End to End Learning in Autonomous Driving Systems



Yang Gao
Trevor Darrell

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-5
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-5.html

January 8, 2020

Samsara EX1071
Samsara v. Motive
IPR2025-00574

Copyright © 2020, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

End to End Learning in Autonomous Driving Systems

by

Yang Gao

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Sergey Levine
Professor Francesco Borrelli

Fall 2019

End to End Learning in Autonomous Driving Systems

Copyright 2019
by
Yang Gao

Abstract

End to End Learning in Autonomous Driving Systems

by

Yang Gao

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Convolutional neural networks have advanced visual perception significantly in recent years. Two major ingredients that enable such a success are the composition of simple modules into a complex network and the end to end optimization. However, such success has not yet revolutionized robotics as much as vision, even if robotics suffer from similar problems as traditional computer vision, i.e. imperfectness of the manual pipeline design of the system.

This thesis investigates using end-to-end learning for the autonomous driving system, a concrete robotic application. End to end learning can produce reasonable driving behaviors, even in the complex urban driving scenarios. Representation learning in end-to-end driving models is crucial, and auxiliary vision tasks such as semantic segmentation can help to form a more informative driving representation especially when training data is limited. Naive convolutional neural networks are usually only capable of doing reactive control and can not involve complex reasoning in a particular scenario. This thesis also studies how to handle scene conditioned driving behavior, which goes beyond the capability of reactive control. Alongside the end-to-end structure, learning methods also play a critical role. Imitation learning methods will acquire meaningful behaviors but usually, the robot can not master the skill. Reinforcement learning, on the contrary, either barely learns anything if the environment is too complex, or it can master the skill otherwise. To get the best of both worlds, this thesis proposes an algorithmically unified method to learn from both demonstration data and the environment.

To my parents, Ying Jin and Pengfei Gao

Contents

Contents **ii**

List of Figures **iv**

List of Tables **vi**

1 Introduction **1**
1.1 Autonomous Driving System 1
1.2 Existing Autonomous Driving Systems 2
1.3 End to End Autonomous Driving Systems 3
1.4 Open Questions in End-To-End Driving 4
1.5 Summary of the Proposed Solution 5

2 End-To-End Driving Models **7**
2.1 Background 7
2.2 Related Work 9
2.3 Deep Generic Driving Networks 10
2.4 The BDDV Dataset 14
2.5 Experiments 16
2.6 Discussion 23

3 Recover Motion from Egocentric Video **24**
3.1 Background 24
3.2 Related Work 26
3.3 Semantically Filtered Structure-from-Motion 28
3.4 Experiment 32
3.5 Discussion 36

4 Perception-Logical Policy **37**
4.1 Background 37
4.2 Related Work 39
4.3 The Perception-Logic Network 41
4.4 Experiments 44

4.5 Limitations 50
4.6 Discussion 51

5 Combining Imitation Learning and Reinforcement Learning 52
5.1 Background 52
5.2 Preliminaries 53
5.3 Soft Advantage Learning on Demonstrations and Rewards 55
5.4 Related Work 59
5.5 Results 60
5.6 Discussion 63

6 Conclusion 64

A Soft Advantage Learning Details 66
A.1 Environments 66
A.2 Effects of Imperfect Demonstrations 66
A.3 Effects of Demonstration Amount 67
A.4 Effects of Reward Choice 67
A.5 Learning from Human Demonstrations 69
A.6 Experiment Details 70

Bibliography 71

List of Figures

1.1 A typical autonomous driving software 2
1.2 The AlexNet architecture 3
1.3 The second place entry in the ILSVRC 2012 challenge 4

2.1 The egomotion formulation of the autonomous driving problem 8
2.2 Architectures that fuse temporal info with visual inputs 11
2.3 The mediated perception, motion reflex and privileged training methods 14
2.4 The example density of the data distribution of the BDDV dataset in a major city 15
2.5 Sample frames from the BDDV dataset 17
2.6 Sample predictions of the FCN-LSTM driving model 19
2.7 Continuous actions predicted by the driving model 21
2.8 Results from three types of driving models 22

3.1 The comparison of a KITTI image and a BDD dashcam image 25
3.2 The keypoint matching comparison of our method versus the standard method . 27
3.3 The first person and third person view of the 3D point reconstruction by the Sf^2M method 30
3.4 Reconstructed trajectories with different methods 31
3.5 Comparison of the Sf2M reconstruction and the GPS trajectory 35
3.6 Video frames of the sample sequence 35

4.1 The perception-logic network that unsupervisely learns logic factors and combines them 38
4.2 The architecture of the perception-logic network 41
4.3 A visualization of the conditional imitation learning method 41
4.4 Sampled images for a qualitative study of Perception-Logic network 47

5.1 Benefits of properly normalized q-values 58
5.2 Sample frames from the Toy Minecraft and the Torcs environment 60
5.3 SAL performances on the Torcs game 61
5.4 Results on learning from imperfect data 63

A.1 More results of the SAL method on imperfect demonstrations 67

A.2 Results on SAL learning from different amount of demonstrations. 68
A.3 The SAL result when the reward is not carefully shaped 69
A.4 SAL performances on the Torcs game with human demonstrations 70

List of Tables

2.1 Comparisons of BDDV dataset with other driving datasets 16
2.2 Results on the discrete feasible action prediction task 18
2.3 The continuous lane following experiment 19
2.4 Comparisons of the privileged training method with the others 20

3.1 Robustness of Sf^{2}M, ORB-SLAM and Libviso2 34

4.1 The performance evaluation of the Perception-Logic network on 2 variables . . . 45
4.2 The performance evaluation of the Perception-Logic network on 3 variables . . . 45
4.3 The gating network prediction accuracy and the driving speed accuracy 47

Acknowledgments

First and foremost, I would like to thank my parents, Ying Jin and Pengfei Gao, who brought me to this world and educate me to be diligent, tenacious and honest. They unconditionally support whatever decision I made, and never ask for anything for their interest. They also encourage me to do challenging things, but they never set any goals that I have to reach. My parents are also great mentors to my research, even if they know very little about computer science, not to say computer vision. My father even learns the news that I don't know about and kept an in-depth discussion about my work in detail. I would like to thank them for everything they did.

I would like to give my equal appreciation to Prof. Trevor Darrell. He is a friendly and nice advisor that made my life in research much easier. I learned a lot from his sense of academic topic importance as well as the precious view on the treasure in the pre-deep-learning age. Trevor is also very supportive of doing any new research direction and gives us the freedom to explore the topic to our interest. Without his open mind, I will not be able to try a lot of exciting stuff. Besides the technical side, I also want to thank Trevor to create such a diverse environment in our group. I learned a lot from those who are from a very different background. Trevor and I also often have many interesting chats about topics other than research, which help make me think of things more broadly, such as politics, law, entrepreneurship, work-life balance and career planning, etc. Trevor is also a great leader: he leads the Berkeley Vision and Learning Center (BVLC), which is then further extended into Berkeley Artificial Intelligence Research (BAIR). He also co-lead the Berkeley Deep Drive (BDD). His figure has demonstrated how to become a good leader and has inspired me to do so in the future.

During my Ph.D., I am also fortunate enough to work with a lot of other mentors. I worked with Prof. Sergey Levine on one of the reinforcement learning projects. I am extremely impressed by him being knowledgeable, hard-working and efficient. I treasure every meeting I had with him. He can propose simple yet effective experiments at every meeting, which is helpful. I have also worked with Dr. Vladlen Koltun and Dr. German Ros at Intel research. Vladlen has a pretty high research standard and has guided me to impactful research directions. German taught me a lot of things from both research and engineering perspective, and I'd like to thank him for bringing up several break-throughs in our project. I also have done a very pleasant internship at Waymo. Waymo was still a project in Google X at that time. I appreciate Abhijit Ogale and Wan-Yen Lo's mentorship during that internship, which is very different from my experience at school. I'd like to especially thank Abhijit for his great help during my job hunting process.

Besides the mentors, I also had a great time with my collaborators. I want to thank Ning Zhang who introduces me to Trevor when I first trying to contact him. She nicely pointed me directly to Trevor's office and it is from then I started the journey in computer vision. I'd like to thank Lisa Anne Hendricks for her guidance when I worked on the first project with Trevor. She always has so much to say and she has such a nice personality. I also want to thank Christian Hane for his great mentorship in the 3D reconstruction project. He is

so knowledgeable in the 3D vision area. He is an extremely friendly advisor and he usually provides useful references for us to follow. I published my first CVPR paper with Oscar Beijbom, who impressed me by his exceptionally organized workflow. Marcel Simon and I have a lot of nice discussions on extensions on the compact bilinear pooling method. I enjoy discussing various experimental phenomena on that topic with him. Last but not least, I want to thanks Huazhe (Harry) Xu for all the collaborations we had during the years. He is a co-author of almost half of my paper and I appreciate many joyful working nights in the lab we spent.

I didn't have enough chance to collaborate with all the people in my group, but I wish I did. Deepak Pathak, Samaneh Azadi and I joined the group in the same year. We share similar thoughts about school work and life. I enjoyed talking with them a lot during my Ph.D. years. Evan Shelhamer helped a lot on the Caffe stuff during my initial years. His academic rigor set a great example for me that I always remind myself. Sayna Ebrahimi joined our group after her Ph.D. She is one of the most hard-working persons I have ever seen. Her academic achievement is well deserved and I wholeheartedly hope she could physically get better as soon as possible. I am also fortunate enough to work with Anna Rohrbach on the XAI project. She is such a responsible leader and friend. I enjoyed spending time hanging out with my Chinese friends in the group, such as Ronghang Hu, Shizhan Zhu, Dequan Wang, Xin Wang, Zhuang Liu, Tete Xiao and Hang Gao. I also had a chance to see many great new faces coming in, such as Seth Park, Parsa Mahmoudieh, Coline Devin, and Medhini Gulganjalli Narasimhan. I learned a lot of new ideas and interesting stuff from them.

I am lucky to advise many brilliant undergrad students. Jingzhao Zhang, who was an undergrad at UC Berkeley, is the first student I worked with. He is remarkably hardworking and eager to do even the most intense work. Although we had bad luck and failed to discover new algorithms during our collaboration, it is still an inspirational experience for us. Later, I worked with two students from Tsinghua University, Ji Lin and the Haoran Ma, both of whom did a brilliant job at their projects. In my final year of the Ph.D., I worked with Boyuan Chen, Brandon Trabucco, Jerry Lin, and Yifei Xing, whom all go to Berkeley. We had a lot of interesting discoveries on two projects. I want to especially thank Boyuan, who is one of the smartest and the most curious people I have ever seen. We also become very good friends: Huazhe Xu and I have successfully transformed Boyuan into a tea-drinking and Lavals-eating person, as we are.

I want to thanks a lot of friends whom I would remember for the rest of my life. They are the source of happiness in my Ph.D. life. Biye Jiang is not only my roommate for four years but also a warm-hearted friend. He not only helped me with the Ph.D. application but also kindly share his ride to school with me during my first two years. You never know how warm-hearted he is under his cool face. Hong Shang and I hang out a lot before he graduated. He is an outdoor person and thus he opened a new world for me. Skiing, paddle-boarding, hiking, and rock climbing, we had so much fun and I also got the chance to know a lot of friends. Xiaofei Zhou and Mao Zhou are my two Berkeley friends that I will visit every time I travel to China. I don't even remember how we three started to have a lot of conversations. Cheng Ju is the funniest person I have ever seen. Except for the funny part, we are the same

type of person, and we shared a lot of jokes that exist only between us. I'd also like to thanks my friend Yi Wu and Huazhe Xu. We started our friendship with an occasional common complaint, and ever since then, we shared a lot of happiness, troubles, concerns. I'd also like to thank one of my closest friends from my junior high, Zhengyu Wang, for listening to a lot of my deep complaints and happiness. He is so trustworthy and kind that I can speak to him without any concern for hours. I'm also grateful to a lot of people that I couldn't possibly enumerate, including the EECS Ph.D. gang, my closest study mates; the ACE gang, my best mates for entrepreneurship chatting; the "what's next dinner" gang, my best mates for late-night party and daily life chats; the go bears gang, the people made me feel home when I started at Berkeley; all of my roommates in the five and a half years, the best people I've ever lived with.

Last but not least, I want to give my final thanks to Hezhu Zhang, whom I am fortunate enough to meet at Berkeley. I'd like to thanks her for giving me unconditionally love and teaching me how to love. We are lovers, friends, brother and sister, mother and child, father and daughter.

Chapter 1

Introduction

1.1 Autonomous Driving System

Autonomous driving (AD) is the task of driving a vehicle without any human intervention in urban and highway conditions. An autonomous driving system is usually equipped with a set of sensors, such as cameras, lidar, radar, and acoustic sensors. With a stream of sensory inputs, the autonomous driving system is required to output low-level controls of the vehicle, usually in the form of continuous steer, throttle and brake values. In this thesis, we focus on the camera sensory input modality, as the driving task can be solved with only vision modality in theory. To be able to navigate on various unseen landscapes, the system must be able to complete the environment state perception, future state prediction, path planning, trajectory generation tasks, where the system can either complete explicitly or implicitly.

Autonomous driving is an interesting problem to study not only because of its potential high impact on how people travel on the road but also because it is a great concrete example of sensorimotor tasks. Sensorimotor learning refers to the task of acting in a physical world, with explicit consideration of raw sensor inputs. AD is an interesting task because it is extremely safety-critical, and we need fundamental innovations to fulfill it. According to the National Safety Council, there are around 1.25 deaths per 100 million human vehicle miles [Wikipedia,]. People usually expect the autonomous driving system would be at least as good as an average human driver. That would require the system to be far more reliable than a non-safety-critical robotic system. In Section 1.3, we will discuss how recent technology advances might be able to improve the performance by a large margin. Besides the safety-critical aspect, autonomous driving also requires the system to complete many challenging tasks, and connect them properly under the hood. For example, the system needs to predict the future environment state, including states of other vehicles and pedestrians. The system also needs to reason about the interactions among all the agents involved in the scene. The problem of properly connecting different components is essential to many sensorimotor systems, as we will take AD system as an example in Section 1.2. The study of the autonomous driving system will potentially lead to many technological advances to

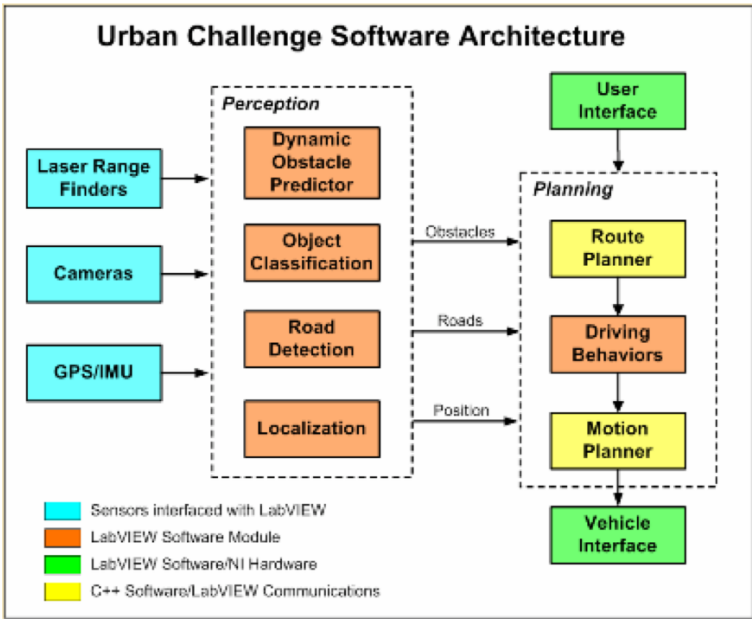


Figure 1.1: The Team Victor Tango's urban driving software stack [Currier,] in the DARPA Urban Challenge.

sensorimotor research in general.

1.2 Existing Autonomous Driving Systems

DARPA held the autonomous driving challenges in 2004, 2005 and 2007. It is also from that time, people have built practical autonomous driving systems. Figure 1.1 shows the software architecture of one of the participating teams in the Urban Challenge. It employs laser range finder, cameras, and GPS/IMU as the perception sensors. The perception stack is a heavily engineered pipeline that detects dynamic obstacles, recognizes the category of the objects, the location of the road, as well as localizes the ego vehicle. The engineered perception interface passes the extracted the obstacles, roads and ego vehicle location to the planning modules. A rule-based planner takes in the output of the perception stack and optional user input, and generate a coarse route planner. Afterward, it generates a refined, detailed low-level trajectory and executes on the car.

Twelves years later, the basic component of the autonomous driving system is very similar to the architecture shown in Figure 1.1. They are still in general consist of a perception stack that output human-defined representations of the driving scenario, and a route planner that incorporates perception's output as well as traffic rules to generate a coarse route, and the low-level motion planner that output the detailed local trajectory. Although each module

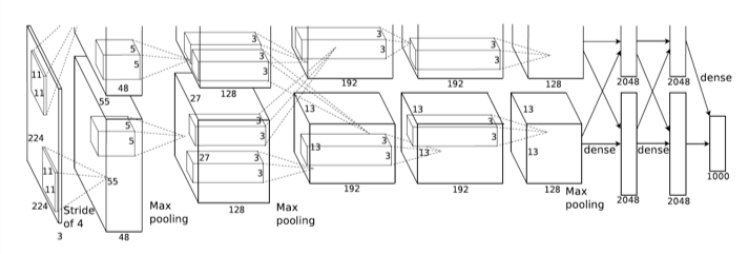


Figure 1.2: The architecture of Alexnet [Krizhevsky et al., 2012], first place in the ILSVRC 2012 challenge.

has improved a lot over time, they is no significant change to the overall architecture.

However, there are many challenges with this design. Some modules still encounter technical challenges. On the recognition part, there are way too many semantic classes to label than other closed world problems, since in the real world, there are a lot of rare object classes. On the planning side, the system has to smartly "violate" some traffic rules when necessary. For example, it needs to cross the double yellow line marking when there is a malfunctioning vehicle ahead. On the behavior prediction side, the system has to use subtle facial expressions, or gestures to guess the intention of the pedestrian. It also needs to reason about multi-agent interaction in a crowded traffic scenario.

Beyond the technical challenges in each module, coordination among submodules is also a big challenge. For example, if the perception module outputs a vehicle ahead of you, but it is flickering across the frames. What should the planning module do in this case? The existing system usually defines some heuristic to handle such problems, such as thresholding and temporal smoothing. However, people have found that managing those coordinations are much harder than expected. In this thesis, we focus on how to tackle this aspect of the problem with inspirations from deep neural networks.

1.3 End to End Autonomous Driving Systems

In recent years, convolutional neural networks have achieved great success in the visual recognition tasks. Figure 1.2 and Figure 1.3 show the ILSVRC 2012 winning and second place method architectures. The second-place entry is a human-designed pipeline, that extract image descriptors such as SIFT [Lowe, 1999], and pool them into a feature vector using the Fisher Vector method [Perronnin and Dance, 2007]. Finally, the pooled feature goes through some classifiers and outputs a classification score. The winning entry is an end-to-end trained convolutional neural network, which is more widely known as AlexNet [Krizhevsky et al., 2012]. The reason why AlexNet is much better than the previous hand-designed descriptor-pooling-classification pipeline is usually attributed to the end-to-end coordination

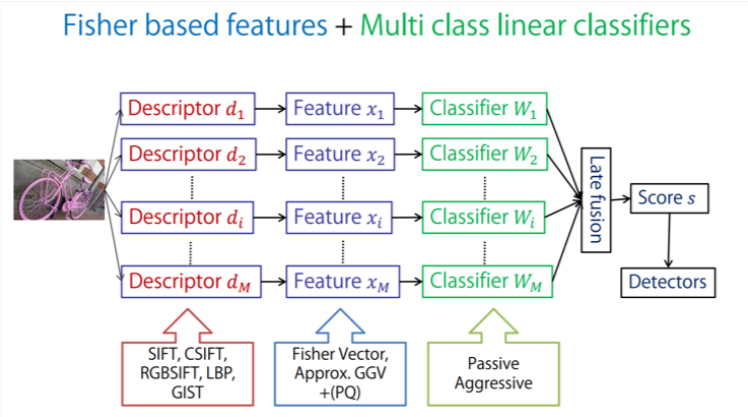


Figure 1.3: The architecture of second place in the ILSVRC 2012 challenge [Gunji et al.,].

among the layers. Although all layers are simple linear convolution filters with the ReLU activation function, they coordinate with each other well by the end to end optimization algorithm.

If we step back and stare at the existing autonomous driving system shown in Figure 1.1, there is a similar hand-designed aspect of the autonomous driving system that leads to suboptimal performance. The question is whether can we improve the autonomous driving performance with this end-to-end philosophy? I.e. use end to end optimization to improve cross-component cooperation. End to end mechanism refers to the philosophy that each module can adapt w.r.t the other components, driven by the task loss. In this thesis, we investigate both the general end-to-end CNN architecture, as well as the task-specific end-to-end architecture (the Perception-Logic Network).

1.4 Open Questions in End-To-End Driving

To design an end to end driving system. There are several open questions.

First, how to design the end to end architecture itself. One obvious choice is to use CNN like architecture, and hope that all the driving-related knowledge can be learned from the data. However, as previous work and we have shown [Bojarski et al., 2016b, Xu et al., 2017, LeCun et al., 2005, Amini et al., 2019], this approach hasn't yet scaled up to complex behaviors such as by-passing a malfunctioning vehicle. Another choice is to design differentiable equivalence of the previous hand-designed systems. However, it is not clear how to define differentiable counterpart of every component in the existing driving system. It might also be the case that one needs to make heavy changes to the overall architecture, to better fit the end-to-end framework.

Second, how to train such an end to end system. Imitation learning is one of the most

widely used methods to train a robotic system, however, imitation learning usually suffers from the distributional shift issue. I.e. the system generates a different distribution of trajectories at test time and thus the agent is not able to generalize well. Reinforcement learning is an alternative method to train such a system. However, sometimes it needs millions of examples, if not billions, to converge. Training such a system in practice is too costly. In this thesis, we explore a method to combine the best of imitation learning and reinforcement learning, to get the best of both worlds.

Last but not least, a human should be able to verify and explain the system. Most of the neural networks are not explainable and hard to verify. In the autonomous driving task, it is highly desirable to develop the system such that it is easy to verify and explain to human beings.

1.5 Summary of the Proposed Solution

In this thesis, we study the problem of end to end driving following the open questions mentioned in Section 1.4. In Chapter2, we take advantage of a newly collected dataset, called Berkeley Deep Drive Video Dataset (BDDV). We formulate the autonomous driving problem as a future ego-motion prediction task. We designed an FCN-LSTM architecture that can directly learn from raw driving videos. This work examines how much work a modern convolutional neural network could do when trained on real-world urban human driving recordings. We have shown that only using the vanilla CNN network, it exhibits behaviors like turning, attend on other traffic participants, as well as react to the traffic lights.

In Chapter 3, we further investigate how to get a more accurate supervision signal for the dataset used in Chapter 2. The dataset used in Chapter 2 is collected by iPhone's video recordings and the GPS/IMU sensory readings. The GPS/IMU fused signal gives the ground truth of the ego-motion of the vehicle. However, the ground truth of the motion is not accurate enough when there are tall buildings around it, or when the vehicle is doing subtle behaviors, such as changing lane. In this work, we investigate the combination of Structure-From-Motion techniques and semantic segmentation methods to acquire accurate ego-motion from the BDDV dataset.

Chapters 2 and 3 both deal with generic convolution neural networks, which is not designed for the autonomous driving task. I.e. they do not have the inductive bias that autonomous driving need. In Chapter 4, we investigate the visuomotor decision-making problem in autonomous driving. In many cases of autonomous driving, the behavior should have a strict logical condition. For example, if the weather is rainy or it is dark, then the agent had better drive more cautiously than usual since the perception capability is limited. When the pre-condition becomes more complex, it is harder for a vanilla CNN to learn those logic-conditioned behaviors well. We propose a Perception-Logic Network that can unsupervised discover logical primitives in the training data, and combine them

with a differentiable logic network. Our method could achieve near-perfect generalization performance during test time.

