.4 – Noise injection during data collection. We show a fragment from an actual driving sequence from the training set. The plot on the left shows steering control [rad] versus time [s]. In the plot, the red curve is an injected triangular noise signal, the green curve is the driver's steering signal, and the blue curve is the steering signal provided to the car, which is the sum of the driver's control and the noise. Images on the right show the driver's view at three points in time (trajectories overlaid post-hoc for visualization). Between times 0 and roughly 1.0, the noise produces a drift to the right, as illustrated in image (a). This triggers a human reaction, from 1.0 to 2.5 seconds, illustrated in (b). Finally, the car recovers from the disturbance, as shown in (c). Only the driver-provided signal (green curve on the left) is used for training.

these transformations.

3.4.5 Training details.

We trained all networks with Adam [53]. We used mini-batches of 120 samples. We balanced the mini-batches, using the same number of samples for each command. Our starting learning rate was 0.0002 and it was multiplied by 0.5 every 50,000 mini-batch iterations. We trained for 294,000 iterations in total. Momentum parameters were set to $\beta_1 = 0.7$ and $\beta_2 = 0.85$. We used no weight decay, but performed 50% dropout after hidden fully-connected layers and 20% dropout on convolutional layers. To further reduce overfitting, we performed extensive data augmentation

by adding Gaussian blur, additive Gaussian noise, pixel dropout, additive and multiplicative brightness variation, contrast variation, and saturation variation. Before feeding a raw 800×600 image to the network, we cropped 171 pixels at the top and 45 at the bottom, and then resized the resulting 800×384 image to a resolution of 200×88.

3.5 Experiments

3.5.1 Simulated Environment

Experimental setup

The use of the CARLA simulator enables running the evaluation in an episodic setup. In each episode, the agent is initialized at a new location and has to drive to a given destination point, given high-level turn commands from a topological planner. An episode is considered successful if the agent reaches the goal within a fixed time interval. In addition to success rate, we measured driving quality by recording the average distance travelled without infractions (collisions or veering outside the lane).

The two CARLA towns used in our experiments are illustrated on Fig. 2.2 from Chapter 2, Section 2.3. Town 1 is used for training, Town 2 is used exclusively for testing. For evaluation, we used 50 pairs of start and goal locations set at least 1 km apart, in each town.

Our training dataset comprises 2 hours of human driving in Town 1 of which only 10% (roughly 12 minutes) contain demonstrations with injected noise. Collecting training data with strong injected noise was quite exhausting for the human driver. However, a relatively small amount of such data proved very effective in stabilizing the learned policy. We only collected training data on the Clear Noon weather.

Results

We compare the `branched` command-conditional architecture, as shown in Figure 3.3(b), with two baseline approaches, as well as several ablated versions of the full architecture. The two baselines are standard imitation learning and goal-conditioned imitation learning. In standard (non-conditional) imitation learning, the action **a** is predicted from the observation **o** and the measurement **m**. In the goal-conditional variant, the controller is additionally provided with a vector pointing to the goal, in the car's coordinate system (the architecture follows Figure 3.3(a)). Ablated versions include: a network with the `command input` architecture instead of `branched` (see Figure 3.3), and three variants of the `branched` network: trained

Model	Success rate		Km per infraction	
	Town 1	Town 2	Town 1	Town 2
Non-conditional	20%	26%	5.76	0.89
Goal-conditional	24%	30%	1.87	1.22
Ours branched	**88%**	**64%**	2.34	1.18
Ours cmd. input	78%	52%	3.97	1.30
Ours no noise	56%	22%	1.31	0.54
Ours no aug.	80%	0%	4.03	0.36
Ours shallow net	46%	14%	0.96	0.42

Table 3.2 – Results in the simulated urban environment. We compare the presented method to baseline approaches and perform an ablation study. We measure the percentage of successful episodes and the average distance (in km) driven between infractions. Higher is better in both cases, but we rank methods based on success. The proposed `branched` architecture outperforms the baselines and the ablated versions.

without noise-injected data, trained without data augmentation, and implemented with a shallower network.

The results are summarized in Table 3.2. The controller that is trained using standard imitation learning only completes 20% of the episodes in Town 1 and 24% in Town 2, which is not surprising given its ignorance of the goal. More interestingly, the goal-conditional controller, which is provided with an accurate vector to the goal at every time step during both training and at test time, is performing only slightly better than the non-conditional controller, successfully completing 24% of the episodes in Town 1 and 30% in Town 2. Qualitatively, this controller eventually veers off the road attempting to shortcut to the goal. This also decreases the number of kilometers the controller is able to traverse without infractions. A simple feed-forward network does not automatically learn to convert a vector pointing to the goal into a sequence of turns.

The proposed `branched` command-conditional controller performs significantly better than the baseline methods in both towns, successfully completing 88% of the episodes in Town 1 and 64% in Town 2. In terms of distance travelled without infractions, in Town 2 the method is on par with baselines, while in Town 1 it is outperformed by the non-conditional model. This difference is misleading: the non-conditional model drives more cleanly because it is not constrained to travel

towards the goal and therefore typically takes a simpler route at each intersection.

The ablation study shown in the bottom part of Table 3.2 reveals that all components of the proposed system are important for good performance. The `branched` architecture reaches the goal more reliably than the `command input` one. The addition of even a small amount of training data with noise in the steering dramatically improves the performance. (Recall that we have only 12 minutes of noisy data out of the total of 2 hours.) Careful data augmentation is crucial for generalization, even within Town 1, but much more so in the previously unseen Town 2: the model without data augmentation was not able to complete a single episode there. Finally, a sufficiently deep network is needed to learn the perceptuomotor mapping in the visually rich and complex simulated urban environment.

3.5.2 Physical System

Experimental setup

The training dataset consists of 2 hours of driving the truck via remote control in a residential area. Figure 3.5 shows a map with the route on which the vehicle was evaluated. The route includes a total of 14 intersections with roughly the same number of `left`, `straight`, and `right`.

We measure the performance in terms of missed intersections, interventions, and time to complete the course. If the robotic vehicle misses an intersection for the first time, it is rerouted to get a second chance to do the turn. If it manages to do the turn the second time, this is not counted as a missed intersection but increases the time taken to complete the route. However, if the vehicle misses the intersection for the second time, this is counted as missed and we intervene to drive the vehicle through the turn manually. Besides missed intersections, we also intervene if the vehicle goes off the road for more than five seconds or if it collides with an obstacle. The models were all evaluated in overcast weather conditions. The majority of training data was collected in sunny weather.

Main results

We select the most important comparisons from the extensive evaluation performed in simulation (Section 3.5.1) and perform them on the physical system. Table 3.3 shows the results of several variants of command-conditional imitation learning: `branched` and `command input` architectures, as well as two ablated models, trained without data augmentation or without noise-injected data. It is evident that the `branched` architecture achieves the best performance. The ablation experiments show the impact of our noise injection method and augmentation strategy. The model trained without noise injection is very unstable, as indicated by the

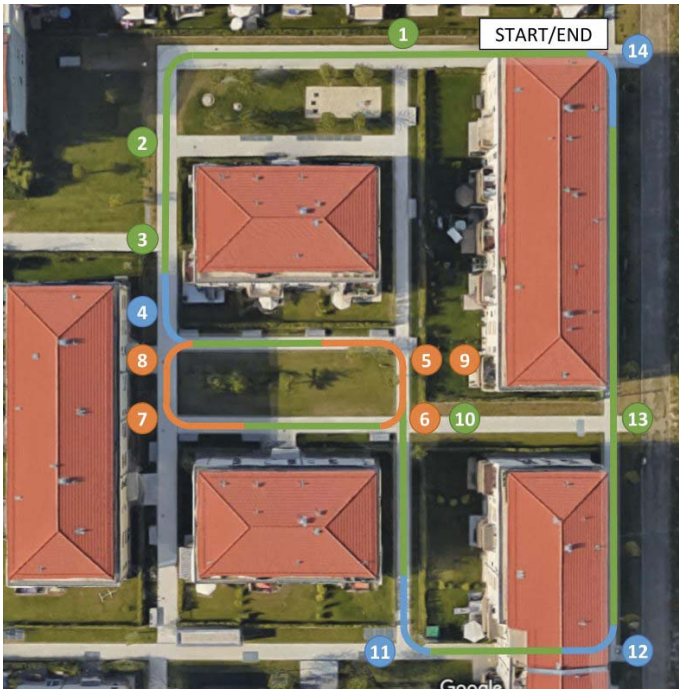


Figure 3.5 – A map of the primary route used for testing the physical system. Intersections traversed by the truck are numbered according to their order along the route. Colors indicate commands provided to the vehicle when it approaches the intersection: blue = `left`, green = `straight`, orange = `right`.

average number of interventions rising from 0.67 to 8.67. Moreover, it misses almost 25% of the intersections and takes double the time to complete the course. The model trained without data augmentation fails completely. The truck misses most intersections and very frequently leaves the lane resulting in almost 40 interventions. It takes more than four times longer to complete the course. This extreme degradation highlights the importance of generalization in real world settings with constantly changing environmental conditions such as weather and lighting. Proper data augmentation dramatically improves performance given limited training data.

Model	Missed turns	Interventions	Time
Ours branched	**0%**	**0.67**	**2:19**
Ours cmd. input	11.1%	2.33	4:13
Ours no noise	24.4%	8.67	4:39
Ours no aug.	73%	39	10:41

Table 3.3 – Results on the physical system. Lower is better. We compare the `branched` model to the simpler `command input` architecture and to ablated versions (without noise injection and without data augmentation). Average performance across 3 runs is reported for all models except for "Ours no aug.", for which we only performed 1 run to avoid breaking the truck.

Generalization to new environments

Beyond the implicit generalization to varying weather conditions that occur naturally in the physical world, we also evaluate qualitatively how well the model generalizes to previously unseen environments with very different appearance. To this end, we run the truck in three environments shown in Figure 3.6. The truck is able to consistently follow the lane in all tested environments and is responsive to commands.

Figure 3.6 – Testing in new environments with very different appearance.

3.6 Discussion

We proposed command-conditional imitation learning: an approach to learning from expert demonstrations of low-level controls and high-level commands. At training time, the commands resolve ambiguities in the perceptuomotor mapping, thus facilitating learning. At test time, the commands serve as a communication channel that can be used to direct the controller.

We applied the presented approach to vision-based driving of a physical robotic vehicle and in realistic simulations of dynamic urban environments. Our results show that the command-conditional formulation significantly improves performance in both scenarios.

While the presented results are encouraging, they also reveal that significant room for progress remains. In particular, more sophisticated and higher-capacity architectures along with larger datasets will be necessary to support autonomous urban driving on a large scale. On Chapter 5 we develop more into this idea of improving conditional imitation learning performance. We hope that the presented approach to making driving policies more controllable will prove useful in such deployment.

Our work has not addressed human guidance of autonomous vehicles using natural language: a mode of human-robot communication that has been explored in the literature [14, 45, 74, 110, 114]. We leave unstructured natural language communication with autonomous vehicles as an important direction for future work.

4 On Offline Evaluation of Vision-based Driving Models

Autonomous driving models should ideally be evaluated by deploying them on a fleet of physical vehicles in the real world. Unfortunately, this approach is not practical for the vast majority of researchers. An attractive alternative is to evaluate models offline, on a pre-collected validation dataset with ground truth annotation. In this chapter, we investigate the relation between various online and offline metrics for evaluation of autonomous driving models. We find that offline prediction error is not necessarily correlated with driving quality, and two models with identical prediction error can differ dramatically in their driving performance. We show that the correlation of offline evaluation with driving quality can be significantly improved by selecting an appropriate validation dataset and suitable offline metrics.

4.1 Introduction

Camera-based autonomous driving can be viewed as a computer vision problem. It requires analyzing the input video stream and estimating certain high-level quantities, such as the desired future trajectory of the vehicle or the raw control signal to be executed. Standard methodology in computer vision is to evaluate an algorithm by collecting a dataset with ground-truth annotation and evaluating the results produced by the algorithm against this ground truth (Figure 4.1(a)). However, driving, in contrast with most computer vision tasks, is inherently active. That is, it involves interaction with the world and other agents. The end goal is to drive well: safely, comfortably, and in accordance with traffic rules. An ultimate evaluation would involve deploying a fleet of vehicles in the real world and executing the model on these (Figure 4.1(b)). The logistical difficulties associated with such an evaluation lead to the question: Is it possible to evaluate a driving model without actually letting it drive, but rather following the offline dataset-centric methodology?

One successful approach to evaluation of driving systems is via decomposition. It stems from the modular approach to driving where separate subsystems

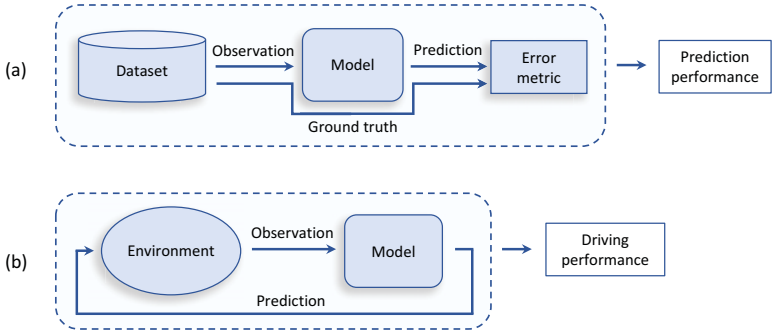


Figure 4.1 – Two approaches to evaluation of a sensorimotor control model. Top: offline (passive) evaluation on a fixed dataset with ground-truth annotation. Bottom: online (active) evaluation with an environment in the loop.

deal with sub-problems, such as environment perception, mapping, and vehicle control. The perception stack provides high-level understanding of the scene in terms of semantics, 3D layout, and motion. These lead to standard computer vision tasks, such as object detection, semantic segmentation, depth estimation, 3D reconstruction, or optical flow estimation, which can be evaluated offline on benchmark datasets [21, 33, 88]. This approach has been extremely fruitful, but it only applies to modular driving systems.

Recent deep learning approaches [11, 119] aim to replace modular pipelines by end-to-end learning from images to control commands. The decomposed evaluation does not apply to models of this type. End-to-end methods are commonly evaluated by collecting a large dataset of expert driving [119] and measuring the average prediction error of the model on the dataset. This offline evaluation is convenient and is consistent with standard practice in computer vision, but how much information does it provide about the actual driving performance of the models?

In this chapter, we empirically investigate the relation between (offline) prediction accuracy and (online) driving quality. We train a diverse set of models for urban driving in realistic simulation [26] and correlate their driving performance with various metrics of offline prediction accuracy. By doing so, we aim to find offline evaluation procedures that can be executed on a static dataset, but at the same time correlate well with driving quality. We empirically discover best practices both in terms of selection of a validation dataset and the design of an error metric. Additionally, we investigate the performance of several models on the real-world Berkeley DeepDrive Video (BDDV) urban driving dataset [119].

Our key finding is that offline prediction accuracy and actual driving quality are surprisingly weakly correlated. This correlation is especially low when prediction is evaluated on data collected by a single forward-facing camera on expert driving trajectories – the setup used in most existing works. A network with very low prediction error can be catastrophically bad at actual driving. Conversely, a model with relatively high prediction error may drive well.