In Section 1.4, we also mentioned the hardness of training an end to end driving system. In Chapter 5, we proposed to combine the best of imitation learning and reinforcement learning to be able to master the skill and learn with a relatively small amount of data at the same time.

Chapter 2

End-To-End Driving Models

Robust perception-action models should be learned from training data with diverse visual appearances and realistic behaviors, yet current approaches to deep visuomotor policy learning have been generally limited to *in-situ* models learned from a single vehicle or simulation environment. We advocate learning a generic vehicle motion model from large scale crowd-sourced video data, and develop an end-to-end trainable architecture for learning to predict a distribution over future vehicle egomotion from instantaneous monocular camera observations and previous vehicle state. Our model incorporates a novel FCN-LSTM architecture, which can be learned from large-scale crowd-sourced vehicle action data, and leverages available scene segmentation side tasks to improve performance under a privileged learning paradigm. We provide a novel large-scale dataset of crowd-sourced driving behavior suitable for training our model, and report results predicting the driver action on held out sequences across diverse conditions.

2.1 Background

Learning perception-based policies to support complex autonomous behaviors, including driving, is an ongoing challenge for computer vision and machine learning. While recent advances that use rule-based methods have achieved some success, we believe that learning-based approaches will be ultimately needed to handle complex or rare scenarios, and scenarios that involve multi-agent interplay with other human agents.

The recent success of deep learning methods for visual perception tasks has increased interest in their efficacy for learning action policies. Recent demonstration systems [Bojarski et al., 2016a, Chen et al., 2015a, LeCun et al., 2005] have shown that simple tasks, such as a vehicle lane-following policy or obstacle avoidance, can be solved by a neural net. This echoes the seminal work by Dean Pomerleau with the CMU NavLab, whose ALVINN network was among the earliest successful neural network models [Pomerleau , 1989].

These prior efforts generally formulate the problem as learning a mapping from pixels to actuation. This end-to-end optimization is appealing as it directly mimics the demonstrated

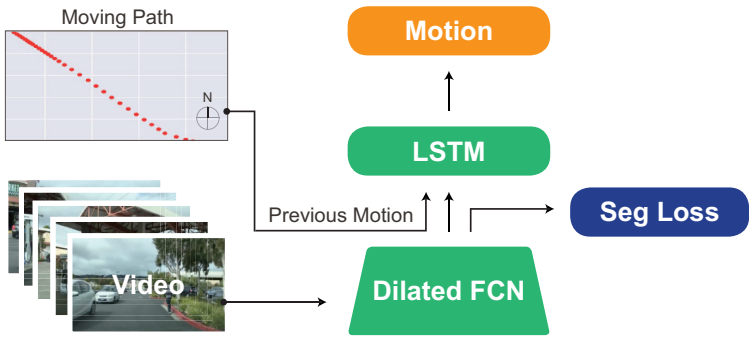


Figure 2.1: Autonomous driving is formulated as a future egomotion prediction problem. Given a large-scale driving video dataset, an end-to-end FCN-LSTM network is trained to predict multi-modal discrete and continuous driving behaviors. Using semantic segmentation as a side task further improves the model.

performance, but is limiting in that it can only be performed on data collected with the specifically calibrated actuation setup, or in corresponding simulations (e.g., as was done in [Pomerleau , 1989], and more recently in [Tzeng et al., 2016, Rusu et al., 2016, Daftry et al., 2016]). The success of supervised robot learning-based methods is governed by the availability of training data, and typical publicly available datasets only contain on the order of dozens to hundreds of hours of collected experience.

We explore an alternative paradigm, which follows the successful practice in most computer vision settings, of exploiting large scale online and/or crowdsourced datasets. We advocate learning a driving model or policy from large scale uncalibrated sources, and specifically optimize models based on crowdsourced dashcam video sources. We release a curated dataset from which suitable models or policies can be learned.

To learn a model from this data, we propose a novel deep learning architecture for learning-to-drive from uncalibrated large-scale video data. We formulate the problem as learning a generic driving model/policy; our learned model is generic in that it learns a predictive future motion path given the present agent state. Presently we learn our model from a corpus of demonstrated behavior and evaluate on held out data from the same corpus. Our driving model is akin to a language model, which scores the likelihood of character or word sequences given certain corpora; our model similarly is trained and evaluated in terms of its ability to score as highly likely the observed behavior of the held out driving sequence. It is also a policy in that it defines a probability distribution over actions conditioned on a state, with the limitation that the policy is never actually executed in the real world or simulation.

This chapter offers four novel contributions. First, we introduce a generic motion approach to learning a deep visuomotor action policy where actuator independent motion plans are learned based on current visual observations and previous vehicle state. Second, we develop a

novel FCN-LSTM which can learn jointly from demonstration loss and segmentation loss, and can output multimodal predictions. Third, we curate and make publicly available a large-scale dataset to learn a generic motion model from vehicles with heterogeneous actuators. Finally, we report experimental results confirming that "privileged" training with side task (semantic segmentation) loss learns egomotion prediction tasks faster than from motion prediction task loss alone[1].

We evaluate our model and compare to various baselines in terms of the ability of the model to predict held-out video examples; our task can be thought of that of predicting future egomotion given present observation and previous agent state history.

While future work includes extending our model to drive a real car, and addressing issues therein involving policy coverage across undemonstrated regions of the policy space (c.f. [Ross et al., 2011]), we nonetheless believe that effective driving models learned from large scale datasets using the class of methods we propose will be a key element in learning a robust policy for a future driving agent.

2.2 Related Work

ALVINN [Pomerleau , 1989] was among the very first attempts to use a neural network for autonomous vehicle navigation. The approach was simple, comprised of a shallow network that predicted actions from pixel inputs applied to simple driving scenarios with few obstacles; nevertheless, its success suggested the potential of neural networks for autonomous navigation.

Recently, NVIDIA demonstrated a similar idea that benefited from the power of modern convolution networks to extract features from the driving frames [Bojarski et al., 2016a]. This framework was successful in relatively simple real-world scenarios, such as highway lane-following and driving in flat, obstacle-free courses.

Instead of directly learning to map from pixels to actuation, [Chen et al., 2015a] proposed mapping pixels to pre-defined affordance measures, such as the distance to surrounding cars. This approach provides human-interpretable intermediate outputs, but a complete set of such measures may be intractable to define in complex, real-world scenarios. Moreover, the learned affordances need to be manually associated with car actions, which is expensive, as was the case with older rule-based systems. Concurrent approaches in industry have used neural network predictions from tasks such as object detection and lane segmentation as inputs to a rule-based control system [Huval et al., 2015].

Another line of work has treated autonomous navigation as a visual prediction task in which future video frames are predicted on the basis of previous frames. [Santana and Hotz, 2016] propose to learn a driving simulator with an approach that combines a Variational Auto-encoder (VAE) [Kingma and Welling, 2014] and a Generative Adversarial Network (GAN) [Goodfellow et al., 2014]. This method is a special case of the more general task of video prediction; there are examples of video prediction models being applied to driving

[1]The codebase and dataset can be found at https://github.com/gy20073/BDD_Driving_Model/

scenarios [De Brabandere et al., 2016, Lotter et al., 2016]. However, in many scenarios, video prediction is ill-constrained as preceding actions are not given as input the model. [Oh et al., 2015, Finn et al., 2016] address this by conditioning the prediction on the model's previous actions. In our work, we incorporate information about previous actions in the form of an accumulated hidden state.

Our model also includes a side- or privileged-information learning aspect. This occurs when a learning algorithm has additional knowledge at training time; i.e., additional labels or meta-data. This extra information helps training of a better model than possible using only the view available at test time. A theoretical framework for learning under privileged information (LUPI) was explored in [Vapnik and Vashist, 2009]; a max-margin framework for learning with side-information in the form of bounding boxes, image tags, and attributes was examined in [Sharmanska et al., 2013] within the DPM framework. Recently [Hoffman et al., 2016] exploited deep learning with side tasks when mapping from depth to intensity data. Below we exploit a privileged/side-training paradigm for learning to drive, using semantic segmentation side labels.

Recent advances in recurrent neural network modeling for sequential image data are also related to our work. The Long-term Recurrent Convolutional Network (LRCN) [Donahue et al., 2015] model investigates the use of deep visual features for sequence modeling tasks by applying a long short-term memory (LSTM) recurrent neural network to the output of a convolutional neural network. We take this approach, but use the novel combination of a fully-convolutional network (FCN) [Long et al., 2015] and an LSTM. A different approach is taken by [Xingjian et al., 2015], as they introduce a convolutional long short-term memory (LSTM) network that directly incorporates convolution operations into the cell updates.

2.3 Deep Generic Driving Networks

We first describe our overall approach for learning a generic driving model from large-scale driving behavior datasets, and then propose a specific novel architecture for learning a deep driving network.

Generic Driving Models

We propose to learn a generic approach to learning a driving policy from demonstrated behaviors, and formulate the problem as predicting future feasible actions. Our driving model is defined as the admissibility of which next motion is plausible given the current observed world configuration. Note that the world configuration incorporates previous observation and vehicle state. Formally, a driving model F is a function defined as:

$$F(s, a) : S \times A \rightarrow \mathbb{R} \tag{2.1}$$

where s denotes states, a represents a potential motion action and $F(s, a)$ measures the feasibility score of operating motion action a under the state s.

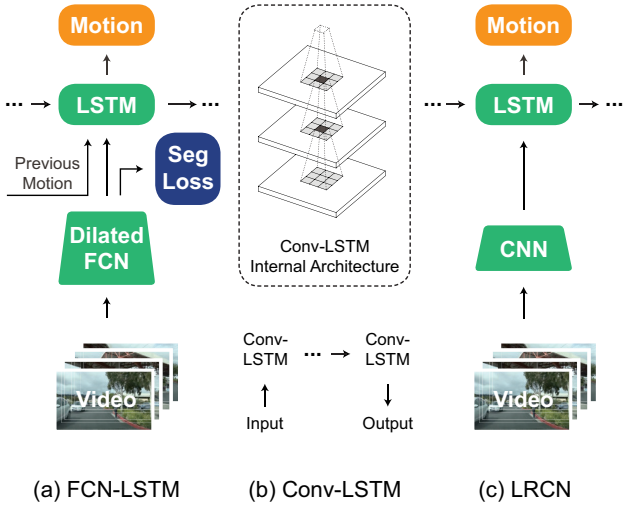


Figure 2.2: Comparison among novel architectures that can fuse time-series information with visual inputs.

Our approach is *generic* in that it predicts egomotion, rather than actuation of a specific vehicle. Our generic models take as input raw pixels and current and prior vehicle state signals, and predict the likelihood of future motion. This can be defined over a range of action or motion granularity, and we consider both discrete and continuous settings in this chapter.[2] For example, the motion action set A could be a set of coarse actions:

$$A = \{\text{straight, stop, left-turn, right-turn}\} \tag{2.2}$$

One can also define finer actions based on the car egomotion heading in the future. In that case, the possible motion action set is:

$$A = \{\vec{v} | \vec{v} \in \mathbb{R}^2\} \tag{2.3}$$

where, $\vec{v}$ denotes the future egomotion on the ground plane.

We refer to $F(s, a)$ as a driving model inspired by its similarity to the classical N-gram language model in Natural Language Processing. Both of them take in the sequence of prior events, such as what the driver has seen in the driving model, or the previously observed tokens in the language model, and predict plausible future events, such as the viable physical actions or the coherent words. Our driving model can equivalently be thought of as a policy from a robotics perspective, but we presently only train and test our model from fixed existing datasets, as explained below, and consequently we feel the language model analogy is the more suitable one.

[2]We leave the most general setting, of predicting directly arbitrary 6DOF motion, also to future work.

FCN-LSTM Architecture

Our goal is to predict the distribution over feasible future actions, conditioned on the past and current states, including visual cues and egomotions. To accomplish our goal, an image encoder is necessary to learn the relevant visual representation in each input frame, together with a temporal network to take advantage of the motion history information. We propose a novel architecture for time-series prediction which fuses an LSTM temporal encoder with a fully convolutional visual encoder. Our model is able to jointly train motion prediction and pixel-level supervised tasks. We can use semantic segmentation as a side task following "previleged" information learning paradigm. This leads to better performance in our experiments. Figure 2.2 compares our architecture (FCN-LSTM) with two related architectures [Donahue et al., 2015, Xingjian et al., 2015].

Visual Encoder

Given a video frame input, a visual encoder can encode the visual information in a discriminative manner while maintaining the relevant spatial information. In our architecture, a dilated fully convolutional neural network [Yu and Koltun, 2015, Donahue et al., 2015] is used to extract the visual representations. We take the ImageNet [Russakovsky et al., 2015] pre-trained AlexNet [Krizhevsky et al., 2012] model, remove POOL2 and POOL5 layers and use dilated convolutions for conv3 through fc7. To get a more discriminative encoder, we finetune it jointly with the temporal network described below. The dilated FCN representation has the advantage that it enables the network to be jointly trained with a side task in an end-to-end manner. This approach is advantageous when the training data is scarce.

Temporal Fusion

We optionally concatenate the past ground truth sensor information, such as speed and angular velocity, with the extracted visual representation. With the visual and sensor states at each time step, we use an LSTM to fuse all past and current states into a single state, corresponding to the state s in our driving model $F(s, a)$. This state is complete, in the sense that it contains all historical information about all sensors. We could predict the physical viability from the state s using a fully connected layer.

We also investigate below another temporal fusion approach, temporal convolution, instead of LSTM to fuse the temporal information. A temporal convolution layer takes in multiple visual representations and convolves on the time dimension with an $n \times 1$ kernel where n is the number of input representations.

Driving Perplexity

Our goal is to learn a future motion action feasibility distribution, also known as the driving model. However, in past work [Pomerleau , 1989, Chen et al., 2015a, Bojarski et al.,

2016a], there are few explicit quantitative evaluation metrics. In this section, we define an evaluation metrics suitable for large-scale uncalibrated training, based on sequence perplexity.

Inspired by language modeling metrics, we propose to use perplexity as evaluation metric to drive training. For example, a bigram model assigns a probability of:

$$p(w_1, \cdots, w_m) = p(w_1)p(w_2|w_1) \cdots p(w_m|w_{m-1})$$

to a held out document. Our model assign:

$$p(a_1|s_1) \cdots p(a_t|s_t) = F(s_1, a_1) \cdots F(s_t, a_t) \tag{2.4}$$

probability to the held out driving sequence with actions $a_1 \cdots a_t$, conditioned on world states $s_1 \cdots s_t$. We define the action predictive perplexity of our model on one held out sample as:

$$perplexity = \exp\Big\{ -\frac{1}{t} \sum_{i=1}^{t} \log F(s_i, a_i) \Big\} \tag{2.5}$$

To evaluate a model, one can take the most probable action predicted $a_{pred} = \text{argmax}_a F(s, a)$ and compare it with the action a_{real} that is carried out by the driver. This is the accuracy of the predictions from a model. Note that models generally do not achieve 100% accuracy, since a driving model does not know the intention of the driver ahead of time.

Discrete and Continuous Action Prediction

The output of our driving model is a probability distribution over all possible actions. A driving model should have correct motion action predictions despite encountering complicated scenes such as an intersection, traffic light, and/or pedestrians. We first consider the case of discrete motion actions, and then investigate continuous prediction tasks, in both cases taking into account the prediction of multiple modes in a distribution when there are multiple possible actions.

Discrete Actions. In the discrete case, we train our network by minimizing perplexity on the training set. In practice, this effectively becomes minimizing the cross entropy loss between our prediction and the action that is carried out. In real world of driving, it's more prevalent to go straight, compared to turn left or right. Thus the samples in the training set are highly biased toward going straight. Inspired by [Zhang et al., 2016], we investigated the weighted loss of different actions according to the inverse of their prevalence.

Continuous Actions. To output a distribution in the continuous domain, one could either use a parametric approach, by defining a family of parametric distribution and regressing to the parameters of the distribution, or one can employ a non-parametric approach, e.g. discretizing the action spaces into many small bins. Here we employ the second approach, since it can be difficult to find a parametric distribution family that could fit all scenarios.

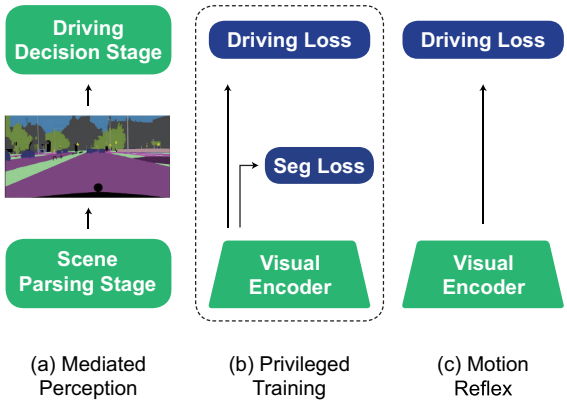


Figure 2.3: Comparison of learning approaches. Mediated Perception relies on semantic-class labels at the pixel level alone to drive motion prediction. The Motion Reflex method learns a representation based on raw pixels. Privileged Training learns from raw pixels but allows side-training on semantic segmentation tasks.

Driving with Privileged Information

Despite the large-scale nature of our training set, small phenomena and objects may be hard to learn in a purely end-to-end fashion. We propose to exploit privileged learning [Vapnik and Vashist, 2009, Sharmanska et al., 2013, Hoffman et al., 2016] to learn a driving policy that exploits both task loss and available side losses. In our model, we use semantic segmentation as the extra supervision. Figure 2.3 summarizes our approach and the alternatives: motion prediction could be learned fully end to end (Motion Reflex Approach), or could rely fully on predicted intermediate semantic segmentation labels (Mediated Perception Approach), in contrast, our proposed approach (Privileged Training Approach) adopts the best of both worlds, having the semantic segmentation as a side task to improve the representation, which ultimately performs motion prediction. Specifically, we add a segmentation loss after fc7, which enforces fc7 to learn a meaningful feature representation. Our results below confirm that even when semantic segmentation is not the ultimate goal, learning with semantic segmentation side tasks can improve performance, especially when coercing a model to attend to small relevant scene phenomena.

2.4 The BDDV Dataset

The Berkeley DeepDrive Video dataset (BDDV) is a dataset comprised of real driving videos and GPS/IMU data. The BDDV dataset contains diverse driving scenarios including cities, highways, towns, and rural areas in several major cities in US. We analyze different properties of this dataset in the following sections and show its suitability for learning a

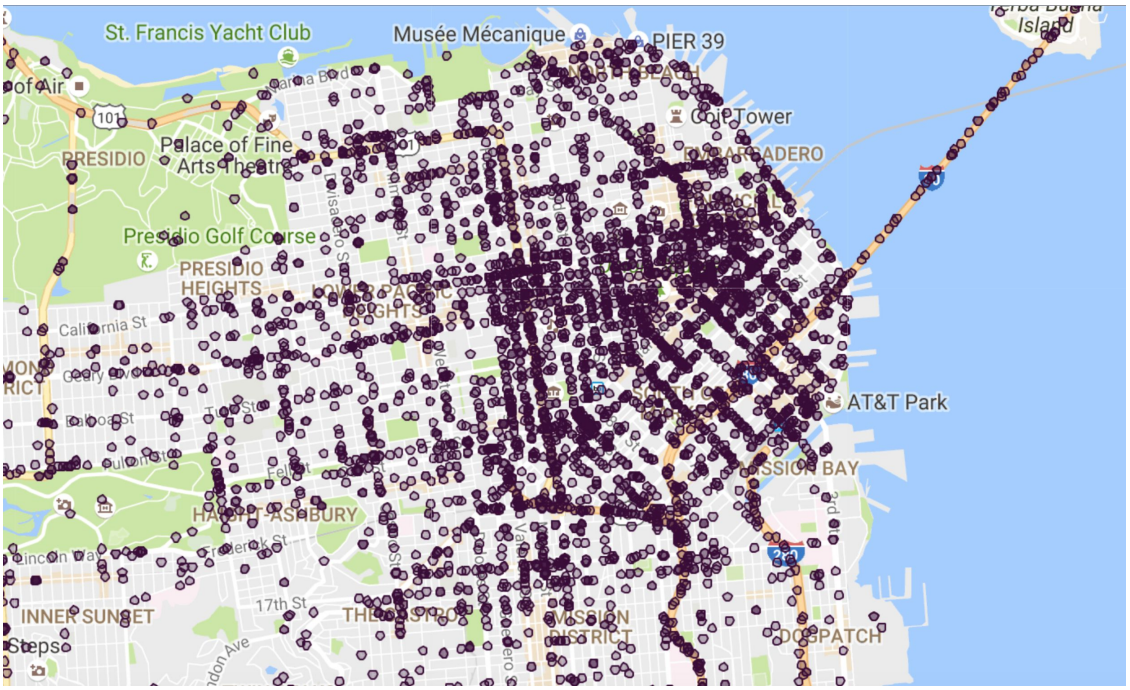


Figure 2.4: Example density of data distribution of BDDV in a major city. Each dot represents the starting location of a short video clip of approximately 40 seconds.

generic driving model in comparison with sets of benchmark datasets including KITTI, Cityscapes, Comma.ai dataset, Oxford Dataset, Princeton Torcs, GTA, each of which varies in size, target, and types of data. A comparison of datasets is provided in Table 2.1.

Scale

BDDV provides a collection of sufficiently large and diverse driving data, from which it is possible to learn generic driving models. The BDDV contains over 10,000 hours of driving dash-cam video streams from different locations in the world. The largest prior dataset is Robotcar dataset [Maddern et al., pear] which corresponds to 214 hours of driving experience. KITTI, which has diverse calibrated data, provides 22 sequences (less than an hour) for SLAM purposes. In Cityscapes, there are no more than 100 hours driving video data provided upon request. To the best of knowledge, BDDV is at least in two orders larger than any benchmark public datasets for vision-based autonomous driving.

Datasets	settings	type	Approx scale	Diversity	Specific Car
KITTI	**city, highway rural area**	**real**	less than 1 hour	one weather condition one city, daytime	Yes
Cityscape	city	**real**	less than 100 hours	**multiple weather conditions** German cities, daytime	Yes
Comma.ai	mostly highway	**real**	7.3 hours	highway, N.A. , **daytime and night**	Yes
Oxford	city	**real**	214 hours	**multiple weather conditions** one city (Oxford), daytime	Yes
Torcs	highway	synthesis	13.5 hours	N.A.	N.A.
GTA	city, highway	synthesis	N.A.	N.A.	N.A.
BDDV(ours)	**city, highway rural area**	**real**	**10k hours**	**multiple cities,daytime and night multiple weather conditions**	**No**

Table 2.1: Comparison of our dataset with other driving datasets.

Modalities

Besides the images, our BDDV dataset also comes with sensor readings of a smart phone. The sensors are GPS, IMU, gyroscope and magnetometer. The data also comes with sensor-fused measurements, such as course and speed. Those modalities could be used to recover the trajectory and dynamics of the vehicle.

Diversity

The BDDV dataset is collected to learn a driving model that is generic in terms of driving scenes, car makes and models, and driving behaviors. The coverage of BDDV includes various driving, scene, and lighting conditions. In Figure 2.5 we show some samples of our dataset in nighttime, daytime, city areas, highway and rural areas. As shown in Table 2.1, existing benchmark datasets are limited in the variety of scene types they comprise. In Figure 2.4 we illustrate the spatial distribution of our data across a major city.