We found two general approaches to increasing this poor correlation between prediction and driving. The first is to use more suitable validation data. We found that prediction error measured in lateral cameras (sometimes mounted to collect additional images for imitation learning) better correlates with driving performance than prediction in the forward-facing camera alone. The second approach is to design offline metrics that depart from simple mean squared error (MSE). We propose offline metrics that correlate with driving performance more than 60% better than MSE.

4.2 Related Work

Vision-based autonomous driving tasks have traditionally been evaluated on dedicated annotated real-world datasets. For instance, KITTI [33] is a comprehensive benchmarking suite with annotations for stereo depth estimation, odometry, optical flow estimation, object detection, semantic segmentation, instance segmentation, 3D bounding box prediction, etc. The Cityscapes dataset [21] provides annotations for semantic and instance segmentation. The BDDV dataset [119] includes semantic segmentation annotation. For some tasks, ground truth data acquisition is challenging or nearly impossible in the physical world (for instance, for optical flow estimation). This motivates the use of simulated data for training and evaluating vision models, as in the SYNTHIA [91], Virtual KITTI [32], and GTA5 datasets [89], and the VIPER benchmark [88]. These datasets and benchmarks are valuable for assessing the performance of different components of a vision pipeline, but they do not allow evaluation of a full driving system.

Recently, increased interest in end-to-end learning for driving has led to the emergence of datasets and benchmarks for the task of direct control signal prediction from observations (typically images). To collect such a dataset, a vehicle is equipped with one or several cameras and additional sensors recording the coordinates, velocity, sometimes the control signal being executed, etc. The Udacity dataset [112] contains recordings of lane following in highway and urban scenarios. The CommaAI dataset [96] includes 7 hours of highway driving. The Oxford RobotCar Dataset [73] includes over 1000 km of driving recoded under varying weather, lighting, and traffic conditions. The BDDV dataset [119] is the largest publicly

available urban driving dataset to date, with 10,000 hours of driving recorded from forward-facing cameras together with smartphone sensor data such as GPS, IMU, gyroscope, and magnetometer readings. These datasets provide useful training data for end-to-end driving systems. However, due to their static nature (passive pre-recorded data rather than a living environment), they do not support evaluation of the actual driving performance of the learned models.

Online evaluation of driving models is technically challenging. In the physical world, tests are typically restricted to controlled simple environments [20, 51] and qualitative results [11, 84]. Large-scale real-world evaluations are impractical for the vast majority of researchers. One alternative is simulation. Due of its logistical feasibility, simulation have been commonly employed for driving research, especially in the context of machine learning. The TORCS simulator [118] focuses on racing, and has been applied to evaluating road following [18]. Rich active environments provided by computer games have been used for training and evaluation of driving models [27]; however, the available information and the controllability of the environment are typically limited in commercial games. The recent CARLA driving simulator [26] allows evaluating driving policies in living towns, populated with vehicles and pedestrians, under different weather and illumination conditions. In this work we use CARLA to perform an extensive study of offline performance metrics for driving.

Although the analysis we perform is applicable to any vision-based driving pipeline (including ones that comprise separate perception [13, 50, 90, 100, 127] and control modules [82]), in this chapter we focus on end-to-end trained models. This line of work dates back to the ALVINN model of Pomerleau [84], capable of road following in simple environments. More recently, LeCun et al. [61] demonstrated collision avoidance with an end-to-end trained deep network. Chen et al. [18] learn road following in the TORCS simulator, by introducing an intermediate representation of "affordances" rather than going directly from pixels to actions. Bojarski et al. [11] train deep convolutional networks for lane following on a large real-world dataset and deploy the system on a physical vehicle. Fernando et al. [30] use neural memory networks combining visual inputs and steering wheel trajectories to perform long-term planning, and use the CommaAI dataset to validate the method. Hubschneider et al. [47] incorporate turning signals as additional inputs to their DriveNet. Codevilla et al. [20] propose conditional imitation learning, which allows imitation learning to scale to complex environments such as urban driving by conditioning action prediction on high-level navigation commands. The growing interest in end-to-end learning for driving motivates our investigation of the associated evaluation metrics.

4.3 Methodology

We aim to analyze the relation between offline prediction performance and online driving quality. To this end, we train models using conditional imitation learning [20], presented on Chapter 3, at the CARLA environment, presented on Chapter 2. We then evaluate the driving quality on goal-directed navigation and correlate the results with multiple offline prediction-based metrics. We now describe the methods used to train and evaluate the models.

4.3.1 Training

Data collection. We collect a training dataset by executing an automated navigation expert in the simulated environment. The expert makes use of privileged information about the environment, including the exact map of the environment and the exact positions of the ego-car, all other vehicles, and pedestrians. The expert keeps a constant speed of 35 km/h when driving straight and reduces the speed when making turns. We record the images from three cameras: a forward-facing one and two lateral cameras facing 30 degrees left and right. In 10% of the data we inject noise in the driving policy to generate examples of recovery from perturbations. In total we record 80 hours of driving data.

Action representation. The most straightforward approach to end-to-end learning for driving is to output the raw control command, such as the steering angle, directly [11, 20]. We use this representation in most of our experiments. The action is then a vector $\mathbf{a} \in \mathbb{R}^3$, consisting of the steering angle, the throttle value, and the brake value. To simplify the analysis and preserve compatibility with prior work [11, 119], we only predict the steering angle with a deep network. We use throttle and brake values provided by the expert policy described above.

Loss function. In most of our experiments we follow standard practice [11, 20] and use mean squared error (MSE) as a per-sample loss:

$$\ell(F(\mathbf{o}_i, \mathbf{c}_i, \boldsymbol{\theta}), \mathbf{a}_i) = \|F(\mathbf{o}_i, \mathbf{c}_i, \boldsymbol{\theta}) - \mathbf{a}_i\|^2. \tag{4.1}$$

We have also experimented with the L1 loss. In most experiments we balance the data during training. We do this by dividing the data into 8 bins based on the ground-truth steering angle and sampling an equal number of datapoints from each bin in every mini-batch. As a result, the loss being optimized is not the average MSE over the dataset, but its weighted version with higher weight given to large steering angles.

Regularization. Even when evaluating in the environment used for collecting the training data, a driving policy needs to generalize to previously unseen views of this

environment. Generalization is therefore crucial for a successful driving policy. We use dropout and data augmentation as regularization measures when training the networks.

Dropout ratio is 0.2 in convolutional layers and 0.5 in fully-connected layers. For each image to be presented to the network, we apply a random subset of a set of transformations with randomly sampled magnitudes. Transformations include contrast change, brightness, and tone, as well as the addition of Gaussian blur, Gaussian noise, salt-and-pepper noise, and region dropout (masking out a random set of rectangles in the image, each rectangle taking roughly 1% of image area). The applied transformations are similar to the ones used on Chapter 3. In order to ensure good convergence, we found it helpful to gradually increase the data augmentation magnitude in proportion to the training step.

Model architecture. We experiment with a feedforward convolutional network, which takes as input the current observation as well as an additional vector of measurements (in our experiments the only measurement is the current speed of the vehicle). This network implements a purely reactive driving policy, since by construction it cannot make use of temporal context. We experiment with three variants of this model. The architecture presented on Chapter 3, Sec. 3.4.1, with 8 convolutional layers, is denoted as "standard". We also experiment with a deeper architecture with 12 convolutional layers and a shallower architecture with 4 convolutional layers.

4.3.2 Performance metrics

Offline error metrics. Assume we are given a validation set $\mathcal{V}$ of tuples $\langle \mathbf{o}_i, \mathbf{c}_i, \mathbf{a}_i, v_i \rangle$, indexed by $i \in V$. Each tuple includes an observation, an input command, a ground-truth action vector, and the speed of the vehicle. We assume the validation set consists of one or more temporally ordered driving sequences. (For simplicity in what follows we assume it is a single sequence, but generalization to multiple sequences is trivial.) Denote the action predicted by the model by $\hat{\mathbf{a}}_i = F(\mathbf{o}_i, \mathbf{c}_i, \boldsymbol{\theta})$. In our experiments, $\mathbf{a}_i$ and $\hat{\mathbf{a}}_i$ will be scalars, representing the steering angle. Speed is also a scalar (in m/s).

Table 4.1 lists offline metrics we evaluate in this chapter. The first two metrics are standard: mean squared error (which is typically the training loss) and absolute error. Absolute error gives relatively less weight to large mistakes than MSE.

The higher the speed of the car, the larger the impact a control mistake can have. To quantify this intuition, we evaluate speed-weighted absolute error. This metric approximately measures how quickly the vehicle is diverging from the ground-truth trajectory, that is, the projection of the velocity vector onto the direction orthogonal to the heading direction.

Table 4.1 – Offline metrics used in the evaluation. δ is the Kronecker delta function, θ is the Heaviside step function, Q is a quantization function (see text for details), $|V|$ is the number of samples in the validation dataset.

Metric name	Parameters	Metric definition
Squared error	–	$\frac{1}{\lvert V\rvert}\sum_{i\in V}\lVert\mathbf{a}_i-\hat{\mathbf{a}}_i\rVert^2$
Absolute error	–	$\frac{1}{\lvert V\rvert}\sum_{i\in V}\lVert\mathbf{a}_i-\hat{\mathbf{a}}_i\rVert_1$
Speed-weighted absolute error	–	$\frac{1}{\lvert V\rvert}\sum_{i\in V}\lVert\mathbf{a}_i-\hat{\mathbf{a}}_i\rVert_1 v_i$
Cumulative speed-weighted absolute error	T	$\frac{1}{\lvert V\rvert}\sum_{i\in V}\lVert\sum_{t=0}^{T}(\mathbf{a}_{i+t}-\hat{\mathbf{a}}_{i+t})v_{i+t}\rVert_1$
Quantized classification error	σ	$\frac{1}{\lvert V\rvert}\sum_{i\in V}\left(1-\delta\left(Q(\mathbf{a}_i,\sigma),Q(\hat{\mathbf{a}}_i,\sigma)\right)\right)$
Thresholded relative error	α	$\frac{1}{\lvert V\rvert}\sum_{i\in V}\theta\left(\lVert\hat{\mathbf{a}}_i-\mathbf{a}_i\rVert-\alpha\lVert\mathbf{a}_i\rVert\right)$

We derive the next metric by accumulating speed-weighted errors over time. The intuition is that the average prediction error may not be characteristic of the driving quality, since it does not take into account the temporal correlations in the errors. Temporally uncorrelated noise may lead to slight oscillations around the expert trajectory, but can still result in successful driving. In contrast, a consistent bias in one direction for a prolonged period of time inevitably leads to a crash. We therefore accumulate the speed-weighted difference between the ground-truth action and the prediction over T time steps. This measure is a rough approximation of the divergence of the vehicle from the desired trajectory over T time steps.

Another intuition is that small noise may be irrelevant for the driving performance, and what matters is getting the general direction right. Similar to Xu et al. [119], we quantize the predicted actions and evaluate the classification error. For quantization, we explicitly make use of the fact that the actions are scalars (although a similar strategy can be applied to higher-dimensional actions). Given a threshold value σ, the quantization function $Q(x,\sigma)$ returns -1 if $x<-\sigma$, 0 if $-\sigma\leq x<\sigma$, and 1 if $x\geq\sigma$. For steering angle, these values correspond to going left, going straight, and going right. Given the quantized predictions and the ground truth, we compute the classification error.

Finally, the last metric is based on quantization and relative errors. Instead of quantizing with a fixed threshold as in the previous metric, here the threshold is adaptive, proportional to the ground truth steering signal. The idea is that for large action values, small discrepancies with the ground truth are not as important as for small action values. Therefore, we count the fraction of samples for which $\lVert\hat{\mathbf{a}}_i-\mathbf{a}_i\rVert\geq\alpha\lVert\mathbf{a}_i\rVert$.

Online performance metrics. We measure the driving quality using three metrics. The first one is the success rate, or simply the fraction of successfully completed navigation trials. The second is the average fraction of distance traveled towards the goal per episode (this value can be negative is the agent moves away form the goal). The third metric measures the average number of kilometers traveled between two infractions. (Examples of infractions are collisions, driving on the sidewalk, or driving on the opposite lane.)

4.4 Experiments

We perform an extensive study of the relation between online and offline performance of driving models. Since conducting such experiments in the real world would be impractical, the bulk of the experiments are performed in the CARLA simulator [26]. We start by training a diverse set of driving models with varying architecture, training data, regularization, and other parameters. We then correlate online driving quality metrics with offline prediction-based metrics, aiming to find offline metrics that are most predictive of online driving performance. Finally, we perform an additional analysis on the real-world BDDV dataset.

4.4.1 Experimental setup

Simulation. We use the CARLA 0.8.4 simulator to evaluate the performance of driving models in an urban environment. We follow the testing protocol presented on Chapter 2. We evaluate goal-directed navigation with dynamic obstacles. One evaluation includes 25 goal-directed navigation trials.

CARLA provides two towns (Town 1 and Town 2) and configurable weather and lighting conditions. We make use of this capability to evaluate generalization of driving methods. We use Town 1 in 4 weathers (Clear Noon, Heavy Rain Noon, Clear Sunset and Clear After Rain) for training data collection, and we use two test conditions: Town 1 in clear noon weather and Town 2 in Soft Rain Sunset weather. The first condition is present in the training data; yet, note that the specific images observed when evaluating the policies have almost certainly not been seen during training. Therefore even this condition requires generalization. The other condition – Town 2 and soft rain sunset weather – is completely new and requires strong generalization.

For validation we use 2 hours of driving data with action noise and 2 hours of data without action noise, in each of the conditions. With three cameras and a frame rate of 10 frames per second, one hour of data amounts to $108K$ validation images.

Real-world data. For real-world tests we use the validation set of the BDDV dataset [119], containing 1816 dashboard camera videos. We computed the offline metrics over the entire dataset using the pre-trained models and the data filtering procedures provided by Xu et al. [119].
Network training and evaluation. All models were trained using the Adam optimizer [53] with minibatches of 120 samples and an initial learning rate of 10^{-4}. We reduce the learning rate by a factor of 2 every $50K$ iterations. All models were trained up to $500K$ iterations. In order to track the evolution of models during the course of training, for each model we perform both online and offline evaluation after the following numbers of training mini-batches: 2K, 4K, 8K, 16K, 32K, 64K, 100K, 200K, 300K, 400K, and 500K.