2.5 Experiments

For our initial experiments, we used a subset of the BDDV comprising 21,808 dashboard camera videos as training data, 1,470 as validation data and 3,561 as test data. Each video is approximately 40 seconds in length. Since a small portion of the videos has duration just under 40 seconds, we truncate all videos to 36 seconds. We downsample frames to 640×360 and temporally downsample the video to 3Hz to avoid feeding near-duplicate frames into our model. After all such preprocessing, we have a total of 2.9 million frames, which is approximately 2.5 times the size of the ILSVRC2012 dataset. To train our model, we used stochastic gradient descent (SGD) with an initial learning rate of 10^{-4}, momentum of 0.99 and a batch size of 2. The learning rate was decayed by 0.5 whenever the training loss plateaus. Gradient clipping of 10 was applied to avoid gradient explosion in the LSTM. The

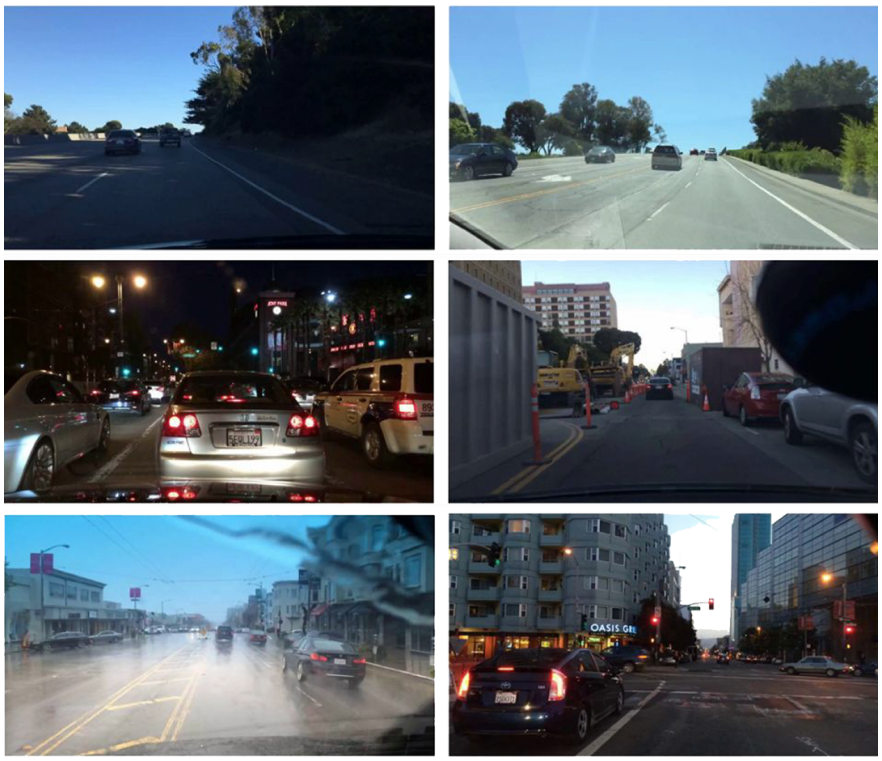


Figure 2.5: Sample frames from the BDDV dataset.

LSTM is run sequentially on the video with the previous visual observations. Specifically, the number of hidden units in LSTM is 64. Models are evaluated using predictive perplexity and accuracy, where the maximum likelihood action is taken as the prediction.

Discrete Action Driving Model

We first consider the discrete action case, in which we define four actions: **straight**, **stop**, **left turn**, **right turn**. The task is defined as predicting the feasible actions in the next 1/3rd of a second.

Following Section 2.3, we minimize perplexity on the training set and evaluate perplexity and accuracy of the maximum likelihood prediction on a set of held out videos. In Table 2.2, we do an ablation study to investigate the importance of different components of our model.

Table 2.2 shows the comparison among a few variants of our method. The Random Guess baseline predicts randomly based on the input distribution. In the speed-only condition, we only use the speed of the previous frame as input, ignoring the image input completely. It achieves decent performance, since the driving behavior is largely predictable from the speed in previous moment. In the "1-Frame" configuration, we only feed in a single image at each timestep and use a CNN as the visual encoder. It achieves better performance than

Configuration	Image	Temporal	Speed	Perplexity	Accuracy
Random-Guess	N.A.	N.A.	No	0.989	42.1%
Speed-Only	N.A.	LSTM	Yes	0.555	80.1%
CNN-1-Frame	CNN	N.A.	No	0.491	82.0%
TCNN3	CNN	CNN	No	0.445	83.2%
TCNN9	CNN	CNN	No	0.411	84.6%
CNN-LSTM	CNN	LSTM	No	0.419	84.5%
CNN-LSTM+Speed	CNN	LSTM	Yes	0.449	84.2%
FCN-LSTM	FCN	LSTM	No	0.430	84.1%

Table 2.2: Results on the discrete feasible action prediction task. We investigated the influence of various image encoders, temporal networks and the effect of speed. Log perplexity (lower is better) and accuracy (higher is better) of our prediction are reported. See Section 2.5 for details.

the two baseline models (random and speed-only). This is intuitive, since human drivers can get a good, but not perfect, sense of feasible motions from a single frame. In the TCNN configuration we study using temporal convolution as the temporal fusion mechanism. We used a fixed length window of 3 (TCNN3) and 9 (TCNN9), which is 1 and 3 seconds in time respectively. TCNN models further improves the performance and the longer the time horizon, the better the performance. However, it needs a fixed size of history window and is more memory demanding than the LSTM based approach. We also explore the CNN-LSTM approach, and it achieves comparable performance as TCNN9. When changing the visual encoder from CNN to FCN, the performance is comparable. However, as we will show later 2.3, a FCN-based visual encoder is vital for learning from privileged segmentation information. We also found that the inverse frequency weighting of the loss function [Zhang et al., 2016] encourages the prediction of rare actions, but it does not improve the prediction perplexity. Thus we do not use this in our methods above.

In Figure. 2.6, we show some predictions made by our model. In the first pair of images (subfig. a&b), the car is going through an intersection, when the traffic light starts to change from yellow to red. Our model has predicted to go straight when the light is yellow, and the prediction changes to stop when the traffic light is red. This indicates that our model has learned how human drivers often react to traffic light colors. In the second pair (c& d), the car is approaching a stopped car in the front. In (c), there is still empty space ahead, and our model predicts to go or stop roughly equally. However, when the driver moves closer to the front car, our model predicts stop instead. This shows that our model has learned the concept of distance and automatically map it to the feasible driving action.



(a) go at yellow light

(b) stop at red light

(c) stop & go equal weight at medium distance

(d) stop when too close to vehicle ahead

Figure 2.6: Discrete actions predicted by our FCN-LSTM model. Each row of 2 images show how the prediction changes by time. The green bars shows the probability of doing that action at that time. The red bars are the driver's action. The four actions from top to bottom are going straight, slow or stop, turn left and turn right.

Table 2.3: Continuous lane following experiment. See Section 2.5 for details.

Configuration	Angle Perplexity
Random Guess	1.86
Linear Bins	-2.82
Log Bins	-3.66
Data-Driven Bins	-4.83

Continuous Action Driving Model

In this section, we investigate the continuous action prediction problem, in particular, lane following. We define the lane following problem as predicting the angular speed of the vehicle in the future 1/3 second. As proposed above, we discretize the prediction domain into bins and turn the problem into a multi-nomial prediction task.

We evaluated three different kinds of binning methods (Table 2.3). First we tried a linear binning method, where we discretize $[-90°, 90°]$ into 180 bins of width $1°$. The linear binning

method	perplexity	accuracy
Motion Reflex Approach	0.718	71.31%
Mediated Perception Approach	0.8887	61.66
Privileged Training Approach	**0.697**	**72.4%**

Table 2.4: Comparison of the privileged training with other methods.

method is reasonable under the assumption that constant controlling accuracy is needed to drive well. Another reasonable assumption might be that constant relative accuracy is required to control the turns. This corresponds to the log bins method. We use a total of 180 bins that is evenly distributed in $logspace(-90°, -1°)$ and $logspace(1°, 90°)$. We also tried a data-driven approach. We first compute the distribution of the drivers' behavior (the vehicle's angular velocity) in the continuous space. Then we discretize the distribution to 180 bins, by requiring each bin having the same probability density. Such data-driven binning method will adaptively capture the details of the driver's action. During training we use a Gaussian smoothing with standard deviation of 0.5 to smooth the training labels in nearby bins. Results are shown in Table 2.3; The data-driven binning method performed the best among all of them, while the linear binning performed worst.

Figure 2.7 shows examples of our prediction on video frames. Sub-figure (a) & (b) shows that our models could follow the curving lane accurately. The prediction has a longer tail towards the direction of turning, which is expected since it's fine to have different degrees of turns. Sub-figure (c) shows the prediction when a car is starting to turn left at an intersection. It assigns a higher probability to continue turning left, while still assigning a small probability to go straight. The probability in the middle is close to zero, since the car should not hit the wall. Close to the completion of the turn (sub-figure (d)), the car could only finish the turn and thus the other direction disappears. This shows that we could predict a variable number of modalities appropriately. In sub-figure (e), when the car is going close to the sidewalk on its right, our model assigns zero probability to turn right. When going to the intersection, the model has correctly assigned non-zero probability to turning right, since it's clear by that time.

Learning with Privileged Information (LUPI)

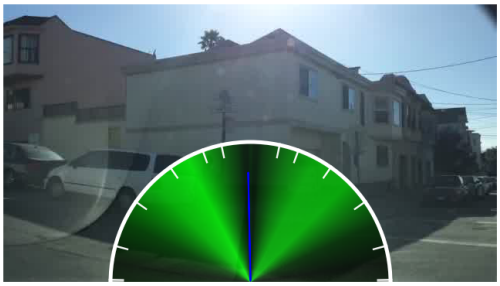
In this section, we demonstrate our LUPI approach on the discrete action prediction task. Following Section 2.3, we designed three approaches: The Motion Reflex Approach refers to the FCN-LSTM approach above. The Privileged Training approach takes the FCN-LSTM architecture and adds an extra segmentation loss after the fc7 layer. We used BDD Segmentation masks as the extra supervision. Since the BDDV dataset only contains the car egomotion and the BDD Segmentation dataset only contains the segmentation of individual images, we pair each video clip with 10 BDD Segmentation images during training.



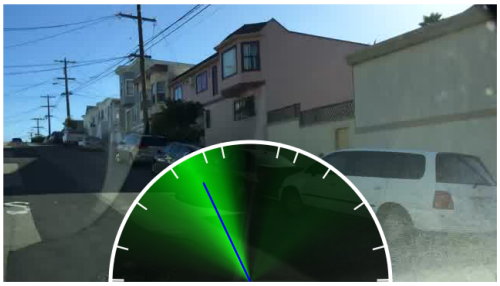
(a) lane following left



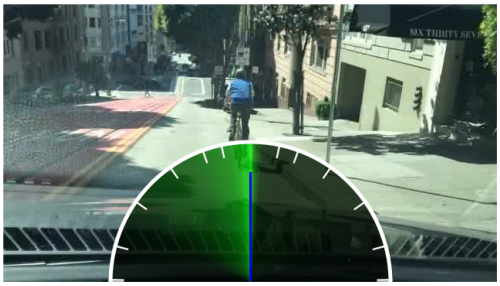
(b) lane following right



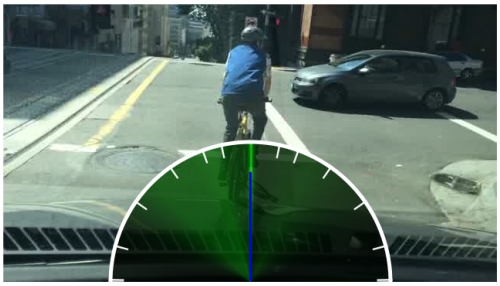
(c) multiple possible actions: turn left or go straight



(d) collapsed to single action after the turn



(e) single sided prediction due to side walk



(f) right turn becomes available at intersection

Figure 2.7: Continuous actions predicted by our model. The green sector with different darkness shows the probability map of going to a particular direction. The blue line shows the driver's action.

The motion prediction loss (or driving loss) and the semantic segmentation loss are weighted equally. For the Mediated Perception Approach, we first compute the segmentation output of every frame in the videos using the Multi-Scale Context Aggregation approach described in [Yu and Koltun, 2015]. We then feed the segmentation results into an LSTM and train the LSTM independently from the segmentation part, mimicking stage-by-stage training. In theory, one would not need side task to improve the performance of a neural network with unlimited data. To simulate a scenario where we only have limited amount of training data,



Figure 2.8: We show one example result in each column from each of the three models. (a) is the Behavior Reflex Approach. (b) is the Mediated Perception Approach and (c) the Privileged Training Approach.

we run experiments on a common subset of 1000 video clips.

As shown in Table 2.4, the Privileged Training approach achieves the best performance in both perplexity and accuracy. These observations align well with our intuition that training on side tasks in an end-to-end fashion improves performance. Figure 2.8 shows an example in which Privileged Training provides a benefit. In the first column, there is a red light far ahead in the intersection. The Privileged Training approach has successfully identified that and predicted stop in (c), while the other two methods fail. In the second column, the car is waiting behind another car. In the frame immediately previous to these frames, the vehicle in front had an illuminated brake light. The second column of images shows the prediction of the three methods when the brake light of the car goes out but the vehicle has not yet started to move. The Privileged Training approach in (c) predicts stop with high probability. The other two methods behave more aggressively and predict going straight with high probability.

2.6 Discussion

We introduce an approach to learning a generic driving model from large scale crowd-sourced video dataset with an end-to-end trainable architecture. It can learning from monocular camera observations and previous egomotion states to predict a distribution over future egomotion. The model uses a novel FCN-LSTM architecture to learn from driving behaviors. It can take advantage of semantic segmentation as side tasks improve performance, following the privileged learning paradigm. To facilitate our study, we provide a novel large-scale dataset of crowd-sourced driving behaviors that is suitable for learning driving models. We investigate the effectiveness of our driving model and the "privileged" learning by evaluating future egomotion prediction on held-out sequences across diverse conditions.

Chapter 3

Recover Motion from Egocentric Video

Current methods for extracting 3D scene structure fail on typical driving sequences collected by consumer-grade cameras. These methods generally rely on geometric cues and have difficulties distinguishing moving from static objects for many scene conditions. Recent advances in pixel-level semantic labeling offer a new approach: leveraging monocular semantic cues prior to reconstruction. We propose static-object Semantic Filtering for Structure-from-Motion, Sf^{2}M, using a fully convolutional network to directly predict the keypoints belonging to the static background in each frame. Our method can be utilized with many existing SfM methods, including both classic and deep learning-based reconstruction algorithms. Leveraging a learned monocular cue to exclude keypoints on moving objects allows our method to succeed with rolling shutter cameras without explicitly modeling the rolling shutter, even in scenes dominated by multiple moving objects with a relatively narrow field-of-view. We report the performance of our method on a recently released crowdsourced driving video dataset, both for scene and camera trajectory reconstruction.

3.1 Background

In this chapter we investigate structure from motion (SfM) on crowdsourced driving video sequences, and show that the addition of a simple yet novel semantic filter leads to successful reconstruction with consumer-grade camera sensors. From a computer vision perspective, the problem of reconstructing camera motion and scene structure is amongst the oldest problems in the field and is generally considered solved in many important cases. It is well known that many classic SfM and simultaneous localization and mapping (SLAM) methods can succeed at visual odometry and 3D reconstruction; somewhat surprisingly, however, we show below that existing solutions fail to handle typical videos from a mobile phone sensor mounted on a vehicle dashboard.

Crowdsourced dashcam videos are more challenging compared to previous visual odometry



Figure 3.1: Comparison of a KITTI image (left) and a BDD dashcam image (right). The BDD image features a narrower field of view and more surrounding moving objects, which makes the monocular visual odometry problem much more challenging than that in the KITTI dataset.

benchmarks for driving scenarios, such as the KITTI odometry benchmark [Geiger et al., 2013]. Several recent publicly available monocular visual odometry methods fail on the crowdsourced dashcam videos in the dataset released by [Xu et al., 2016], including DSO [Engel et al., 2016], ORB-SLAM [Mur-Artal et al., 2015], Libviso2 [Geiger et al., 2011], and OpenSFM [Adorjan,]. Compared to KITTI, the crowdsourced videos have much denser traffic, they are recorded by rolling shutter cameras, there are motion blur and jitter, and the camera is generally mounted at a lower height and, hence, a larger fraction of the image is covered by other road users (c.f. Fig. 3.1). Moreover, neither camera mounting locations nor camera calibrations are available. Among all the challenging factors, our experiments indicate that moving objects in the scene create the biggest problem.

We propose a learning-based approach that is able to overcome these limitations, and can reconstruct road scenes without a complex camera setup, calibration of each individual camera sensor, nor explicit modeling of the rolling shutter. We argue that semantic understanding of the image should be a primary component for correspondence estimation in visual odometry. While some SfM models have included object recognition cues, as discussed below, in general, previous systems only make use of geometric cues and/or appearance consistency when forming keypoint correspondences, as in [Engel et al., 2016, Mur-Artal et al., 2015, Geiger et al., 2011, Adorjan, , Klein and Murray, 2007]. Such methods are prone to forming an incorrect hypothesis on moving objects.

Traditionally, robust estimation using RANdom SAmpling and Consensus (RANSAC) has been used to discount the effect of moving objects. However, in the case of uncalibrated dashcam videos, fairly generous inlier thresholds may be required due to the uncertainty induced by the unknown calibration and rolling shutter artifacts; with sequences where moving objects occupy a large proportion of the scene, this can lead to erroneous reconstructions. While one solution would be to try to model all the effects in the image formation process, such that very strict inlier thresholds can be used to reject moving objects, this becomes very challenging for large collections of videos captured with different cameras in a crowdsourced

manner. Moreover, motion estimation with more complicated camera models that model the rolling shutter is generally more difficult and requires more point correspondences. We propose a complementary solution, making the correspondence estimation more robust from the beginning. Our method, which we call Semantic Filtered Structure-from-Motion, or Sf^{2}M, simply rejects keypoint correspondences when they are lying on objects which potentially move (c.f. Fig. 3.2). In our experiments we show that this is a very powerful yet simple way of making SfM and SLAM more robust, allowing us to reconstruct crowdsourced dashboard video sequences using lightly modified state-of-the-art SfM or SLAM systems.

We compare Sf^{2}M to baseline visual odometry methods on the BDD dashcam video dataset [Xu et al., 2016]. Baseline methods significantly under-perform our method on a majority of videos in this dataset. While we build and will make publicly available a reference implementation of our method using the OpenSfM framework, our approach is general and potentially applicable to a wide range of SfM and SLAM approaches.

3.2 Related Work

Structure from Motion has attracted considerable attention in computer vision since the early stages of the field, and existing techniques are able to reconstruct large scale 3D scenes, e.g., [Wu et al., 2011, Snavely et al., 2006, Snavely et al., 2008, Agarwal et al., 2010, Frahm et al., 2010]. However, although we have a thorough theoretical understanding of the geometry [Hartley and Zisserman, 2003, Pollefeys et al., 1999] of the problem, we still face two practical problems that keep SfM from working robustly: finding reliable correspondences in real scenes and reconstructing scenes which contain moving objects. Various feature detectors and descriptors (e.g., [Lowe, 1999, Bay et al., 2006, Mur-Artal et al., 2015]) have been proposed to robustly detect and match keypoints between images. Yet in practice, these methods may not be sufficient to resolve difficulties with moving objects, rolling shutters, and other challenges present in crowdsourced video data. We conjecture that humans can find correspondences in those images based on high level semantic cues; therefore we propose a semantic convolutional network filter to identify unlikely correspondences.

Surprisingly, to the best of our knowledge, no previous work has taken advantage of static semantic information to improve the robustness of initial correspondence matching in SfM. Semantic information has been exploited in previous SfM approaches, but in a post-hoc manner. Several papers [Alcantarilla et al., 2016, Song et al., 2016, Song and Chandraker, 2015, Vineet et al., 2015, Kundu et al., 2014] assume that a reconstruction of the static scene can be acquired using SfM and subsequently use semantic cues to refine the reconstruction by tracking moving objects or augmenting the reconstruction with semantic information. For example, [Song et al., 2016] first produces a 3D reconstruction, which is subsequently used with a bounding box detector to further estimate the ground plane and retrieve the 3D vehicle detections. In our work we consider situations where state-of-the-art methods already fail on the initial reconstruction of the static environment. [Salas-Moreno et al., 2013, Bao and Savarese, 2011] use monocular object detections to augment the information acquired

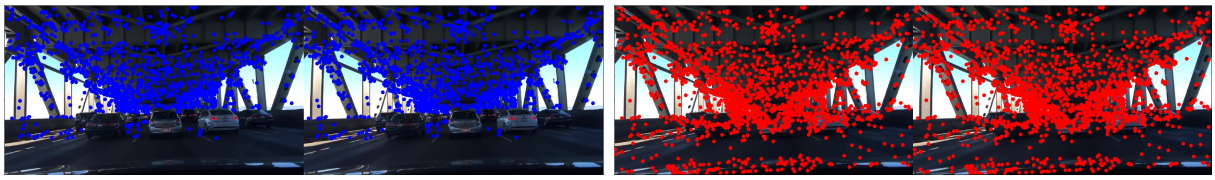


Figure 3.2: Keypoints matching comparison of our method (left) versus the standard structure from motion method (right) [Snavely et al., 2006, Snavely et al., 2008]. Matches are obtained after nearest neighbor matching between two images, followed by the ratio test and RANSAC fundamental matrix estimation filtering. Our method has excluded matches on all moving objects, thus improving the overall robustness of the system. Matches are between adjacent video frames. Correspondences are omitted for visualization purpose.

from keypoint matches. However, this means that the objects are assumed to be static, which is not the case in general environments. The method closest to ours is [Kundu et al., 2010], which designs a motion segmentation procedure to segment out different rigid body motions in the scene based on geometric cues. In our work we utilize semantic information to aid the SfM computation.

Multi-body SfM [Ozden et al., 2010, Fitzgibbon and Zisserman, 2000, Kundu et al., 2011, Roussos et al., 2012] is an extension of SfM that enables the reconstruction of multiple rigidly moving objects. The main idea is to cluster tracks into several independently moving objects and apply traditional multi-view geometry on each object. The presence of multiple motions has long been recognized to be a challenge. Without the help of semantics, it is considered hard both in theory and in practice to recover the camera motion, even under a relaxed rigid multi-body assumption [Ozden et al., 2010, Fitzgibbon and Zisserman, 2000]. Recently, semantic cues have been utilized for optical flow computation [Sevilla-Lara et al., 2016, Bai et al., 2016], further motivating our approach. With our method, we can separate the static and dynamic parts of the images so that we can use different SfM algorithms to focus on different parts of the scene.

Note that many of these SfM systems take an unordered collection of images as input and the focus is reconstructing the structure, and, hence, often only a subset of the input images is reconstructed. However, for Simultaneous Localization and Mapping (SLAM) problems, images are sequential and the goal is to reconstruct the entire trajectory, hence, all images need to be registered. Many SLAM systems have been proposed in the literature, a few recent examples being [Engel et al., 2014, Mur-Artal et al., 2015, Engel et al., 2016, Song and Chandraker, 2015]. They generally exploit the smooth motion assumption to track feature points through the image sequence. Some works use learning-based techniques to improve egomotion estimation. [Hartmann et al., 2014] proposes predicting how matchable an extracted SIFT feature is in order to make the matching procedure more efficient. DeMon [Ummenhofer et al., 2016] replaces the traditional motion estimation between two frames with a purely

learning-based approach. Using these stronger assumptions about the motion allows for a more complete reconstruction of the trajectory within SLAM systems as compared to SfM, which generally makes no assumptions about the motion.

Semantic image segmentation is one of the classic problems in computer vision. Researchers used to design features by hand to classify pixel categories [Shotton et al., 2008]. With the advent of representation learning by convolutional networks, significant performance improvement was obtained [Long et al., 2015, Ronneberger et al., 2015, Badrinarayanan et al., 2015, Yu and Koltun, 2015, Yu et al., 2017]; with state of the art methods the majority of the static pixels can be detected reliably. These methods provide semantic labeling of the image pixels, which is complementary to the geometrical understanding obtained by SfM. We build on the recent success of semantic image segmentation [Long et al., 2015, Yu and Koltun, 2015], and use pixel semantics as a filter to improve the robustness of correspondences used in SfM.

3.3 Semantically Filtered Structure-from-Motion

One of the key steps in SfM and SLAM systems is finding the relative pose (rotation and translation) between two images. The predominant paradigm for this task is finding a sufficiently large set of corresponding points in both images using keypoint detectors and feature matching, and subsequently using algorithms such as the 5 point or 8 point algorithm to find the essential or the fundamental matrix, respectively [Hartley and Zisserman, 2003]. Once a good transformation is found, a least squares solution that takes into account all established keypoint correspondences can be computed for a more accurate estimate. The key to a robust SfM or SLAM system is being able to reliably and accurately estimate these pairwise transformations.

We propose an SfM method that explicitly incorporates an end-to-end learned Fully Convolutional Neural Network model to predict matching confidence. The FCN is used as a semantic filter in our formulation, so we refer to our method as Semantic Filtered Structure-from-Motion, or Sf^{2}M.

Matching with Semantics

Given two images I_1 and I_2, we want to find points in the set P_1 and their corresponding points in set P_2 and vice versa, where P_1 and P_2 are sets of all the matchable points in I_1 and I_2. Formally, we use a **matchability score** $\mathbf{M}_{I_1 I_2}(i_1, i_2)$ to denote our preference over all pairs of points between $i_1 \in P_1$ and $i_2 \in P_2$. Traditionally, the matchability score only depends on the appearance of two images at i_1 and i_2:

$$\mathbf{M}_{I_1 I_2}(i_1, i_2) = \frac{1}{\|\mathbf{f}_{I_1}(i_1), \mathbf{f}_{I_2}(i_2)\|},$$

where $\mathbf{f}_I(i)$ is a function that calculates local image descriptors at location i for image I and $\|\cdot\|$ is a distance metric for the pair of descriptors. For example, most SfM pipelines use SIFT [Lowe, 2004, Lowe, 1999] as local descriptors and a corresponding L_2 distance metric. Retrieving the nearest neighbor of a descriptor in another image corresponds to a special kind of matchability score ranking.

We propose that matching only based on local low level image descriptors is insufficient. Instead, the matching process has to take higher level semantic information into account. We propose that the matchability score should have the form:

$$\mathbf{M}_{I_1 I_2}(i_1, i_2) = \frac{\mathbf{S}_{I_1 I_2}(i_1, i_2)}{\|\mathbf{f}_{I_1}(i_1), \mathbf{f}_{I_2}(i_2)\|}, \tag{3.1}$$

where $\mathbf{S}_{I_1 I_2}$ is a learnable function that outputs pairwise semantic compatibility between all matches. The semantic compatibility provides a much higher level of global information than the local descriptors, such as the class of objects and physical knowledge of whether the object can be moving.

Matching FCN

In this section, we develop a factorized semantic term based on a fully convolutional network approach to pixel-level category prediction. The semantic term $\mathbf{S}_{I_1 I_2}(i_1, i_2)$ takes the semantics of both images, as well as their interaction into account. Our factored term drops the pairwise semantic interaction:

$$\mathbf{S}_{I_1 I_2}(i_1, i_2) = \mathbf{p}_{I_1}(i_1) \cdot \mathbf{p}_{I_2}(i_2),$$

where $\mathbf{p}_I(i)$ gives dense semantic score at pixel i for image I. The $\mathbf{p}_I(i)$ could be easily implemented by an Fully Convolutional Network.

The Dilated FCN is a recently proposed network architecture for semantic segmentation [Yu and Koltun, 2015], which achieves dense outputs by using dilated convolution. We take a dilated VGG-16 [Yu and Koltun, 2015] architecture and train it with semantic segmentation annotation provided by Cityscapes [Cordts et al., 2016]. We define the $\mathbf{p}_I(i)$ function by:

$$\mathbf{p}_I(i) = 1_{[\mathbf{s}_I(i) \in S]}$$

where $\mathbf{s}_I(i)$ is the semantic segmentation output at the pixel corresponding to keypoint i, S is a set of object classes that is always static.

We thus learn $\mathbf{p}$ as a segmentation network defined by classes which have the potential for motion. In practice, this choice shows strong results, as presented in our experimental evaluation below; however, it will inevitably miss some static objects, such as parked vehicles. We explored a bootstrap method to further distinguish moving versus static within a single object class, using points labeled with re-projection errors to finetune the model. Somewhat surprisingly, our experiments with this approach failed to significantly improve performance,

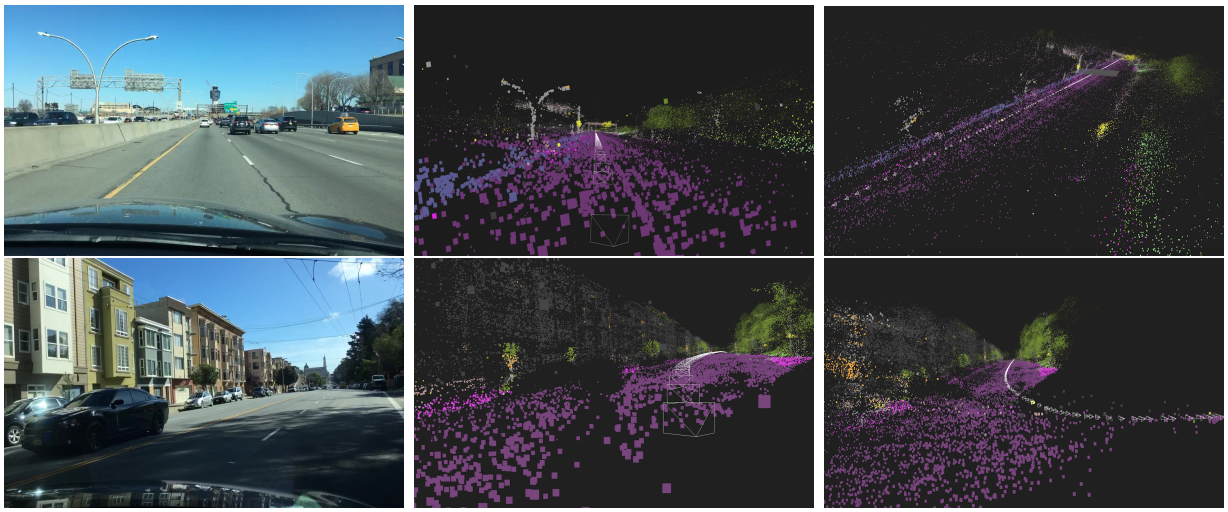


Figure 3.3: Left: Frames from two videos in the BDD Dataset. Middle: The 3D point cloud reconstructed by Sf^{2}M, seen from camera view. Right: The 3D point cloud reconstructed by Sf^{2}M, seen from side-viewan alternate viewpoint. Points are colored by the segmentation label. The road is purple, vegetation is green, poles are light gray, traffic signs are light yellow, and buildings are dark gray.

reinforcing the message of this chapter that monocular semantics are a powerful cue for reconstructing the challenging type of data we are exploring. We hypothesize that the frontal motion of the camera, along with keypoints being too far away, makes the triangulation non-trivial. We additionally note that some datasets do include pixel level labels which distinguish moving from non-moving vehicles, and future work is to finetune our model with such labels and see what effect that has on the performance.

Base SfM Model

We experiment below using our framework with a traditional SfM pipeline, based on OpenSFM. Our base SfM model is comprised of the following steps. First, a set of sparse image keypoints, along with their descriptors, are extracted from the images. Then, for all pairs of images which are within a sequential window, all feature points are exhaustively matched. A distance ratio test and fundamental matrix estimation with RANSAC [Fischler and Bolles, 1981] are used to prune bad matches and find a motion estimate. After that, all 2D keypoints that correspond to a potential single 3D point are collected together, which is called a *track*. Finally, a reconstruction is grown by incrementally adding images by resection, and bundle adjustment is applied to improve the quality further. We note that our overall framework is generic, and can be broadly developed incorporating any SfM or visual odometry method that allows a pairwise matching confidence term to be integrated into the

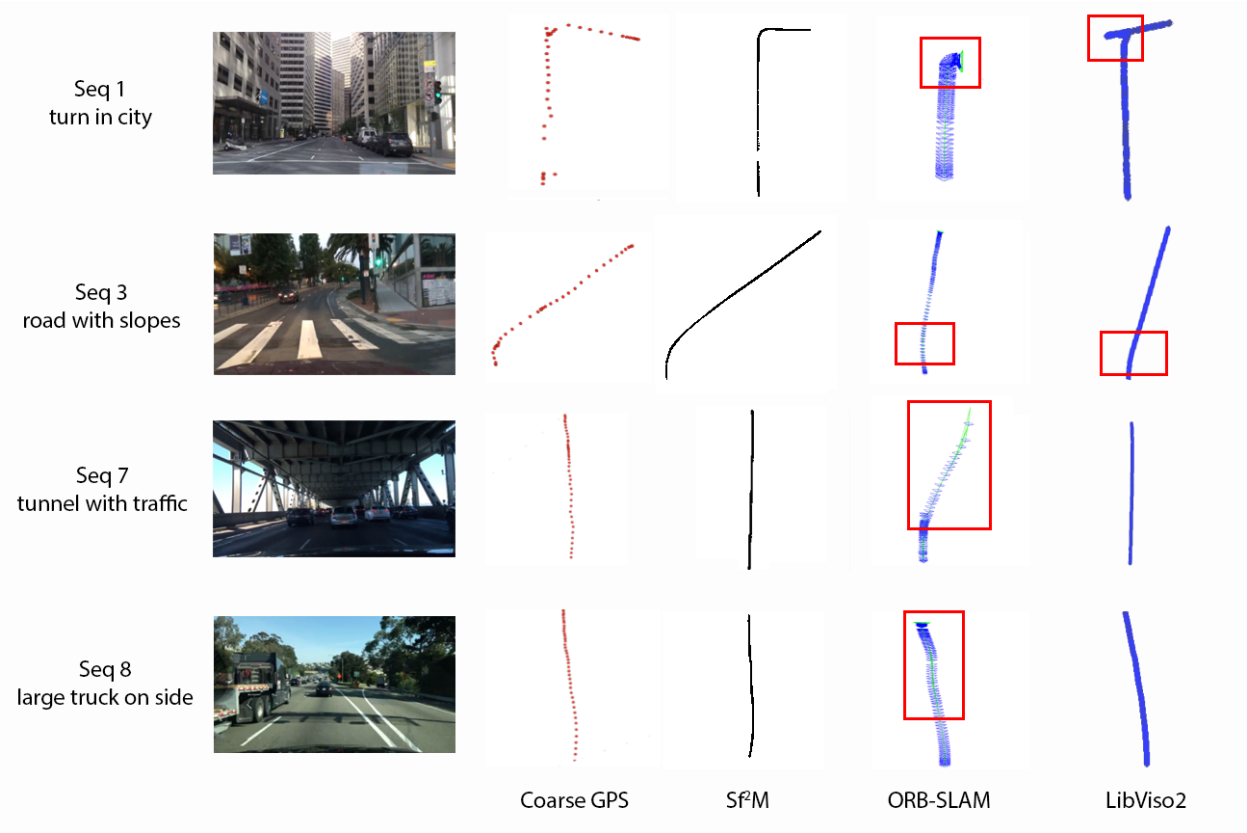


Figure 3.4: Reconstructed Trajectories with different methods. Red boxes highlight the errors. Our method outputs a convincing reconstruction for all the trajectories. ORB-SLAM [Mur-Artal et al., 2015] reconstructs the straight line in the beginning well, but fail after turning. Libviso2 [Geiger et al., 2011] is robust in a coarse manner but misses most of details. Coarse GPS are also shown for reference. The four visualizations are obtained from the corresponding software interfaces. Detailed Analysis is shown in Section3.4

reconstruction process. This is more straightforward in OpenSFM than in some frameworks, but it is uncommon that an SfM framework does not allow weighted matchings at least in theory, we note.

While OpenSFM is traditionally designed for unordered collections we adopt the following changes to apply it to videos. Since a video is inherently sequential, we limit the keypoint matching to a temporal neighborhood of 10 images. SIFT features are employed to extract around 4000 keypoints per image. To avoid the degenerate case of little or no motion, we adaptively downsample the sequence based on the inlier ratio of homography between adjacent frames. Specifically, we keep a frame if the homography inlier ratio between a current frame and the previously kept frame is less than 0.85.

3.4 Experiment

Experiment Setup

We aim to study the robustness of our proposed method on challenging driving videos. The recently released BDD Video Dataset [Xu et al., 2016] provides a suitable testbed.The videos were recorded by recent iPhones with rolling shutter cameras at 30fps and 1280×720 resolution. All videos are 40 seconds long. The videos are challenging for 3D reconstruction. The city scenes are complicated and covered by moving objects such as heavy traffic and crowded pedestrians. Rolling shutter artifacts are visible in some highway scenes, where the vehicle moves at high speed. In some videos, the camera jitters seriously due to road surface vibrations. Precise geometric calibration for each camera is not available. Instead, we provide a single approximate calibration for all videos. From an iPhone5's camera calibration, we set camera intrinsics as $f_x = f_y = 1200$, $c_x = 640, c_y = 360$ and skew coefficient as 0, distortion correction are omitted. All the following experiments adopt the same calibration.

We experiment with a dilated FCN [Yu and Koltun, 2015] model pre-trained on Cityscapes [Cordts et al., 2016] with 19-category segmentation, by specifying pedestrians, the sky, and all types of vehicles as matching outliers.

Baselines

We investigate ORB-SLAM and Libviso2 as two baselines. We choose these baselines as they have shown strong performance on driving sequences and are designed to reconstruct video sequences. Standard SfM systems are not suitable as baselines as they are usually optimized for unordered image collections and do not exploit the sequential nature of the data.

ORB-SLAM [Mur-Artal et al., 2015] is a recent effort in the SLAM community that combines many recent advances in an indirect framework. The default settings of ORB-SLAM fail on most of our videos. We have to increase the number of ORB keypoints from 2000 to 6000. To get better results, we also increased the number of pyramid levels from 8 to 12. Other parameters such as scale factor between levels in the pyramid and FAST [Rosten and Drummond, 2005, Rosten and Drummond, 2006] threshold are kept as default.

Libviso2 [Geiger et al., 2011] is especially tuned for driving scenarios, such as the KITTI visual odometry benchmark. We tried to adapt it to the BDD Video Dataset but the default setting fails on all sequences. We change the non-maximum suppression threshold to 1 pixel. We also increase the RANSAC iteration limit to 10000, which experimentally improves the matching stability. In addition, we increase the motion threshold to 40, to ensure resection will only happen when there is enough motion. The multi stage parameter is kept as 0, since we do not observe much difference by having multiple stages.

Visualization

As a first qualitative result, we show the 3D scene reconstruction obtained after running Sf^{2}M on two BDD Dataset videos. In Figure 3.3, notice the 3D point cloud reconstructions of the videos, colored according to the segmentation labels. We present the point cloud from both the camera view and a side view, to illustrate that our reconstructions look reasonable from different angles. In the first row, one can easily identify various different objects: the road, vegetation, street signs, and a well-defined lamppost. In the second example, notice how the 3D reconstruction accurately detects the left side buildings and the right side trees, which are visible in the image as well. These two visual examples provide an initial demonstration of the efficiency of Sf^{2}M.

Robustness Evaluation

We randomly select videos from the BDD Video Dataset [Xu et al., 2016], representing a wide range of dashcam videos recorded in the wild. The videos contain very challenging cases, including city scenes with heavy traffic and lane changing, and highway scenes with few feature keypoints, rolling shutter effects, and significant camera shaking. As we are interested in having a system which generates accurate trajectories even in challenging conditions we focus our evaluation on the robustness of each method when running on the BDD dataset.

Measures such as scale drift and accumulated rotational drift do not quantify the robustness of the methods, since these measures assume the reconstruction does not fail, i.e. outputs a plausible trajectory for all the frames. Some common failures include:

- Multiple reconstructions are returned, i.e. the camera trajectory is broken into several parts.
- Incorrect speed. This happens when large moving objects are predominant in the images and get tracked. They are then assumed to be the static background by the reconstruction algorithm.
- Erroneous rotations. This easily happens when keypoints from both the static background and a moving object are tracked at the same time.

With these failure cases present, we use the following measures to quantify the robustness of the reconstruction algorithms: the number of reconstructed frames until the first failure, and the number of reconstructed frames in total. Tracking failures are identified by manually analyzing the trajectories. If no keypoints are tracked or the reconstructed trajectory deviates significantly from the GPS signal (more than 10 meters or 30 degrees) we consider it a failure.

Although ORB-SLAM has taken advantage of many recent advances, it still fails when there are a large number of moving vehicles, especially when moving vehicles are close to the camera, such as sequences 2, 6, 7 and 8 (Table 3.1). When encountering a large number of moving vehicles, ORB will gradually lose its original tracks on the static objects and start to focus on the vehicles. After that, ORB will wrongly infer super slow motion and, thus, fail.

Table 3.1: Robustness of Sf^{2}M, ORB-SLAM and Libviso2 on 8 randomly selected video sequences. Each pair of numbers in the table denotes how many frames have succeeded before the first failure, and how many frames have succeeded in the whole video. Each video has 1200 frames in total, thus 1200/1200 means a robust reconstruction on all frames.

Sequence Id	1	2	3	4	5	6	7	8
Scenario	city	city	city	city	city	highway	bridge	highway
Traffic Level	light	heavy	medium	light	medium	light	heavy	heavy
Characteristics	turns	lane change	slopes	simple	long stop	large cars	many cars	large cars
Sf^2M	1200/1200	1200/1200	1200/1200	1200/1200	1078/1177	1200/1200	1200/1200	1200/1200
ORB-SLAM	104/104	352/352	290/1195	1200/1200	1185/1185	355/410	480/480	455/455
Libviso2	746/1065	60/938	1200/1200	1200/1200	1200/1200	0/200	1200/1200	96/1001

On the other hand, it succeeds in the simple cases, such as sequences 3, 4 and 5, where there is only slight to medium level of traffic.

Experimentally, Libviso2 is slightly more robust than ORB-SLAM. We attribute this to its scene specific optimization. However, it suffers from the same moving vehicle problem, as the ORB-SLAM: the motion is underestimated when the camera is in a crowd of moving vehicles. The failed sequences (2, 6, 8) are similar to that of ORB-SLAM.

Our method, on the other hand, explicitly excludes all moving vehicles and thus works much more robustly than the other methods. Among the subset of videos in Table 3.1, our method succeeds on all of them.

Effect of Moving Vehicles

In Figure 3.4, we show results for which all methods output a smooth trajectory (we exclude cases with large motion discontinuities). However, even in these cases the reconstructions can be erroneous due to moving objects in the scene. For example, in Sequence 8 (Fig. 3.4), a large truck is on the side of the image and most methods, including ORB-SLAM, track a large portion of features on the moving truck, since it is the dominant object in the scene and has rich texture. Since the truck is moving slowly and steadily, all methods return a reconstruction; however the erroneously tracked points induce a large error, i.e. the reconstruction algorithm considers the slowly moving object as static, which leads to wrong pose estimations.

Comparison with GPS+IMU Fusion

The crowdsourced video dataset that we are using does not contain accurate ground truth trajectories. The only trajectory data that is contained in the dataset is consumer grade GPS and IMU data, which is recorded with the sensors built into the mobile phones used for the data collection. This type of trajectory data is in practice not very accurate, e.g. lane changes cannot be identified. Despite that it still contains important information. By adding

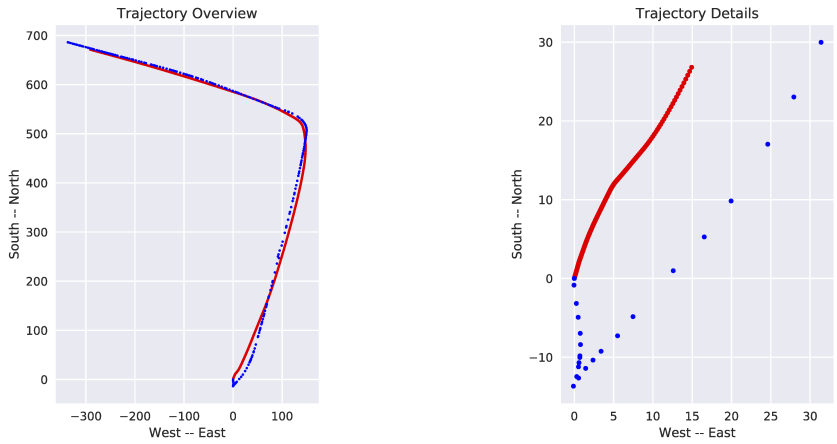


Figure 3.5: Both our reconstructed trajectory (blue) and GPS (red) trajectory are shown in this figure (left). At the starting point, zoomed-in trajectories are shown (right). The GPS trajectory does not contain the backing up and lane changing maneuver at the beginning of the sequence while our reconstruction recovers this motion.



Figure 3.6: Three frames extracted from the starting point of the sequence reconstructed in Fig. 3.5. Visually, it is clear that the car first backs up and then changes to the other side of the road to go forward. This maneuver is correctly recovered by our algorithm but not present in the GPS+IMU trajectory.

the coarse GPS locations as a soft constraint (l_2 loss) to the bundle adjustment absolute scale can be recovered and scale drift [Strasdat et al., ,Song and Chandraker, 2014,Song and Chandraker, 2015] can be effectively avoided.

In Figure 3.5, we compare the reconstructions recovered from our approach using the visual information and the GPS+IMU data with a trajectory acquired by fusing only the GPS+IMU signal without using the visual information. Two scenarios are analyzed, lane changes and backing up. In both cases the trajectory from GPS+IMU does not contain

the maneuver, while in the trajectory reconstructed with our approach these details are recovered. As an example, Fig. 3.6 shows the backing up maneuver conducted by the car at the beginning of the sequence, which is correctly recovered by our algorithm. The fact that GPS+IMU is unable to recover such maneuvers underlines the importance of being able to use the visual information to robustly recover accurate trajectories in crowedsourced driving data.

3.5 Discussion

We presented a method for reconstructing scenes in 3D from a single moving camera suitable for large-scale crowdsourced driving videos. We demonstrated that state-of-the-art methods fail on this type of data due to the challenges posed by the rolling shutter, motion blur, and, most importantly, the many moving objects which are present in general driving sequences. We developed a semantic filtering method that employs a Fully Convolutional Network (FCN) to directly predict the keypoints belonging to the static background in each frame. This allows us to robustly reconstruct the trajectories and scene structures from driving videos acquired in a crowdsourced manner, recorded with mobile phones mounted on a dashboard. As such, our method succeeds even in scenes dominated by moving objects. In future, semantic information can be further used to improve and densify the 3D reconstructions to eventually form a complete system for road scene reconstruction and understanding from crowdsourced data. Our work can help bridge 2D and 3D scene semantic analysis and introduce geometric properties back to segmented images. This will make crowedsourced driving data an important asset for research on self-driving applications in challenging city environments.

Chapter 4

Perception-Logical Policy

Human driving behavior can not be explained solely by traffic rules, rather, people take a lot of *scene factors* into account, e.g. driving slower than speed limit when many pedestrians are around. Autonomous driving system should obtain those behaviors as well, since it not only ensures natural interactions with non-autonomous divers and pedestrians, but also improves driving safeness. There are numerous scene factors that are related to driving, for which labeling each of them extensively can be a cumbersome job. We propose to leverage only demonstrative driving data to unsupervisely learn the reactions to those scene factors. Moreover, scene conditioned driving behavior have complex logical dependencies on the scene factors. We propose to combine the learned scene factors with a logic network, to finally output the driving behaviors. We end-to-end train the perception and the logic network, as retrieving intermediate supervision signals is expensive. Our experiments show that the proposed *Perception-Logic* network can unsupervisely learn meaningful scene factors and generalize almost perfectly in terms of scene conditioned behavior. The driving performance is significantly better than strong state-of-the-art baselines.

4.1 Background

Human vehicle driving is a complex behavior. People not only follow written traffic rules, but also tend to behave differently in various contexts. For example, people would drive as fast as the speed limit on an empty road. However, when passing through an urban narrow road with lots of pedestrians on the sidewalk, he probably won't drive as fast as the speed limit, even if there are no other vehicles on the road. The reason is simple and rather intuitive: it is more safe to drive slowly on the urban setting just in case some pedestrian suddenly goes off the sidewalk, while it is more efficient to drive fast in an empty, rural environment. We use the term *Scene Conditioned Driving Behavior* to refer to this phenomenon. It is an important but often ignored rule for the autonomous driving system to make an appropriate driving behavior according to the current scenario.