4.4.2 Evaluated models

We train a total of 45 models. The parameters we vary can be broadly separated into three categories: properties of the training data, of the model architecture, and of the training procedure. We vary the amount and the distribution of the training data. The amount varies between 0.2 hours and 80 hours of driving. The distribution is one of the following four: all data collected from three cameras and with noise added to the control, only data from the central camera, only data without noise, and data from the central camera without noise. The model architecture variations amount to varying the depth between 4 and 12 layers. The variations in the training process are the use of data balancing, the loss function, and the regularization applied (dropout and the level of data augmentation). A complete list of parameters varied during the evaluation is provided on Table 4.2

4.4.3 Correlation between offline and online metrics

We start by studying the correlation between online and offline performance metrics on the whole set of evaluated models. We represent the results by scatter plots and correlation coefficients. To generate a scatter plot, we select two metrics and plot each evaluated model as a circle, with the coordinates of the center of the circle equal to the values of these two metrics, and the radius of the circle proportional to the training iteration the model was evaluated at. To quantify the correlations, we use the standard sample Pearson correlation coefficient, computed over all points in the plot. In the figures below, we plot results in generalization conditions (Town 2, unseen weather). We focus our analysis on the well-performing models, by discarding the 50% worst models according to the offline metric.
The effect of validation data. We first plot the (offline) average steering MSE versus the (online) success rate in goal-directed navigation, for different offline validation

Training data	
Amount	The amount of training data (in hours of driving footage)
Distribution	Distribution of the data used for training
3 cam. + noise	Data from three cameras, with action noise in 10% of data
3 cam.	Data from three cameras, without action noise
1 cam. + noise	Data only from the central camera, with action noise
1 cam.	Data only from the central camera, without action noise
Model architecture	
Architecture	The architecture of the network
Shallow	4-layer convolutional network
Standard	8-layer convolutional network
Deep	12-layer convolutional network
Training procedure	
Regularization	Regularization techniques applied during training
None	No regularization
Mild Dropout	Dropout: 50% in the FC layers of the measurements and control modules
High Dropout	Dropout: 50% in all FC layers
Drop. + aug.	Dropout and random transformations applied to input images
Balancing	If data balancing w.r.t. steering angles is applied during training
Loss	Type of loss function used for training
MSE (L2)	Regression with MSE loss
L1	Regression with absolute error (L1)

Table 4.2 – Parameters of driving models explored in the evaluation.

datasets. We vary the number of cameras used for validation (just a forward-facing camera or three cameras including two lateral ones) and the presence of action noise in the validation set. This experiment is inspired by the fact that the 3-camera setup and the addition of noise have been advocated for training end-to-end driving models [11, 20, 26, 119].

The results are shown in Figure 4.2. The most striking observation is that the correlation between offline prediction and online performance is weak. For the basic setup – central camera and no action noise – the absolute value of the correlation coefficient is only 0.39. The addition of action noise improves the correlation to 0.54. Evaluating on data from three cameras brings the correlation up to 0.77. This shows that a successful policy must not only predict the actions of an expert on the expert's trajectories, but also for observations away from the expert's trajectories. Proper validation data should therefore include examples of recovery from perturbations.

Offline metrics. Offline validation data from three cameras or with action noise may not always be available. Therefore, we now aim to find offline metrics that are predictive of driving quality even when evaluated in the basic setup with a single forward-facing camera and no action noise.

Figure 4.3 shows scatter plots of offline metrics described in Section 4.3.2, versus the navigation success rate. MSE is the least correlated with the driving success rate: the absolute value of the correlation coefficient is only 0.39. Absolute steering

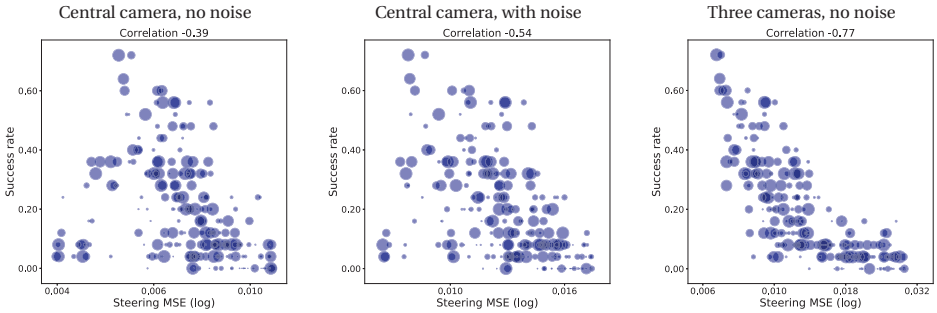


Figure 4.2 – Scatter plots of goal-directed navigation success rate vs. steering MSE when evaluated on data from different distributions. We evaluate the models in the generalization condition (Town 2) and we plot the 50% best-performing models according to the offline metric. Sizes of the circles denote the training iterations at which the models were evaluated. We additionally show the sample Pearson correlation coefficient for each plot. Note how the error on the basic dataset (single camera, no action noise) is the least informative of the driving performance.

error is more strongly correlated, at 0.61. Surprisingly, weighting the error by speed or accumulating the error over multiple subsequent steps does not improve the correlation. Finally, quantized classification error and thresholded relative error are also more strongly correlated, with the absolute value of the correlation coefficient equal to 0.65 and 0.64, respectively.

Online metrics. So far we have looked at the relation between offline metrics and a single online metric – success rate. Is success rate fully representative of actual driving quality? Here we compare the success rate with two other online metrics: average fraction of distance traveled towards the goal and average number of kilometers traveled between two infractions.

Figure 4.4 shows pairwise scatter plots of these three online metrics. Success rate and average completion are strongly correlated, with a correlation coefficient of 0.8. The number of kilometers traveled between two infractions is similarly correlated with the success rate (0.77), but much less correlated with the average completion (0.44). We conclude that online metrics are not perfectly correlated and it is therefore advisable to measure several online metrics when evaluating driving models. Success rate is well correlated with the other two metrics, which justifies its use as the main online metric in our analysis.

Case study. We have seen that even the best-correlated offline and online metrics have a correlation coefficient of only 0.65. Aiming to understand the reason for this

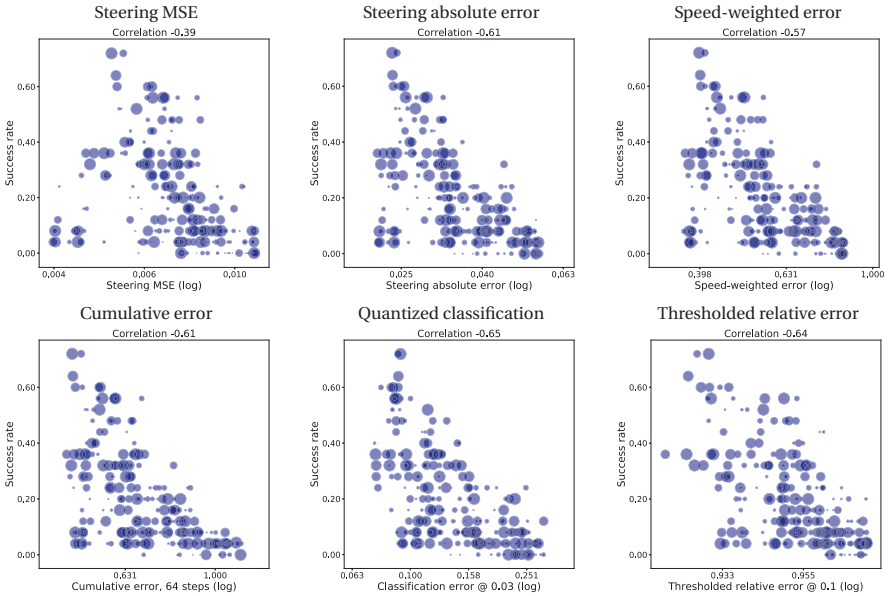


Figure 4.3 – Scatter plots of goal-directed navigation success rate vs. different offline metrics. We evaluate the models in the generalization condition (Town 2) and we plot the 50% best-performing models according to the offline metric. Note how correlation is generally weak, especially for mean squred error (MSE).

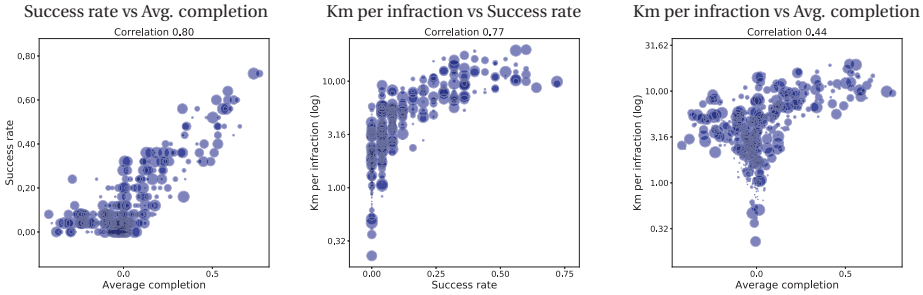


Figure 4.4 – Scatter plots of online driving quality metrics versus each other. The metrics are: success rate, average fraction of distance to the goal covered (average completion), and average distance (in km) driven between two infractions. Success rate is strongly correlated with the other two metrics, which justifies its use as the main online metric in our analysis.

Model	Average accuracy				Weighted with speed	
	All data	Straight	Stop	Turns	All data	Turns
Feedforward	78.0	90.0	72.0	32.4	80.7	27.7
CNN + LSTM	81.8	90.2	78.1	49.3	83.0	43.2
FCN + LSTM	83.3	90.4	80.7	50.7	83.6	44.4

Table 4.3 – Detailed accuracy evaluations on the BDDV dataset. We report the 4-way classification accuracy (in %) for various data subsets and varying speed.

remaining discrepancy, here we take a closer look at two models which achieve similar prediction accuracy, but drastically different driving quality. The first model was trained with the MSE loss and forward-facing camera only. The second model used the L1 loss and three cameras. We refer to these models as Model 1 and Model 2, respectively.

Figure 4.5 (top left) shows the ground truth steering signal over time (blue), as well as the predictions of the models (red and green, respectively). There is no obvious qualitative difference in the predictions of the models: both often deviate from the ground truth. One difference is a large error in the steering signal predicted by Model 1 in a turn, as shown in Figure 4.5 (top right). Such a short-term discrepancy can lead to a crash, and it is difficult to detect based on average prediction error. The advanced offline metrics evaluated above are designed to be better at capturing such mistakes.

Figure 4.5 (bottom) shows several trajectories driven by both models. Model 1 is able to drive straight for some period of time, but eventually crashes in every single trial, typically because of wrong timing or direction of a turn. In contrast, Model 2 drives well and successfully completes most trials. This example illustrates the difficulty of using offline metrics for predicting online driving behavior.

4.4.4 Real-world data

Evaluation of real-world urban driving is logistically complicated, therefore we restrict the experiments on real-world data to an offline evaluation. We use the BDDV dataset and the trained models provided by [119]. The models are trained to perform 4-way classification (accelerate, brake, left, right), and we measure their classification accuracy. We evaluate on the validation set of BDDV.

The offline metrics we presented above are designed for continuous values and cannot be directly applied to classification-based models. Yet, some of them can be adapted to this discrete setting. Table 4.3 shows the average accuracy, as well as several additional metrics. First, we provide a breakdown of classification

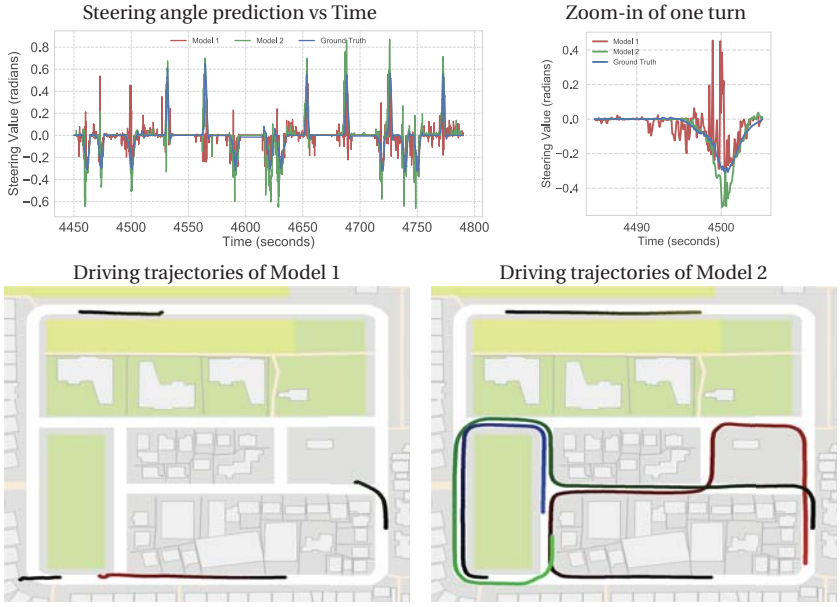


Figure 4.5 – Detailed evaluation of two driving models with similar offline prediction quality, but very different driving behavior. Top left: Ground-truth steering signal (blue) and predictions of two models (red and green) over time. Top right: a zoomed fragment of the steering time series, showing a large mistake made by Model 1 (red). Bottom: Several trajectories driven by the models in Town 1. Same scenarios indicated with the same color in both plots. Note how the driving performance of the models is dramatically different: Model 1 crashes in every trial, while Model 2 can drive successfully.

accuracy by subsets of the data corresponding to different ground truth labels. The prediction error in the turns is most informative, yielding the largest separation between the best and the worst models. Second, we try weighting the errors with the ground-truth speed. We measure the resulting metric for the full validation dataset, as well as for turns only. These metrics reduce the gap between the feedforward and the LSTM models.

4.4.5 Detailed evaluation of models

Scatter plots presented in the previous sections indicate general tendencies, but not the performance of specific models. Here we provide a more detailed evaluation of several driving models, with a focus on several parameters: the amount of training data, its distribution, the regularization being used, the network architecture, and the loss function. We evaluate two offline metrics – MSE and the thresholded relative error (TRE) – as well as the goal-directed navigation success rate. For TRE we use the parameter $\alpha = 0.1$.

The results are shown in Table 4.4. In each section of the table all parameters are fixed, except for the parameter of interest. (Parameters may vary between sections.) Driving performance is sensitive to all the variations. Larger amount of training data generally leads to better driving. Training with one or three cameras has a surprisingly minor effect. Data balancing helps in both towns. Regularization helps generalization to the previously unseen town and weather. Deeper networks generally perform better. Finally, the L1 loss leads to better driving than the usual MSE loss. This last result is in agreement with Figure 4.3, which shows that absolute error is better correlated with the driving quality than MSE.

Next, for each of the 6 parameters and each of the 2 towns we check if the best model chosen based on the offline metrics is also the best in terms of the driving quality. This simulates a realistic parameter tuning scenario a practitioner might face. We find that TRE is more predictive of the driving performance than MSE, correctly identifying the best-driving model in 10 cases out of 12, compared to 6 out of 12 for MSE. This demonstrates that TRE, although far from being perfectly correlated with the online driving quality, is much more indicative of well-driving models than MSE.