The scene conditioned driving behavior is highly dependent on the current scenario, and

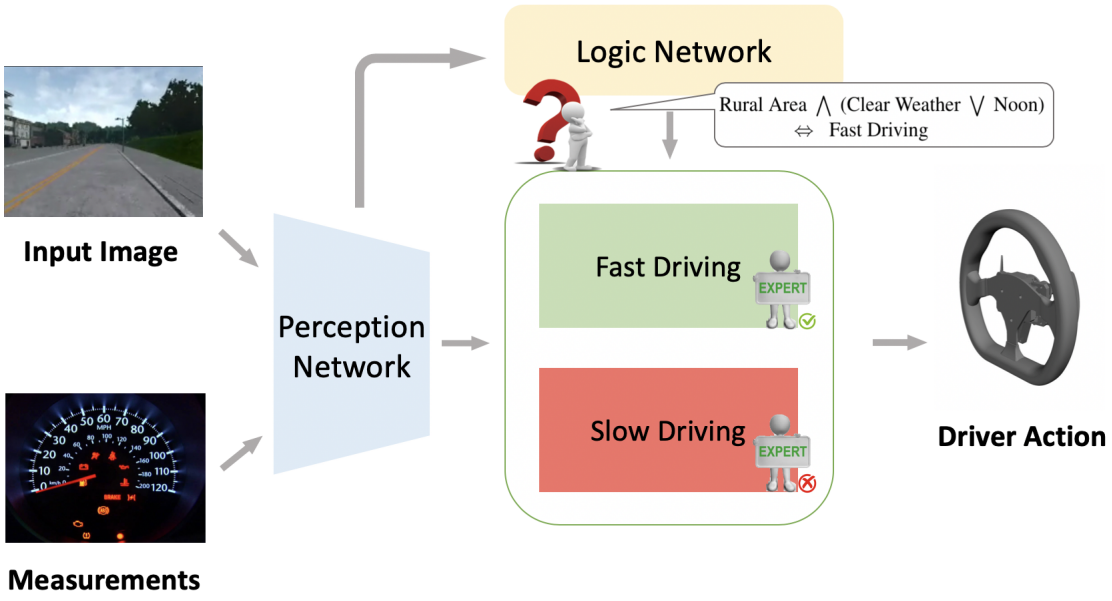


Figure 4.1: Human driving behavior varies dramatically according to the current scenes. For example, people will drive much more cautiously in a rainy weather, compared to a sunny weather. Those scene conditioned behavior often depends on multiple scene factors, and the dependency between the behavior and the scene factors might be connected through a complex logic rule. For example, we may drive faster only if there is no pedestrian nearby and the weather is not rainy and it is not dark. To solve this problem, we propose a Perception-Logic Network that unsupervisely learns the scene factors and combine them with a logic network. In terms of executing the scene specific driving behaviors, we find that our method significantly outperforms strong baselines.

might change dramatically with the presence of a scenario factor. As in the previous example, it is fine to drive fast without any vehicles on the road; however, it is better to drive cautiously when there are lots of pedestrians on the sidewalk, even if the road is empty. But if we note that the sidewalk is well separated from the main road by a traffic road barrier, then we can still drive according to the speed limit. Those conditions can be summarized with a single logical clause, namely, drive-fast if and only if empty-road and (has-barrier or no-pedestrian). Ideally, the autonomous driving system can strictly follow the aforementioned rule.

One way to solve this problem is to label every aspect of the driving scenario we care about, and code the driving system with the trained classifier. However, in practice, it is extremely time-consuming to label training data with all the attributes that matter. Due to the inherent complexity of the driving behavior, in the end of the day, there will be too many attributes that is requires prohibitive human labor. In this chapter, we propose to

learn those factors in an unsupervised manner. Given a set of expert driving videos, and the scene conditioned rules we need to follow, we would like to unsupervisely learn each factor from the images, and ground them to the factors in the logical formula. Most importantly, we would like the learned model to strictly follow the logic rules.

Most of the previous learning based driving agents [Pomerleau, 1989a, Bojarski et al., 2016b, Chen et al., 2015b, Amini et al., 2019, Codevilla et al., 2018] do not use any explicitly designed network architecture to strictly follow the logical rules. Rather, many of them rely on the generalizability of a generic CNN to follow the rules presented in the training data [Xu et al., 2017]. A well-established way is to use Mixture-of-Expert [and others Jacobs, Robert A and Jordan, Michael I and Nowlan, Steven J and Hinton, 1991] to handle this; however, we empirically show that a simple MoE might easily fail when the scene become complicated. Many of the previous works [Dong et al., 2019, Evans and Grefenstette, 2018] as well as our experiment show that a generic CNN generalizes poorly when there are complex logic patterns in the data.

In this chapter, we propose a *Perception-Logic Network* that can unsupervisely learn and ground logical factors directly from raw images. To achieve this goal, we designed a neural network architecture that takes in images, and unsupervisely output intermediate logic factors, which is later assembled according to the logic formula using a neural network. The proposed perception-logic network can be end-to-end trained with the demonstration data. We also proposed a batch-diversity loss to make sure we are learning the correct logic concepts. We have shown significant improvements in terms of following the logic rules compared to a number of baselines, and at the same time, our model also outputs explainable intermediate decision logic factors.

4.2 Related Work

Autonomous Driving The seminal work of Pomerleau [Pomerleau, 1989a] proposed to use learning method and neural networks to automatically learn driving policies from the data. Recently, the learning based autonomous driving has received a great attention. Nvidia [Bojarski et al., 2016b, Xu et al., 2017] adapted modern CNN to the driving task and used 3 cameras to provide both positive expert demonstrations and negative correction samples. Codevilla et al. [Codevilla et al., 2018] proposed conditional imitation learning, which resolved the inherent multi-modality of the driving task by assuming a high level command and a corresponding network architecture. [Chen et al., 2015b] proposed an intermediate level of abstraction, i.e. driving affordance, to combine the best of machine learning and classical control methods. Sauer et al. [Sauer et al., 2018] combined the affordance learning with the conditional imitation learning idea, which results in further improvements of the method. Beyond those works, people also have been making progress in architectures [Amini et al., 2019], driving models [Xu et al., 2017], planning algorithms [Bansal et al., 2018], etc.

However, none of the previous works deal with perceptual-logic reasoning on the driving domain: most of them use some form of CNN, and the intermediate neurons within those

CNN do not correspond to physical concepts. As a result, the logic behind the training data do not generalize well to the test data, as we will show in the scene conditioned driving experiments. We proposed a perception-logic network that can unsupervisely discover physical meaningful concepts and combine them with a logic network.

Perception and Logic Joint Models

Recently, people have extensively studied how to use learning method for logic tasks, such as inductive learning and logic reasoning [Richardson and Domingos, 2006, Kersting et al., 2000, Kimmig et al., 2012]. However, most of the aforementioned methods assume we have already abstracted the domain (image, language, or knowledge base) to a few logical variables. We are interested in how to complete a task with logic involved from raw perceptual inputs, such as images. Both Dong et al. [Dong et al., 2019] and Evans & Grefenstette [Evans and Grefenstette, 2018] designed end-to-end differentiable models that can learn logic rules from the data. They both demonstrate how their designed method could be applied to simple image domains, such as MNIST [and others LeCun, Yann and Bottou, L{\'e}on and Bengio, Yoshua and Haffner, 1998]. However, the ∂ILP method [Evans and Grefenstette, 2018] needs to pre-train the perception CNN on digit classification, violating our assumption that no previous knowledge is required. NLM [Dong et al., 2019] claims to be able to learn perception representations and logic rules in an end-to-end manner, but only on simple image domains.

Many works [Hu et al., 2018, Hu et al., 2019, Mascharka et al., 2018] in vision and language domain jointly learns over raw sensory data, such as images, and reason with logic on top of it. Neural Module Networks [Andreas et al., 2016] propose to compose per-instance inference network by composing neural modules together on the fly. The neural modules are assembled by instructions from the parsing of natural language sentences. Later works have extended it in many aspects, such as learning the language parser end-to-end [Hu et al., 2017a], extends into the multi-task RL setting [Andreas et al., 2017]. Some works [Santoro et al., 2017, Hu et al., 2017b] deal with special kinds of reasoning, i.e. pairwise relation reasoning, because they are common in many vision and language tasks. Though related, those methods differ from ours because they utilize rich language priors to instruct the logic part.

Architectures Part of our proposed *Perception-Logic Network* architecture is similar to Mixture of Experts (MOE) model [and others Jacobs, Robert A and Jordan, Michael I and Nowlan, Steven J and Hinton, 1991, Jordan and Jacobs, 1994]. MOE architectures aim to divide the output concept into several easier concepts, and at the same time, use a switching network to automatically switch from those experts. MOE can only deal with simple switching logic, while our proposed method can in theory handle any complex pre-defined logic rule. The vanilla MOE method suffer from "mode collapse", i.e. when learning the experts and switching network at the same time, the switching network tends to assigning higher weight to the one or few better performing experts. Those experts with higher switching weights are trained more rapidly than the other experts. The two process reinforce each other. As a result the switching network will only select the best experts. Many works have been done to alleviate this issue [Shazeer et al., 2017]. We found that this issue is even more severe when we have a logic network, since the order of the experts matter in our case. We proposed the *diversity loss on logic network* to resolve this issue.

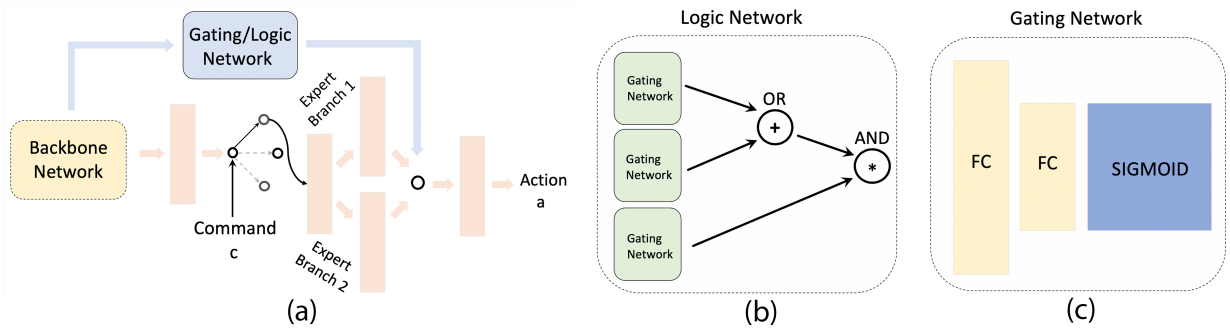


Figure 4.2: (a) The overall architecture of our method. The backbone is the same as the Conditional Imitation Learning baseline. The logic and gating network architectures are shown in (b) and (c) respectively. (b) The Logic network architecture. The logic network takes input from the scene factor prediction and combines them with continuous approximation to logical operations. (c) The Gating network architecture. The gating network outputs a continuous approximation to the discrete logic factor with the Sigmoid layer.

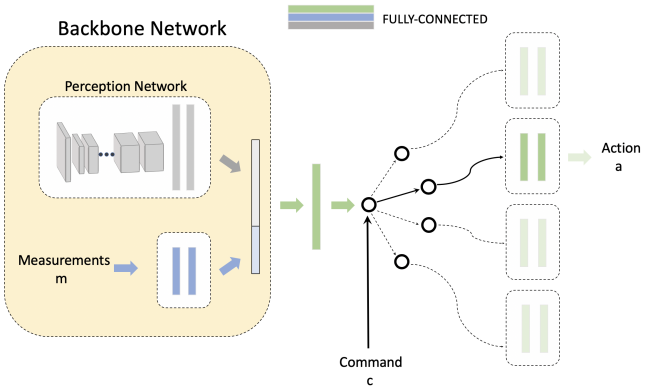


Figure 4.3: Visualization of the conditional imitation learning [Sauer et al., 2018] baseline. The CIL method assumes a high level navigational input to the network and the neural network has one branch for each high level command. The high level command selects the branch during training and testing time.

4.3 The Perception-Logic Network

In this section, we will describe the scene conditioned driving problem formulation, and our proposed *Perception-Logic Network* to solve this problem.

Problem Formulation

Let's first formulate the scene conditioned driving problem. At each timestep t, we have a representation of the whole driving scenario s_t, such as the current image observation. We would like to learn from a set of expert driving demonstrations, denoted by:

$$\{(s_t, a_t)\}$$

where a_t is the corresponding expert actions. In practice, a large amount of expert demonstration can be easily collected by recording human driving behaviors. The expert driving behaviors are scene conditioned, i.e. the behavior of the driver differs across scenarios. Here, we assume the scene can be described by a set of k Boolean factors, denoted by:

$$\{x_1, x_2, \cdots, x_k\}$$

such as: is this a rural area (x_1) or an urban area ($\neg x_1$), or is it a rainy weather (x_2) or a sunny weather ($\neg x_2$). We do not assume knowing the annotations for those scene factors, since in practice, it is cumbersome to annotate a large amount of possible scene factors for many images. On the contrary, it took much less man hours to provide rules, under which condition to perform a certain driving style. For example, we should drive more cautiously when it is rainy or it is a crowded urban area. Those conditions can be described by Boolean functions of the scene factors, such as $\neg x_1 \vee x_2$ for the case above. We assume we have a set of rules, that each of them describes a particular driving style. Those rules can be represented by r Boolean functions, $f_i(x_1, \cdots, x_k), i = 1 \cdots r$, where when f_i is true, one should follow the corresponding driving style.

The unsupervised scene conditioned driving problem is defined as: given a set of driving demonstrations $\{(s_t, a_t)\}$, and the corresponding set of scene conditioned symbolic rules $f_i(x_1, \cdots, x_k), i = 1 \cdots r$, to learn a driving agent $a = \pi(s)$ that exhibits the scene conditioned driving behavior. Note that the problem is unsupervised in terms of the scene factors $x_1 \cdots x_k$, since they are not provided as labeled data.

Perception-Logic Network

It is hard for a vanilla CNN to learn the exact behaviors when the data is generated by an inherent process involving complex logic [Dong et al., 2019]. As shown in our experiment section, the vanilla CNN method almost completely fails to capture the complex logic behind the scenes. In this section, we propose the *Perception-Logic Network* that allows the model to learn meaningful scene factors and compose them according to the logic rules. Besides the model architecture, we also proposed two techniques to make training the network more stable, i.e. a diversity loss on logic network and a batch wise logic-value re-normalization. The experiments show that our method achieves almost perfect action generalization in terms of scene conditioned behavior modeling.

Figure 4.2 shows our proposed solution to the aforementioned problem. We proposed to use Conditional Imitation Learning (CIL) [Sauer et al., 2018] as our backbone network. In

the autonomous driving application, the agent sometimes have multiple potential actions to take at an intersection, such as turning left, turning right and going straight. The usual regression method will not work in this scenario, since there are multiple ground-truth labels. CIL assumes we are given the navigational command (LEFT, RIGHT or STRAIGHT) during training and testing time, and it proposes to use multiple branches in the neural network to deal with each command separately. Our method modifies each branch of the backbone in a similar manner, thus for the remainder of this section, we only describe our method in terms of one branch.

Scene Factors The proposed *Perception-Logic Network* outputs the probability of each of the k scene factors being true. This is achieved by using k Sigmoid functions following the backbone. Each of the k scene factors lies in the range of 0 to 1, and they can be seen as a continuous approximation to the discrete Boolean variables. There is no supervised loss attached to those k scene factors, rather, they are trained in an end-to-end manner. Figure 4.2 (c) shows the gating network used to output one scene factor. Figure 4.2 (b) shows the case for outputting 3 scene factors. Those gating networks are connected to the backbone network.

Logic Network We propose to use a neural network to approximate logical computations among Boolean variables, as shown in Figure 4.2 (b). In particular, we propose to use $x \times y$, $\min(x + y, 1)$ and $1 - x$ to approximate the logical computations of $x \wedge y$, $x \vee y$ and $\neg x$. Similarly, these can be seen as continuous approximations. We compose r logic networks, based on the r input logical rules. The network will output true if the agent should drive according to a particular scene. Abusing the notation a little bit, we represent the outputs of those logic network as $f_i, i = 1 \cdots r$. Figure 4.2 (b) shows one instance of the logic network, and Figure 4.2 (a) uses two logic network, one for each expert branch.

Scene Behavior Experts and Experts Fusion We also output r scene experts driving behaviors $a_1, a_2, \cdots, a_r$, denoting how to drive under each of r possible scenarios. Those r scene experts are fused by the r logic network outputs: $a = \sum_{i=1}^{r} a_i \times f_i$. We note that there is no training loss for each expert, since we do not get that supervision. Instead, the r scene experts are also automatically learned from the data.

Diversity Loss on Logic Network We found that simply using the above mentioned architecture suffers from the "mode collapse" issue [Shazeer et al., 2017]. The scene factors typically converge to true or false all the time, no matter what is the input. The network degenerates to the vanilla CIL in this case. We propose to fix this by enforcing a *diversity loss* on the logic network. Assuming we know the empirical distribution of each scene factors, for example, 70% of the days are sunny days and the remaining 30% are rainy days. We can compute the expected distribution of any intermediate logic variables in the logic network. We propose to enforce the empirical distribution of each input and intermediate logic variable v of the logic network by:

$$L_d = \left| ||\mathbf{v}||_2^2 - ||\mathbf{t}||_2^2 \right| + \left| ||1 - \mathbf{v}||_2^2 - ||1 - \mathbf{t}||_2^2 \right|$$

where the loss is computed over a training batch of size B. $\mathbf{v}$ is a size B column vector of the

values v in this batch. $\mathbf{t}$ is the target vector composed of $p_v B$ 1s and $(1-p_v)B$ 0s. p_v is the expected probability of v being true. This loss makes sure the distribution of a single node matches the expectation. We need both the v and $1-v$ side losses because otherwise the optimizer can easily find non 0/1 solutions to v values in a batch to minimize the diversity loss.

Note that the diversity loss is applied on **every** node of the logic network, including both the input unsupervised learned scene factors and the intermediate logic variable during the computation of the logic network. We name the diversity loss on the input logic factors as *Gating Network Diversity Loss*, and name the diversity loss on the intermediate computations as *Switch Weight Diversity Loss*. We will ablate the two components of the whole diversity loss in the experiment section.

Batch Logic Value Re-Normalization Since we use a continuous approximation to the discrete logic value, the approximated logic values will accumulate error after a few logical operations. This problem is similar to the inaccuracy problem when computing using analog circuit. For example, if the two inputs to an `and` gate are 0.6 and 0.6, the continuous output would be 0.36, i.e. `False`. However, the correct output should be `True`. We propose to resolve this issue by using *Batch Logic Value Re-Normalization*. The intuition is that although 0.36 is smaller than the standard threshold 0.5, however, it is much larger than the continuous `and` result of a `False` value, such as the result of 0.1 and 0.9. We propose to normalize each of the output of a logic operation with:

$$o_i^{norm} = \frac{o_i - o_{min}}{o_{max} - o_{min}}$$

where o_i is one logic output in the current batch, and o_{min}, o_{max} are the min and max logic output in this batch.

Other Losses Besides the diversity loss, we also apply an MSE regression loss to regress the action output a. In this work, the action is a 3-dimensional vector, including the steer, throttle and brake. We weight each action prediction by 1.0 and each diversity loss on the logic network node by 0.001.

4.4 Experiments

Experimental Setup

We use the CARLA [Dosovitskiy et al., 2017] simulator, which is one of the closest proxy to the real world driving environment, to train and evaluate our model. We collect our training data from a simulated internal autonomous driving agent. Our data is collected in an episodic fashion that in each episode, the agent is initialized at a random location and set to drive to a certain destination. During data collection, the autonomous driving agent is given explicit commands, such as follow, straight, left and right and is supposed to make corresponding actions. We choose CARLA 0.8.4 version specifically for our experimentation.

	Logic Network	GND Loss	SWD Loss	Speed Accuracy	Weather Accuracy	Temporal Accuracy	Collision Percentage	Intersection Percentage
Two Logic Factors (Clear vs. Rainy & Noon vs. Sunset)	-	-	-	76%	N/A	N/A	27%	3%
	✓	-	-	77%	51.15%	51.47%	0%	0%
	✓	✓	-	90%	70.99%	72.21%	0%	0%
	✓	✓	✓	99%	85.87%	86.86%	0%	0%

Table 4.1: Evaluation Performance of Logic expression with 2 Variables. Our logic network consists of a *weather* factor and a *temporal* factor. **GND Loss** represents the use of a Gating Network Diversity Loss; **SWD Loss** represents the use of a Switch Weight Diversity Loss. We use **Speed Accuracy** to indicate the accuracy of the model action. We use **Weather Accuracy** to indicate how well the model tells apart different weather conditions, such as *Clear* vs. *Sunny*. We use **Temporal Accuracy** to indicate how well the model tells apart different time conditions, such as *Noon* vs. *Sunset*. **Collision Percentage** indicates the number of collision frames over total frames. **Intersection Percentage** measures the number of frames the driving agent intersects the wrong lane.

	Logic Network	GND Loss	SWD Loss	Speed Accuracy	Weather Accuracy	Temporal Accuracy	Landscape Accuracy	Collision Percentage	Intersection Percentage
Three Logic Factors (Clear vs. Rainy & Noon vs. Sunset & Rural vs. Urban)	-	-	-	43%	N/A	N/A	N/A	37%	0%
	✓	-	-	45%	50.82%	52.46%	53.91%	18%	0%
	✓	✓	-	48%	55.47%	64.06%	53.13%	18%	0%
	✓	✓	✓	99%	78.34%	70.83%	85.88%	10%	0%

Table 4.2: Evaluation Performance of logic expression with 3 Variables. We incorporate an additional *landscape* factor in our logic network. We use **Landscape Accuracy** to indicate how well the model tells apart different geographical conditions, such as *Rural* vs. *Urban*.

Our training data consists of 7 different configurations to learn a robust policy: two different camera positions (height of 1.4m and 1.8m above the ground), two different image sizes (longer side of images rescaled to 700 and 800 pixel) and three camera rotational pitches (angle of -5, 0, and 5 degrees with the ground). We capture training data in the form of a stream of images together with measurements (e.g. speed values) across each time step. Each image is of resolution 576×768. To simulate real-life driving, we also add spike noise to the action (eg. throttle, steering) of our Carla agent. We pre-process the collected images by downsampling images to 144×192 resolution. We also do data augmentation as mentioned in [Codevilla et al., 2017] that apply a random subset of a large set of transformations with randomly sampled magnitudes, including change in contrast, brightness, region dropout and Gaussian noise.

All experiments were trained on a NVIDIA Tesla m40 GPU with 24GB memory storage and 3072 CUDA cores. We utilize the ADAM optimizer for training with learning rate 1e-4 for all of our experiments.

Data Collection

We setup two scenarios to test our proposed method, one with two scene factors and the other with three scene factors. In the two factor experiments, we consider the weather scene factor (either rainy or clear) and the temporal factor (either noon or sunset). If it is a rainy weather and at sunset, we would like to drive more carefully. We ask the agent to drive slower in this particular combination of the scene factors. In the other settings, we drive normally. The logic rule in this setting is: rainy `and` sunset $\iff$ drive_normally. We set normal driving speed to be 20 km/h, and slow driving to be 10 km/h.

As our experiments show, even for this simple `and` logic, both of the vanilla CNN baseline and the Mixture of Experts baseline already fail. To test how our method performs in a more logical complex setting, we also collect a 3-scene-factor dataset. In this more complex example, we include a new landscape factor: rural or urban. Since rural area are usually populated less than the urban area, we change the scene conditioned driving behavior to: rural `and` (clear `or` noon) $\iff$ drive_normally. i.e. we will drive normally (comparatively faster), if and only if we are in an rural area and it is sunny or it is noon, otherwise we will drive slowly. Here, we use the same two driving behaviors (drive normally vs. slowly) as the two factor dataset.

Our training data consists of 20000 images for each configuration. Since there are 7 configurations, there are a total of 140000 training images and corresponding driving commands in our training set.

Architectures

To conduct fair comparisons among experiments, we utilize a shared perception network as a backbone architecture for all models evaluated:

The perception network consists of 8 convolution and 2 fully connected layers. The convolution layers start with a channel size of 32 and is increased by a factor of two for every two convolution layers. We also alternate between a stride of 1 and 2 for adjacent convolution layers. Each convolution layer has a kernel size of 3, besides the leading one which has a kernel size of 5. Each convolution layer is followed by a batch-normalization layer, a dropout layer, and a nonlinear activation function. For dropout we uniformly choose a dropout rate of 0.1. We use ReLU or rectified linear unit as our activation function. The final output of the convolution layers is then fed into two side-by-side fully connected layers with final output dimension 512.

The input images are passed through the perception network pipeline. The speed measurements are passed through a normal multi-layer fully connected network and has an output dimension of 128. Then, we concatenate the encoded image vector with speed vector, and use an additional dense layer to reduce the concatenated vector to a dimension of 512, which is then used as the input vector to each expert branch. Each expert branch is simply a two-layer fully connected network with an output dimension of 512.



Figure 4.4: Four sampled images for the qualitative study. The first two images are sampled from the rural area and the last two images are sampled from the urban area.

Model	Image 1				Image 2				Image 3				Image 4			
	WT	TP	TN	SP	WT	TP	TN	SP	WT	TP	TN	SP	WT	TP	TN	SP
G	1	1	1	20	1	1	1	20	1	1	0	10	1	1	0	10
A	-	-	-	10	-	-	-	10	-	-	-	10	-	-	-	10
B	-	-	-	10	-	-	-	10	-	-	-	10	-	-	-	20
C	0	1	1	10	0	1	1	10	1	1	1	10	1	1	1	10
D	1	1	1	20	1	1	1	20	1	1	0	10	1	1	0	10

Table 4.3: Gating Network prediction accuracy and driving speed accuracy. The row with Model G is the ground-truth label for weather Gating Network (**WT**), Temporal Gating Network (**TP**), Town Gating Network (**TN**) and output driving speed (**SP**) in unit km/h. For weather gating, **1** represents clear and **0** represents rainy; for temporal gating, **1** represents noon and **0** represents sunset; for town gating, **1** represents rural area and **0** represents urban area and for speed, **10** represents slow driving at 10km/h and **20** represents fast driving at 20km/h, which is the speed of experts' driving behaviour in our training data. The row with Model A is the output of single branch baseline. The row with Model B is the output of Mixture-of-Experts(MoE). The row with Model C is the output of our double branch network without any diversity loss and the row with Model D is the output of our proposed network with diversity loss and batch logic value re-normalization.

We use L-2 loss between the network output actions and the expert's actions collected from CARLA simulator.

Training

Besides the details mentioned above, for all the methods, we train the networks for 60k iterations in an end-to-end manner. To speedup training, we have a scheduled learning rate that decreases when the loss plateaus with a factor of 1/3. Also, we stop the training process when the prediction accuracy of each logical factor and the overall driving action loss converge. Otherwise, the model may overfit and affect the driving behaviour of our autonomous driving

agent.

Moreover, as our diversity loss is batch-wise and its effectiveness is theoretically guaranteed by Central Limit Theorem, we set our training batch size to a large number, e.g. 128. Also, for each batch, we randomly sample data from our training dataset so that the data with different scene conditions are evenly distributed in each batch.

Baselines and Ablations

We compare the proposed method with the following baselines and ablated versions:
Single Branch Network Our first baseline is the Conditional Imitation Learning architecture proposed in [Codevilla et al., 2017]. We call it a single branch network because there is only one branch for each command. Each branch learns a mapping from images and driving command to the expert's driving behavior. This baseline is shown in Figure 4.3.
Mixture of Experts We also compare to the mixture of experts [Jordan and Jacobs, 1994] baseline. We use two experts in this setting. We also use a generic neural network as the gating network. This method is equivalent to our method without the logic network. By comparing to the MOE method, we could investigate the importance of the logic network.
Perception-Logic without SWD Loss We also would like to ablate the diversity loss on the network. This method ignores the Switch Weight Diversity Loss, such that we have the logic network and the Gating Network Diversity Loss (GND).
Perception-Logic without Diversity Loss Similar to the Perception-Logic without SWD Loss, we leave out the whole diversity loss, including both the SWD and the GND Loss. This ablative method will show the role of the diversity loss.

Evaluation Metrics

To evaluate each method, we propose to use multiple evaluation metrics. The most critical evaluation metric is how the agent follows the scene conditioned driving behavior. In our setting, the scene conditioned behavior is driving normally (20 km/h), or driving slowly (10 km/h). We use a *speed accuracy* to measure the quality of the scene conditioned behavior. The computation of speed accuracy is similar to a classification accuracy: we round the current speed to 10 km/h or 20 km/h, and count how many frames during testing has the correct speed.

We would also like to see how our unsupervised scene factor learning works. Specifically, we would like to know the answer to the following question: Does the unsupervisely learned scene factor correspond to the ground truth factor? Fortunately, we can easily verify this in the Carla simulator. We use the *factor accuracy* to measure them. In our 3-factor experiment, we measure the factor accuracy for weather, time of the day and the landscape.

We would also like to evaluate how much collision our agent experienced during the testing. Following previous work [Dosovitskiy et al., 2017], we use the *collision percentage* and *intersection percentage* to measure the driving quality in general. They refer to the

number of collision frames over total frames, and the number of vehicle invading other lanes over total frames respectively.

Quantitative Results

We present results of two-factor and three-factor environmental settings in this section. For each condition, we compare our *Perception-Logic Model* with one baseline model as well as several ablated versions of the full architecture. The baseline model consists of a single-branched command-conditional network as proposed by [Codevilla et al., 2017]. The ablated versions of the *Perception-Logic Network* consists of: a model with only Gating Network and no diversity loss, a model with Gating Network and diversity loss per gate only. We utilize 5 metrics to consider the performance of each model on the two-factor setting, which consists of Speed Accuracy, Weather Accuracy, Temporal Accuracy, Collision Percentage, and Intersection Percentage. We introduce a sixth metric for the three-factor setting to account for the additional landscape factor.

Two-factor Results and Analysis The results of the previously discussed 2-factor and 3-factor settings are presented in Tables 4.1 and 4.2 respectively. For the two-factor setting, the single-branch baseline achieves a speed accuracy of 76% and also has a relatively high error rate. It is more prone to make wrong decisions as seen in a collision percentage of 27% and intersection percentage of 3%, most likely due to the model's inability to learn different driving styles across multiple environment settings.

On the contrary, all models implemented the Gating Network achieves a 0% collision percentage and a 0% intersection percentage. The model that only implements the Gating Network achieves an on-par performance with the baseline model in speed accuracy, while avoids making any erroneous behavior as compared to the baseline model. This shows the ability of the Gating Network to capture and separate general driving rules from environment-specific driving styles. However, this model only learns a 51.15% and 51.47% accuracy in predicting weather and temporal factors respectively, which is in fact similar to a random switch.

During experimentation, we find that the use of a diversity loss contributes significantly to the model performance by improving logic-learning for our Gating Network. We conclude that the diversity is one of the key component of our method. The model that implements the Gating Network as well as a Gating Network Diversity Loss (**GND**) achieves an increased speed accuracy of 90% while also learns a 70.99% and 72.21% accuracy in predicting weather and temporal factors respectively.

Our *Perception-Logic Model*, on the other hand, achieves a superior speed accuracy of 99%. The use of diversity loss across outputs of different logic operation helps maintain a close relationship between the input data distribution and the output distribution. In turn, we are able to produce a 85.87% accuracy on weather prediction, as well as a 86.86% accuracy on temporal prediction.

Three-factor Results and Analysis For the three-factor setting, we add an additional landscape factor into the environment. Due to the increased complexity of logic present in the

training data, all models produce relatively worse results than their two-factor counterparts. The baseline, for instance, achieves a speed accuracy of only 43% while being more likely to make incorrect actions with a collision percentage of 37%. However, the use of a Gating Network still manages to eliminate most cases of incorrect behavior, decreasing the collision percentage from 37% to at most 18%. Similar to the two-factor setting, the use of both the Gating Network Diversity Loss (**GND**) and the Switch Weight Diversity Loss (**SWD**) greatly improves weather and temporal predictions, as well as the newly introduced landscape variable prediction. Our *Perception-Logic Model* achieves accuracy of 99%, 78.34%, 70.83%, 85.88% for speed, weather, temporal and landscape respectively.

Qualitative Results

In Table 4.3, the single branch baseline tends to prefer driving slowly even in a sunny noon of a rural area. Also, the naive Mixture-of-Experts network has similar performance as the single branch network as one gating network is hard to learn the complex logical rule corresponding to each visual scenario. After adding our proposed logic network, including a gating network for each logic variable with continuous approximation of logical operations, the network still suffers from mode collapse of Mixture-of-Experts as the output of each gating network aren't restrained to match the expected distribution. Thus, the row with Model C predicts weather and town incorrectly sometimes, causing the speed output to be incorrect. For instance, in both Image 1 and 2 of Figure 4.4, the weather is clear, but the model without diversity loss predicts it as rainy; in both Image 3 and 4 of Figure 4.4, the town is of an urban setting, but the model without diversity loss predicts it as a rural environment. And as shown in the last row, by adding diversity losses for each gating network and output after each continuous approximation of logical operation, we achieve perfect accuracy in terms of driving speed and significantly improve the prediction accuracy of each logic variable.

4.5 Limitations

First, our proposed method needs a pre-defined logic rule to manage the connections of different neural logic modules. This is theoretically hard to solve as the search space for the rules can be exponentially large in terms of the number of logic variables. Moreover, providing logic rules are much less time-consuming and realistic than labeling every driving images, which makes our method practical in the real-world.

Second, our method might have the problem of extending to a *very* large number of logical variables. This is understandable that as the number of logic variables increases, their possible combinations grow factorially, making it extremely hard to find the correct ordering of the variables to fit the pre-determined logic rule.

4.6 Discussion

In this chapter, we study how to utilize visual information more wisely in order to improve the accuracy of driving behaviour, and unsupervisely capture important logical decision factors in images. We propose a new architecture, called the *Perception-Logic Network* and systematically evaluate the performance of our model in comparison to the state-of-art single branch network and mixture-of-experts network. We find that our network produces significantly better results than other baselines and ablated models.

In the future, we'd like to explore better methods and architectures to further improve our logic factor prediction accuracy and understand why an imperfect prediction of logic factors still leads to an almost perfect driving behavior. Additionally, we believe our *Perception-Logic Network* is easy to generalize to other domains, such as natural language processing, which we also leave as our future work.

Chapter 5

Combining Imitation Learning and Reinforcement Learning

Robust real-world learning should benefit from both demonstrations and interactions with the environment. Current approaches to learning from demonstration and reward usually perform supervised learning on expert demonstration data and use reinforcement learning to further improve performance based on the reward received from the environment. These two tasks have divergent losses which are difficult to jointly optimize, and such methods can be very sensitive to suboptimal demonstrations. We propose a unified reinforcement learning algorithm, Soft Advantage Learning (SAL), that effectively normalizes the Q-function, reducing the Q-values of actions unseen in the demonstration data. SAL learns an initial policy network from demonstrations and refines the policy in the environment, surpassing the demonstrator's performance. Crucially, both learning from demonstration and interactive refinement use the same objective, unlike prior approaches that combine distinct supervised and reinforcement losses. This makes SAL robust to suboptimal demonstration data, since the method is not forced to mimic all of the examples in the demonstration. We show that our unified reinforcement learning algorithm can learn robustly and outperform existing baselines when evaluated on several realistic driving games.

5.1 Background

Deep reinforcement learning (RL) has achieved significant success on many complex sequential decision-making problems. However, RL algorithms usually require a large amount of interactions with an environment to reach good performance [Kakade et al., 2003]; initial performance may be nearly random, clearly suboptimal, and often rather dangerous in real-world settings such as autonomous driving. Learning from demonstration is a well-known alternative, but typically does not leverage reward, and presumes relatively small-scale noise-free demonstrations. We develop a new robust algorithm that can learn value and policy functions from state, action and reward (s, a, r) signals that either come from imperfect

demonstration data or the environment.

Recent efforts toward policy learning which does not suffer from a suboptimal initial performance generally leverage an initial phase of supervised learning and/or auxiliary task learning. Several previous efforts have shown that demonstrations can speed up RL by combining imitation loss with reinforcement learning loss [Hester et al., 2017, Rajeswaran et al., 2017, Sun et al., 2018, Nair et al., 2018, Večerík et al., 2017], yet these methods presume near-optimal demonstrations. Prior works that also leverage imperfect demonstrations assumes extra information is available, such as confidence scores on the demonstrated actions [Wu et al., 2019]. [Jaderberg et al., 2016] and [Shelhamer et al., 2016] obtained improved initialization via auxiliary task losses (e.g., predicting environment dynamics) in a self-supervised manner; policy performance is still initially random with these approaches.

A simple combination of several distinct losses can learn from demonstrations; however, it is more appealing to have a single principled loss that is applicable to learning both from the demonstration and from the environment. Our approach, Soft Advantage Learning (SAL), when given a set of demonstrations consisting of transitions (s, a, r, s') and the corresponding MDP definition, uses a unified loss function to process both off-line demonstration data and on-line experience based on the underlying maximum entropy reinforcement learning framework [Toussaint, 2009, Haarnoja et al., 2017a, Schulman et al., 2017]. This learning protocol is widely applicable in a variety of domains, including robotics and autonomous driving, where demonstration data is available but additional interaction is necessary for the mastery of skills. Our method also enables robust learning from corrupted, or imperfect demonstrations, because it does not assumes the optimality of the demonstrations.

We demonstrate our approach in a toy Minecraft Game with discrete states and tabular Q functions as a proof-of-concept experiment. We also evaluate the proposed method on a 3D simulated environment, Torcs , where inputs are raw images and Q functions are approximated by neural networks. Our experimental results outperform previous approaches on driving tasks with only a modest amount of demonstrations while tolerating significant noise in the demonstrations.

In summary, we propose the Soft Advantage Learning (SAL) method. It utilizes a unified objective, capable of learning from both demonstrations and environments, that outperforms methods including the ones with an explicit supervised imitation loss. Moreover, unlike the other methods that purely imitate from the demonstrations, our proposed method is robust to noisy demonstrations.

5.2 Preliminaries

In this section, we will briefly review the reinforcement learning techniques that our method is built on, including maximum entropy reinforcement learning and the soft Q-learning.

Maximum Entropy Reinforcement Learning

The reinforcement learning problem we consider is defined by a Markov decision process (MDP) [Thie, 1983]. Specifically, the MDP is characterized by a tuple $< \mathbb{S}, \mathbb{A}, \mathrm{R}, \mathrm{T}, \gamma >$, where $\mathbb{S}$ is the set of states, $\mathbb{A}$ is the set of actions, $R(s,a)$ is the reward function, $T(s,a,s') = P(s'|s,a)$ is the transition function and γ is the reward discount factor. An agent interacts with the environment by taking an action at a given state, receiving the reward, and transiting to the next state.

In the standard reinforcement learning setting [Sutton and Barto, 1998], the goal of an agent is to learn a policy π_{std}, such that an agent maximizes the future discounted reward:

$$\pi_{std} = \underset{\pi}{\operatorname{argmax}} \sum_t \gamma^t \mathop{\mathbb{E}}_{s_t, a_t \sim \pi} [R_t]$$

Maximum entropy policy learning [Ziebart, 2010, Haarnoja et al., 2017a] uses an entropy augmented reward. The optimal policy will not only optimize for discounted future rewards, but also maximize the discounted future entropy of the action distribution:

$$\pi_{ent} = \underset{\pi}{\operatorname{argmax}} \sum_t \gamma^t \mathop{\mathbb{E}}_{s_t, a_t \sim \pi} [R_t + \alpha H(\pi(\cdot|s_t))]$$

, where α is a weighting term to balance the importance of the entropy. Unlike previous attempts that only adds the entropy term at a single time step, maximum entropy policy learning maximizes the discounted future entropy over the whole trajectory. Maximum entropy reinforcement learning has many benefits, such as better exploration in multi-modal problems and connections between Q-learning and the actor-critic method [Haarnoja et al., 2017a, Schulman et al., 2017].

Soft Value Functions and Soft Q-Learning

Since the maximum entropy RL paradigm augments the reward with an entropy term, the definition of the value functions becomes

$$Q_\pi(s,a) = R_0 + \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi} \sum_{t=1}^{\infty} \gamma^t (R_t + \alpha H(\pi(\cdot|s_t)))$$

and

$$V_\pi(s) = \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi} \sum_{t=0}^{\infty} \gamma^t (R_t + \alpha H(\pi(\cdot|s_t)))$$

, where π is a policy that value functions evaluate on. Given the state-action value function $Q^*(s,a)$ of the optimal policy, [Ziebart, 2010] shows that the optimal state value function and the optimal policy could be expressed as:

$$V^*(s) = \alpha \log \sum_a \exp(Q^*(s,a)/\alpha)$$

Algorithm 1: Soft Advantage Learning (SAL) from Demonstrations and Environments

θ: parameters for the rapid Q network, θ': parameters for the target Q network, $\mathcal{D}$: demonstrations collected by human or a trained policy network, T: target network update frequency, $\mathcal{M}$: replay buffer, k: number of steps to train on the demonstrations

```
for step t ∈ {1, 2, ...} do
  if t ≤ k then
    Sample a mini-batch of transitions from D
  else
    Start from s, sample a from π_Q, execute a, observe (s',r) and store (s, a, r, s') in M
    Sample a mini-batch of transitions from M
  end if
  Update θ with gradient in Equation (5.6)
  if t mod T = 0 then
    θ' ← θ
  end if
end for
```

$$\pi^*(a|s) = \exp((Q^*(s,a) - V^*(s))/\alpha)$$

With the entropy augmented reward, one can derive the soft versions of Q-learning [Haarnoja et al., 2017b]. The soft Q-learning gradient is given by

$$\nabla_\theta Q_\theta(s,a)(Q_\theta(s,a) - \hat{Q}(s,a))$$

, where $\hat{Q}(s,a)$ is a bootstrapped Q-value estimate obtained by $R(s,a)+\gamma V_Q(s')$. Here, $R(s,a)$ is the reward received from the environment, V_Q is computed from $Q_\theta(s,a)$ as mentioned above.

5.3 Soft Advantage Learning on Demonstrations and Rewards

We aim to design a reinforcement learning algorithm that, when given a set of demonstrations consisting of transitions (s,a,r,s') and the corresponding MDP definition, can both perform entirely off-policy learning from these demonstrations and continue to improve from on-policy experience when deploying in the environment. This learning protocol is natural and desirable in a range of domains, including robotics and autonomous driving, where demonstration data is available but additional interaction is needed to achieve mastery on a given task. Furthermore, off-policy pre-training from demonstrations can in principle achieve safer and more reliable environment interaction during the first few on-policy episodes.

Although prior off-policy RL algorithms such as Q-learning could in theory be used for this purpose, they tend to work poorly when learning only from off-policy demonstrations, as we show in our experiments. The intuition behind this is that if the Q-function in these methods is trained only on good data, it can hardly understand why the action taken is appropriate: it will assign a high Q-value to correct actions, but will not necessarily assign a low Q-value to other alternative actions. Prior methods based on soft optimality, such as soft Q-learning, also suffer from this problem. However, the framework of soft optimality *does* provide us with a natural mechanism to mitigate this problem if we modify it to learn values and advantages, rather than using a soft Q-learning style objective. This amounts to *normalizing* the Q-function over the actions. Our approach, which we call Soft Advantage Learning (SAL), trains a Q-function by separately supervising state values and advantages, which has a normalizing effect that reduces the Q-values of actions that were not observed in the demonstrations, while still performing Bellman error minimization. In other words, without data to indicate otherwise, SAL will opt to follow the demonstrations.

Soft Advantage Learning

We propose a unified method, Soft Advantage Learning (SAL), that can learn from both from the demonstrations and the environments. SAL parameterizes a Q-function, and expresses the value function and advantage function in terms of the Q-function. Specifically:

$$V_Q(s) = \alpha \log \sum_a \exp(Q(s,a)/\alpha) \tag{5.1}$$

$$A_Q(s,a) = Q(s,a) - V_Q(s) \tag{5.2}$$

The subscripts Q of V_Q and A_Q emphasize that these terms are derived from the Q values. The SAL algorithm minimizes the following objective:

$$\frac{1}{2}(V_Q(s) - \hat{V}(s))^2 + \frac{1}{2}(A_Q(s,a) - \hat{A}(s,a))^2, \tag{5.3}$$

where $\hat{V}(s)$ and $\hat{A}(s,a)$ are the one step backup values obtained from a target Q network [Mnih et al., 2015] by:

$$\hat{V}(s) = \mathop{\mathbb{E}}_{a \sim \pi_Q^{target}} \left[R(s,a) + \gamma V_Q^{target}(s') \right] + \alpha H(\pi_Q^{target}(\cdot|s)) \tag{5.4}$$

$$\hat{A}(s,a) = R(s,a) + \gamma V_Q^{target}(s') - V_Q^{target}(s) \tag{5.5}$$

Here, π_Q is the policy induced by the Q values, as in Section 5.2. Taking the gradient of the objective, we get:

$$\begin{aligned}
&\nabla V_Q(s)(V_Q(s) - \hat{V}(s)) + \nabla A_Q(s,a)(A_Q(s,a) - \hat{A}(s,a)) \\
=&\nabla V_Q(s)(V_Q(s) - \hat{V}(s)) + (\nabla Q(s,a) - \nabla V_Q(s)) \\
&\quad ((Q(s,a) - V_Q(s)) - (\hat{Q}(s,a) - V_Q^{target}(s))) \\
\approx&\nabla V_Q(s)(V_Q(s) - \hat{V}(s)) + (\nabla Q(s,a) - \nabla V_Q(s)) \\
&\quad (Q(s,a) - \hat{Q}(s,a)) \qquad (5.6)
\end{aligned}$$

where we define $\hat{Q}(s,a) = R(s,a) + \gamma V_Q^{target}(s')$ and make an assumption that $V_Q(s) \approx V_Q^{target}(s)$, which is reasonable when the target network is not too far from the current network. Equation (5.6) is the update we use in practice. We also use samples from the demonstration and replay buffer. The full method is summarized in Algorithm 1.

The optimal $Q^*(s,a)$ and the induced V_Q^* and A_Q^* are the minimizers of the proposed objective. The objective contains two terms: the advantage error term and the state value function error term. Q value normalization falls out from the advantage term, which we analyze further in the next section. The state value function error term ensures that the learned Q function satisfies the Bellman equation.

Analysis of the Method

We provide a discussion as well as empirical evidence in this section to explain why SAL has the *normalization effect* on the Q values that makes it well-suited for learning off-policy from demonstrations without any explicit supervised loss. The gradient of the advantage fitting term is approximately

$$(\nabla Q(s,a) - \nabla V_Q(s))(Q(s,a) - \hat{Q}(s,a))$$

Comparing this with the soft Q learning update:

$$\nabla Q(s,a)(Q(s,a) - \hat{Q}(s,a))$$

our update mainly differs in an extra term in the gradient: $-\nabla_\theta V_Q(s)$. This term falls out naturally when we compute the gradient of the advantage error. Intuitively, this term will decrease $V_Q(s)$ when increasing $Q(s,a)$ and vice versa, since $\nabla_\theta Q(s,a)$ and $-\nabla_\theta V_Q(s)$ have different signs. Because $V_Q(s) = \alpha \log \sum_a \exp(Q(s,a)/\alpha)$, decreasing $V_Q(s)$ will prevent $Q(s,a)$ from increasing for the actions that are not in the demonstrations. Thus the aforementioned normalization effect emerges with the extra $\nabla_\theta V_Q(s)$ term.