4.5 Additional results

On section 4.4, we evaluate the models in the generalization condition (Town 2) and we plot 50% best-performing models according to the offline metric. Here we show results in the training condition (Town 1) and show plots with all models, not

Parameter	Value	MSE Town 1	MSE Town 2	TRE @ 0.1 Town 1	TRE @ 0.1 Town 2	Success rate Town 1	Success rate Town 2
Amount of training data	0.2 hours	0.0086	0.0481	0.970	0.985	0.44	0.00
	1 hour	0.0025	0.0217	0.945	0.972	0.44	0.04
	5 hours	**0.0005**	**0.0093**	0.928	0.961	0.60	**0.08**
	25 hours	0.0007	0.0166	**0.926**	**0.958**	**0.76**	0.04
Type of training data	1 cam., no noise	0.0007	**0.0066**	**0.922**	0.947	**0.84**	0.04
	1 cam., noise	0.0009	0.0077	0.926	**0.946**	0.80	**0.20**
	3 cam., no noise	**0.0004**	0.0086	0.928	0.953	**0.84**	0.08
	3 cam., noise	0.0007	0.0166	0.926	0.958	0.76	0.04
Data balancing	No balancing	0.0012	**0.0065**	0.907	**0.924**	0.88	0.36
	With balancing	**0.0011**	0.0066	**0.891**	0.930	**0.92**	**0.56**
Regularization	None	0.0014	0.0092	**0.911**	0.953	**0.92**	0.08
	Mild dropout	0.0010	0.0074	0.921	0.953	0.84	0.20
	High dropout	**0.0007**	0.0166	0.926	0.958	0.76	0.04
	High drop., data aug.	0.0013	**0.0051**	0.919	**0.931**	0.88	**0.36**
Network architecture	Shallow	**0.0005**	0.0111	0.936	0.963	0.68	0.12
	Standard	0.0007	0.0166	**0.926**	0.958	**0.76**	0.04
	Deep	0.0011	**0.0072**	0.928	**0.949**	**0.76**	**0.24**
Loss function	L2	**0.0010**	0.0074	0.921	0.953	0.84	0.20
	L1	0.0012	**0.0061**	**0.891**	**0.944**	**0.96**	**0.52**

Table 4.4 – Detailed evaluation of models in CARLA. “TRE” stands for thresholded relative error, “Success rate” for the driving success rate. For MSE and TRE lower is better, for the success rate higher is better. We mark with bold the best result in each section. We highlight in green the cases where the best model according to an offline metric is also the best at driving, separately for each section and each town. Both MSE and TRE are not necessarily correlated with driving performance, but generally TRE is more predictive of driving quality, correctly identifying 10 best-driving models out of 12, compared to 6 out of 12 for MSE.

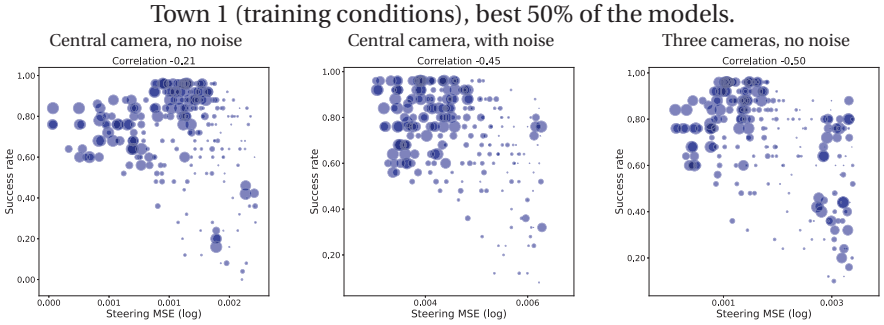


Figure 4.6 – Scatter plots of goal-directed navigation success rate vs steering absolute error when evaluated on data from different distributions. Town 1 (training conditions), best 50% of the models.

only best-performing ones.

Figures 4.6 and 4.7 show scatter plots of online vs offline metrics with 50% best models, evaluated in Town 1. Figure 4.8 shows scatter plots of online driving quality metrics, evaluated in Town 1. Figures 4.9 and 4.10 show scatter plots of online vs offline metrics with all models, evaluated in Town 1. Figures 4.11 and 4.12 show scatter plots of online vs offline metrics with all models, evaluated in Town 2.

4.6 Conclusion

We investigated the performance of offline versus online evaluation metrics for autonomous driving. We have shown that the MSE prediction error of expert actions is not a good metric for evaluating the performance of autonomous driving systems, since it is very weakly correlated with actual driving quality. We explore two avenues for improving the offline metrics: modifying the validation data and modifying the metrics themselves. Both paths lead to improved correlation with driving quality.

Our work takes a step towards understanding the evaluation of driving models, but it has several limitations that can be addressed in future work. First, the evaluation is almost entirely based on simulated data. We believe that the general conclusion about weak correlation of online and offline metrics is likely to transfer to the real world; however, it is not clear if the details of our correlation analysis will hold in the real world. Performing a similar study with physical vehicles operating in rich real-world environments would therefore be very valuable. Second, we focus on the correlation coefficient as the measure of relation between two quantities.

Figure 4.7 – Scatter plots of goal-directed navigation success rate vs different offline metrics. Town 1 (training conditions), best 50% of the models.

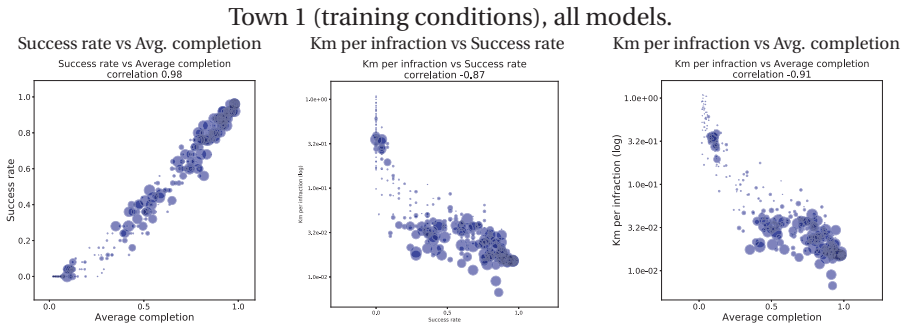


Figure 4.8 – Scatter plots of online driving quality metrics versus each other. The metrics are: success rate, average fraction of distance to the goal covered (average completion), and average distance (in km) driven between two infractions. Town 1 (training conditions), all models.

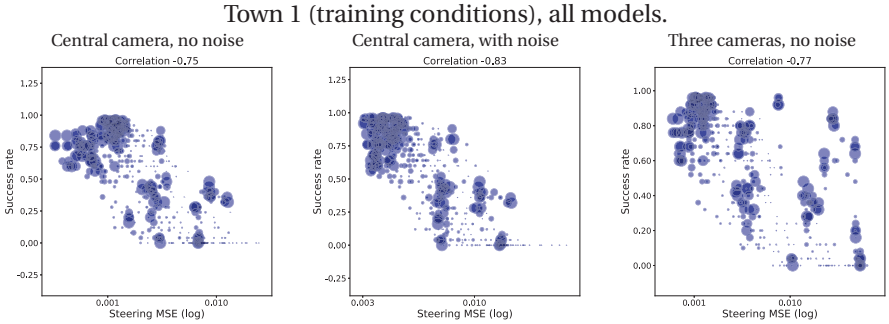


Figure 4.9 – Scatter plots of goal-directed navigation success rate vs steering absolute error when evaluated on data from different distributions. Town 1 (training conditions), all models.

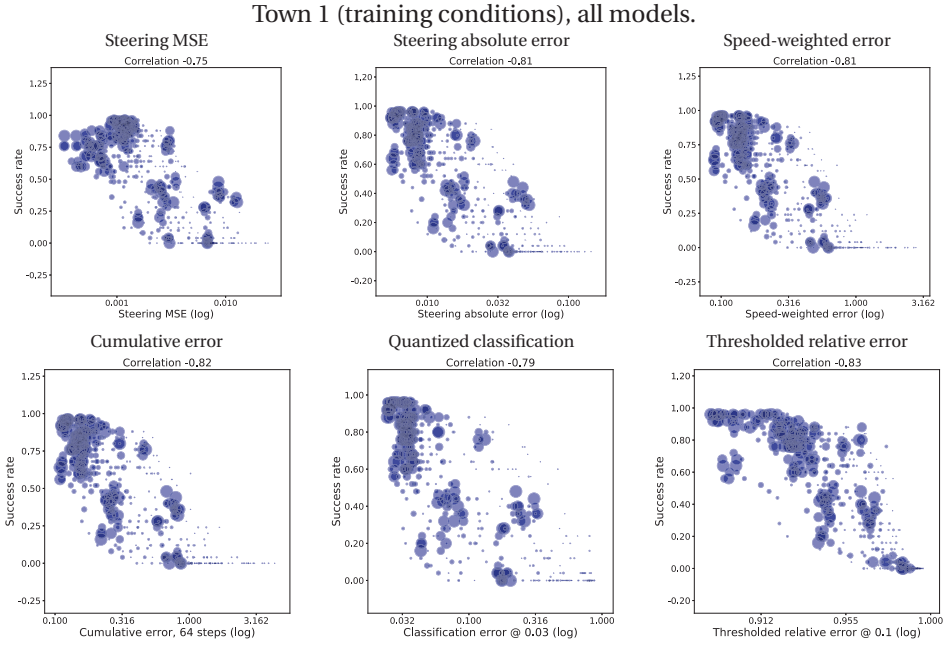


Figure 4.10 – Scatter plots of goal-directed navigation success rate vs different offline metrics. Town 1 (training conditions), all models.

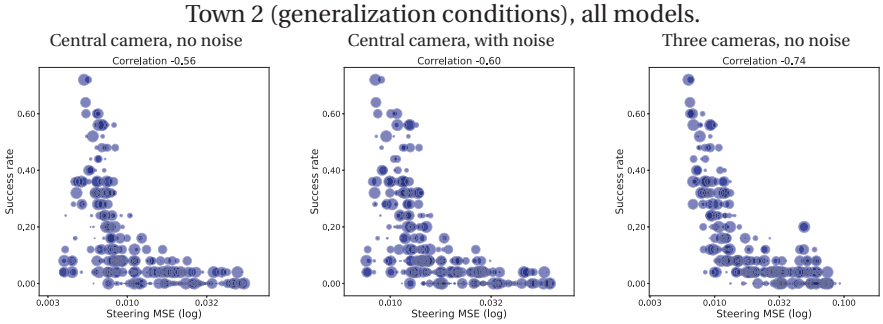


Figure 4.11 – Scatter plots of goal-directed navigation success rate vs steering absolute error when evaluated on data from different distributions. Town 2 (generalization conditions), all models.

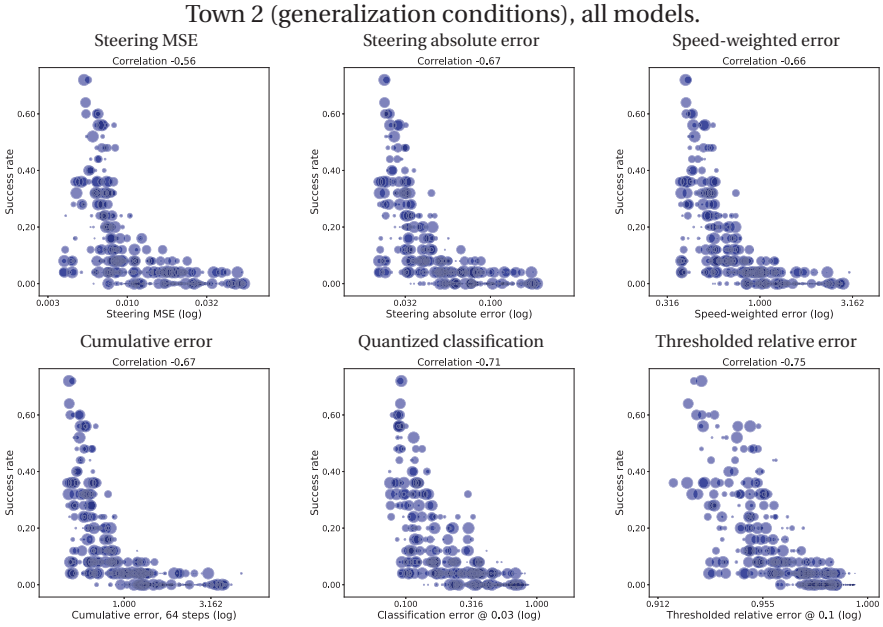


Figure 4.12 – Scatter plots of goal-directed navigation success rate vs different offline metrics. Town 2 (generalization conditions), all models.

Correlation coefficient estimates the connection between two variables to some degree, but a finer-grained analysis may be needed provide a more complete understanding of the dependencies between online and offline metrics. Third, even the best offline metric we found is far from being perfectly correlated with actual driving quality. Designing offline performance metrics that are more strongly correlated with driving performance remains an important challenge.

5 Exploring the Limitations of Behavior Cloning for Autonomous Driving

Figure 5.1 – Driving scenarios from our new benchmark where the agent needs to react to dynamic changes in the environment, handle clutter (only part of the environment is causally relevant), and predict complex sensorimotor controls (lateral and longitudinal). We show that Behavior Cloning yields state-of-the-art policies in these complex scenarios and investigate its limitations.

Driving requires reacting to a wide variety of complex environmental conditions and agent behaviors. Explicitly modeling each possible scenario is unrealistic. In contrast, imitation learning can, in theory, leverage data from large fleets of human-driven cars. Behavior cloning in particular has been successfully used to learn simple visuomotor policies end-to-end, but scaling to the full spectrum of driving behaviors remains open. In this chapter, we propose a new benchmark to experimentally investigate the scalability and limitations of this approach. We show that behavior cloning leads to state-of-the-art results, executing complex lateral and longitudinal maneuvers without these reactions being explicitly programmed. However, we confirm well-known generalization issues (due to dataset bias and overfitting), new ones (due to dynamic objects and the lack of a causal model), and training instability; problems requiring further research before behavior cloning can scale to real-world driving.

5.1 Introduction

End-to-end imitative systems can suffer a domain shift between the off-line training experience and the on-line behavior [93]. This problem, however, can be addressed

in practice by data augmentation [11, 20]. Nonetheless, in spite of the early and recent successes of behavior cloning for end-to-end driving [11, 17, 20, 61, 84], it has not yet proved to scale to the full spectrum of driving behaviors, such as reacting to multiple dynamic objects.

In this chapter, we propose to use our new benchmark, called *NoCrash*, presented on Chapter 2 and perform a large scale analysis of end-to-end behavioral cloning systems in complex driving conditions not studied in this context before. We perform a large scale off-line training and on-line evaluation in over 80 hours of driving under several different conditions. We describe a strong Conditional Imitation Learning baseline, an expansion to the one presented on Chapter 3, that significantly improves upon state of the art modular [65], affordance based [97], and reinforcement learning [67] approaches, both in terms of generalization performance in training environments and unseen ones.

Despite its positive performance, we identify limitations that prevent behavior cloning from successfully graduating to real-world applications. First, although generalization performance should scale with training data, generalizing to complex conditions is still an open problem with a lot of room for improvement. In particular, we show that no approach reliably handles dense traffic scenes with many dynamic agents. Second, we report generalization issues due to dataset biases and the lack of a causal model. We indeed observe diminishing returns after a certain amount of demonstrations, and even characterize a degradation of performance on unseen environments. Third, we observe a significant variability in generalization performance when varying the initialization or the training sample order, similar to on-policy RL issues [46]. We conduct experiments estimating the impact of ImageNet pre-training and show that it is not able to fully reduce the variance. This suggests the order of training samples matters for off-policy Imitation Learning, similar to the on-policy case [123].

This chapter is organized as follows. Section 5.2 describes related work, Section 5.3 our strong behavior cloning baseline, Section 5.4 our experimental results, and Section 5.5 our conclusion.

5.2 Related Work

Behavior cloning for driving dates back to the work of Pomerleau [84] on lane following, later followed by other approaches [61], including going beyond driving [2, 107]. The distributional shift between the training and testing distributions is the main known limitation of this approach, which might require *on-policy* data collection [92, 93], obtained by the learning agent. Nonetheless, recent works have proposed effective *off-policy* solutions, for instance by expanding the space

of image/action pairs either using noise [20, 60], extra sensors [11], or modularization [66, 97]. We show, however, that there are other limitations important to consider in complex driving scenarios, in particular dataset bias and high variance, which both harm scaling generalization performance with training data.

Dataset bias is a core problem of real-world machine learning applications [7, 111] that can have dramatic effects in a safety-critical application like autonomous driving. Imitation learning approaches are particularly sensitive to this issue, as the learning objective might be dominated by the main modes in the training data. Going beyond the original CARLA benchmark [26], we use our new NoCrash benchmark to quantitatively assess the magnitude of this problem on generalization performance for more realistic and challenging driving behaviors.

High variance is a key problem in powerful deep neural networks, and we show that high performance behavior cloning models are particularly suffering from this. This problem is related to sensitivity to both initialization and sampling order [79], reproducibility issues in Reinforcement Learning [46, 72], and the need to move beyond the i.i.d. data assumption towards curriculum learning [9] for sensorimotor control [5, 123].

Driving benchmarks fall in two main categories: off-line datasets, e.g., [33, 44, 96, 119], or on-line environments. We focus here on on-line benchmarks, as visuomotor models performing well in dataset-based evaluations do not necessarily translate to good driving policies [19]. Driving is obviously a safety-critical robotic application. Consequently, for safety and to enable reproducibility, researchers focus on using photo-realistic simulation environments. In particular, the CARLA open-source driving simulator [26] is emerging as a standard platform for driving research, used in [20, 66, 67, 77, 97]. Note, however, that transferring policies from simulation to the real-world is an open problem [71] out of the scope of this chapter, although recent works have shown encouraging results [77, 120].

5.3 Behavior Cloning

In this section, we first describe the behavior cloning framework we use, its limitations, and a robustified baseline that tries to tackle these issues.

5.3.1 Conditional Imitation Learning

Behavior cloning [64, 84, 93, 98] is a form of supervised learning that can learn sensorimotor policies from off-line collected data. The only requirements are pairs of input sensory observations associated with expert actions. We use an expanded formulation for self-driving cars called Conditional Imitation Learning, proposed

on Chapter 3. It uses a high-level navigational command $\mathbf{c}$ that disambiguates imitation around multiple types of intersections. Given an expert policy with access to the environment state x, we can execute this policy to produce a dataset, $D = \{\langle \mathbf{o}_i, \mathbf{c}_i, \mathbf{a}_i \rangle\}_{i=1}^{N}$, where $\mathbf{o}_i$ are sensor data observations, $\mathbf{c}_i$ are high-level commands (e.g., take the next right, left, or stay in lane) and $\mathbf{a}_i = \pi^*(x_i)$ are the resulting vehicle actions (low-level controls). Observations $\mathbf{o}_i = \{i, v_m\}$ contain a single image i and the ego car speed v_m [20] added for the system to properly react to dynamic objects on the road. Without the speed context, the model cannot learn if and when it should accelerate or brake to reach a desired speed or stop.

We want to learn a policy F parametrized by $\boldsymbol{\theta}$ to produce similar actions to the reference policy based only on observations $\mathbf{o}$ and high-level commands $\mathbf{c}$. The best parameters $\boldsymbol{\theta}^*$ are obtained by minimizing an imitation cost ℓ:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_i \ell\big(F(\mathbf{o}_i, \mathbf{c}_i; \boldsymbol{\theta}), \mathbf{a}_i\big). \tag{5.1}$$

In order to evaluate the performance of the learned policy $F(\mathbf{o}_i, \mathbf{c}_i; \boldsymbol{\theta})$ on-line at test time, we assume access to a score function giving a numeric value expressing the performance of the function approximator F on a given benchmark (cf. chap. 2 section 2.4).

5.3.2 Limitations

In addition to the distributional shift problem [93], behavior cloning presents some key limitations.

Bias in Naturalistic Driving Datasets. The appeal of behavior cloning lies in its simplicity and theoretical scalability, as it can indeed learn by imitation from large off-line collected demonstrations (e.g., using driving logs from manually driven production vehicles). It is, however, susceptible to dataset biases like all learning methods. This is exacerbated in the case of imitation learning of driving policies, as most of real-world driving consists in either a few simple behaviors or a heavy tail of complex reactions to rare events. Consequently, this can result in performance degrading as more data is collected, because the diversity of the dataset does not grow fast enough compared to the main mode of demonstrations. This phenomenon was not clearly measured before. Using our new *NoCrash* benchmark (chapter 2, section 2.4.2), we confirm it may happen in practice.

Causal Confusion. Related to dataset bias, end-to-end behavior cloning can suffer from causal confusion [39]: spurious correlations cannot be distinguished from

true causes in observed training demonstration patterns unless an explicit causal model or on-policy demonstrations are used. Our new *NoCrash* benchmark confirms the theoretical observation and toy experiments of [39] in realistic driving conditions. In particular, we identify a typical failure mode due to a subtle dataset bias: the *inertia problem*. When the ego vehicle is stopped (e.g., at a red traffic light), the probability it stays static is indeed overwhelming in the training data. This creates a spurious correlation between low speed and no acceleration, inducing excessive stopping and difficult restarting in the imitative policy. Although mediated perception approaches that explicitly model causal signals like traffic lights do not suffer from this theoretical limitation, they still under-perform end-to-end learning in unconstrained environments, because not all causes might be modeled (e.g., some potential obstacles) and errors at the perception layer (e.g., missed detections) are irrecoverable.

High variance. With a fixed off-policy training dataset, one would expect CIL to always learn the same policy in different runs of the training phase. However, the cost function is optimized via Stochastic Gradient Descent (SGD), which assumes the data is independent and identically distributed [12]. When training a reactive policy on snapshots of longer human demonstrations included in the training data, the i.i.d. assumption does not hold. Consequently, we might observe a high sensitivity to the initialization and the order in which the samples are seen during training. We confirm this in our experiments, finding an overall high variance due to both initialization and sampling order, following the decomposition in [79]:

$$Var(\pi) = E_D\big[Var_I(F|D)\big] + Var_D\big(E_I[F|D]\big), \tag{5.2}$$

where I denotes the randomness in initialization. Because the function F is evaluated on-line in simulated environments, we evaluate in practice the variance of the score on the test benchmark, and report results when freezing the initialization and/or varying the sampling order for different training datasets D (including of varying sizes).

5.3.3 Model

In order to explore the aforementioned limitations of behavior cloning, we propose a robustified CIL model designed to improve on the model from Chapter 3 while remaining strictly off-policy. Our network architecture, called CILRS, is shown in Figure 5.2. We describe our enhancements below.

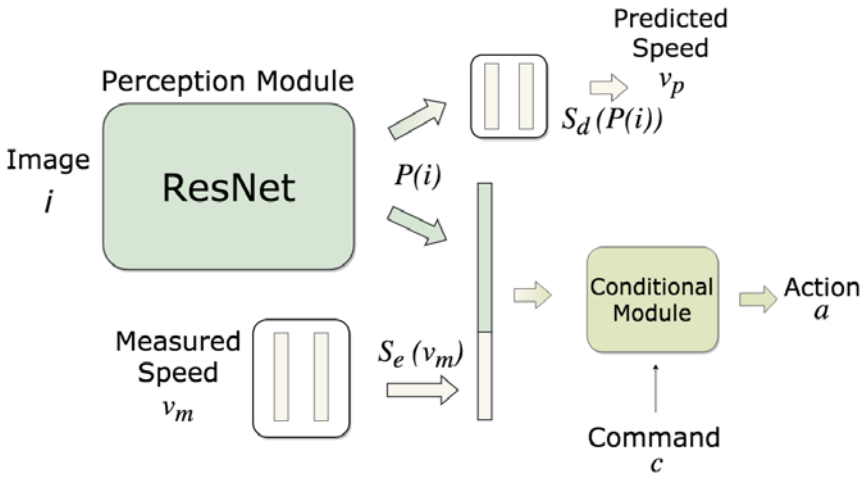


Figure 5.2 – Our proposed network architecture, called CILRS, for end-to-end urban driving, based on the one presented on Chapter 3. A ResNet perception module processes an input image to a latent space followed by two prediction heads: one for controls and one for speed.

Deeper Residual Architecture. We use a ResNet34 architecture [43] for the perception backbone $P(i)$. In the presence of large amounts of data, using deeper architectures can be an effective strategy to improve performance [43]. In particular, it can reduce *both* bias *and* variance, maintaining in particular a constant variance due to training set sampling with both network width and depth [79]. For end-to-end driving, the choice of architecture has been mostly limited to small networks so far [11, 20, 97] to avoid overfitting on limited datasets. In contrast, we notice that bigger models have better generalization performance on learning reactions to dynamic objects and traffic lights in complex urban environments.

Speed Prediction Regularization. To cope with the inertia problem without an explicit mapping of potential causes or on-policy interventions, we jointly train a sensorimotor controller with a network that predicts the ego vehicle's speed. Both neural networks share the same representation via our ResNet perception backbone. Intuitively, what happens is that this joint optimization enforces the perception module to have speed related features into the learned representation. This reduces the dependency on input speed as the only way to get dynamics of the

scene, leveraging instead visual cues that are predictive of the car's velocity (e.g., free space, curves, traffic light states, etc).

Other changes. We use L1 as loss function ℓ instead of the mean squared error (MSE), as it is more correlated to driving performance [19]. As our *NoCrash* benchmark consists of complex realistic driving conditions in the presence of dynamic agents, we collect demonstrations from an expert game AI using privileged information to drive correctly (i.e. mostly respecting rules of the road and not crashing into any obstacle). Robustness to heavy noise in the demonstrations is beyond the scope of our work, as we aim to explore limitations of behavior cloning methods *in spite of* good demonstrations. Finally, we pre-trained our perception backbone on ImageNet to reduce initialization variance and benefit from generic transfer learning, a standard practice in deep learning seldom explored for behavior cloning.

5.4 Experiments

In this section we detail our protocol for model training and briefly show that it is capable of being on par with the state-of-the-art. We also explore several corner cases to stress the limitations of the behavior cloning approach.

5.4.1 Training Details

As the usual methodology for the training autonomous driving controllers, we collect an exhaustive amount of data on a single town of the CARLA simulated environment. More details about the dataset can be found at the appendix A.1. From this dataset we trained our controller using 10 hours of expert demonstrations. We found that augmentation was not as crucial as for the setup used on Chapter 3 and previous works [20, 66]. The only regularization we found important for performance was using a 50% dropout after the last convolutional layer. Any larger dropout led us to under-fitting models. All models were trained using Adam [53] with minibatches of 120 samples and an initial learning rate of 0.0002. At each iteration, a minibatch is sampled randomly from the entire dataset and presented to the network for training. If we detect that the training error hasn't decreased for over 1000 iterations we divide the learning rate by 10. We used a 2 hours validation dataset to discover when to stop the training process We validate every 20k iterations and if the validation error increases for three iterations we stale the training process and use this checkpoint to test on the benchmarks, both CARLA and *NoCrash*. To build the validation dataset we used the recommendations from the conclusions obtained on Chapter 4.

Scenario	Task	CIL[20]	CIRL[67]	CAL[97]	MT[66]	CILR	CILRS
Training Conditions	Straight	98	98	**100**	96	94	96
	One Turn	89	97	**97**	87	92	92
	Navigation	86	93	92	81	88	**95**
	Nav. Dynamic	83	82	83	81	85	**92**
New Weather	Straight	98	**100**	**100**	**100**	96	96
	One Turn	90	94	**96**	88	**96**	**96**
	Navigation	84	86	90	88	94	**96**
	Nav. Dynamic	82	80	82	80	92	**96**
New Town	Straight	97	**100**	93	**100**	92	96
	One Turn	59	71	82	81	81	**84**
	Navigation	40	53	70	**72**	60	69
	Nav. Dynamic	38	41	64	53	55	**66**
New Weather & Town	Straight	80	**98**	94	96	92	96
	One Turn	48	80	72	82	92	**92**
	Navigation	44	68	68	78	88	**92**
	Nav. Dynamic	42	62	64	62	82	**90**

Table 5.1 – Comparison with the state of the art on the CoRL 2017 benchmark (chapter 2, section 2.4.1). The "CILRS" version corresponds to our CIL-based ResNet using the speed prediction branch, whereas "CILR" is without this speed prediction. These two models and CIL are the only ones that do not use any extra supervision or online interaction with the environment during training. The table reports the percentage of successfully completed episodes in each condition, selecting the best seed out of five runs.

5.4.2 Comparison with the state of the art

We compare our results using both the original CARLA benchmark [26] and *NoCrash* benchmark (cf. chapter 2 section 2.4). We compare two versions of our method: "CILRS" (our CIL extension with a ResNet architecture and speed prediction, as described in section 5.3), and a version without the speed prediction branch noted "CILR". We compare our method with the original CIL from [20], Chapter 3, and three state-of-the-art approaches: CAL [97], MT [66], and CIRL [67]. In contrast to end-to-end behavior cloning, these methods enforce some modularization that require extra information at training time, such as affordances (CAL), semantic segmentation (MT), or extra on-policy interaction with the environment (CIRL). Our approach only requires a fixed off-policy dataset of demonstrations.

We show results on the original CARLA benchmark in Table 5.1 and results on *NoCrash* benchmark in Table 5.2. While most methods perform well in most conditions on the original CARLA benchmark, they all perform significantly worse

Scenario	Task	CIL[20]	CAL[97]	MT[66]	CILR	CILRS
Training Conditions	Empty	79 ± 1	81 ± 1	84 ± 1	92 ± 1	$\mathbf{97} \pm 2$
	Normal	60 ± 1	73 ± 2	54 ± 2	72 ± 5	$\mathbf{83} \pm 0$
	Cluttered	21 ± 2	$\mathbf{42} \pm 3$	13 ± 4	28 ± 1	$\mathbf{42} \pm 2$
New Weather	Empty	83 ± 2	85 ± 2	58 ± 2	$\mathbf{98} \pm 1$	96 ± 1
	Normal	55 ± 5	68 ± 5	40 ± 6	69 ± 4	$\mathbf{77} \pm 1$
	Cluttered	13 ± 4	33 ± 2	7 ± 2	27 ± 3	$\mathbf{47} \pm 5$
New Town	Empty	48 ± 3	36 ± 6	41 ± 3	60 ± 2	$\mathbf{66} \pm 2$
	Regular	27 ± 1	26 ± 2	22 ± 0	42 ± 2	$\mathbf{49} \pm 5$
	Cluttered	10 ± 2	9 ± 1	7 ± 1	12 ± 2	$\mathbf{23} \pm 1$
New Weather & Town	Empty	24 ± 1	25 ± 3	57 ± 0	66 ± 2	$\mathbf{90} \pm 2$
	Normal	13 ± 2	14 ± 2	32 ± 2	54 ± 2	$\mathbf{56} \pm 2$
	Cluttered	2 ± 0	10 ± 0	14 ± 2	13 ± 4	$\mathbf{24} \pm 8$

Table 5.2 – Results on our *NoCrash* benchmark (chapter 2 sec. 2.4.2. Mean and standard deviation on three runs, as CARLA 0.8.4 has significant non-determinism.

on *NoCrash*, especially when trying to generalize to new conditions. This confirms the usefulness of *NoCrash* in terms of exploring the limitations of driving policy learning due to its more challenging nature.

In addition, our proposed CILRS model significantly improves over the state of the art, e.g., +9% and +26% on CARLA "Nav. Dynamic" in training and new weather & town conditions respectively, +10% and +24% on *NoCrash* Regular traffic in training and new weather & town conditions respectively. As showed on [26], new town conditions are harder than new weather & town and the improvements of the results was smaller. The significant improvements in generalization conditions, both w.r.t. CIL and mediated approaches, confirm that our improved end-to-end behavior cloning architecture can effectively learn complex general policies from demonstrations alone. Furthermore, our ablative analysis shows that speed prediction is helpful: CILR can indeed be up to −14% worse than CILRS on *NoCrash*.

5.4.3 Analysis of Limitations

Although clearly above the state of the art, our improved CILRS architecture nonetheless sees a strong degradation of performance similar to all other methods in the presence of challenging driving conditions. We investigate how this degradation relates to the limitations of behavior cloning mentioned in Section 5.3.2 by using the *NoCrash* benchmark, in particular to better evaluate the interaction of the agents

with dynamic objects.

Generalization in the presence of dynamic objects. Limited generalization was previously reported for end-to-end driving approaches [26]. In our experiments, we observed additional, and more prominent, generalization issues when the control policies have to deal with dynamic objects. Table 5.2 indeed shows a large drop in performance as we change to tasks with more traffic, e.g., −55% and −66% from Empty to Dense traffic in *NoCrash* training / new conditions respectively. In contrast, results in Empty town only degrade by −7% when changing to a new environment and weather. Therefore, the learned policies have a much harder time dealing robustly with a large number of vehicles and pedestrians. Furthermore, this impacts all policy learning methods, including those using additional supervision or on-policy demonstrations, often even more than our proposed CILRS method.

Driving Dataset Biases. Figure 5.3 evaluates the effect of the amount of training demonstrations on the learned policy. Here we compare models trained with 2, 10, 50 and 100 hours of demonstrations. The plots show the mean success rate and standard deviation over four different training cycles with different random seeds. Our best results on most of the scenarios were obtained by using only 10 hours of training data, in particular on the "Dense Traffic" tasks and novel conditions such as New Weather and New Town. These results quantify a limitation described in Section 5.3.2: the risk of overfitting to data that lacks diversity.

This is here exacerbated by the limited spatial extent and visual variety of our environment, including in terms of dynamic objects. We indeed observed that some types of vehicles tend to elicit better reactions from the policy than others. The more common the vehicle model and color, the better the trained agent reacts to it. This raises ethical challenges in automated driving, requiring further research in fair machine learning for decision-making systems [7].

We also observed considerable change in the success rate results when not using ImageNet initialization, as show in Figure 5.6. The highest success rate becomes the ones trained on 100 hours of demonstrations. However, these later results are still below what can be achieved with less data and ImageNet pre-training, specially on the dense traffic tasks.

In order to assure that the the used architectures do not underfit or overfit to the data, we perform ablation studies with deeper and shalower architectures. On Figure 5.7, we compare the results of the 8 layer convolutional model used on Chapter 3 and several ResNet configurations using our new dataset. First, we noticed that the 8 convs model obtained worse results than the ones reported as CIL at Table 5.1. This happened since we trained the 8 convolutions architecture with the more

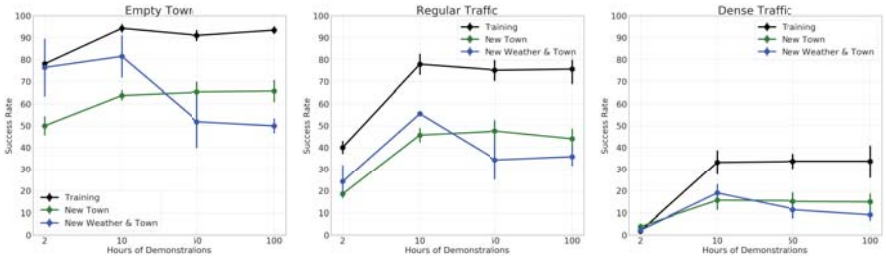


Figure 5.3 – The importance of data for improving the results. We can see a that due to biases present on data that the results get either staled or worse after you increase the amount of data.

complex CARLA100 dataset. The model did not have enough capacity to capture the more complex actions and only fitted the several segments where the demonstrator stands still in front of traffic lights. This shows that higher capacity models are able to better learn different sub-tasks. However, for the deeper ResNet50 the results get worse showing that there is still possibility for even further overfitting without eliminating the dataset bias.

Causal confusion and the inertia problem. The main problem we observe caused by bias is the inertia problem stemming from causal confusion, as detailed in Section 5.3.2. Figure 5.4 shows the percentage of episodes that failed due to the agent staying still, without any intention to use the throttle, for at least 8 seconds before the timeout. Our results show the percentage of episodes failed due to that inertia problem increases with the amount of data used for training. We proposed to use a speed prediction branch as part of our CILRS model (cf. Figure 5.2) to mitigate this problem. Figure 5.8 shows the percentage of successes for the New Weather & Town conditions on different tasks with and without speed prediction. We observe that the speed prediction branch can substantially improve the success rate thanks to its regularization effect. It is, however, not a final solution to this problem, as we still observe instances of the inertia problem after using this approach.

High Variance. Repeatability of the training process is crucial for enhancing trust in end-to-end models. Unfortunately, we can still see drastic changes in the learned policy performance due to the variance caused by initialization and data sampling (cf. Section 5.3.2). Figure 5.9 compares the cause of episode termination for two models where the only difference is the random seed during training. The Model S1 has a much higher chance of ending episodes due to vehicle collisions. Qualitatively,

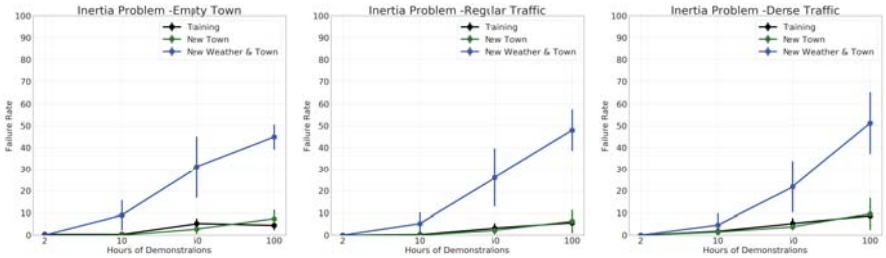


Figure 5.4 – The percentage of episode that failed due to the inertia problem. We can see that by increasing the amount of data, this bias is further enforced degrading specially the generalization capabilities of the models.

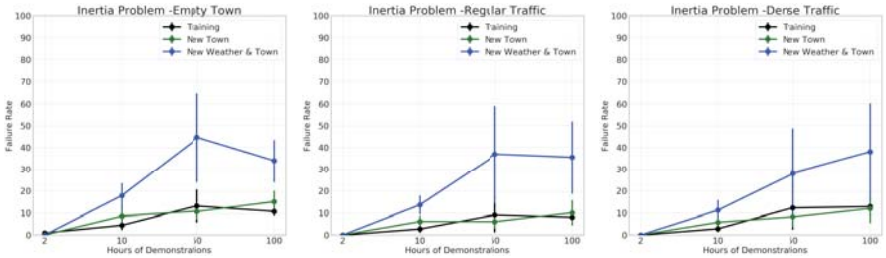


Figure 5.5 – Percentage of episodes ended by the "inertia problem" on different conditions. We report the mean and the standard deviation over four different trainings. We compare models with different amounts of training data and **without** image-net pre-training. We can see that the inertia problem becomes more prominent with more data.

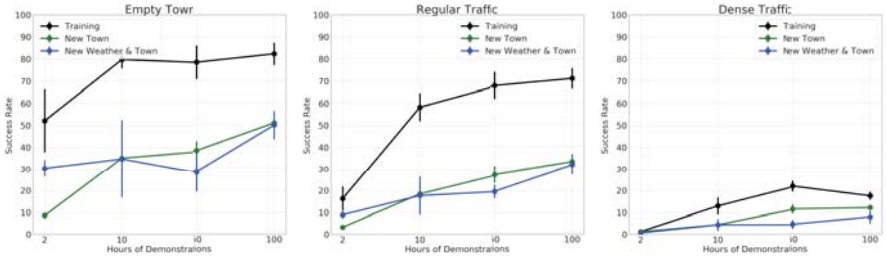


Figure 5.6 – The importance of data **without** ImageNet pre-training. We can see that the results improve with more data but not significant. We can see also one case of worse results as from 50 to 100 hours on the New Weather & Town conditions with Dense Traffic.

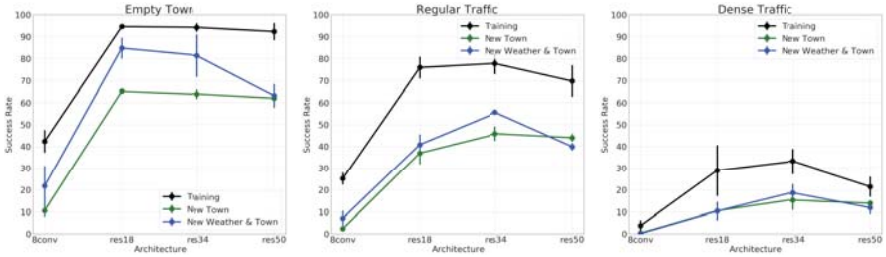


Figure 5.7 – Ablative analysis between different architectures. The eight convolutions architecture,"8conv", proposed by Codevilla [20] obtained poor results on the more complex CARLA100 datasets. ResNet based deeper architectures, "res18" and "res34", were able to improve the results. However, when testing ResNet 50 we notice a significant drop on the quality of the results.

it seemed to have learned a less general braking policy and was more prone to rear-end collisions with other vehicles. On the other hand, Model S2 is able to complete more episodes and is less likely to fail due to vehicle crashes. However, we can see that it times out more, showing a tendency to stop a lot, even in non-threatening situations. This can be seen by analyzing the histograms of the throttle applied by both models during the benchmark, as shown in Figure 5.10. We can see a tendency for throttles of higher magnitude on Model S1.

As off-policy imitation learning uses a static dataset for training, this randomness comes from the order in which training data is sampled and the initialization of the random weights. This can possibly define which minima the models converges to. Table 5.3 quantifies the effect of initialization on the success rate of driving tasks by computing the variance expressed in Equation 5.2. The expected policy score was computed by averaging twelve different training runs. We also consider the variance with and without ImageNet initialization. We can see that the success rate can change by up to 42% for tasks with dynamic objects. ImageNet initialization tends to reduce the training variability, mainly due to smaller randomness on initialization but also due to a more stable learned policy.

5.4.4 Reacting to Traffic Lights

We show that other policies can also emerge for some of the models trained. In Table 5.4 we show the percentage of traffic lights that were crossed on green light for different models. This number is computed for the "Empty Town" task from the dynamic urban scenarios benchmark. The original CIL model trained with older

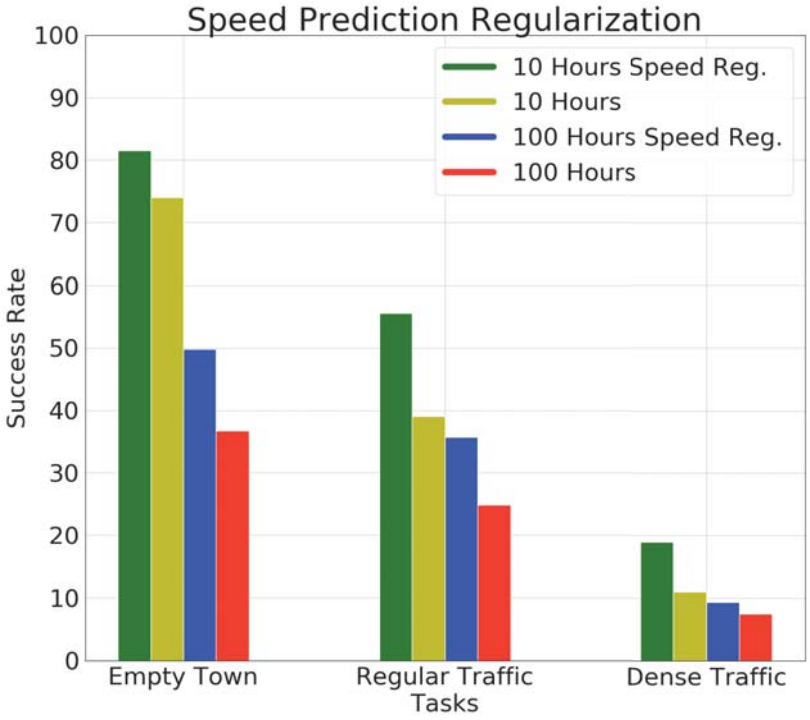


Figure 5.8 – Comparison between the results with and without the speed prediction and different amounts of training demonstrations. We report the results only for the case were highest generalization is needed (New Weather and Town).

data [20] represents an effective policy that was trained without demonstrations of stopping for red traffic light, so the number is actually a lower bound. The 8 convolutions is a model with the same architecture as the CIL model but trained to react to traffic lights. We can see that this model probably was having some reaction to traffic lights, but very poorly. On the other hand, CILRS, having only 47% of traffic light violations, is clearly stopping for a significant amount of red traffic lights. This result is even more expressive considering the version using 100 hours of training data which did only 27% of traffic light violations. However, when we analyze generalization conditions, Tab. 5.4 bottom, we see there is an ample room for improvement.

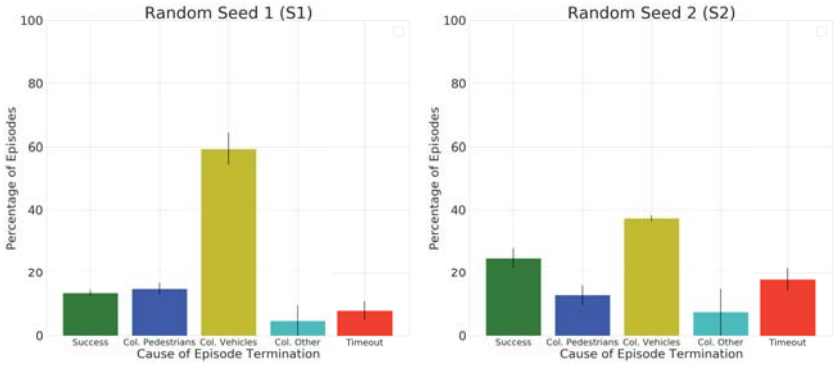


Figure 5.9 – Comparison of the cause of episode termination for two models with identical parameters but different random seeds evaluated in the *NoCrash* benchmark. The models were evaluated under a series of goal directed episodes and the plot shows the success rate for each termination condition. The episodes were ran under "New Weather & Town" conditions of the "Dense Traffic" task. The models correspond to the Res34 10 hours of training data and ImageNet initialization.

	Task	Variation
CILRS	Empty	23%
	Normal	26%
	Cluttered	42%
CILRS ImNet	Empty	4%
	Normal	12%
	Cluttered	38%

Table 5.3 – Estimated variance of the success rate of CILRS on *NoCrash* computed by training 12 times the same model with different random seeds. The variance is reduced by fixing part of the initial weights with ImageNet pre-training.

5.4.5 Main Causes of Failure

On Table 5.5 we show some of our models compared with some of the literature with regard to their cause of failure. We specify the percentage of episodes that

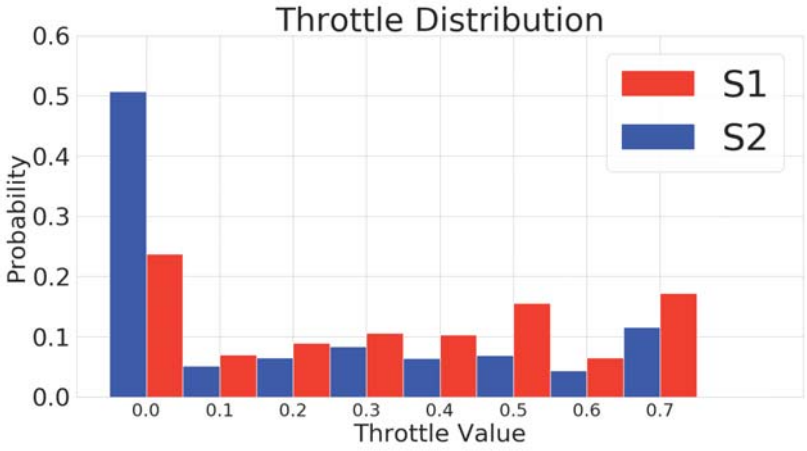


Figure 5.10 – Probability distribution of having certain throttle values comparing models with two different random seeds but trained with same hyper-parameters and data. We can perceive that S1 (red) is much more likely to have a higher throttle value.

(a) Input Image (b) Layer 1 (c) Layer 2

Figure 5.11 – Activation maps showing the increased selectivity for traffic lights in the ResNet34 case (bottom) compared to the standard 8 convolution architecture (top). For the ResNet34, layer 1, refers to the attention maps obtained after a full ResNet block.

ended due to different causes of crash, due to timeout of the task or if the main cause is if the controller stopped and did not continued again (inertia problem).

Condition	Models	Traffic Light Violations
Training Conditions	CIL	83
	8 Conv	71
	CILRS 10	47
	CILRS 100	**27**
New town & weather	CIL	82
	8 Conv	81
	CILRS 10 hours	**64**
	CILRS 100 hours	78

Table 5.4 – Percentage of times the agents burned a traffic light (lower is better) in the "Empty" conditions of the *NoCrash* benchmark.

5.5 Conclusion

Our new driving dataset (CARLA100), benchmark (*NoCrash*), and end-to-end sensorimotor architecture (CILRS) indicate that behavior cloning on large scale off-policy demonstration datasets can vastly improve over the state of the art in terms of generalization performance, including mediated perception approaches with additional supervision. This is thanks to using a deeper residual architecture with an additional speed prediction target and good regularization.

Nonetheless, our extensive experimental analysis has shown that some big challenges remain open. First of all, the amount of dynamic objects in the scene directly hurts all policy learning methods, as multi-agent dynamics are not directly captured. Second, the self-supervised nature of behavior cloning enables it to scale to large datasets of demonstrations, but with diminishing returns (or worse) due to driving-specific dataset biases that require explicit treatment, in particular biases that create causal confusion (e.g., the inertia problem). Third, the large variance resulting from initialization and sampling order indicates that running multiple runs on the same off-policy data is key to identify the best possible policies. This is part of the broader deep learning challenges regarding non-convexity and initialization, curriculum learning and training stability.

		Training					New Weather				
Task	Metric	CAL	MT	CIL	CILRS 10	CILRS 100	CAL	MT	CIL	CILRS 10	CILRS 100
Empty Town	Success	84.00	81.00	79.00	**97.33**	96.33	86.00	85.33	59.00	**98.67**	**98.67**
	Col. Pedestrian	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Col. Vehicles	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Col. Other	9.00	**11.67**	11.00	1.33	1.33	10.00	9.33	**33.00**	0.00	0.00
	Timeout	7.00	**7.33**	10.00	1.33	2.33	4.00	5.33	**8.00**	1.33	1.33
Regular Traffic	Success	57.00	74.00	61.50	83.33	**87.33**	58.00	68.00	40.00	77.33	**80.00**
	Col. Pedestrian	7.00	3.33	**9.50**	4.00	2.00	6.00	8.67	**15.00**	2.00	6.67
	Col. Vehicles	**26.00**	6.00	16.00	7.67	4.00	**30.00**	7.33	17.00	17.33	8.67
	Col. Other	7.00	11.33	7.00	4.67	3.67	2.00	10.67	**21.00**	2.67	4.00
	Timeout	3.00	5.33	6.00	0.33	3.00	4.00	5.33	**7.00**	0.67	0.67
Dense Traffic	Success	16.00	**42.67**	22.00	**42.67**	41.67	18.00	33.33	6.00	**47.33**	38.00
	Col. Pedestrian	14.00	13.67	20.50	**24.33**	22.33	12.00	18.00	15.00	12.00	14.67
	Col. Vehicles	**57.00**	22.33	49.50	18.33	20.67	68.00	18.00	**69.00**	26.00	34.00
	Col. Other	10.00	12.33	4.50	**13.33**	12.33	0.00	**20.00**	12.00	11.33	12.00
	Timeout	3.00	**9.00**	3.50	1.33	3.00	2.00	**10.67**	0.00	3.33	1.33

		New Town					New Weather & Town				
Task	Metric	CAL	MT	CIL	CILRS 10	CILRS 100	CAL	MT	CIL	CILRS 10	CILRS 100
Empty Town	Success	48.67	36.33	41.67	66.00	**72.33**	57.33	25.33	24.00	**90.67**	55.33
	Col. Pedestrian	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Col. Vehicles	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Col. Other	45.33	**56.67**	51.00	21.33	20.00	33.33	64.00	**66.67**	5.33	2.67
	Timeout	6.00	7.00	7.33	**12.67**	7.67	9.33	10.67	9.33	4.00	**42.00**
Regular Traffic	Success	27.67	26.00	22.00	**49.67**	49.00	32.00	14.67	13.33	**56.67**	42.67
	Col. Pedestrian	6.00	3.33	4.33	**8.33**	4.33	**7.33**	2.00	1.33	6.67	2.00
	Col. Vehicles	30.00	9.00	**34.67**	8.33	12.67	32.67	11.33	**36.67**	22.67	18.67
	Col. Other	30.33	**51.33**	33.00	21.67	23.67	22.67	**65.33**	44.00	8.00	8.67
	Timeout	6.00	10.33	6.00	**12.00**	10.33	5.33	6.67	4.67	6.00	**28.00**
Dense Traffic	Success	10.67	9.00	7.33	**23.00**	21.00	14.67	10.67	2.67	**24.67**	12.67
	Col. Pedestrian	7.00	8.33	9.67	**15.33**	12.33	8.00	3.33	4.67	**14.00**	8.67
	Col. Vehicles	46.33	27.67	**55.67**	39.00	35.00	46.67	38.00	**57.33**	37.33	34.00
	Col. Other	28.33	**40.33**	24.67	17.33	22.67	20.00	**35.33**	31.33	7.33	7.33
	Timeout	7.67	**14.67**	2.67	5.33	9.00	10.67	12.67	4.00	16.67	**37.33**

Table 5.5 – Analysis of the causes of episode end for different methods. We show the results for all tasks, and weather conditions. The columns for a single method-/task/condition should sum to 1. For each cause of episode end we highlight the method with higher probability. The reported results are the average over three different runs of the benchmark.

6 Conclusions and Future work

6.1 Conclusions

In this PhD dissertation we have addressed the problem of end-to-end learning of driving policies. We focused the studies on some topics we considered necessary for the development of this area. The topics are: Simulation, Controllability and Evaluation. By working on each of these topics we were then able to explore the limits of end-to-end driving through behavior cloning. We show that besides the main limitation of the distributional shift, other problems can affect the learned policy such as lack of reproducibility and bias on the datasets. However, we also showed some substantial improvement of end-to-end driving for urban driving on a simulated environment.

Chapter 2 presented the CARLA simulator. This simulator provide tools to develop systems for autonomous driving and test them on controlled scenarios. Further, the simulator allows us detailed analysis of the performance of each system. This simulator has been already used by several researchers and was a key tool for the other contributions of this thesis.

Chapter 3 presented a way to make networks controllable by conditioning them to a high level command. We showed that a branched architecture is a better alternative then a more naive command input option. We evaluated our proposal both on a physical hardware and on the CARLA simulator. This work has been already used by several other authors including some extensions [97] [67].

Chapter 4 shows that evaluating driving models by comparing the driving model predictions with the ground truth of some dataset is weekly correlated to the actual driving. We also showed that depending on how the dataset was collected this correlation can be improved. Collecting data augmented with additional cameras, and adding perturbations to the demonstrator, improves the correlation to the actual driving since the dataset will have comprised a bigger state-space. We also found that MSE was particularly uncorrelated to driving. Using L1, or the proposed threshold relative error, improves the correlation.

Finally, Chapter 5 shows that by using large scale evaluation of a model on

the CARLA environment, we were able to give a more detailed look to potential limitations of end-to-end driving IL. First there is a big variability when training this kind of model. Further, dataset biases can easily burden the improvement that could be obtained by leveraging bigger amounts of data. We opened this discussion and proposed some techniques to help improving the performance of IL models on this direction.

Learning to drive end-to-end is a very powerful technique since it allows leveraging data with little necessity for extra annotation. However, there are complications with respect to evaluation and generalization that must be still addressed in order to allow the use of such techniques for the full spectrum of real world applications. We could conclude that the use of simulation is fundamental and promising for the development of end-to-end driving agents through imitation. Simulation allows the development of replicable benchmarks that can be run in parallel and allow quickly testing of models of different natures. And as we showed on Chapter 4 using offline datasets is not a good evaluation strategy for end-to-end driving agents. Adding integration with a higher level planning, as we did on Chapter 3, allow end-to-end driving agents to perform more complex trajectories. This leads to a better use of end-to-end IL techniques as well as a more realistic evaluation of its usability for the real world applications.

6.2 Future Perspective

Simulation The simulator version explained on this paper is limited to controlling a single vehicle and to simple urban environment conditions. By having the possibility to script the behavior of each vehicle one could produce much more complex conditions to be tested. This is already released on the newest CARLA version. Since CARLA is under constant update and now have several contributors, please check the github repository https://github.com/carla-simulator/carla for an updated state of the simulator.

Controllability The set of commands that were used for this matter were quite limited, this happened due to the limitations of the simulated environment that we had. However, it would be interesting to extend this idea for more complex scenarios incorporating more complex intersections with multiple exits. Also it would be interesting to evaluate the necessity of a lane change command or if this behavior can emerge when conditioned to a certain policy.

Evaluation It would be interesting to propose a way to collect datasets that were highly correlated to actual driving. This would have a big impact on industry

and how to evaluate driving models. Also it is interesting to evaluate if the more correlated metrics, such as the threshold relative error, also produce a more sensible loss function.

Limitations The limitations can probably addressed by not using pure behavior cloning. Purely imitating a referees driver produces end-to-end driving agents that lack higher level knowledge of the objectives of driving such as traffic rules and safety. By adding traffic and safety rules to the objective function it would be possible to obtain models that generalize better. Still, behavior cloning has a much bigger applicability to the industry since it is very easy to train. A different training strategy that detects wrong causalities from data could be very beneficial for this type of training.

A Appendix

A.1 CARLA100

We describe all the contents present on the CARLA100 dataset. Note that for training our model we only used RGB sensor data, the ego-vehicle forward speed, the high level turn intentions for the conditional imitation learning and the ego vehicle controls.

A.1.1 Expert Demonstrator

We collect a training dataset, here referred as CARLA100, by executing an automated navigation expert in the simulated environment. The expert has access to privileged information about the simulation state, including the exact map of the environment and the exact positions of the ego-car, all other vehicles, and pedestrians.

The path driven by the expert is calculated using a planner. This planner uses an A* algorithm to determine the path to reach a certain goal. This path is then converted into waypoints used by a PID controller to generate the throttle, brake, and steering for the expert demonstrator. The expert drives steadily on the center of the lane, keeping a constant speed of 35 km/h when driving straight and reducing the speed when making turns to about 15 Km/h.

In addition, the expert is programmed to react to visible pedestrians when required to prevent collisions. The expert reduces its speed proportionally to the collision distance when the pedestrian is over 5 meters away and less than 15 meters away, or breaking to full stop when the pedestrian is less than 5 meters away.

The proposed demonstrator also reduces its speed to follow lead cars. The expert stops when the leading vehicle is closer than 5 meters. For our data collection process the expert never performs lane changes or overtakes.

To improve diversity, realism, and increase the number of visited state-action pairs, we add noise to the ego car controls. This reduces the difference between offline training and online testing scenarios [60]. We input noise to the expert demonstrator in a similar way as proposed by [20]. The noise simulates a gradual

drift away from the desired trajectory of the experts. However, for training, the drift is not used, but only the reactions performed by the expert. The added noise signal is detailed on Chapter 3, Section 3.4.3

A.1.2 Content

The dataset collection is divided into goal directed episodes were the expert goes from a start position into some goal position while stopping for dynamic obstacles. In total, we collected 2373 episodes with different characteristics. The entire dataset was collected on Town01. Each episode has the following features:

- Number of Pedestrians: the total number of spawned pedestrians on the town. This number is randomly sampled from the interval [50, 100].
- Number of Vehicles: the total number of spawned vehicle on the town. This number is randomly sampled from the interval [30, 70].
- Spawned seed for pedestrians and vehicles: the random seed used for the CARLA object spawning process.
- Weather: the used weather for the episode which is sampled from the set: *Clear Noon, Clear Noon After Rain, Heavy Rain Noon, Clear Sunset*.

Each episode lasted from 1-5 minutes partitioned in simulation steps of 100 ms. For each step, we store data divided into two different categories, sensor data, stored as PNG images, and measurement data, stored as json files.

For the sensor data we have the different camera sensors used: RGB camera, depth camera, and semantic segmentation pseudo sensor. For each sensor we record data in three positions: aligned with the car center, rotated 30 degrees to the left and rotated 30 degrees to the right.

As measurements, we have data measured from the ego-vehicle, the world status, and from all the other non player agents. From the ego-vehicle and the world status the following data was collected:

- Step Number: the step number of the current step, starts at zero and is incremented by one for every simulation step.
- Game Time-stamp: the time that has passed since the simulation has started.
- Position: the world position of the ego-vehicle. It is expressed as a three dimensional vector $[x, y, z]$ in meters.
- Orientation: the orientation of the vehicle with respect to the world. Expressed as Euler Angles (row, pitch and yaw).

- Acceleration: the acceleration vector of the ego-vehicle with respect to the world.
- Forward Speed: an scalar expressing the linear forward speed of the vehicle.
- Intentions: a signal that is proportional to the effect that the dynamic objects on the scene are having on the ego car actions. We use three different intention signals: stopping for pedestrians, stopping for cars and stopping for traffic lights. For example, an intention of 1 for stopping for pedestrian means that the ego car totally stopped for a pedestrians that is less than 5 meters away. An intention of the same class of 0.5 means that the expert noticed a pedestrians and has reduced its speed to a certain extent. An intention of 0 means there are no pedestrians nearby in the field of view of the expert.
- High Level Commands: the high level indication stating what the ego-vehicle would do on the next intersection: go straight, turn left, turn right, or do not care (the ego vehicle could pick any option). Each of these commands are encoded as a integer number. 2 is do not care, 3 for turn left, 4 for turn right, 5 for go straight.
- Waypoints: a set containing the 10 future positions the vehicle would assume.
- Steering Angle: the current steering angle of the vehicle's steering wheel.
- Throttle: the current pressure on the throttle pedal.
- Brake: the current pressure on the brake pedal.
- Hand Brake: if the hand brake is activated.
- Steer Noise: the current steering angle in the vehicle considering the noise function.
- Throttle Noise: the current pressure on the throttle pedal considering the noise function.
- Brake Noise: the current pressure on the brake pedal considering the noise function. The noise function is described on Chapter 3.

For each of the non-player agents (pedestrians, vehicles, traffic light), the following information is provided:

- Unique ID: an unique identifier of this agent.
- Type: if it is a pedestrian, a vehicle or a traffic light.

- Position: the world position of the agent. It is expressed as a three dimensional vector $[x, y, z]$ in meters.
- Orientation: the orientation of the agent with respect to the world. Expressed as Euler angles (row, pitch and yaw).
- Forward Speed: an scalar expressing the linear forward speed of the agent.
- State: only for traffic lights, contains the state of the traffic light, it its either red, yellow or green.

A.2 Scientific Articles

A.2.1 International Conferences

- **On Offline Evaluation of Vision-based Driving Models**, ***F. Codevilla***, *A. López, V. Koltun, A. Dosovitskiy,* **In Proc. of European Conference on Computer Vision (ECCV), Munich, Germany, 2018**
- **End-to-end driving via conditional imitation learning**, ***F. Codevilla***, *M Müller, A Dosovitskiy, A López, V Koltun,* **In Proc. of the International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018**
- **CARLA: An open urban driving simulator**, *A Dosovitskiy, G Ros,* ***F Codevilla***, *A López, V Koltun,* **In Proc. of the Conference on Robotics Learning (CoRL), Mountain View, USA, 2017**

A.3 Contributed Code and Datasets

- **CARLA 100 Dataset**
- **Benchmark Codes**

  https://github.com/carla-simulator/driving-benchmarks
- **Data Collector**

  https://github.com/carla-simulator/data-collector
- **Framework for training imitation learning networks**

  https://github.com/felipecode/coiltraine

Bibliography

[1] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *International Journal of Robotics Research*, 29(13), 2010.

[2] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 1–8, Cambridge, MA, USA, 2006. MIT Press.

[3] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004.

[4] National Highway Traffic Safety Administration. Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115, 2015. [Online; accessed 03-Feb-2019].

[5] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[6] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5), 2009.

[7] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning.

[8] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2), 2003.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[10] Daniel H. Biedermann, Matthias Ochs, and Rudolf Mester. Evaluating visual ADAS components on the COnGRATS dataset. In *IEEE Intelligent Vehicles Symposium*, 2016.

[11] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv:1604.07316*, 2016.

[12] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[13] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sebastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving.

[14] Alexander Broad, Jacob Arkin, Nathan Ratliff, Thomas Howard, and Brenna Argall. Real-time natural language corrections for assistive robotic manipulators. *International Journal of Robotics Research*, 2017.

[15] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[16] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, 2012.

[17] 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.

[18] Chenyi Chen, Ari Seff, Alain L. Kornhauser, and Jianxiong Xiao. DeepDriving: Learning affordance for direct perception in autonomous driving. In *International Conference on Computer Vision (ICCV)*, 2015.

[19] Felipe Codevilla, Antonio M Lopez, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. In *European Conference on Computer Vision (ECCV)*, pages 236–251, 2018.

[20] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[21] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] Bruno Castro da Silva, George Konidaris, and Andrew G. Barto. Learning parameterized skills. In *ICML*, 2012.

[23] C.R. de Souza, A. Gaidon, Y. Cabon, and A.M. Lopez. Procedural generation of videos to train deep action recognition networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *ICRA*, 2014.

[25] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. In *ICLR*, 2017.

[26] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, 2017.

[27] Sayna Ebrahimi, Anna Rohrbach, and Trevor Darrell. Gradient-free policy architecture search and adaptation. In *Conference on Robot Learning (CoRL)*, 2017.

[28] Peter Englert, Alexandros Paraschos, Jan Peters, and Marc Peter Deisenroth. Model-based imitation learning by probabilistic trajectory matching. In *ICRA*, 2013.

[29] Epic Games. Unreal Engine 4. https://www.unrealengine.com, 2017.

[30] Tharindu Fernando, Simon Denman, Sridha, Sridharan, and Clinton Fookes. Going deeper: Autonomous steering with neural memory networks. In *International Conference on Computer Vision (ICCV) Workshops*, 2017.

[31] Uwe Franke. Autonomous driving. In *Computer Vision in Vehicle Technology*. 2017.

[32] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[33] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[34] David Geronimo, Antonio M Lopez, Angel D Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):1239–1258, 2009.

[35] Alessandro Giusti, Jerome Guzzi, Dan Ciresan, Fang-Lin He, Juan Pablo Rodriguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jurgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2016.

[36] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[37] R. Haeusler and D. Kondermann. Synthesizing real world stereo challenges. In *German Conference on Pattern Recognition (GCPR)*, 2013.

[38] Vladimir Haltakov, Christian Unger, and Slobodan Ilic. Framework for generation of synthetic ground truth data for driver assistance applications. In *German Conference on Pattern Recognition (GCPR)*, 2013.

[39] P. Hamm, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. In *"Neural Information Processing Systems Imitation Learning and its Challenges in Robotics Workshop (NeurIPS ILR)*, 2018.

[40] Ankur Handa, Richard A. Newcombe, Adrien Angeli, and Andrew J. Davison. Real-time camera tracking: When is high frame-rate best? In *European Conference on Computer Vision (ECCV)*, 2012.

[41] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[45] Sachithra Hemachandra, Felix Duvallet, Thomas M. Howard, Nicholas Roy, Anthony Stentz, and Matthew R. Walter. Learning models for following natural language directions in unknown environments. In *ICRA*, 2015.

[46] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[47] Christian Hubschneider, Andre Bauer, Michael Weber, and J. Marius Zollner. Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In *Intelligent Transportation Systems Conference (ITSC) Workshops*, 2017.

[48] Shervin Javdani, Siddhartha S. Srinivasa, and J. Andrew Bagnell. Shared autonomy via hindsight optimization. In *RSS*, 2015.

[49] Joe Schneider Jerry Albright, Alex Bell and Chris Nyce. Automobile Insurance in the Era of Autonomous Driving. https://home.kpmg/content/dam/kpmg/pdf/2016/05/kpmg-automobile-insurance-in-era-autonomous.pdf, 2015. [Online; accessed 03-Feb-2019].

[50] Xiaojie Jin, Huaxin Xiao, Xiaohui Shen, Jimei Yang, Zhe Lin, Yunpeng Chen, Zequn Jie, Jiashi Feng, and Shuicheng Yan. Predicting scene parsing and motion dynamics in the future. In *Neural Information Processing Systems (NIPS)*, 2017.

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

[52] Nidhi Kalra and Susan M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182 – 193, 2016.

[53] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representation (ICLR)*, 2015.

[54] Harrison Kinsley. Explorations of using python to play Grand Theft Auto 5. https://github.com/Sentdex/pygta5, 2016.

[55] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11), 2013.

[56] Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4), 2012.

[57] George Konidaris, Scott Kuindersma, Roderic A. Grupen, and Andrew G. Barto. Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research*, 31(3), 2012.

[58] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*, 2016.

[59] Roger Lanctot. Accelerating the future: The economic impact of the emerging passenger economy. 2017.

[60] Michael Laskey, Anca Dragan, Jonathan Lee, Ken Goldberg, and Roy Fox. Dart: Optimizing noise injection in imitation learning. In *Conference on Robot Learning (CoRL)*, 2017.

[61] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. Off-road obstacle avoidance through end-to-end learning. In *Neural Information Processing Systems (NIPS)*, 2005.

[62] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. In *International Conference on Machine Learning (ICML)*, 2016.

[63] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML*, 2013.

[64] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *International Symposium on Experimental Robotics (ISER)*, 2017.

[65] L. Li, Z. Liu, O. Ozgüner, J. Lian, Y. Zhou, and Y. Zhao. Dense 3D semantic slam of traffic environment based on stereo vision. In *Intelligent Vehicles Symposium (IV)*, 2018.

[66] Zhihao Li, Toshiyuki Motoyoshi, Kazuma Sasaki, Tetsuya Ogata, and Shigeki Sugano. Rethinking self-driving: Multi-task knowledge for better generalization and accident explanation ability. *arXiv preprint arXiv:1809.11100*, 2018.

[67] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *European Conference on Computer Vision (ECCV)*, pages 584–599, 2018.

[68] Peng Liu, Run Yang, and Zhigang Xu. How safe is safe enough for self-driving vehicles? *Risk analysis*.

[69] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg. SSD: single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016.

[70] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[71] A.M. Lopez, G. Villalonga, L. Sellart, G. Ros, D. Vázquez, J. Xu, J. Marin, and A. Mozafari. Training my car to see using virtual worlds.

[72] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[73] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 2017.

[74] Cynthia Matuszek, Liefeng Bo, Luke Zettlemoyer, and Dieter Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *AAAI*, 2014.

[75] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[76] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 2015.

[77] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladen Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

[78] Matthias Müller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In *European Conference on Computer Vision (ECCV)*, 2016.

[79] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

[80] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2015.

[81] World Health Organization. Global status report on road safety 2018. https://apps.who.int/iris/bitstream/handle/10665/276462/9789241565684-eng.pdf?ua=1, 2108. [Online; accessed 03-Feb-2019].

[82] Brian Paden, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55.

[83] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009.

[84] Dean Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Neural Information Processing Systems (NIPS)*, 1988.

[85] Nathan D. Ratliff, James A. Bagnell, and Siddhartha S. Srinivasa. Imitation learning for locomotion and manipulation. In *International Conference on Humanoid Robots*, 2007.

[86] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[87] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.

[88] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *International Conference on Computer Vision (ICCV)*, 2017.

[89] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, 2016.

[90] German Ros, Sebastian Ramos, Manuel Granados, Amir Bakhtiary, David Vazquez, and Antonio M. López. Vision-based offline-online perception paradigm for autonomous driving. In *Winter conf. on Applications of Computer Vision (WACV)*, 2015.

[91] Germán Ros, Laura Sellart, Joanna Materzynska, David Vázquez, and Antonio M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[92] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[93] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[94] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. Learning monocular reactive UAV control in cluttered natural environments. In *ICRA*, 2013.

[95] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[96] Eder Santana and George Hotz. Learning a driving simulator. *arXiv:1608.01230*, 2016.

[97] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. *arXiv preprint arXiv:1806.06498*, 2018.

[98] Stefan Schaal, Auke Jan Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B*, 358(1431), 2003.

[99] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *ICML*, 2015.

[100] Lukas Schneider, Marius Cordts, Timo Rehfeld, David Pfeiffer, Markus Enzweiler, Uwe Franke, Marc Pollefeys, and Stefan Roth. Semantic stixels: Depth is not enough. In *Intelligent Vehicles Symposium (IV)*, 2016.

[101] Alireza Shafaei, James J. Little, and Mark Schmidt. Play and learn: Using video games to train computer vision models. In *British Machine Vision Conference (BMVC)*, 2016.

[102] Shai Shalev-Shwartz and Amnon Shashua. On the sample complexity of end-to-end training vs. semantic abstraction training. *arXiv preprint arXiv:1604.06915*, 2016.

[103] David Silver, J. Andrew Bagnell, and Anthony Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *International Journal of Robotics Research*, 29(12), 2010.

[104] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 2016.

[105] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[106] John Skinner, Sourav Garg, Niko Sünderhauf, Peter I. Corke, Ben Upcroft, and Michael Milford. High-fidelity simulation for evaluating robotic vision performance. In *Intelligent Robots and Systems (IROS)*, 2016.

[107] Sai Prashanth Soundararaj, Arvind K Sujeeth, and Ashutosh Saxena. Autonomous indoor helicopter flight using a single onboard camera. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5307–5314. IEEE, 2009.

[108] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2), 1999.

[109] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[110] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth J. Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011.

[111] A Torralba and AA Efros. Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1521–1528. IEEE Computer Society, 2011.

[112] Udacity. https://github.com/udacity/self-driving-car.

[113] R. Vedantam, X. Lin, T. Batra, C.L. Zitnick, and D. Parikh. Learning common sense through visual abstraction. In *International Conference on Computer Vision (ICCV)*, 2015.

[114] Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth J. Teller. Learning semantic maps from natural language descriptions. In *RSS*, 2013.

[115] Q. Wang, L. Chen, and W. Tian. End-to-end driving simulation via angle branched network. *arXiv:1805.07545*, 2018.

[116] Ren Wu, Shengen Yan, Yi Shan, Qingqing Dang, and Gang Sun. Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*, 2015.

[117] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: revisiting the ResNet model for visual recognition. *arXiv:1611.10080*, 2016.

[118] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. TORCS, The Open Racing Car Simulator. http://www.torcs.org.

[119] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[120] Luona Yang, Xiaodan Liang, Tairui Wang, and Eric Xing. Real-to-virtual domain unification for end-to-end autonomous driving. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[121] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong. LocNet: global localization in 3D point clouds for mobile vehicles. In *Intelligent Vehicles Symposium (IV)*, 2018.

[122] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representation (ICLR)*, 2016.

[123] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In *AAAI*, 2017.

[124] Zichao Zhang, Henri Rebecq, Christian Forster, and Davide Scaramuzza. Benefit of large field-of-view cameras for visual odometry. In *International Conference on Robotics and Automation (ICRA)*, 2016.

[125] J. Zhu, Y. Ai, B. Tian, D. Cao, and S. Scherer. Visual place recognition in long-term and large-scale environment based on CNN feature. In *Intelligent Vehicles Symposium (IV)*, 2018.

[126] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu. Traffic sign detection and classification in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[127] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[128] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

[129] Brian D. Ziebart, Andrew L. Maas, Anind K. Dey, and J. Andrew Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *UbiComp*, 2008.