Figure 5.1 shows a sample of the learned Q values for DQN and SAL. It empirically confirms that our method has the normalization effect, while DQN could assign higher Q values to the actions that are not demonstrated. Moreover, the true Q values are on the scale of 100, since the reward each step is around 1.0 for a well trained agent, and the discount

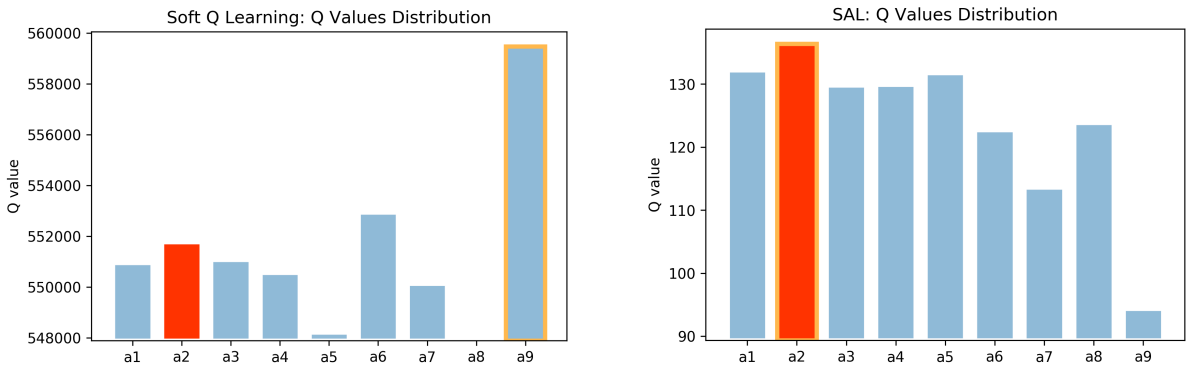


Figure 5.1: Benefits of properly normalized q-values: typical predicted $Q(s_0, \cdot)$ values (y-axis) of DQN and SAL on a state s_0 for actions a_1 through a_9 (x-axis). Action 2 is the demonstrated action (red bar). The actions with maximum Q values would be executed during test time (yellow border). DQN has excessively large wrong predicted Q values, and the demonstrated action does not receive the largest Q value and thus it will execute the wrong action. SAL does not have those issues. Both DQN and SAL are trained on the TORCS game demonstration set, without environment interactions and these two figures show Q values from the actual experiments.

factor is 0.99. SAL has the correct Q values, while DQN predicts excessive large wrong Q values, due to the lack of normalization. This is a typical phenomenon for any state, and statistically our method assigns the largest Q value to the demonstrated action 81% of time, while DQN does so only 8% of the time.

Besides the desirable normalization effect, SAL is also less sensitive to noisy demonstrations. Since SAL is an RL algorithm, it maximizes the reward, and thus poor demonstration with a low reward will be ignored by the learned policy. One could also see this with similar analysis as above. When there is a negative reward in the demonstrations, $Q(s, a)$ tends to decrease and $V_Q(s)$ tends to increase, hence having the normalizing behavior acts in the reverse direction.

Moreover, SAL provides a single and principled approach to learn from both demonstrations and environments. It also avoids the use of imitation learning. Therefore, besides its natural robustness to imperfect demonstrations, SAL does not suffer from any trade-offs due to the divergent objective of supervised learning and reinforcement learning. For example, SAL can improve based upon suboptimal demonstrations when starting to interact with the environment, while imitation learning based method usually gets stuck with the demonstrated suboptimal behavior. See Section 5.5 for details.

5.4 Related Work

Maximum entropy reinforcement learning. Maximum entropy reinforcement learning has been explored in a number of prior works [Todorov, 2008, Toussaint, 2009, Ziebart et al., 2008], including several recent works that extend it into a deep reinforcement learning setting [Nachum et al., 2017, Haarnoja et al., 2017a, Schulman et al., 2017, Haarnoja et al., 2018]. However, most of those works do not deal with the learning from demonstration settings. [Haarnoja et al., 2017a] proposes maximum entropy RL methods to learn from environments, where the objective is minimizing the Bellman error of the soft Q function. [Schulman et al., 2017] points out a connection between Q learning and policy gradient. The Soft Actor-Critic method [Haarnoja et al., 2018] parametrizes $V(s)$, $Q(s,a)$ and $\pi(a|s)$ separately and optimize for Bellman error of the soft Q function, KL divergence between π and policy induced by Q, as well as square error between $V(s)$ and value computed from π and Q.

PCL [Nachum et al., 2017] is the only prior work that studies the learning from demonstration task with the maximum entropy RL framework, where the loss is derived from the sum of Bellman error across multiple steps. We empirically outperforms PCL, which will be shown in the experiment section.

We would like to emphasize that although our SAL method shares the same max entropy RL framework with several other methods [Haarnoja et al., 2018, Haarnoja et al., 2017b, Nachum et al., 2017], the actual algorithm is distinct from them. Soft Q learning [Haarnoja et al., 2017b] and PCL [Nachum et al., 2017] minimize one step and multi-step Bellman error respectively. SAC [Haarnoja et al., 2018] is an actor-critic method under the soft RL framework. However, our SAL method proposes a novel value (V) plus advantage (A) error minimization objective, which is different from all previous methods.

Learning from demonstrations. Most of the prior learning from demonstration efforts [Osa et al., 2018, Schaal, 1997] assume the demonstrations are perfect, *i.e.* the ultimate goal is to copy the behaviors from the demonstrations. Imitation learning is one of such approaches, example applications including [Pomerleau, 1989b, Xu et al., 2016, Bojarski et al., 2016a, Torabi et al., 2018].

Instead of assuming that the demonstrations are perfect, our method allows imperfect demonstrations. Our method *learns* which part of the demonstrations is good and which part is bad, unlike the methods that simply imitate the demonstrated behaviors. We follow the Reinforcement Learning with Expert Demonstrations (RLED) framework [Chemali and Lazaric, 2015, Kim et al., 2013, Piot et al., 2014], where both rewards and actions are available in the demonstrations. The extra reward in the demonstrations allows our method to be aware of poor behaviors in the demonstrations. DQfD [Hester et al., 2017] is a recent method that also uses rewards in the demonstrations. It combines an imitation hinge loss with the Q-learning loss in order to learn from demonstrations and transfer to environments smoothly. Due to the use of the imitation loss, DQfD is more sensitive to noisy demonstrations, as we show in the experiment section.

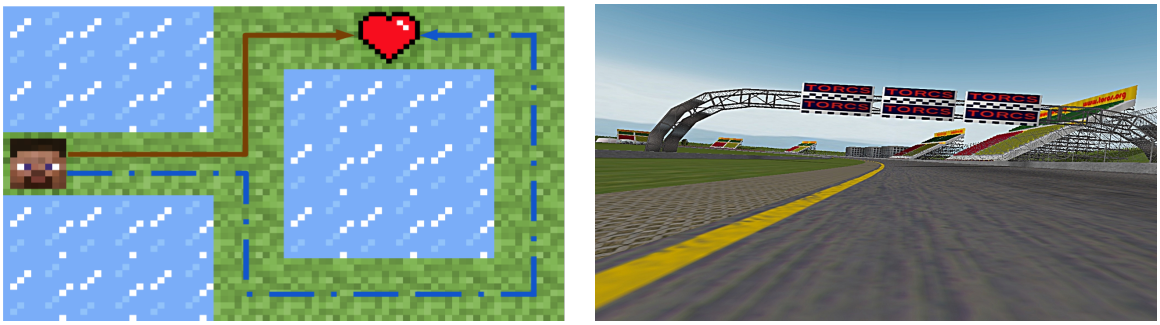


Figure 5.2: Sample frames from the Toy Minecraft and Torcs environments.

Another line of work assumes we have access to the expert oracle, i.e. we can query the optimal action of any state. DAGGER [Ross et al., 2011] is one of the representative algorithms under those assumptions. Recently, algorithms that try to unify DAGGER and RL have been proposed, such as Deeply AggreVaTeD [Sun et al., 2017] which tries to improves the sample complexity when we have a sub-optimal expert. Truncated Horizon Policy Search (THOR) [Sun et al., 2018] more explicitly interpolates between RL and IL and achieves superior performance when the oracle is sub-optimal. Since we only have a set of demonstrations rather than an oracle, those works are orthogonal to our approach.

Off-policy learning. It is tempting to apply various off-policy methods to the problem of learning from demonstration, such as policy gradient variants [Gu et al., 2017, Gu et al., 2016, Degris et al., 2012, Wang et al., 2016], Q-learning [Watkins and Dayan, 1992] and Retrace [Munos et al., 2016]. However, we emphasize that off-policy learning and learning from demonstration are different problems. For most of the off-policy methods, their convergence relies on the assumption of visiting each (s, a) pair infinitely many times. In the learning from demonstration setting, the samples are highly biased and off-policy method can fail to learn anything from the demonstrations, as we explained the Q-learning case in Section 5.3.

5.5 Results

Our experiments address two questions: (1) Can SAL benefit from both demonstrations and rewards? (2) Is SAL robust to ill-behaved demonstrations? We compare our algorithm with DQfD [Hester et al., 2017], which has been shown to learn efficiently from demonstrations and to preserve performance while acting in an environment. Other baseline methods include supervised behavioral cloning method, Q-learning, soft Q-learning, policy gradient with importance sampling weighting, PCL and Q-learning, soft Q-learning without demonstrations. Implementation details and hyperparameters are included in the Supplementary Material.

We evaluate our result in a grid world, toy Minecraft, and in a 3D simulated environments, Torcs as shown in Figure 5.2. See Supplementary Material for environment details.

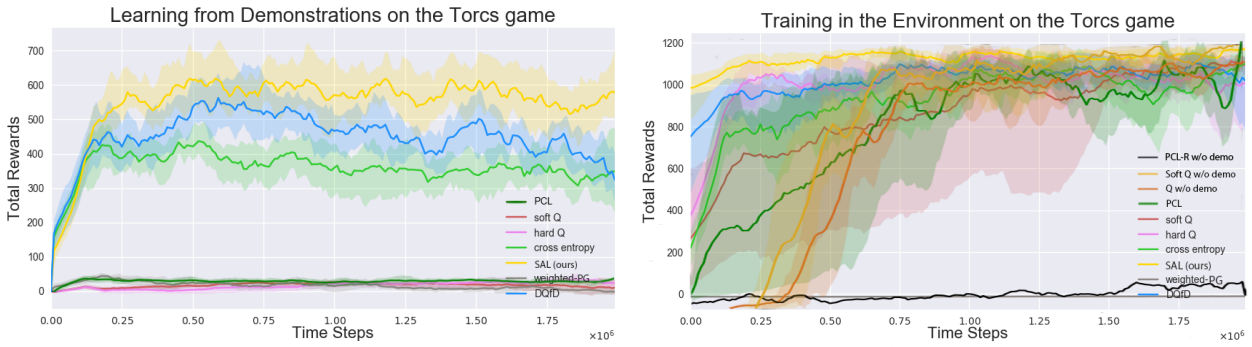


Figure 5.3: Performances on the Torcs game. The x-axis shows the training iterations. The y-axis shows the average total rewards. Solid lines are average values over 10 random seeds. Shaded regions correspond to one standard deviation. The left figure shows the performance for each agent when they only learn from demonstrations, while the right one shows the performance for each agent when they interact with the environments after learning from demonstrations. Our method consistently outperforms other methods in both cases.

Comparisons

We compare our approach with the following methods:

- **DQfD:** the method proposed by [Hester et al., 2017]. For the learning from demonstration phase, DQfD combines a hinge loss with a temporal difference (TD) loss. For the finetuning-in-environment phase, DQfD combines a hinge loss on demonstrations and a TD loss on both the demonstrations and the policy-generated data. To alleviate over-fitting issues, we also include weight decay following the original paper.

- **Q-learning:** the classic DQN method [Mnih et al., 2015]. We first train DQN with the demonstrations in a replay buffer and then finetune in the environment with regular Q-learning. Similar to DQfD, we use a constant exploration ratio of 0.01 in the finetuning phase to preserve the performance obtained from the demonstrations. We also train from scratch a baseline DQN in the environment, without any demonstration.

- **Soft Q-learning:** similar to the Q-learning method, but with an entropy regularized reward. This is the method proposed by [Haarnoja et al., 2017b, Schulman et al., 2017]. We also include the soft Q-learning trained without demonstration, as another baseline.

- **Behavior cloning with Q-learning:** the naive way of combining cross-entropy loss with Q-learning. First we perform behavior cloning with cross-entropy loss on the demonstrations. Then we treat the logit activations prior the softmax layer as an initialization of the Q function and finetune with regular Q-learning in the environment.

- **Soft actor-critic with importance sampling:** An actor-critic method on entropy regularized reward with the importance sampling weighting. The importance weighting

term is used to correct the action distribution mismatch between the demonstration and the current policy.

- **Path Consistency Learning (PCL):** the PCL [Nachum et al., 2017] method that minimizes the soft path consistency loss. The method proposed in the original paper (denoted as *PCL-R*) does not utilize a target network. We find that PCL-R does not work when it is trained from scratch in the visually complex environment. We stabilize it by adding a target network (denoted as *PCL*), similar to [Haarnoja et al., 2017b].

Experiments on Toy Minecraft

To understand the basic properties of our proposed method, we design the toy Minecraft environment. In this experiment, the state is simply the location of the agent. We use a tabular Q function. With those settings, we hope to reveal some differences between our SAL algorithm and algorithms that incorporate supervised loss.

As shown in Figure 5.2, there are only two paths to reach the goal. In terms of the discounted reward, the shorter path is more favorable. To make the problem more interesting, we provide the longer suboptimal path as the demonstrations. We found that in the learning from demonstration phase, both DQfD and SAL have learned the suboptimal path since both methods do not have access to the environment and could not possibly figure out the optimal path. When the two methods finetune their policies in the environment, SAL succeeds in finding the optimal path, while DQfD stucks with the suboptimal one. It is because DQfD has the imitation loss, thus preventing it from deviating from the original solution.

Comparison to Other Methods

We compare our SAL method with other methods on 300k transitions. The demonstrations are collected by a trained Q-learning expert policy. We execute the policy in the environment to collect demonstrations. To avoid deterministic executions of the expert policy, we sample an action randomly with probability 0.01.

To explicitly compare different methods, we show separate figures for performances on the demonstrations and inside the environments. In Fig 5.3, we show that our method performs better than other methods on demonstrations. When we start finetuning, the performance of our method continues to increase and reaches peak performance faster than other methods. DQfD [Hester et al., 2017] has similar behavior to ours but has lower performance. Behavior cloning learns well on demonstrations, but it has a significant performance drop while interacting with environments. All the methods can ultimately learn by interacting with the environment but only our method and DQfD start from a relatively high performance. Without the demonstration data, Q-learning (Q w/o demo) and soft Q-learning (soft-Q w/o demo) suffer from low performance during the initial interactions with the environment. The original PCL-R method (PCL-R w/o demo) fails to learn even when trained from

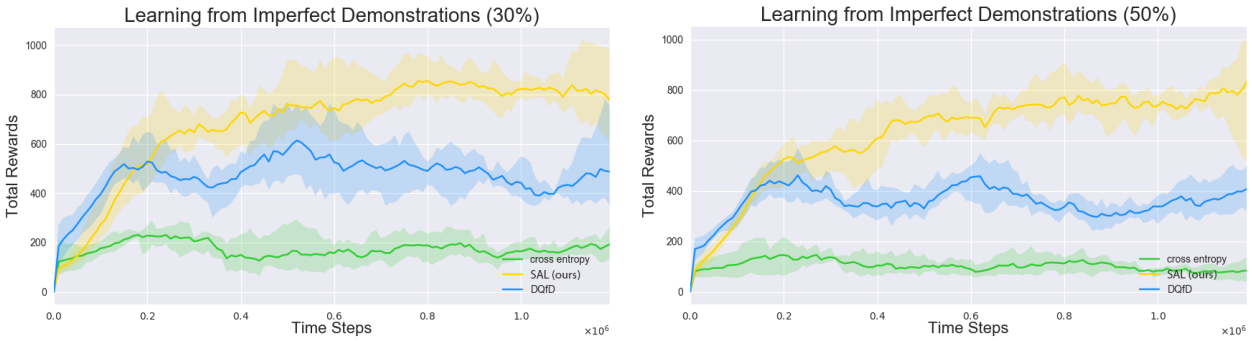


Figure 5.4: Learning from imperfect data when the imperfectness is 30% and 50%. Our SAL method does not clone suboptimal behaviors and thus outperforms DQfD and behavior cloning. The solid lines are mean reward averaged over 10 random seeds; the error bar is one standard deviation. We leave the 80% imperfectness plot in the Supplementary Material.

scratch in the environments. The improved PCL method (PCL) is not able to learn on the demonstrations, but it can learn in the environment.

Effects of Imperfect Demonstrations

In the real world, collected demonstrations might be far from optimal. To study this phenomenon in a principled manner, we collect a few versions of demonstrations with varying degrees of noise. When collecting the demonstrations with the trained Q agent, we corrupt a certain percentage of the demonstrations by choosing non-optimal actions ($argmin_a Q(s,a)$). The data corruption process is conducted while interacting with the environment; therefore, the error will affect the collection of the following steps. We get 3 sets of $\{30\%, 50\%, 80\%\}$ percentage of imperfect data. In the left of Fig. 5.4, we show that our method performs well compared with DQfD and behavior cloning methods. Supervised behavior cloning method is heavily influenced by the imperfect demonstrations. DQfD is also heavily affected, but not as severely as the behavior cloning. SAL is robust because it does not imitate the suboptimal behaviors. The results for 50% and 80% percentage of imperfect data are similar, and they are available in the appendix.

5.6 Discussion

We proposed a Soft Advantage Learning algorithm for reinforcement learning from demonstrations. Our algorithm provides a unified approach for learning from reward and demonstrations, and is robust to potentially suboptimal demonstration data. An agent can be fine-tuned with rewards after training on demonstrations by simply continuing to perform the same algorithm on the on-policy data. Our algorithm preserves and improves the behaviors learned from demonstrations while receiving reward through interaction with an environment.

Chapter 6

Conclusion

As we have mentioned in the open question part (Section 1.4) in the introduction, there are two main categories of challenges in using end-to-end learning for the autonomous driving task. I.e. the **architectural design** and the **training scheme**. On one hand, an end to end naive neural-network-based policy is unlikely to learn everything necessary for an autonomous agent, such as reasoning about decisions logically, obeying the traffic rule strictly and understanding the physics of various road obstacles. On the other hand, the current non-end-to-end system suffers a lot from the coordination problem among the modules, such as propagating the multi-modality and uncertainty in behavior prediction part to the planning part and semantic information loss when abstracting an object by a few attributes. Architectural design is the problem of how to design an autonomous system that can have the benefits of both the pipeline style system and that of an end to end style system. In this thesis, we explored enabling the autonomous system to do logical reasoning. In the future, we still need to extend the system to enable more aforementioned capabilities. The other problem, i.e. training scheme, refers to how the system is tuned. We had an initial attempt to solve such a problem by combining imitation learning with reinforcement learning. However, it is still unclear how to apply this algorithm in a real-world system, including the problem of non-differentiable components, reward design, and safety-aware training, etc. When this challenge is tackled, we probably could rely less on human engineers to change the code every time the system fails.

The goal of studying an autonomous driving system is to use it as a concrete example of the generic sensorimotor problem. On a bigger scope, the generic sensorimotor control problem does have some quite different properties than the autonomous driving application we mentioned here. For example, a generic robot would encounter many more scenarios than a driving agent. People try to engineer the knowledge required by an autonomous driving agent, however, we find it hard to cover all of the knowledge, such as the long tail distribution of object categories. The general robot will encounter many more scenarios and the robot needs to learn from the environment by itself since a human can not possibly teach everything it needs. The other important difference is the representation learning of an autonomous driving agent is largely mitigated by human knowledge. Human knows that the

3D structure of the world and the object-based representation could do a large percent of the job for driving, although a systematic representation of uncertainty and multi-modality is not well designed yet. In the more generic robot tasks, the representation might be completely unknown. For example, when manipulating a towel, it is hard for a human to tell what is the internal representation for an arbitrary shape towel. Being aware of the domain knowledge we used in driving, and trying to solve the bigger sensorimotor problem instead, might help us to solve the driving task better.

Appendix A

Soft Advantage Learning Details

A.1 Environments

Toy Minecraft: The toy Minecraft is a customized grid world environment. As shown in the main text, the agent starts from the left and would like to reach the final goal (marked as a heart). The agent can walk on the green grass and go into the blue water ends the episode. The input to the agent is its current (x, y) location. At each step, the agent can move *Up*, *Down*, *Left* or *Right*. It gets a reward of 1 when reaching the goal, 0 otherwise. For more details, please refer to the OpenAI gym Frozen-Lake environment [Brockman et al., 2016].

Torcs: Torcs is an open-source racing game that has been used widely as an experimental environment for driving. The goal of the agent is to drive as fast as possible on the track while avoiding crashes. We use an oval two-lane racing venue in our experiments. The input to the agent is an 84×84 gray scale image. The agent controls the vehicle at 5Hz, and at each step, it chooses from a set of 9 actions which is a Cartesian product between {left, no-op, right} and {up, no-op, down}. We design a dense driving reward function that encourages the car to follow the lane and to avoid collision with obstacles. [1]

A.2 Effects of Imperfect Demonstrations

See Figure A.1 for more results for imperfect demonstrations when the amount of noise varies.

[1] $reward = (1 - \mathbb{1}_{damage})[(\cos\theta - \sin\theta - lane\_ratio) \times speed] + \mathbb{1}_{damage}[-10]$, where $\mathbb{1}_{damage}$ is an indicator function of whether the vehicle is damaged at the current state. $lane\_ratio$ is the ratio between distance to lane center and lane width. θ is the angle between the vehicle heading direction and the road direction.

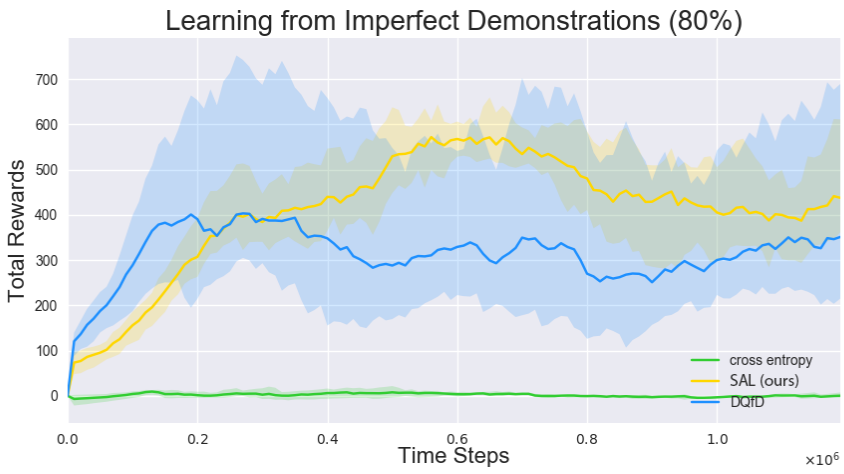


Figure A.1: More results when introducing imperfect demonstrations. The figure shows the case for 80%. Our SAL method is highly robust to noisy demonstrations.

A.3 Effects of Demonstration Amount

In this section, we show comparisons between our method and other methods with different amounts of demonstration data. We use a trained agent to collect three sets of demonstrations which include 10k, 150k, and 300k transitions each. In the experiments, we find that our algorithm performs well when the amount of data is large and is comparable to supervised methods even with a limited amount of data. In Fig. A.2 , we show when there are extremely limited amounts of demonstration data (10k transitions or 30 minutes of experience), our method performs on par with supervised methods. Fig. A.2 also shows the results for 150k and 300k transitions: our method outperforms the baselines by a large margin with 300k transitions. In summary, our method can learn from small amounts of demonstration data and dominates in terms of performance when there is sufficient amount of data.

A.4 Effects of Reward Choice

In the experiments in the main text, we adopt a natural reward: it maximizes speed along the lane, minimizes speed perpendicular to the lane and penalizes when the agent hits anything. However, very informative rewards are not available under many conditions. In this section, we study whether our method is robust to a less informative reward. We change the reward function to be the square of the speed of an agent, irrespective of the speed's direction. This reward encourages the agent to drive fast, however, it is difficult to work with because the agent has to learn by itself that driving off-road or hitting obstacles reduce its

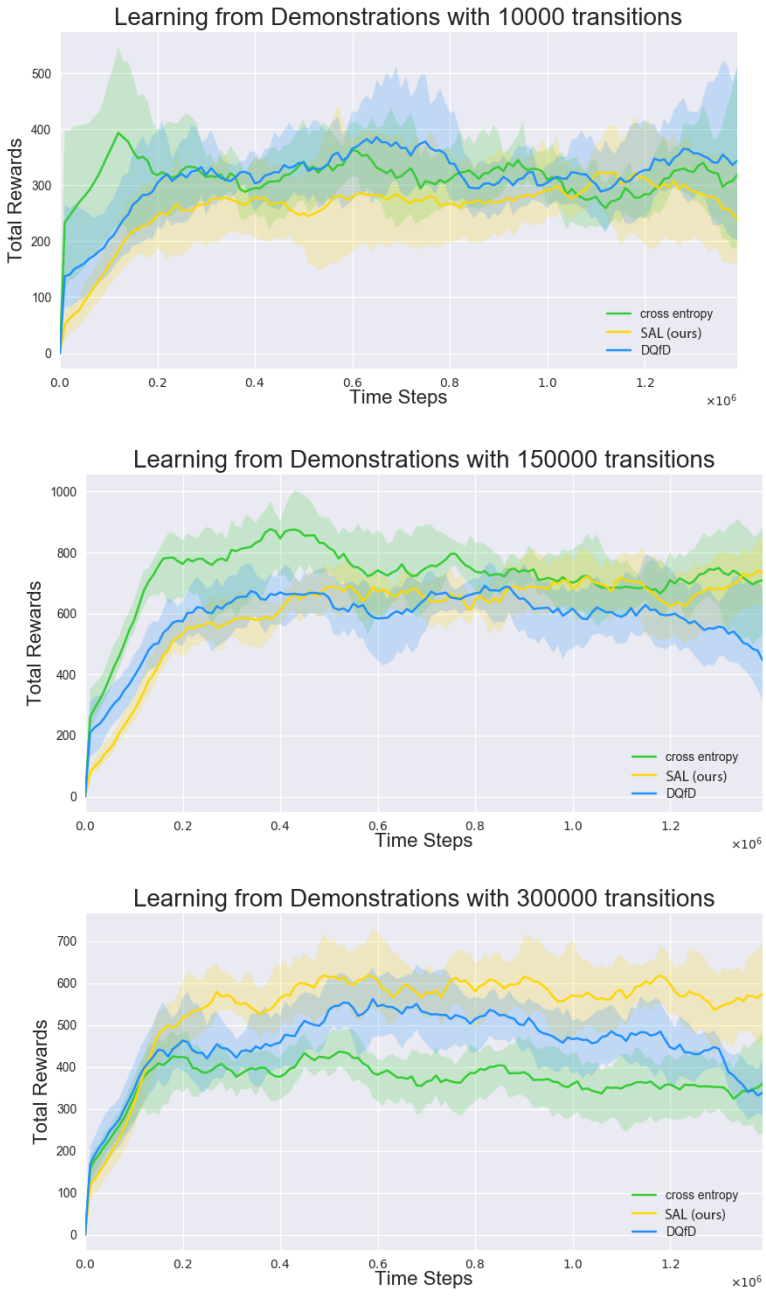


Figure A.2: Learning from different amount of demonstrations (10k, 150k, 300k). Even with only 30 minutes (10k transitions) of experience, our method could still learn a policy that is comparable with supervised learning methods. SAL method achieves superior performance with a large amount of demonstrations.

future speed. It is also hard because $speed^2$ has a large numerical range. Figure A.3 shows that SAL method still performs the best at convergence, while DQfD suffers from severe performance degeneration.

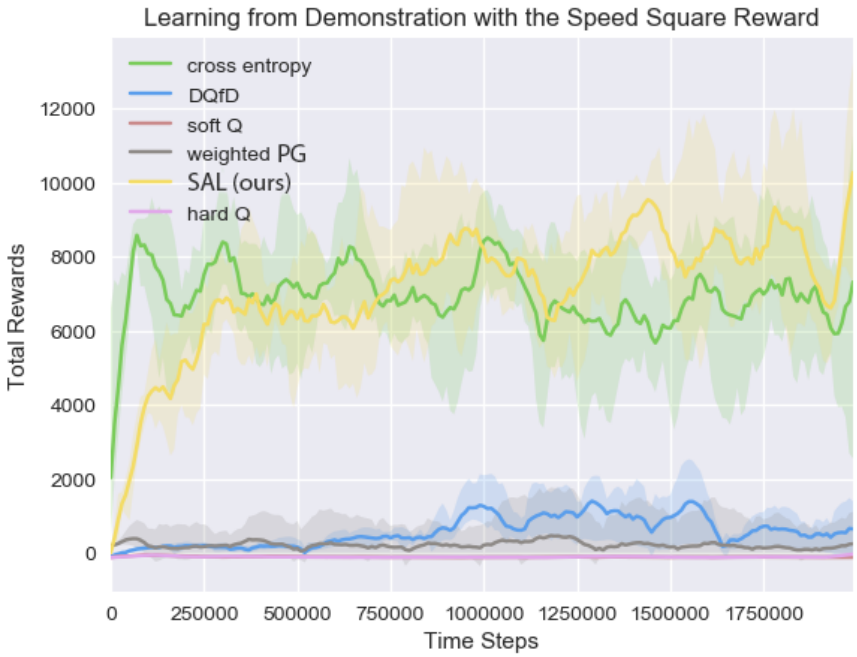


Figure A.3: We compare SAL with other methods when only learning from the demonstrations. We use a different reward: $speed^2$. Our method still performs the best.

A.5 Learning from Human Demonstrations

For many practical problems, such as autonomous driving, we might have a large number of human demonstrations, but no demonstration available from a trained agent at all. In contrast to a scripted agent, humans usually perform actions diversely, both from multiple individuals (e.g. conservative players will slow down before a U-turn; aggressive players will not) and a single individual (e.g. a player may randomly turn or go straight at an intersection). Many learning from demonstration methods do not study this challenging case, such as [Ho and Ermon, 2016]. We study how different methods perform with diverse demonstrations. To collect human demonstrations, we asked 3 non-expert human players to play TORCS for 3 hours each. Human players control the game with the combination of four arrow keys, at 5Hz, the same rate as the trained agent. In total, we collected around 150k transitions. Among them, 4.5k transitions are used as a validation set to monitor the Bellman error. Comparing with data collected from a trained agent, the data is more diverse

and the quality of the demonstrations improves naturally when the players get familiar with the game.

In Fig. A.4, we observe that the behavior cloning method performs much worse than SAL and DQfD. DQfD initially is better than our method but later is surpassed by the SAL method quickly, which might be caused by the supervised hinge loss being harmful when demonstrations are suboptimal. Similar to the policy generated demonstrations case, PCL, hard Q-learning and soft Q-learning do not perform well.

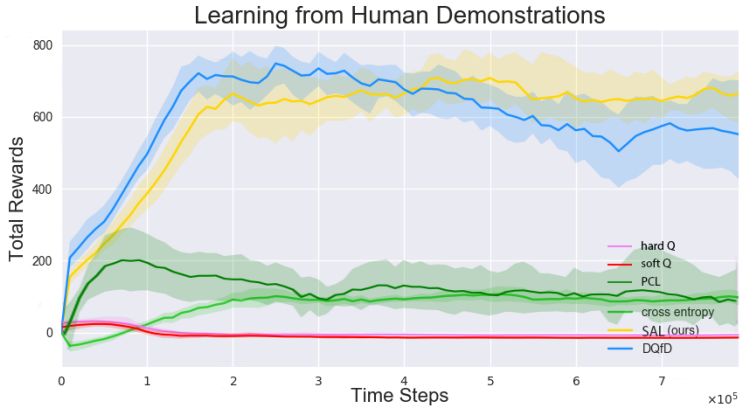


Figure A.4: Performances on the Torcs game with human demonstrations. DQfD performs well in the beginning, but overfits in the end. The behavior cloning method is much worse than SAL and DQfD. Our SAL method performs best at convergence.

A.6 Experiment Details

Network Architecture: We use the same architecture as in [Mnih et al., 2015] to parametrize $Q(s,a)$. With this Q parametrization, we also output and $V_Q(s)$ by $V_Q(s) = \alpha \log \sum_a \exp(Q(s,a)/\alpha)$.

Hyper-parameters: We use a replay buffer with a capacity of 1 million steps and update the target network every 10K steps. Initially, the learning rate is linearly annealed from 1e-4 to 5e-5 for the first 1/10 of the training process and then it is kept as a constant (5e-5). Gradients are clipped at 10 to reduce training variance. The reward discount factor γ is set to 0.99. We concatenate the 4 most recent frames as the input to the neural network. For the methods with an entropy regularizer, we set α to 0.1, following [Schulman et al., 2017]. We truncate the importance sampling weighting factor $\beta = \min\left\{\frac{\pi_Q(a|s)}{\mu(a|s)}, c\right\}$ at 10, i.e., $c = 10$ for the importance weighted soft actor-critic method.

Bibliography

[Adorjan,] Adorjan, M. " opensfm ein kollaboratives structure-from-motion system"; betreuer/in (nen): M. wimmer, m. birsak; institut für computergraphik und algorithmen, 2016; abschlussprüfung: 02.05. 2016.

[Agarwal et al., 2010] Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Reconstructing rome. *Computer*, 43(6):40–47.

[Alcantarilla et al., 2016] Alcantarilla, P. F., Stent, S., Ros, G., Arroyo, R., and Gherardi, R. (2016). Street-view change detection with deconvolutional networks. In *Robotics: Science and Systems*.

[Amini et al., 2019] Amini, A., Rosman, G., Karaman, S., and Rus, D. (2019). Variational End-to-End Navigation and Localization. pages 8958–8964.

[and others Jacobs, Robert A and Jordan, Michael I and Nowlan, Steven J and Hinton, 1991] and others Jacobs, Robert A and Jordan, Michael I and Nowlan, Steven J and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3:79—-87.

[and others LeCun, Yann and Bottou, L{\'e}on and Bengio, Yoshua and Haffner, 1998] and others LeCun, Yann and Bottou, L{\'e}on and Bengio, Yoshua and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278—-2324.

[Andreas et al., 2017] Andreas, J., Klein, D., and Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. *34th International Conference on Machine Learning, ICML 2017*, 1:229–239.

[Andreas et al., 2016] Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Neural module networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December(Figure 1):39–48.

[Badrinarayanan et al., 2015] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*.

[Bai et al., 2016] Bai, M., Luo, W., Kundu, K., and Urtasun, R. (2016). Exploiting semantic information and deep matching for optical flow. In *European Conference on Computer Vision*, pages 154–170.

[Bansal et al., 2018] Bansal, M., Krizhevsky, A., and Ogale, A. (2018). ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. pages 1–20.

[Bao and Savarese, 2011] Bao, S. Y. and Savarese, S. (2011). Semantic structure from motion. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2025–2032. IEEE.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

[Bojarski et al., 2016a] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016a). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

[Bojarski et al., 2016b] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016b). End to End Learning for Self-Driving Cars. pages 1–9.

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[Chemali and Lazaric, 2015] Chemali, J. and Lazaric, A. (2015). Direct policy iteration with demonstrations. In *IJCAI*, pages 3380–3386.

[Chen et al., 2015a] Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015a). Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730.

[Chen et al., 2015b] Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015b). DeepDriving: Learning affordance for direct perception in autonomous driving. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 International Conference on Computer Vision, ICCV 2015(Figure 1):2722–2730.

[Codevilla et al., 2018] Codevilla, F., Miiller, M., Lopez, A., Koltun, V., and Dosovitskiy, A. (2018). End-to-End Driving Via Conditional Imitation Learning. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4693–4700.

[Codevilla et al., 2017] Codevilla, F., Müller, M., Dosovitskiy, A., López, A., and Koltun, V. (2017). End-to-end driving via conditional imitation learning. *CoRR*, abs/1710.02410.

[Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes dataset for semantic urban scene understanding. *arXiv preprint arXiv:1604.01685*.

[Currier,] Currier, P. *Team Victor Tango's Odin: Autonomous Driving Using NI LabVIEW in the DARPA Urban Challenge*.

[Daftry et al., 2016] Daftry, S., Bagnell, J. A., and Hebert, M. (2016). Learning transferable policies for monocular reactive MAV control. In *International Symposium on Experimental Robotics*.

[De Brabandere et al., 2016] De Brabandere, B., Jia, X., Tuytelaars, T., and Van Gool, L. (2016). Dynamic filter networks. *arXiv preprint arXiv:1605.09673*.

[Degris et al., 2012] Degris, T., White, M., and Sutton, R. S. (2012). Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*.

[Donahue et al., 2015] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634.

[Dong et al., 2019] Dong, H., Mao, J., Lin, T., Wang, C., Li, L., and Zhou, D. (2019). Neural Logic Machines.

[Dosovitskiy et al., 2017] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. (CoRL).

[Engel et al., 2016] Engel, J., Koltun, V., and Cremers, D. (2016). Direct sparse odometry. *arXiv preprint arXiv:1607.02565*.

[Engel et al., 2014] Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849.

[Evans and Grefenstette, 2018] Evans, R. and Grefenstette, E. (2018). Learning explanatory rules from noisy data. *IJCAI International Joint Conference on Artificial Intelligence*, 2018-July:5598–5602.

[Finn et al., 2016] Finn, C., Goodfellow, I., and Levine, S. (2016). Unsupervised learning for physical interaction through video prediction. *arXiv preprint arXiv:1605.07157*.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[Fitzgibbon and Zisserman, 2000] Fitzgibbon, A. W. and Zisserman, A. (2000). Multibody structure and motion: 3-d reconstruction of independently moving objects. In *European Conference on Computer Vision*, pages 891–906. Springer.

[Frahm et al., 2010] Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., et al. (2010). Building rome on a cloudless day. In *European Conference on Computer Vision*, pages 368–381. Springer.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

[Geiger et al., 2011] Geiger, A., Ziegler, J., and Stiller, C. (2011). Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

[Gu et al., 2016] Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. (2016). Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*.

[Gu et al., 2017] Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., Schölkopf, B., and Levine, S. (2017). Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. *arXiv preprint arXiv:1706.00387*.

[Gunji et al.,] Gunji, N., Higuchi, T., Yasumoto, K., Muraoka, H., Ushiku, Y., Harada, T., and Kuniyoshi, Y. *Scalable Multiclass Object Categorization with Fisher Based Features*.

[Haarnoja et al., 2017a] Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017a). Reinforcement learning with deep energy-based policies. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1352–1361, International Convention Centre, Sydney, Australia. PMLR.

[Haarnoja et al., 2017b] Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017b). Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*.

[Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[Hartmann et al., 2014] Hartmann, W., Havlena, M., and Schindler, K. (2014). Predicting matchability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9–16.

[Hester et al., 2017] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., et al. (2017). Learning from demonstrations for real world reinforcement learning. *arXiv preprint arXiv:1704.03732*.

[Ho and Ermon, 2016] Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573.

[Hoffman et al., 2016] Hoffman, J., Gupta, S., and Darrell, T. (2016). Learning with side information through modality hallucination. In *In Proc. Computer Vision and Pattern Recognition (CVPR)*.

[Hu et al., 2018] Hu, R., Andreas, J., Darrell, T., and Saenko, K. (2018). Explainable neural computation via stack neural module networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11211 LNCS:55–71.

[Hu et al., 2017a] Hu, R., Andreas, J., Rohrbach, M., Darrell, T., and Saenko, K. (2017a). Learning to Reason: End-to-End Module Networks for Visual Question Answering. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-October(Figure 1):804–813.

[Hu et al., 2019] Hu, R., Rohrbach, A., Darrell, T., and Saenko, K. (2019). Language-Conditioned Graph Networks for Relational Reasoning.

[Hu et al., 2017b] Hu, R., Rohrbach, M., Andreas, J., Darrell, T., and Saenko, K. (2017b). Modeling relationships in referential expressions with compositional modular networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:4418–4427.

[Huval et al., 2015] Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*.

[Jaderberg et al., 2016] Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

[Jordan and Jacobs, 1994] Jordan, M. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6:181—-214.

[Kakade et al., 2003] Kakade, S. M. et al. (2003). *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England.

[Kersting et al., 2000] Kersting, K., De Raedt, L., and Kramer, S. (2000). Interpreting Bayesian logic programs. *Proceedings of the The Seventeenth National Conference on*

Artificial Intelligence (AAAI '2000) workshop on learning statistical models from relational data, (X):138–155.

[Kim et al., 2013] Kim, B., massoud Farahmand, A., Pineau, J., and Precup, D. (2013). Learning from limited demonstrations. In *Advances in Neural Information Processing Systems*, pages 2859–2867.

[Kimmig et al., 2012] Kimmig, A., Bach, S. H., Broecheler, M., Huang, B., and Getoor, L. (2012). A Short Introduction to Probabilistic Soft Logic. *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, (1):1–4.

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *stat*, 1050:10.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kundu et al., 2010] Kundu, A., Krishna, K. M., and Jawahar, C. (2010). Realtime motion segmentation based multibody visual slam. In *Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing*, pages 251–258. ACM.

[Kundu et al., 2011] Kundu, A., Krishna, K. M., and Jawahar, C. (2011). Realtime multibody visual slam with a smoothly moving monocular camera. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2080–2087. IEEE.

[Kundu et al., 2014] Kundu, A., Li, Y., Dellaert, F., Li, F., and Rehg, J. M. (2014). Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision*, pages 703–718. Springer.

[LeCun et al., 2005] LeCun, Y., Muller, U., Ben, J., Cosatto, E., and Flepp, B. (2005). Off-road obstacle avoidance through end-to-end learning. In *NIPS*, pages 739–746.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

[Lotter et al., 2016] Lotter, W., Kreiman, G., and Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Maddern et al., pear] Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (to appear). 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*.

[Mascharka et al., 2018] Mascharka, D., Tran, P., Soklaski, R., and Majumdar, A. (2018). Transparency by Design: Closing the Gap between Performance and Interpretability in Visual Reasoning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4942–4950.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Munos et al., 2016] Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. (2016). Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[Nachum et al., 2017] Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*.

[Nair et al., 2018] Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE.

[Oh et al., 2015] Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in Atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871.

[Osa et al., 2018] Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., Peters, J., et al. (2018). An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179.

[Ozden et al., 2010] Ozden, K. E., Schindler, K., and Van Gool, L. (2010). Multibody structure-from-motion in practice. *IEEE transactions on pattern analysis and machine intelligence*, 32(6):1134–1141.

[Perronnin and Dance, 2007] Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

[Piot et al., 2014] Piot, B., Geist, M., and Pietquin, O. (2014). Boosted bellman residual minimization handling expert demonstrations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 549–564. Springer.

[Pollefeys et al., 1999] Pollefeys, M., Koch, R., and Van Gool, L. (1999). Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25.

[Pomerleau , 1989] Pomerleau , D. (1989). Alvinn: An autonomous land vehicle in a neural network. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 1*. Morgan Kaufmann.

[Pomerleau, 1989a] Pomerleau, D. a. (1989a). Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems 1*, pages 305–313.

[Pomerleau, 1989b] Pomerleau, D. A. (1989b). Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313.

[Rajeswaran et al., 2017] Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. (2017). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

[Richardson and Domingos, 2006] Richardson, M. and Domingos, P. (2006). Markov logic networks. In *Machine Learning*.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

[Ross et al., 2011] Ross, S., Gordon, G. J., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 1, page 6.

[Rosten and Drummond, 2005] Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

[Roussos et al., 2012] Roussos, A., Russell, C., Garg, R., and Agapito, L. (2012). Dense multibody motion estimation and reconstruction from a handheld camera. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 31–40. IEEE.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Rusu et al., 2016] Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. (2016). Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*.

[Salas-Moreno et al., 2013] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359.

[Santana and Hotz, 2016] Santana, E. and Hotz, G. (2016). Learning a driving simulator. *arXiv preprint arXiv:1608.01230*.

[Santoro et al., 2017] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. *Advances in Neural Information Processing Systems*, 2017-December:4968–4977.

[Sauer et al., 2018] Sauer, A., Savinov, N., and Geiger, A. (2018). Conditional Affordance Learning for Driving in Urban Environments. pages 1–15.

[Schaal, 1997] Schaal, S. (1997). Learning from demonstration. In *Advances in neural information processing systems*, pages 1040–1046.

[Schulman et al., 2017] Schulman, J., Abbeel, P., and Chen, X. (2017). Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*.

[Sevilla-Lara et al., 2016] Sevilla-Lara, L., Sun, D., Jampani, V., and Black, M. J. (2016). Optical flow with semantic segmentation and localized layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3889–3898.

[Sharmanska et al., 2013] Sharmanska, V., Quadrianto, N., and Lampert, C. H. (2013). Learning to rank using privileged information. In *International Conference on Computer Vision (ICCV)*, pages 825–832. IEEE.

[Shazeer et al., 2017] Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538.

[Shelhamer et al., 2016] Shelhamer, E., Mahmoudieh, P., Argus, M., and Darrell, T. (2016). Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*.

[Shotton et al., 2008] Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[Snavely et al., 2006] Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM.

[Snavely et al., 2008] Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.

[Song and Chandraker, 2014] Song, S. and Chandraker, M. (2014). Robust scale estimation in real-time monocular sfm for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1566–1573.

[Song and Chandraker, 2015] Song, S. and Chandraker, M. (2015). Joint sfm and detection cues for monocular 3d localization in road scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3734–3742.

[Song et al., 2016] Song, S., Chandraker, M., and Guest, C. C. (2016). High accuracy monocular sfm and scale correction for autonomous driving. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):730–743.

[Strasdat et al.,] Strasdat, H., Montiel, J., and Davison, A. J. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*.

[Sun et al., 2018] Sun, W., Bagnell, J. A., and Boots, B. (2018). Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*.

[Sun et al., 2017] Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. (2017). Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3309–3318. JMLR. org.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

[Thie, 1983] Thie, P. R. (1983). *Markov decision processes*. Comap, Incorporated.

[Todorov, 2008] Todorov, E. (2008). General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4286–4292. IEEE.

[Torabi et al., 2018] Torabi, F., Warnell, G., and Stone, P. (2018). Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*.

[Toussaint, 2009] Toussaint, M. (2009). Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056. ACM.

[Tzeng et al., 2016] Tzeng, E., Devin, C., Hoffman, J., Finn, C., Abbeel, P., Levine, S., Saenko, K., and Darrell, T. (2016). Adapting deep visuomotor representations with weak pairwise constraints. In *Workshop on the Algorithmic Foundations of Robotics*.

[Ummenhofer et al., 2016] Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., and Brox, T. (2016). Demon: Depth and motion network for learning monocular stereo. *arXiv preprint arXiv:1612.02401*.

[Vapnik and Vashist, 2009] Vapnik, V. and Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5):544–557.

[Večerík et al., 2017] Večerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*.

[Vineet et al., 2015] Vineet, V., Miksik, O., Lidegaard, M., Nießner, M., Golodetz, S., Prisacariu, V. A., Kähler, O., Murray, D. W., Izadi, S., Pérez, P., et al. (2015). Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 75–82. IEEE.

[Wang et al., 2016] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2016). Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Wikipedia,] Wikipedia. *Transportation safety in the United States*.

[Wu et al., 2011] Wu, C. et al. (2011). Visualsfm: A visual structure from motion system.

[Wu et al., 2019] Wu, Y.-H., Charoenphakdee, N., Bao, H., Tangkaratt, V., and Sugiyama, M. (2019). Imitation learning from imperfect demonstration. *arXiv preprint arXiv:1901.09387*.

[Xingjian et al., 2015] Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810.

[Xu et al., 2016] Xu, H., Gao, Y., Yu, F., and Darrell, T. (2016). End-to-end learning of driving models from large-scale video datasets. *arXiv preprint arXiv:1612.01079*.

[Xu et al., 2017] Xu, H., Gao, Y., Yu, F., and Darrell, T. (2017). End-to-end learning of driving models from large-scale video datasets. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:3530–3538.

[Yu and Koltun, 2015] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

[Yu et al., 2017] Yu, F., Koltun, V., and Funkhouser, T. (2017). Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*.

[Zhang et al., 2016] Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. *arXiv preprint arXiv:1603.08511*.

[Ziebart, 2010] Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

[Ziebart et al., 2008] Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA.