End-to-end Learning for Robust Decision Making

by

Alexander Andre Amini

S.B., Massachusetts Institute of Technology (2017)
S.M., Massachusetts Institute of Technology (2018)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author . . . .
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by . . . .
Daniela L. Rus
Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

End-to-end Learning for Robust Decision Making

by

Alexander Andre Amini

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Because the physical world is complex, ambiguous, and unpredictable, autonomous agents must be engineered to exhibit a human-level degree of flexibility and generality — far beyond what we are capable of explicitly programming. Such realizations of autonomy are capable of not only reliably solving a particular problem, but also anticipating what could go wrong in order to strategize, adapt, and continuously learn. Achieving such rich and intricate decision making requires rethinking the foundations of intelligence across all stages of the autonomous learning lifecycle.

In this thesis, we develop new learning-based approaches towards dynamic, resilient, and robust decision making of autonomous systems. We advance robust decision making in the wild by addressing critical challenges that arise at all stages, stemming from the data used for training, to the models that learn on this data, to the algorithms to reliably adapt to unexpected events during deployment. We start by exploring how we can computationally design rich, synthetic environments capable of simulating a continuum of hard to collect, out-of-distribution edge-cases, amenable for use during both training and evaluation. Taking this rich data foundation, we then create efficient, expressive learning models together with the algorithms necessary to optimize their representations and overcome imbalances in under-represented and challenging data. Finally, with our trained models, we then turn to the deployment setting where we should still anticipate that our system will be faced with entirely new scenarios that they have never encountered during training. To this end, we develop adaptive and uncertainty-aware algorithms for estimating model uncertainty, and exploiting its presence to realize generalizable decision making, even in the presence of unexpected events.

Thesis Supervisor: Daniela L. Rus
Title: Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science

This doctoral thesis has been examined by a Committee of the Department of Electrical Engineering and Computer Science as follows:

Professor Daniela Rus . . . .
Chairperson, Thesis Supervisor
Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science

Professor Pulkit Agrawal. . . . .
Member, Thesis Committee
Assistant Professor of Electrical Engineering and Computer Science

Professor Sertac Karaman . . . .
Member, Thesis Committee
Associate Professor of Aeronautics and Astronautics

Professor John Leonard. . . . .
Member, Thesis Committee
Samuel C. Collins Professor of Mechanical Engineering

Acknowledgments

This thesis would not be possible without the unwavering support of many people.

First, I would like to extend my deepest gratitude towards my advisor, Professor Daniela Rus. Daniela has given me her unconditional, unwavering, and never-ending support, both professionally and personally. I first approached Daniela as an undergraduate student, when I pitched ideas to her on how we could build a new generation of end-to-end learning systems, which ultimately comprised the foundations of this thesis. Despite being only an undergraduate student with a passion for work at the edge of her lab, she was immediately receptive, supporting me to tackle big problems at an early stage in my research career. Her guidance over the years has been critical to my development and growth as a scientist, researcher, mentor, teacher, and person. Daniela has provided me the means, support, and freedom to ask and tackle challenging questions in my research. In particular, her enthusiasm and commitment to solving impactful problems in science and engineering, and to do so through a creative, interdisciplinary, and collaborative approach, have inspired me to strive for the same goals in my research and to continue to direct my works towards the betterment of our society. I would also like to thank Daniela for meaning so much to me personally. Thank you, Daniela, for pushing me to always keep the gradient.

I also thank the three outstanding members of my thesis committee, Prof. Pulkit Agrawal, Prof. Sertac Karaman, and Prof. John Leonard, for their critical advice and guidance towards the completion of my PhD and this thesis.

My scientific growth and development have been shaped by my research mentors, who have lent their expertise and made my research possible: Prof. Sangeeta Bhatia, Prof. Connor Coley, Dr. Cait Crawford, Prof. Alan Edelman, Mr. Kieran Gallagher, Prof. Igor Gilitschenski, Prof. Radu Grosu, Prof. Song Han, Prof. Thomas Henzinger, Prof. Berthold Horn, Dr. Larry Jackel, Prof. John Leonard, Prof. Jeffrey Lipton, Prof. Aleksander Madry, Prof. Wojciech Matusik, Dr. Urs Muller, Mr. Oliver Murphy, Prof. Liam Paull, Dr. Artem Provodin, Prof. Carlo Ratti, Dr. Stanislav Sobolevsky, Prof. Suvrit Sra, Prof. Russ Tedrake, and Dr. Yuji Yoshimura.

My research would not have been possible without my extraordinary close collaborators, with whom I have worked on several projects and personally become very close to throughout my PhD. To Dr. Ramin Hasani, Dr. Mathias Lechner, Zhijian Liu, Dr. Wilko Schwarting, Dr. Ava Soleimany, Dr. Andy Spielberg, and Tsun-Hsuan (Johnson) Wang, I extend my sincere appreciation and gratitude for a wonderful journey, and look forward to hopefully many more collaborations in the future. Thank you all for making research such a fun and enlivening experience.

I have had the privilege and honor of mentoring and collaborating with many outstanding undergraduate and graduate students, who have pushed me to think critically and to grow as a scientist and mentor. Thank you to Elaheh Ahmadi, Matt Beveridge, Jagdeep Bhatia, Baptiste Bouvier, William Chen, Jordan Docter, Tom Dudzik, Shinjini Ghosh, Alex Gu, Alex Knapp, Alvin Li, Sadhana Lolla, Natasha Maniar, David Matthews, Julia Moseyko, Jacob Phillips, Roshni Sahoo, Suraj Srinivasan, Daniela Velez, Charles Vorbach, Diana Voronin, Selena Zhang, and Catherine Zheng, for enriching my PhD experience so completely.

I sincerely thank the members of the Distributed Robotics Lab at MIT as well as all of my collaborators for providing such an amazing research environment: Murad Abu-Khalaf, Brandon Araki, Yutong Ban, Cenk Baykal, James Bern, Noam Buckman, Veevee Cai, Makram Chahine, Lillian Chin, Joseph DelPreto, Stephanie Gil, Samuel Goldman, Sophie Gruenbacher, Josie Hughes, Robert Katzschmann, Shuguang Li, Xiao Li, Lucas Liebenwein, Felix Naser, Teddy Ort, Alyssa Pierson, Aaron Ray, John Romanishin, Guy Rosman, Tim Seyde, Ryan Truby, Paul Tylkin, Alex Wallar, Wei Xiao, Annan Zhang. Thank you for the inspiring discussions and joyous moments, which have together made for a wonderful PhD experience overall. I want to especially thank Mieke Moran for her constant efforts in making research in the lab possible.

I express my most profound gratitude to my parents, Lisa and Sean, and my siblings, Ami, Ashley, and Andrew, for their unconditional love, support, and care. Lastly, a very special thanks is due to my wife Ava, for her endless love, encouragement, and kindness, and for truly supporting me through every step of this journey. This accomplishment would not have been possible without them.

Dedicated to my family

Ava, Lisa, Sean, Ami, Ashley, and Andrew

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent years, deep neural networks (NNs) have enabled breakthrough performances across a wide range of high-dimensional and high-complexity applications such as object detection (Szegedy et al., 2013a; Redmon et al., 2016; Zhao et al., 2019), facial recognition (Balaban, 2015; Farfade et al., 2015), speech recognition (Graves et al., 2013; Chorowski et al., 2015; Schneider et al., 2019), time-series forecasting (Qiu et al., 2014; Rangapuram et al., 2018), and natural language processing (Otter et al., 2020). Inspired by these successes on modular tasks, there has been increasing interest in building *end-to-end* learning models in areas of autonomous navigation (Bojarski et al., 2016; Smolyanskiy et al., 2017), robot control (Levine et al., 2016), machine translation (Vaswani et al., 2017), and strategic game play (Mnih et al., 2015; Silver et al., 2017). These approaches aim to bypass rigid intermediate definitions of modular components, and thereby push the limits of flexibility within learning-based decision making systems.

Despite these empirical successes of NNs to date, state-of-the-art solutions that are ultimately deployed into the wild remain crippled by extreme brittleness. Most notably, even small, carefully chosen changes to their inputs can have catastrophic effects on the performance of the model and yield arbitrarily incorrect errors in the system (Szegedy et al., 2013b; Goodfellow et al., 2014). These perturbations, referred to as adversarial attacks, are often imperceptible to human vision systems despite having such a profound impact on their artificial counterparts. In more practical settings, it has been shown that real-world realizations of these failure modes also exist through the form of 3D modifications of the environment (Kurakin et al., 2018; Athalye et al., 2018), basic image perturbations (Engstrom et al., 2018), distribution shift (Koh et al., 2021), and even susceptibility to algorithmic bias (Buolamwini and Gebru, 2018).

Unlike artificial NN systems, natural learning systems excel at generalizing learned skills beyond the original data distribution (Hassabis et al., 2017) and reasoning about the underlying causal structure of the task (Peters et al., 2017). At their core, machine learning systems rely on key statistical assumptions about the data they encounter during deployment. Namely, an underlying assumption to machine learning is that the data observed during testing is drawn from the same, independent and identically distributed dataset (*i.i.d.*) as the training data (James et al., 2013). When deploying end-to-end decision making systems within closed-loop feedback settings, this assumption is rarely respected, thereby resulting in unreliable performance and an overall lack of guarantees during deployment.

To this end, prior works have targeted improving the robustness of end-to-end learning based systems for decision making at a variety of different stages of the learning process, ranging from the data that are used for training, to the model representations that are learned, to the algorithms for understanding trust and uncertainty at deployment time. Data augmentation techniques attempt to bridge the train-test gap by applying perturbations to training data in order to increase their coverage space (Perez and Wang, 2017; Shorten and Khoshgoftaar, 2019). Such perturbations are commonly simple transformations of the data (Cubuk et al., 2019) with limited geometric grounding of reality and of the task at hand, thereby limiting their utility for broad generalization capabilities. Modular decomposition of the learning problem in to differentiable components (Karkus et al., 2019) aims to inject structure and expressivity into the model but fails to effectively communicate failure modes from one component to the remainder. Existing approaches to uncertainty estimation (Gal and

Ghahramani, 2015; Lakshminarayanan et al., 2017) aim to quantify these notions of trust during deployment but present inflexible solutions that do not respect compute constraints of real-time deployment settings.

Overall, we currently lack a unified approach to building robust decision making agents for deployable autonomy in the wild. To this end, we are motivated by the vision of advancing the science of engineered autonomy through theoretically-grounded technical advancements for new capabilities and realizations of the theory through robust and rigorous experiments.

1.2 Vision

Autonomous agents are uniquely capable of expanding the reach of humans deep into extreme, safety-critical environments through exploration, adaptation, and decision making. Because the physical world is complex and unpredictable, autonomous agents must learn to exhibit a human-level degree of flexibility and generality – far beyond what we are capable of explicitly programming. Achieving such rich and intricate control requires rethinking the foundations of intelligence across all stages of the learning lifecycle of an autonomous agent: from the data it perceives, to the models underlying its perception and behavior, to its deployment in the wild.

However, today's leading statistical learning algorithms were largely conceived within the confines of restrictive datasets and thus provide narrow and unsustainable solutions to predictive tasks. As a result, these approaches fail to deliver safe, generalizable, and robust decision-making in intricate, dynamic, and uncertain environments. Unlike other forms of learning which have been tailored for passive, static, and supervised datasets, learning capable machines in a closed-loop world presents a unique set of research challenges. As an agent executes actions, its decisions have a direct impact on its future observations, creating a dynamically evolving dataset, and thus requiring learning beyond an isolated static framework.

The objective of this thesis is to contribute towards advancing the science and engineering of autonomy and advance its application to safe decision making for autonomous agents (Figure 1-1). These settings require dynamic and resilient autonomy for general reasoning and robustness: solutions that not only reliably solve the particular problem, but also anticipate what could go wrong in order to strategize, adapt, and continuously learn. Realizing such properties requires new methods for machine learning, planning, and autonomy that fundamentally and directly connect data to decisions in a robust and reliable way.

1.3 Challenges

In order to achieve our vision of robust decision making in the wild, we have to overcome fundamental challenges at the intersection of machine learning and robotics. Below, we discuss key technical challenges that this thesis aims to tackle.

1.3.1 Edge-cases and Rare Events

Modern artificial intelligence (AI) and machine learning (ML) algorithms rely on optimizing "average-case" performance on a fixed dataset. Unfortunately, in the vast majority of dynamic autonomy problems the most safety-critical predictions occur on the *least common* parts of the data, and not the most common scenarios where the majority of the data is collected. Because AI models are optimized based on their average performance, these low-density regions of the training data will be overshadowed by the common and repetitive portions of the data. Importantly, ML models trained on very large datasets can easily get stuck in local minima where they over-optimize performance on the most common datapoints and divert attention away from potentially more challenging under-represented samples. Since these samples are lower density, incurring worse performance on them is not catastrophic in the global optimization view, especially if the model diverts that lost learning capacity to the over-represented examples. The low-density samples of interest are commonly referred to as "edge-cases" as they fall on the *edge* of the data distribution.

It is often difficult and prohibitively costly to obtain high accuracy and performance on the edge-cases on the long tail of a data distribution. This ultimately presents a foundational training and testing problem in ML. Because edge-cases often correspond to rare events, it is challenging to train a robust and performant model with very few examples of these events. When testing the model, diagnosing gaps in the model's knowledge space is again fundamentally flawed without sufficient test data of these edge-cases to evaluate performance on. This results in ML models that can silently fail and be brittle on the most challenging examples, without practitioners easily being able to diagnose the problem before deployment.

In order to overcome such gaps in the data distribution, the most common approach is to collect a larger training dataset, with the hope that as we increase the training dataset size, we encounter more of these edge cases that can be used for learning. The fundamental issue underlying this approach is that by blindly increasing our dataset size, we not only mine more edge cases, but we also further increase the data on the already over-represented regions as well. Furthermore, in many safety critical situations, it is prohibitively dangerous and expensive to collect data on edge cases. For example, to train an autonomous vehicle to drive, it is critical to have training examples of the vehicle recovering from near-crash and collision situations. Collecting such data in reality is not practical as it would directly conflict with the safety of the surrounding environment. Instead, we need to consider approaches to overcome this limitation through (1) principled data generation and simulation; and/or (2) improved model representations capable of generalizing to under-represented edge-cases.

1.3.2 Bridging the Sim-to-Real Gap

Generating new data in under-represented and challenging regions of the training data distribution is a promising approach to overcome the central challenge of mining edge cases and effectively learning from the long tail of the data. This approach requires firstly identifying and parameterizing regions of under-representation where we need more data. This can be achieved through density estimation approaches on our dataset, which shifts the challenge now to generating novel and diverse data in these regions of interest that are sufficiently relevant to learning. Simulation is a common tool to tackle this problem and aims to synthesize data based on a set of parameters of our environment and desired scenario. However, because simulators do not perfectly reflect the physical world where our models will ultimately be deployed, this approach incurs the infamous sim-to-real gap, where our model is not able to transfer knowledge acquired from simulated data back into reality during eventual deployment.

Sim-to-real refers to the idea of transferring skills learned within a simulation engine to the real world. Solving sim-to-real is a grand challenge in robotics and learning because it opens up the potential for infinite data generation in simulation while not incurring the safety costs that we have for data collection in the real world. Equally as important, unlocking simulation as a viable data source allows for extensive validation of our models after training and before deployment, to effectively evaluate them and score their performance on edge-cases before unleashing them into the wild.

Solving sim-to-real for high-dimensional and complex data sources such as perception data is a grand challenge in the community. Traditional simulation engines rely on defining a model of the world (including the physics, other agents, and their environment) that approximates reality. Despite extensive resources devoted to developing high-fidelity simulation engines over many decades, we still have not achieved simulators of the quality needed to bridge the sim-to-real gap. Furthermore, it is also debatable if reaching such a level is possible, as perfectly modeling the real-world through simulation could be viewed as an "AI-complete" task (*i.e.*, solving this task is equivalent to solving the central AI problem of making computers as intelligent as people). In this thesis, we seek to create new approaches to data generation and synthesis which do not rely on manually crafting simulation environments but instead are built directly from real world data. Such an approach has the potential to bypass traditional sim-to-real challenges that plague robot learning problems while still enabling targeted generative collection of novel edge-case data that can be used for training and evaluation.

1.3.3 Interpretability and Decomposability

Even with ideal datasets and evaluation environments, deep NNs still present a lack of provable training guarantees. This has led them to be regarded as "black box" systems that fail to provide explanations of their internal workings. Building systems that not only perform well, but also can be trusted for full scale deployment is a key challenge of modern AI. While model interpretability is typically seen as a possible solution to this problem, many works do not present a clear definition of what it means to be interpretable nor the concrete principles of how their methods enable greater interpretability. Achieving trustworthy AI through interpretability can be characterized into two categories. Firstly, we can develop models that are inherently more interpretable (*i.e.*, by enforcing sparsity, monotonicity, bio-inspiration, or other structure). Secondly, we can build a set of algorithms for probing the underlying mechanism to interpret existing complex models.

There exist many works focused on visual and/or textual interpretations of the decision process. These range from highlighting regions of the input which maximally contribute to the output (Selvaraju et al., 2016; You et al., 2016; Bojarski et al., 2016), to building additional generative models to infer linguistic interpretations of the decision (Harwath and Glass, 2017). While these approaches are empirically powerful, the need for mathematically grounded formulations of system explainability and models that posses these capabilities are still needed.

Unlike interpretability and explainability, which focus on the output decision of the system, decomposability focuses on understanding the underlying internal processes or mechanisms of a given model to answer questions like: "*how does this model work?*". Specifically, if a complex model can be systematically decomposed into interpretable components where all or a subset of all inputs, features, and operations can be understood intuitively, then the AI system, as a whole, becomes more trustworthy as well. Decomposing the system into smaller interpretable pieces allows us to also understand if mismatched objectives (*i.e.*, the system is not optimizing the true objective) or hidden biases (*i.e.*, imbalances during learning) are present.

In this thesis, we aim to formulate a set of foundational principles of model interpretability and develop new computational models that are decomposable into more compact neural processing units to enable greater robustness and trustworthiness.

1.3.4 Generalization to Challenging Scenarios

When building a robust decision making system, we require not only diverse datasets that capture edge-cases and expressive models that are reliable, but also the learning and optimization algorithms to combine these pieces together to compile a trained model capable of generalizing to deployment scenarios. The main goal of generalization is to ensure that the AI systems that we develop are able to perform well even on unseen test data that may be different from their training distribution.

Learning a model capable of generalizing is a central challenge in machine learning as a whole. The challenge is exacerbated by the fact that many state-of-the-art ML models (*i.e.*, deep NNs) are highly overparameterized and, for many tasks, have enough capacity to completely overfit their data even when the labels are replaced with complete random noise (Zhang et al., 2021). Building a generalizable neural-based model ultimately reduces down to injecting the correct priors into the learning model and algorithm in order to help guide and constrain the search space of possible solutions.

Furthermore, challenges brought on through data imbalance and long-tailed training distributions lead to manifestations of failed generalization in the form of algorithmic bias and predictive brittleness. These issues move beyond traditional generalization challenges as they are also rooted in train-test distributional shifts (*i.e.*, when the deployment and testing environment does not match the distribution of the training environment). Building learning based systems that are able to adaptively inspect their own weaknesses and training flaws to adjust and guide their learning procedure is critical to achieve robustness and generalization of learned decision making in practice.

1.3.5 Quantifying Trust in AI

Perhaps at the root of building a robust AI decision making system in the wild lies in the ability to quantify model confidence and, consequently, one's trust in the model. While traditional machine learning algorithms such as linear regression, support vector machines, or even Bayesian classifiers have clear and intuitive definitions of uncertainty, similar algorithms for estimating uncertainty of deep NNs, which have billions or even trillions of parameters, remains a largely unsolved problem.

There have been significant advances in Bayesian deep learning, where NNs are probabilistically reformulated using Bayes rule by placing a prior distribution over every network weight. Model uncertainty is then defined as the likelihood of observing the current set of weights given the training data. In practice, analytically evaluating this posterior is intractable, but works have explored approximations through Monte Carlo sampling (Gal and Ghahramani, 2016b, 2015), variational inference (Graves, 2011), and ensemble methods (Lakshminarayanan et al., 2017). However, a key challenge of uncertainty *calibration* remains. We aim to build the computational methods capable of not only estimating a relative measure of uncertainty, but also generating calibrated absolute measures that can be used to compare across multiple models, neural networks, and tasks.

Achieving a reliable metric of model confidence is only one level of this grand challenge. While this metric allows us as humans to inspect the system post hoc (*i.e.*, after real-time execution), it is unclear how to propagate and build these quantifiable metrics back into the learning algorithm or downstream computational systems, which may not be learning-based or data-driven. This raises key questions in how we can develop new algorithms capable of ingesting these measures of upstream confidence and adaptively changing their own decision-making capabilities *online*, after training and during deployment. This requires reasoning about the task environment and predictive uncertainties in a principled manner to achieve this level of flexibility and adaptation during deployment.

1.4 Contributions

This thesis proposes to address these challenges through theoretically-grounded technical advancements in end-to-end machine learning and realizations of the theory through robust and rigorous experiments on autonomous agents deployed in the wild – ultimately moving closer towards a world with adaptive autonomous agents capable of learning to interact in complex, uncertain, and extreme scenarios, supporting people with cognitive and physical tasks (Figure 1-1). This thesis advances the vision of end-to-end learning to control by:

[Aim 1] ***Data***: Computationally designing rich synthetic environments with a range of realistic attributes amenable for high-fidelity data generation;

[Aim 2] ***Model***: Creating efficient, expressive, and interpretable learning models and algorithms that are robust against unseen perturbations and distributional shifts;

[Aim 3] ***Deploy***: Developing adaptive, uncertainty-aware, and grounded end-to-end control algorithms for robust deployment.

By exploiting the interdependence of these contributions, this thesis advances towards realizing generalizable autonomy and decision making in the physical world (Figure 1-1). In the following subsections, each of the above contributions are outlined in greater detail.

1.4.1 *Data*: High-fidelity synthetic environments for data generation

Collecting the necessary amount of rich and complex data amenable to learning is often both prohibitively costly as well as too dangerous for most practical real-world autonomous systems. The curation of synthetic, yet photorealistic, learning environments presents a promising approach for safely exposing agents to challenging and dangerous edge cases. Through the creation of data-driven simulation engines capable of synthesizing entire virtual worlds directly from real-data, we have enabled closed-loop training and direct real-world transfer of autonomous controllers learned entirely within our simulation platform. We initially focus on autonomous driving and summarize our contributions as follows:

1. ***VISTA***, an open-source photorealistic, scalable, data-driven simulator for synthesizing a continuum of new perceptual inputs locally around an existing dataset of stable human collected driving data;

2. A framework for translating real-world multi-modal data (*i.e.*, 2D RGB images, 3D LiDAR pointclouds, asynchronous event-based streams) to a simulated perception-control API spanning a diversity of compatible environments with varying complexity, lighting, weather, and road types;

3. An end-to-end learning pipeline for training autonomous lane-stable controllers using only visual inputs and sparse reward signals, without explicit supervision using ground truth human control labels; and

4. Experimental validation that agents trained in ***VISTA*** can be deployed directly into the real world with zero-shot policy transfer and achieve more robust recovery compared to previous state-of-the-art learning-based models.

These results are based on the following papers:

- (Amini et al., 2020a): A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150, 2020a

- (Amini et al., 2021): A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus. VISTA 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2021

- (Wang et al., 2021): T.-H. Wang, A. Amini, W. Schwarting, I. Gilitschenski, S. Karaman, and D. Rus. Learning interactive driving policies via data-driven simulation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2021

Full code release for the VISTA data-driven simulation engine and all code associated to these contributions are available here: `vista.csail.mit.edu`.

Results for this contribution are presented in Chapter 3.

1.4.2 *Model*: Robust representations for decision making

In order to reason in the presence of ambiguous decisions, environmental noise, or imperfect sensing, autonomous agents need to learn representations that are both highly expressive and robust. By incorporating greater structure and priors within learned representations, we have created robust models capable of learning directly from human demonstrations, adapting on the fly based on their surroundings, and handling new unseen scenarios that are encountered. These contributions can be summarized as follows:

1. A new class of time-continuous recurrent neural network models which exhibit stable and bounded behavior, yield superior expressivity within the family of neural ordinary differential equations, and give rise to improved performance on time-series control tasks;

2. Demonstration of these neural systems scaling to complex learning tasks in real-world autonomous driving and flight, and improving generalizability, causality, and robustness compared with orders-of-magnitude larger black-box learning systems; and

3. A tunable learning algorithm which estimates underlying latent distributions to improve the quality of representations on under-represented edge-cases by adjusting the respective sampling probabilities of low-density samples.

These results are based on the following papers:

- (Amini et al., 2018b): A. Amini, W. Schwarting, G. Rosman, B. Araki, S. Karaman, and D. Rus. Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 568–575. IEEE, 2018b

- (Amini et al., 2019b): A. Amini, A. P. Soleimany, W. Schwarting, S. N. Bhatia, and D. Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019b

- (Hasani et al., 2020a): R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu. Liquid Time-constant Networks. In *Proceedings of the Seventeenth AAAI Conference on Artificial Intelligence*, 2020a

- (Lechner et al., 2020a): M. Lechner, R. Hasani, A. Amini, T. A. Henzinger, D. Rus, and R. Grosu. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2(10):642–652, 2020a

- (Vorbach et al., 2021): C. Vorbach, R. Hasani, A. Amini, M. Lechner, and D. Rus. Causal navigation by continuous-time neural networks. *Advances in Neural Information Processing Systems*, 34, 2021

Results for this contribution are presented in Chapter 4, 5, and 6.

1.4.3 *Deploy*: Uncertainty-aware autonomy for robust deployment

In reality, autonomous systems will routinely be placed in scenarios where they lack sufficient knowledge or potentially are even uncertain about their existing knowledge. Achieving safe autonomy that effectively interfaces with humans requires agents that are aware of their own uncertainties and can intelligently introspect to resolve ambiguities. This thesis presents new scalable methods for uncertainty estimation in deep neural networks and leverages these methods to design uncertainty-aware controllers capable of self-reflecting on their own uncertainty to guide and improve decisions. These contributions can be summarized as follows:

1. A novel and scalable method for learning epistemic and aleatoric uncertainty on regression problems, without sampling during inference or training with out-of-distribution data;

2. Robustness and calibration evaluation of learned uncertainties on out-of-distribution and adversarially perturbed test input data;

3. An uncertainty-aware fusion algorithm that directly learns prediction uncertainties and adaptively integrates predictions from neighboring frames to achieve robust autonomous control;

4. Deployment of our system on a full-scale autonomous vehicle and demonstration of navigation and improved localization despite noisy inputs as well as robustness in the presence of entirely out-of-distribution sensor failures.

These results are based on the following papers:

- (Amini et al., 2019a): A. Amini, G. Rosman, S. Karaman, and D. Rus. Variational end-to-end navigation and localization. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019a

- (Amini et al., 2020c): A. Amini, W. Schwarting, A. Soleimany, and D. Rus. Deep Evidential Regression. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020c

- (Liu et al., 2021): Z. Liu, A. Amini, S. Zhu, S. Karaman, S. Han, and D. Rus. Efficient and robust lidar-based end-to-end navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021

Results for this contribution are presented in Chapter 7 and 8.

1.5 Thesis Outline

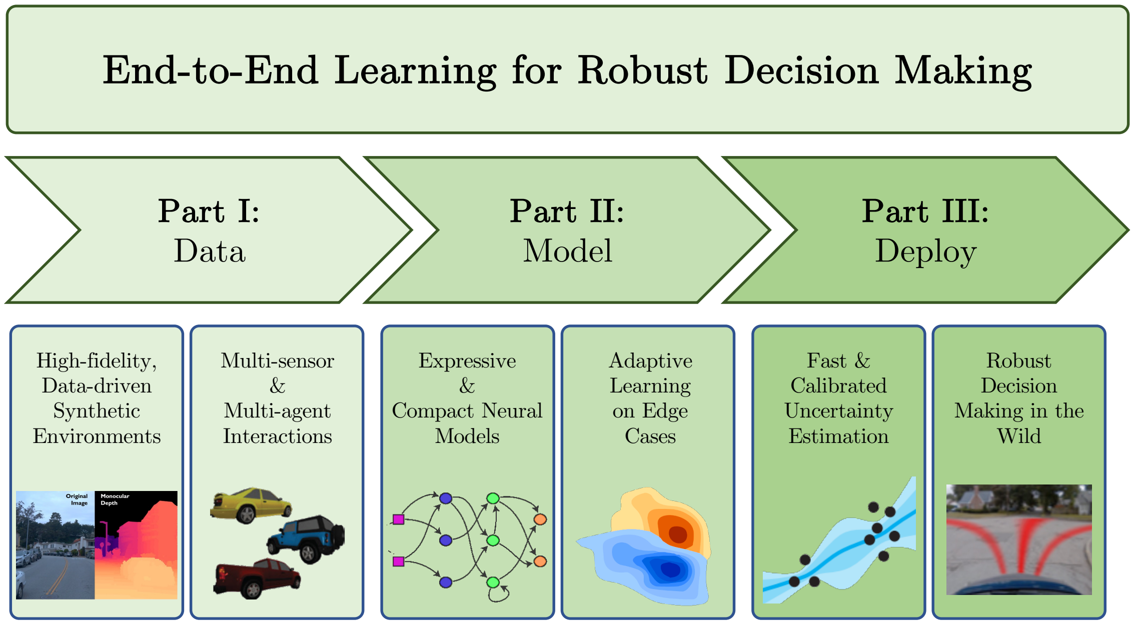

End-to-End Learning for Robust Decision Making
Part I: Data
Part II: Model
Part III: Deploy
High-fidelity, Data-driven Synthetic Environments
Original Image
Monocular Depth
Multi-sensor & Multi-agent Interactions
Expressive & Compact Neural Models
Adaptive Learning on Edge Cases
Fast & Calibrated Uncertainty Estimation
Robust Decision Making in the Wild

Figure 1-1: **Thesis overview.** An overview of the thesis research, which is organized around three key components of the autonomous learning lifecycle.

The ultimate aim of this thesis is to contribute towards advancing the science and engineering of learned autonomy and advance its application to safe, robust decision making for autonomous agents. This thesis is structured according to the outline shown in Figure 1-1. We develop solutions to key challenges at each stage of the autonomous learning and decision making lifecycle, ranging from the data that is used for training, to the models that are learned, to algorithms that ensure robust deployment.

In Part I, we develop new data-driven approaches for large-scale synthesis of challenging edge-cases which are infeasible to safely collect prior to training. Chapter 3 introduces a novel data-driven simulation engine that can construct entire virtual worlds for mining such edge-cases and building more comprehensive training dataset that bridge the gap to reality.

In Part II, we focus on designing more expressive models and learning algorithms capable of ingesting these rich datasets and distilling them into actionable representations. In Chapter 4, we focus on the development of a class of highly-expressive continuous-time neural networks. In Chapter 5, we explore their ability to scale to high-complexity tasks while maintaining compactness. Given this data and model framework, we then transition in Chapter 6 to developing adaptive learning algorithms capable of overcoming persistent imbalances in the representational capacity and achieve algorithmic debiasing.

In Part III, we finally turn to developing algorithms for our trained models as they enter the deployment phase, where they need to account for sudden and unexpected events on which they may not have been explicitly trained to handle. In Chapter 7, we design a single-shot uncertainty estimation technique for quantifying the uncertainty of data-driven models in order to efficiently and reliably identify out-of-distribution events and untrustworthy predictions. In Chapter 8, we demonstrate how these learned confidences can be integrated back into the model's decision making process in order to build uncertainty-awareness and ultimately overcome these types of unexpected events that may occur during deployment.

Finally, in Chapter 9, we summarize the contributions of this thesis, provide perspectives on remaining limitations as well as lessons learned, and discuss potential future directions to overcome these limitations.

Chapter 2

Related Work

2.1 Learning Autonomous Control Policies

2.1.1 Policy Learning

Traditional planning and control approaches for navigation (Alcalá et al., 2020; Carrau et al., 2016; Galceran et al., 2017; Liniger and Lygeros, 2019; Schwarting et al., 2018) typically do not make use of high dimensional inputs and use game-theoretic approaches for interaction modelling (Schwarting et al., 2019b; Williams et al., 2017; Liniger and Lygeros, 2020; Schwarting et al., 2019a). End-to-end navigation approaches can learn driving policies from image inputs; however, in the context of driving using real-world data, past works are largely restricted to imitation learning (IL) (Bojarski et al., 2016; Xu et al., 2017; Amini et al., 2019a; Lechner et al., 2020a; Hawke et al., 2020).

Imitation learning describes the task of learning an observation-action mapping from human demonstrations (Schaal, 1999). This objective can either be achieved via behavior cloning, which directly learns from observation-action pairs, or indirectly via inverse reinforcement learning (Ng et al., 2000) which first constructs a reward function from an optimal policy. The most dominant behavior cloning paradigm is based on the DAgger framework (Ross et al., 2011), which iterates over the steps of expert data collection, supervised learning, and cloned policy evaluation. State-aware imitation learning (Schroecker and Isbell, 2017) further adds a secondary objective to the learning task to bias the policy towards states where more training data is available. Recently, imitation learning methods have been adapted to domains where the environment dynamics of the expert and learned policy mismatch (Desai et al., 2020).

More recent imitation learning one-shot methods pre-train policies via meta-learning to adapt to a task, such that task-specific behavior can be cloned with as little as a single demonstration (Duan et al., 2017; Yu et al., 2018). Alternatively, generative adversarial imitation learning phrases the behavior cloning problem as a min-max optimization problem between a generator policy and discriminator classifier (Ho and Ermon, 2016). (Baram et al., 2017) extended the method by making the human expert policy end-to-end differentiable. The method has been further adapted to imperfect (Wu et al., 2019) and incomplete demonstrations (Sun and Ma, 2019).

2.1.2 Mapless and Multimodal Navigation

Map-based navigation of autonomous vehicles relies heavily on pre-collected high-definition (HD) maps using either LiDAR (Burgard et al., 2008; Levinson and Thrun, 2010) and/or vision (Wolcott and Eustice, 2014) to precisely localize the robot (Leonard and Durrant-Whyte, 1991; Montemerlo et al., 2002). These HD maps are expensive to create, large to maintain, and difficult to process efficiently. Recent works on "map-lite" approaches (Ort et al., 2018, 2019) require only sparse topometric maps, which are orders of magnitude smaller and publicly available (Haklay and Weber, 2008). However, these approaches are largely rule-based and do not leverage modern advances in end-to-end learning of control representations. Our work builds on the advances of end-to-end learning of reactionary control (Bojarski et al., 2016; Xu et al., 2017) and navigation (Amini et al., 2019a; Codevilla et al., 2018; Hawke et al., 2020), and furthermore extends beyond 2D sensing. Recent works have investigated multimodal fusion of vision and depth to improve control (Xiao et al., 2020b; Patel et al., 2017; Bohez et al., 2017). However, these approaches project 3D data onto 2D depth maps to leverage 2D convolutions, thereby losing significant geometric information.

2.2 Simulation and Learning in Robotics

Learning in simulation allows for greater algorithmic flexibility ranging from IL (Rhinehart et al., 2018; Xiao et al., 2020a; Codevilla et al., 2018), to reinforcement learning (RL) (Pan et al., 2017; Dosovitskiy et al., 2017; Amini et al., 2020a; Wang et al., 2021; Fuchs et al., 2021), and guided policy learning (GPL) (Levine and Koltun, 2013; Chen et al., 2020).

2.2.1 Model-based Simulation

The use of simulation for learning and robotics has exploded in recent years. Model-driven simulators rely on predefined models of scenery and underlying physics (Coumans and Bai, 2016–2021; Tassa et al., 2020; Todorov et al., 2012a; Tedrake and the Drake Development Team, 2019; Wymann et al., 2000). Most simulation engines for reinforcement learning and robotics focus on modelling the underlying physics (Coumans and Bai, 2016–2021; Tassa et al., 2020; Todorov et al., 2012a). Recently, simulators based on video-game engines (*e.g. CARLA* (Dosovitskiy et al., 2017), *AirSim* (Shah et al., 2017), *TDW* (Gan et al., 2020), or *FlightGoggles* (Guerra et al., 2019)) also offer a high-fidelity stream of visual data mainly focusing on navigation tasks.

However, training agents with these model-based simulation engines without sacrificing real-world performance and robustness is a long-standing goal in many areas of robotics (Andrychowicz et al., 2018; Mahler et al., 2019; Sadeghi and Levine, 2016; Bewley et al., 2018). Several works have demonstrated transferable policy learning using domain randomization (Tobin et al., 2017) or stochastic augmentation techniques (Bruce et al., 2018) on smaller mobile robots.

However, synthetically generated images are typically still insufficient to enable zero shot transfer of learned policies and require techniques such as domain randomization (Loquercio et al., 2020; Tobin et al., 2017) to actually achieve real-world testing deployment. Style transformation, such as adding realistic textures to synthetic images with deep generative models, has been used to facilitate transfer of learned policies from model-based simulation engines into the real world (Pan et al., 2017; Bewley et al., 2018). While these approaches can successfully transfer low-level details such as textures or sensory noise, they are unable to capture and transfer the critical higher-level semantic complexities (such as vehicle or pedestrian behaviors) required to train robust autonomous controllers.

2.2.2 Data-driven Simulation

A philosophically different approach for achieving the goal of visually and physically realistic simulators is using real-world data for building the simulator. This is inspired by elaborate data augmentation (Abu Alhaija et al., 2018), weakly-supervised learning (Remez et al., 2018), and generative video manipulation (Menapace et al., 2021) techniques. Typically this sacrifices some of the environment editing abilities in favor of maintaining photorealism. Improving learning for autonomous driving (Li et al., 2019) is one of the main applications of these ideas focusing on individual sensors such as camera (Chen et al., 2021) or LiDAR (Manivasagam et al., 2020; Wang et al., 2020). Simulators such as those in *Gibson* (Xia et al., 2018), *Habitat* (Savva et al., 2019), or *RoboTHOR* (Deitke et al., 2020) offer reconstructed real-world environments for embodied AI research. The work presented in Chapter 3 follows the philosophy of data-driven simulation to develop a modular, scalable data generation engine that produces photorealistic and dynamic synthetic environments ultimately enabling zero-shot policy transfer to real-world platforms.

2.2.3 Cross-Sensor Transfer

Numerous works consider augmenting sensing modalities and simulating different modalities via, often learned, sensor fusion. *Monocular Depth Prediction* trains a neural network to act as a depth sensor using monocular image data (Godard et al., 2017; Fu et al., 2018). Similarly, *Depth Completion* combines cameras with sparse depth from LiDAR to simulate a dense depth sensor (Xu et al., 2019; Zhao et al., 2021).

In event-based vision, recent work combines the use of classical and event-based cameras for simulating higher frame-rate cameras (Tulyakov et al., 2021) and depth predictions (Hidalgo-Carrió et al., 2020; Gehrig et al., 2021), or focuses on translating between these two modalities (Rebecq et al., 2018, 2019; Gehrig et al., 2020a; Paredes-Valles and de Croon, 2021). ***VISTA*** provides unified simulation framework that jointly simulates a diverse set of sensors while supporting novel view synthesis for each sensor modality.

2.3 Time-Continuous Neural Models

2.3.1 Time-Continuous Models

Time-continuous networks have become unprecedentedly popular. This is due to the manifestation of several benefits such as adaptive computations, better continuous time-series modeling, memory, and parameter efficiency (Chen et al., 2018a). A large number of alternative approaches have tried to improve and stabilize the adjoint method (Gholami et al., 2019), to use neural ODEs in specific contexts (Rubanova et al., 2019; Lechner et al., 2019), and to characterize them better (Dupont et al., 2019; Durkan et al., 2019; Jia and Benson, 2019; Hanshu et al., 2020; Holl et al., 2020; Quaglino et al., 2020). In this thesis, we build on this line of research with expressive neural ODEs and propose a new ODE model to improve the expressivity and performance of time-continuous models.

2.3.2 Measures of Expressivity

Many works have tried to address why deeper networks and particular architectures perform well in particular settings in order to assess the boundary between the approximation capability of shallow versus deep networks. In this context, previous works proposed counting the linear regions of NNs as a measure of expressivity (Montufar et al., 2014; Pascanu et al., 2013b), showed that there exists a class of radial functions that smaller networks fail to produce (Eldan and Shamir, 2016), and studied the exponential expressivity of NNs by transient chaos (Poole et al., 2016). These methods are compelling; however, they are bound to particular weight configurations of a given network in order to lower-bound expressivity (Serra et al., 2017; Gabrié et al., 2018; Hanin and Rolnick, 2018, 2019; Lee et al., 2019a). The work in (Raghu et al., 2017) introduced an interrelated concept which quantifies expressivity by trajectory length.

2.3.3 Causal Learning

The dominant approach toward learning causal models are graphical methods (Russell and Norvig, 2002; Ruggeri et al., 2007), which try to model cause-effect relationships as a directed graph (Pearl, 2009; Pearl et al., 2009). Bayesian networks further combine graphical models with Bayesian methods (Ruggeri et al., 2007; Kemp et al., 2010) to decompose the learning problem into a set of Bayesian inference sub-problems. (Weichwald et al., 2020) showed the effectiveness of such an approach for learning causal structures in nonlinear time-series via a set of linear models. Continuous-time Bayesian networks further adapted the idea to modeling cause-effect relationships in time-continuous processes (Nodelman et al., 2002, 2003; Gopalratnam et al., 2005; Nodelman, 2007). A different approach for causal modeling of time-continuous processes is to learn ODEs, which under certain conditions imply a structured causal model (Rubenstein et al., 2016). In the work presented in Chapter 4, we describe a class of continuous models that has the ability to account for interventions and therefore captures the causal structures from data.

2.4 Underrepresentation and Bias

Learning in the presence of data imbalance (*e.g.*, class or feature imbalance; underrepresented edge cases) is one of the key challenges underlying all machine learning systems. Interventions that seek to address this challenge and thereby improve fairness into ML pipelines generally fall into one of three categories: those that use data pre-processing before training, in-processing during training, and post-processing after training (Friedler et al., 2018).

2.4.1 Resampling for Class Imbalance

Resampling approaches have largely focused on addressing class imbalances (Chawla et al., 2002; More, 2016; Zhou and Liu, 2006), as opposed to biases within individual classes. For example, over-sampling the minority class with a nearest-neighbors based approach (Chawla et al., 2002) and duplicating instances of the minority class as in (Lu et al., 1998) have been used as pre-processing steps for mitigating class imbalance, yet are not capable of running adaptively during training itself. Further, applying these approaches to debiasing variabilities *within* a class would require *a priori* knowledge of the latent structure to the data, which necessitates manual annotation of the desired features. Instead we seek to debias variability within a class automatically during training with a model that learns the latent structure from scratch in an unsupervised manner. We present this approach in Chapter 6.

2.4.2 Generating Debiased Data

Recent approaches have utilized generative models (Sattigeri et al., 2018) and data transformations (Calmon et al., 2017) to generate training data that is more 'fair' than the original dataset. For example, (Sattigeri et al., 2018) used a generative adversarial network (GAN) to output a reconstructed dataset similar to the input but more fair with respect to certain attributes. Preprocessing data transformations that mitigate discrimination, as in (Calmon et al., 2017), have also been proposed, yet such methods are not learned adaptively during training nor do they provide realistic training examples. In contrast to these works, our approach (Chapter 6 does not rely on artificially generated data, but rather uses a resampled, more representative subset of the original dataset for debiasing.

2.4.3 Clustering to Identify Bias

Supervised learning approaches have also been used to identify underrepresented regions and characterize biases in imbalanced data sets. Specifically, k-means clustering has been employed to identify clusters in the input data prior to training and to inform resampling the training data into a smaller set of representative examples (Nguyen et al., 2008). However, this method does not extend to high dimensional data like images or to cases where there is no notion of a data 'cluster', and relies on significant pre-processing. We seek to overcome these limitations by learning the latent structure using an unsupervised variational approach.

2.5 Uncertainty in Deep Learning

We build on a large history of epistemic uncertainty estimation (Kendall and Gal, 2017; Papadopoulos and Haralambous, 2011; Osband et al., 2016; Hafner et al., 2020; Lakshminarayanan et al., 2017; Gal and Ghahramani, 2016a) and modelling probability distributions using NNs (Nix and Weigend, 1994; Bishop, 1994; Gilitschenski et al., 2019; Kingma et al., 2015). Studies across domains have demonstrated the importance of focusing explicitly on epistemic uncertainty (Soleimany et al., 2021); here, we contextualize our contributions in new methods for estimating epistemic uncertainty.

2.5.1 Bayesian Deep Learning

In Bayesian deep learning, priors are placed over network weights, and distributions over network weights are estimated using variational inference (Kingma et al., 2015), due to the fact that exact computation of the associated posterior is intractable. As a result, approximations via dropout (Gal and Ghahramani, 2016a; Molchanov et al., 2017; Gal et al., 2017; Amini et al., 2018c), ensembling (Lakshminarayanan et al., 2017; Pearce et al., 2018), or other approaches (Blundell et al., 2015; Hernández-Lobato and Adams, 2015) have been derived. However, all these methods rely on expensive sampling to estimate predictive variance, limiting their utility for iterative active learning procedures, scans of very large sensory datasets, and in simulation. In contrast, training a deterministic NN to place uncertainty priors directly over the predictive distribution itself would require only a single forward pass to estimate uncertainty.

2.5.2 Sampling-based Approaches

Sampling-based approaches, such as model ensembling (Lakshminarayanan et al., 2017) and dropout sampling (Gal and Ghahramani, 2016a), are broadly accepted as state of the art for epistemic uncertainty quantification in NNs. These methods are model agnostic, easy to implement, and generalizable to a range of settings and model types. Briefly, these sampling-based approaches yield *estimates* of epistemic uncertainty by generating a set of predictions (e.g., across models in an ensemble), and then considering the variance across those predictions to estimate model uncertainty (Lakshminarayanan et al., 2017; Gal and Ghahramani, 2016a). However, these approaches only generate approximations to the underlying uncertainty functions via stochastic sampling, incurring computational costs and runtimes that are routinely an order of magnitude higher than those of single models. As we demonstrate in Chapter 7, evidential learning can address these limitations in their ability to directly learn grounded representations of epistemic uncertainty without the need for sampling (Sensoy et al., 2018; Malinin and Gales, 2018).

2.5.3 Prior Networks and Evidential Models

While neural networks have been trained to output probabilities, for example with Softmax goodfellow2016softmax for classification or Gaussian distributions (MVE) (Nix and Weigend, 1994) for regression, these approaches estimate the probability of an output but neglect the model's uncertainty (i.e., the epistemic uncertainty) associated with that output. Existing Bayesian and evidential learning methods attempt to tackle the question of epistemic uncertainty estimation in NNs. For example, a large focus within Bayesian inference is on placing prior distributions over hierarchical models to estimate uncertainty (Gelman et al., 2006, 2008). Our methodology, presented in Chapter 7, relates to evidential deep learning for classification (Sensoy et al., 2018) and prior networks (Malinin and Gales, 2018, 2019), which place Dirichlet priors over discrete classification predictions. However, these works either rely on regularizing divergence to a fixed, well-defined prior (Sensoy et al., 2018; Tsiligkaridis, 2019), require OOD training data (Malinin and Gales, 2018; Malinin, 2019; Chen et al., 2018b; Hafner et al., 2020), or can only estimate aleatoric uncertainty by performing density estimation (Gast and Roth, 2018; Gurevich and Stuke, 2020). We tackle these limitations with a particular focus on continuous regression learning tasks where this divergence regularizer is not well-defined.

Part I

Data

Chapter 3

High-fidelity Synthetic Environments for Data Generation

3.1 Introduction

End-to-end (i.e., perception-to-control) trained neural networks for autonomous control have shown great promise, for example in lane stable driving for autonomous vehicles (Pomerleau, 1989; Bojarski et al., 2016; Amini et al., 2019a). However, these methods require vast amounts of training data that are time consuming and expensive to collect, and are furthermore limited in their ability to learn robust models at scale. In autonomous driving, learned end-to-end control policies and modular perception components in a driving pipeline require capturing training data from all necessary edge cases, such as recovery from off-orientation positions or even near collisions. This is not only prohibitively expensive but also potentially dangerous (Kendall et al., 2018). Training and evaluating robotic controllers in simulation (Dosovitskiy et al., 2017; Tedrake and the Drake Development Team, 2019; Shah et al., 2018) has emerged as a potential solution to the need for more data and increased robustness to novel situations, while also avoiding the time, cost, and safety issues of current methods for real world data collection. However, transferring policies learned in simulation into the real world still remains an open research challenge.

This section describes an end-to-end simulation and training engine capable of

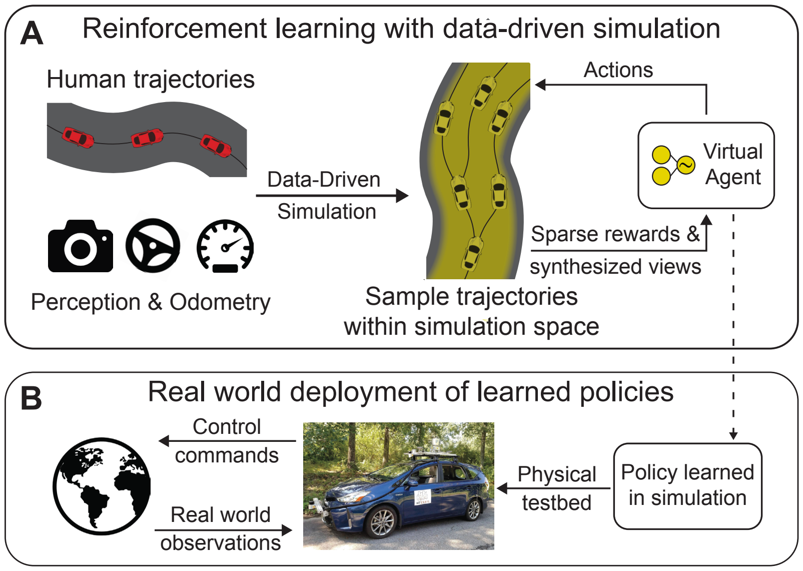

A
Reinforcement learning with data-driven simulation
Human trajectories
Data-Driven Simulation
Actions
Virtual Agent
Sparse rewards & synthesized views
Perception & Odometry
Sample trajectories within simulation space
B
Real world deployment of learned policies
Control commands
Real world observations
Physical testbed
Policy learned in simulation

Figure 3-1: **Training and deployment of policies from data-driven simulation.** From a single human collected trajectory our data-driven simulator (***VISTA***) synthesizes a space of new possible trajectories for learning virtual agent control policies (A). Preserving photorealism of the real world allows the virtual agent to move beyond imitation learning and instead explore the space using methods like reinforcement learning with only sparse rewards and guided policy learning. Learned policies not only transfer directly to the real world (B), but also outperform state-of-the-art end-to-end methods trained using imitation learning.

training real-world autonomous lane-stable controllers agents entirely in simulation, without any prior knowledge of human driving or post-training fine-tuning. We demonstrate trained models can then be deployed directly in the real world, on roads and environments not encountered in training. Our engine, termed ***VISTA***: *Virtual Image Synthesis and Transformation for Autonomy*, synthesizes a continuum of driving trajectories that are photorealistic and semantically faithful to their respective real world driving conditions (Figure 3-1), from a small dataset of human collected driving trajectories. ***VISTA*** allows a virtual agent to not only observe a stream of sensory data from stable driving (*i.e.*, human collected driving data), but also from a simulated band of new observations from off-orientations on the road. Given visual observations of the environment (*e.g.*, camera images), our system learns a lane-stable control policy over a wide variety of different road and environment types, as opposed to current end-to-end systems (Bojarski et al., 2016; Amini et al., 2019a;

Codevilla et al., 2018; Bewley et al., 2018) which only imitate human behavior. This is a major advancement as there does not currently exist a scalable method for training autonomous vehicle control policies that go beyond imitation learning and that generalize to previously unseen road and complex, near-crash situations.

VISTA synthesizes training data for a broad range of vehicle positions and orientations from real driving data; performs data generation across three distinct sensor modalities –RGB cameras, LiDAR sensors, and event-based cameras; and provides in-painted ado vehicles for learning robust driving policies that involve multi-agent encounters and interactions. The engine is capable of generating a continuum of novel, photorealistic trajectories and environmental states. This variety ensures agent policies learned in ***VISTA*** benefit from autonomous exploration of the feasible driving space, including scenarios in which the agent can recover from near-crash off-orientation positions. Such positions are a common edge-case in autonomous driving and are difficult and dangerous to collect training data for in the real world.

Using ***VISTA***, we demonstrate that learned policies derived directly from simulation can be directly transferred to a full-scale autonomous vehicle. We demonstrate the ability to train and test perception-to-control policies across each of the sensor types and to deploy them onboard a full-scale autonomous vehicle. The policies learned in ***VISTA*** exhibit sim-to-real transfer without modification and greater robustness than those trained exclusively on real-world data.

In this chapter, we present the key capabilities of ***VISTA*** , summarized as:

1. A photorealistic, scalable, data-driven engine for synthesizing a continuum of new perceptual inputs locally around an existing dataset of stable human collected driving data;

2. End-to-end pipelines for learning autonomous control policies across three distinct sensor modalities, for both lane-stable control and interactive driving tasks;

3. Experimental validation that agents trained in ***VISTA*** can be deployed directly in the real-world and achieve improved robustness compared to previous state-

of-the-art imitation learning models trained exclusively on real data.

3.2 Data-Driven Simulation for Robot Learning

3.2.1 Building Virtual Worlds from Data

Simulation engines for training robust, end-to-end autonomous vehicle controllers must address the challenges of photorealism, real-world semantic complexities, and scalable exploration of control options, while avoiding the fragility of imitation learning and preventing unsafe conditions during data collection, evaluation, and deployment. Our data-driven simulator, ***VISTA***, synthesizes photorealistic and semantically accurate local viewpoints as a virtual agent moves through the environment (Figure 3-2). ***VISTA*** uses a repository of sparsely sampled trajectories collected by human drivers. For each trajectory through a road environment, ***VISTA*** synthesizes views that allow virtual agents to drive along an infinity of new local trajectories consistent with the road appearance and semantics, each with a different view of the scene.

Upon receiving an observation of the environment at time t, the agent commands a desired steering curvature, κ_t, and velocity, v_t to execute at that instant until the next observation. We denote the time difference between consecutive observations as Δt. ***VISTA*** maintains an internal state of each agent's position, (x_t, y_t), and angular orientation, θ_t, in a global reference frame. The goal is to compute the new state of the agent at time, $t + \Delta t$, after receiving the commanded steering curvature and velocity. First, ***VISTA*** computes the changes in state since the last timestep,

$$\begin{aligned}\Delta\theta &= |v_t \cdot \Delta t| \cdot \kappa_t, \\ \Delta\hat{x} &= \left(1 - \cos\left(\Delta\theta\right)\right)/\kappa_t, \\ \Delta\hat{y} &= \sin\left(\Delta\theta\right)/\kappa_t.\end{aligned} \tag{3.1}$$

VISTA updates the global state, taking into account the change in the agent's orien-

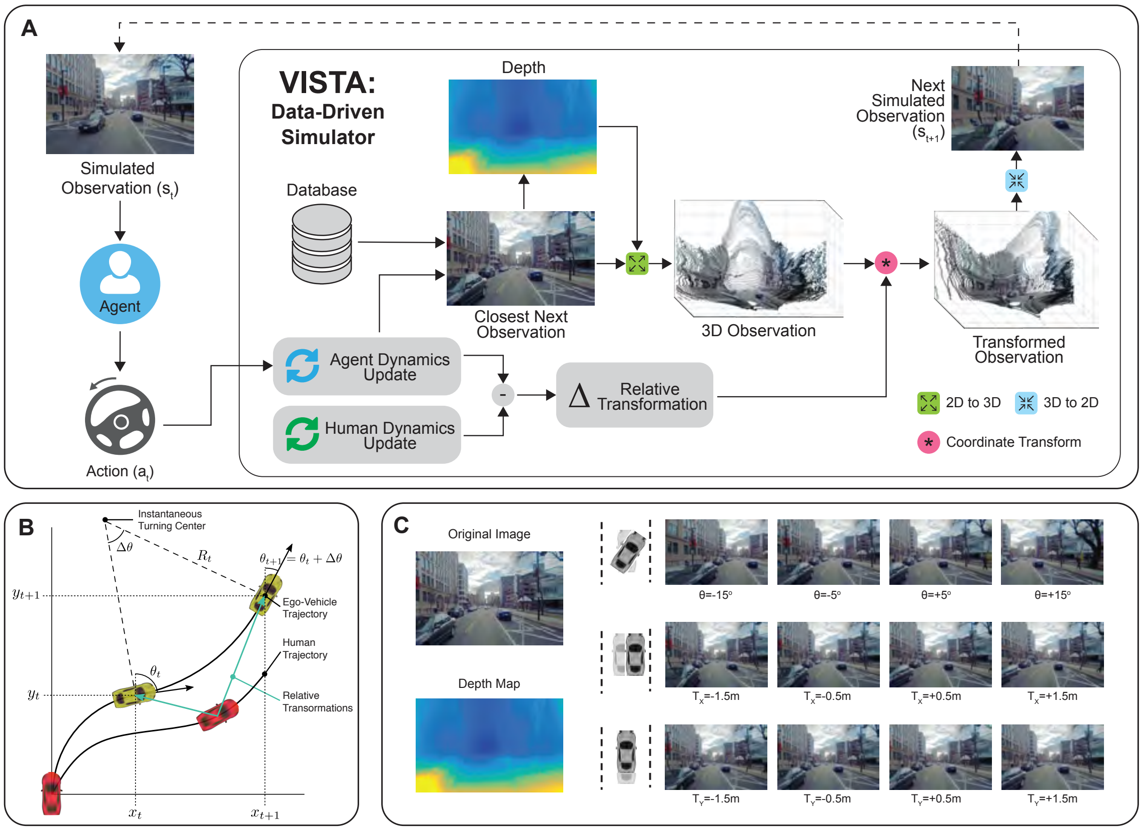


Figure 3-2: **Simulating novel viewpoints for learning.** Schematic of an autonomous agent's interaction with the data-driven simulator (A). At time step, t, the agent receives an observation of the environment and commands an action to execute. Motion is simulated in ***VISTA*** and compared to the human's estimated motion in the real world (B). A new observation is then simulated by transforming a 3D representation of the scene into the virtual agent's viewpoint (C).

tation, by applying a 2D rotational matrix before updating the position in the global frame:

$$\theta_{t+\Delta t} = \theta_t + \Delta\theta, \tag{3.2}$$

$$\begin{bmatrix} x_{t+\Delta t} \\ y_{t+\Delta t} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} \cos(\theta_{t+\Delta t}) & -\sin(\theta_{t+\Delta t}) \\ \sin(\theta_{t+\Delta t}) & \cos(\theta_{t+\Delta t}) \end{bmatrix} \begin{bmatrix} \Delta\hat{x} \\ \Delta\hat{y} \end{bmatrix}.$$

This process is repeated for both the virtual agent who is navigating the environment and the replayed version of the human who drove through the environment in the real world. Now in a common coordinate frame, ***VISTA*** computes the relative displacement by subtracting the two state vectors. Thus, ***VISTA*** maintains estimates of the lateral, longitudinal, and angular perturbations of the virtual agent with respect to the closest human state at all times (Figure 3-2B).

3.2.2 Multi-sensor Simulation

RGB Image Camera

VISTA is scalable as it does not require storing and operating on 3D reconstructions of entire environments or cities. Instead, it considers only the observation collected nearest to the virtual agent's current state. Simulating virtual agents over real road networks spanning thousands of kilometers requires several hundred gigabytes of monocular camera data. Figure 3-2C presents view synthesis samples. From the single closest monocular image, a depth map is estimated using a convolutional neural network using self-supervision of stereo cameras (Godard et al., 2017). Using the estimated depth map and camera intrinsics, our algorithm projects from the sensor frame into the 3D world frame. After applying a coordinate transformation to account for the relative transformation between virtual agent and human, the algorithm projects back into the sensor frame of the vehicle and returns the result to the agent as its next observation. To allow some movement of the virtual agent within the ***VISTA*** environment, we project images back into a smaller field-of-view than the collected data (which starts at 120°). Missing pixels are inpainted using a bilinear sampler, although we acknowledge more photorealistic, data-driven approaches (Liu et al., 2018) that could also be used. ***VISTA*** is capable of simulating different local rotations ($\pm 15°$) of the agent as well as both lateral and longitudinal translations (± 1.5m) along the road. As the free lateral space of a vehicle within its lane is typically less than 1m, ***VISTA*** can simulate beyond the bounds of lane-stable driving. Note that while we focus on data-driven simulation for lane-stable driving, the presented approach is also applicable to end-to-end navigation (Amini et al., 2019a) learning by stitching together collected trajectories to learn through arbitrary intersection configurations.

LiDAR Pointcloud

LiDAR sensors play a central role in modern autonomy pipelines due to their accuracy in measuring geometric depth information and robustness to environmental changes like illumination. Unlike cameras which return structured grid-like images, the LiDAR sensor captures a sparse pointcloud of the environment. Here, every point is represented by a 4-tuple: (x, y, z, i), where (x, y, z) is the position of the point in 3D Cartesian space and i is the intensity feature measurement of that point. Given a virtual agent's position in the environment, along with a relative transformation (rotation $R \in \mathbb{R}^{3\times 3}$ and translation $t \in \mathbb{R}^3$) to the nearest human collected pointcloud, Ψ, the goal of ***VISTA*** is to synthesize a novel LiDAR pointcloud, Ψ', which appears to originate from the virtual agent's relative position.

Since Ψ is represented in 3D Cartesian space, a naive solution would be to directly apply the relative transformation of the agent (R, t) to Ψ as a rigid transformation: $\Psi' = R\,\Psi + t$. However, this approach will fail for several reasons. The pointcloud obtained from a LiDAR sensor has a specific ring pattern pattern originating at the sensor's optical center. Applying a rigid transformation to the points will not only transform the individual points, but in doing so, also transform and break the ring structure inherently defining the sensor's location. Instead, to preserve the sensor structure we must recast LiDAR rays, from the new sensor location, into the scene and estimate new readings. Furthermore, the naively transformed pointcloud will very likely have points which may have been visible in the original scan, but become occluded in the new viewpoint and thus need to be rejected to maintain line-of-sight properties of the sensor. To overcome these issues, we (1) cull the now occluded points, (2) create a dense representation of the sparse pointcloud, and (3) sample from the dense representation according to a sensor-specific prior. We outline the algorithm below in detail.

First, we implement a GPU-accelerated culling technique to operate on our sparse transformed pointcloud, Ψ'. We start by projecting Ψ' into 2D polar coordinates,

$$\alpha = \arctan\left(\frac{\Psi'_y}{\Psi'_x}\right); \quad \beta = \arcsin\left(\frac{\Psi'_z}{d}\right); \quad d = \|\Psi'\|_2 \tag{3.3}$$

where (α, β) are the yaw and pitch angles of the rays connecting each of the points, and d are the distances along each ray. Now, the entire pointcloud, Ψ' is represented as a sparse 2D image (without loss of information) over (α, β) with d being the color or value of each pixel. To cull out points within our image, the distance of each pixel is compared to the average distance of its surrounding "cone" of neighboring rays. If the average distance of neighboring rays is less than the depth of the current pixel, the point is occluded and is removed from the sparse image. Figure 3-3 visualizes the large effect of our culling algorithm and the qualitative improvement it has on transformed pointclouds.

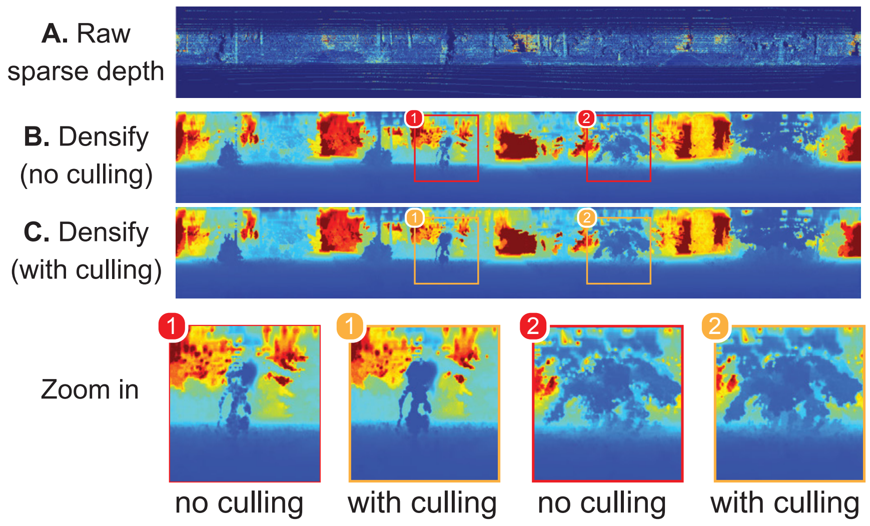


Figure 3-3: **Culling occluded points.** Transformed sparse scenes (A) will have points which should be rejected (culled) before rendering to avoid blending of foreground and background (B). Our culling algorithm (C) is lightweight, GPU-accelerated, and does not rely on raycasting a scene mesh.

With our sparse and culled pointcloud, we need to build a dense representation of the scene to sample a new cast of LiDAR rays and generate the novel viewpoint. To densify our sparse representation we train a UNet architecture (Ronneberger et al., 2015) to learn a dense output of the scene. Training data for our densification network is generated using a 2D linear interpolator. We found that using a data-driven approach to densification yielded smoother, more natural qualitative results over strict rule-based interpolation (`scipy.interpolate`). Furthermore, the resulting model is easily GPU-parallelizable to achieve significant speedups ($\sim 100\times$ faster).

Finally, we sample sparse points from our dense representation to form the novel view pointcloud. To determine sampling locations we can construct a prior, Ω, over the existing ray cast angles of the sensor in our dataset. The ray vectors for the sensor are largely fixed over time, as they are built into the hardware of the sensor, but can have some slight variations or drops based on the environment. Sampling ω from the prior yields a collection of rays, $\{(\alpha_i, \beta_i)\}$, to cast and collect point readings from. Furthermore, the prior, Ω, will respect several desirable properties of the sensor which can also be user specified such as the quantity and density of the LiDAR rays. Since we are still operating in polar coordinate image space, ω is equivalent to a binary mask image denoting where in our dense image should be sampled. With our new, sampled polar image we can invert the transform in Eq. 3.3 to represent our data back in the desired 3D Cartesian space. Figure 3-4 visualizes the different stages in the rendering pipeline, through the dense representation of the scene (A,B) as well as the result after sampling and reprojecting back to 3D cartesian space (C).

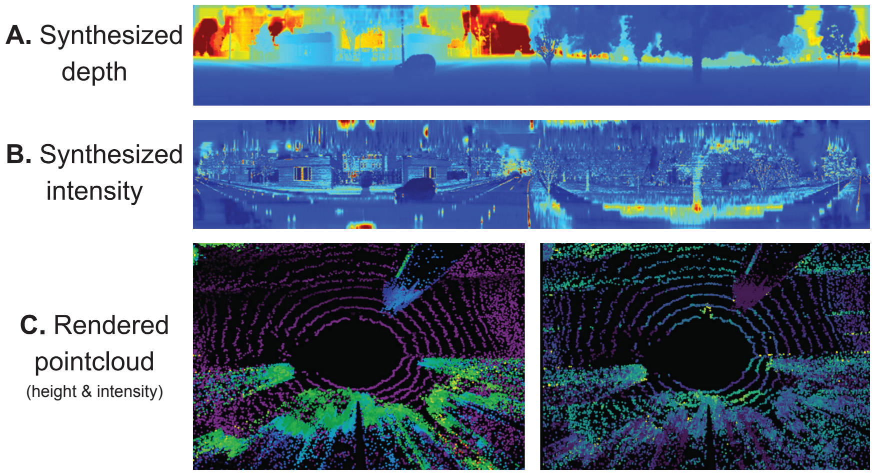


Figure 3-4: **LiDAR novel view-synthesis.** Simulating a lateral translation of 1m off the road. Dense representations of depth (A) and intensity (B) are estimated from the sparse transformation. Sparse pointclouds (C) are rendered by sampling the dense representation according to the sensor prior.

Event-based Camera Stream

Event-based cameras are asynchronous, continuous-time sensors that detect brightness changes of the scene. An event is emitted when brightness change exceeds a certain threshold at a pixel location, and is described as a 4-tuple of pixel coordinate, timestamp, and polarity. The polarity is a binary value that indicates whether brightness change is positive or negative. Conceptually, event camera data can be viewed as the derivative of regular RGB camera data with additional advantages of much higher operating frequency ($> 10,000$Hz) and dynamic range. Given its similarity to RGB cameras, event data can be simulated by taking the derivative with respect to time over interpolated RGB frames (Rebecq et al., 2018; Gehrig et al., 2020b). Our proposed method extends prior work to additionally handle (1) non-aligned camera projection across RGB and event cameras; and (2) novel view synthesis according to vehicle's ego-motion. Simulating events from RGB instead of event data allows applying ***VISTA*** to existing datasets which mostly contain RGB sequences but not event data (Figure 3-5).

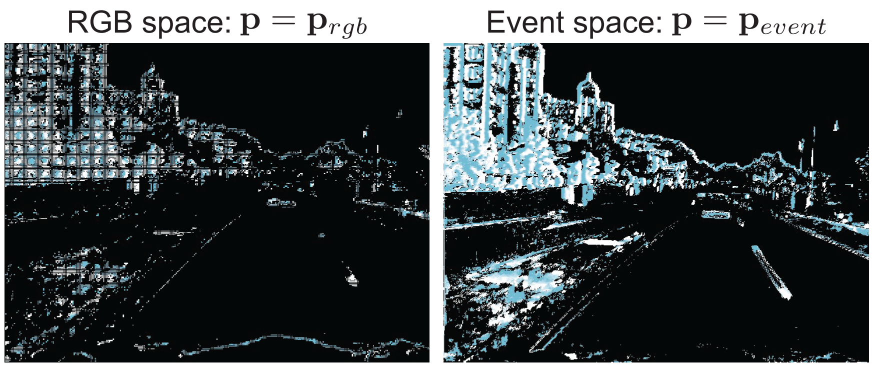

RGB space: $\mathbf{p} = \mathbf{p}_{rgb}$
Event space: $\mathbf{p} = \mathbf{p}_{event}$

Figure 3-5: **Different pixel spaces for event generation**. Events can be generated by estimating brightness change in RGB camera image space or virtual event camera image space.

To capture the instantaneous change of pixel intensity, we need to first construct a continuous representation of RGB image stream. Given two consecutive RGB frames I_{t_1}, I_{t_2} and bidirectional optical flow $F_{t_1 \rightarrow t_2}, F_{t_2 \rightarrow t_1}$, this can be achieved by arbitrary-time frame interpolation (Jiang et al., 2018),

$$I_{t_1+k\Delta t} = f_{\text{interp}}(I_{t_1},\ I_{t_2},\ F_{t_1+k\Delta t\to t_1},\ F_{t_1+k\Delta t\to t_2}) \tag{3.4}$$

$$F_{t_1+k\Delta t\to t_1} = -(1-k\Delta t)k\Delta t F_{t_1\to t_2} + (k\Delta t)^2 F_{t_2\to t_1} \tag{3.5}$$

$$F_{t_1+k\Delta t\to t_2} = (1-k\Delta t)^2 F_{t_1\to t_2} - k\Delta t(1-k\Delta t)F_{t_2\to t_1} \tag{3.6}$$

where $k \in [0, \frac{t_2-t_1}{\Delta t}]$ and $k\Delta t$ specifies the time interval to be simulated in between. The arbitrary-time flow is derived from temporal consistency going forward ($t_1 \to t_1 + k\Delta t$) and backward ($t_2 \to t_1 + k\Delta_t$) in time with local smoothness assumption. The interpolation function f_{interp} is implemented by a neural network that handles visibility issue from both directions. We refer the reader to (Jiang et al., 2018) for more details.

With this, we now apply an event generation model (Mueggler et al., 2017; Gallego et al., 2017),

$$(\mathbf{p}, t_k, \rho) \text{ if } \rho\big(\ln I(\mathbf{p}, t_k) - \ln I(\mathbf{p}, t_k - \Delta t)\big) \geq c_k \tag{3.7}$$

where $\mathbf{p}$ is the pixel coordinate, $\rho \in \{-1, 1\}$ is polarity and $c_k \sim \mathcal{N}(\mu_c, \sigma_c)$ is the contrast threshold sampled from a Gaussian to simulate noise. The temporal granularity of event generation is determined by Δt, which yields more accurate simulation with smaller values until saturating at subpixel displacement of optical flow. Furthermore, adaptive sampling (Rebecq et al., 2018) is used to jointly achieve accuracy and efficiency,

$$\Delta t = \frac{t_2 - t_1}{\min\{\max_{\mathbf{p}} \max\{F_{t_1\to t_2}(\mathbf{p}), F_{t_2\to t_1}(\mathbf{p})\} - 1, \Delta t_{\max}\}} \tag{3.8}$$

where $\Delta t_{\max}$ sets an upper bound to computational resources required for simulation.

Thus far, the only thing yet to be defined is the space that pixel coordinate $\mathbf{p}$ lives in. It can be image space of either RGB camera (input) or novel-view event camera (output), which are related by reprojection,

$$\mathbf{p}_{event} = K_{event} T_{event}^{novel} T_{rgb}^{event} D(\mathbf{p}_{rgb}) K_{rgb}^{-1} \mathbf{p}_{rgb} \tag{3.9}$$

where K_{event}, K_{rgb} are intrinsics for event and RGB cameras, D is depth, and T is transformation across two poses. We explicitly factorize T_{rgb}^{event} since the control commands reference at existing sensors in the dataset. In event generation model (3.7), using $\mathbf{p}_{rgb}$ involves generating events in RGB image space from the dataset and reprojecting pixel coordinates to event image space. We implement a bilinear sampler with thresholding to handle non-integer pixels after reprojection. On the other hand, using $\mathbf{p}_{event}$ renders the scene based on RGB dataset in event camera space and generate events without pixel reprojection. We argue that $\mathbf{p} = \mathbf{p}_{event}$ may be a better option since it casts the subpixel issue of reprojection (3.9) from interpolating in pixel coordinate space as in $\mathbf{p} = \mathbf{p}_{rgb}$ to interpolating in color/intensity space of meshes during RGB image rendering. The comparison can be seen in Figure 3-5.

3.2.3 Multi-agent Interaction

Multi-agent Rendering

VISTA supports the rendering of multiple agents in a simulated environment in order to enable studies and exploration of multi-agent interactions and dynamics (Figure 3-6). Every agent's current state is associated with a pre-collected frame according to the closest distance from its pose to the pose of the data-collection vehicle. We compensate for the local transform between the data and the agent leveraging the camera extrinsics during rendering combined with the use of approximate depth information. We project the image from the data coordinate frame, i.e. the data viewpoint p_s, to the agent's viewpoint p_t,

$$p_t = K T_{v_2 \to t} T_{v_1 \to v_2} T_{s \to v_1} D_s(p_s) K^{-1} p_s, \quad T_{s \to v_1} = T_{v_2 \to t}^{-1}, \tag{3.10}$$

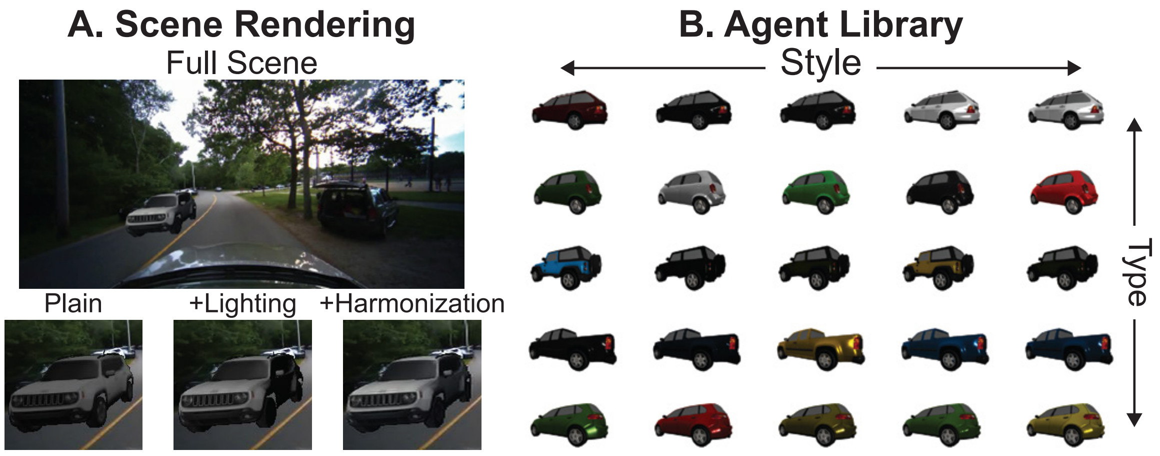


Figure 3-6: **Multi-agent rendering.** (A) Example simulated scene with an oncoming agent. Different levels of rendering compare the effect of lighting and harmonization. (B) A variety of agents are available during simulation with different styles, body material, specularity, and color.

where K is the camera intrinsic, $D_s(p_s)$ is the depth of the data at point p_s, $T_{s\to v_*}$ is the transform from camera to vehicle body, and $T_{v_1\to v_2}$ is the transform from the vehicle body of the data to that of agent. We explicitly model the transform using vehicle body $T_{v_1\to v_2}$ since it simplifies placing new meshes in the scene and checking for collisions among meshes. Note that the yaw difference in $T_{s\to v_*}$ may induce a bias of how the mesh is placed towards the left or right, which is of great importance with other agents' present in the scene.

Each agent is embodied by a mesh randomly sampled from a parametrizable vehicle mesh library, which allows randomization over different car models, a set of physically-realistic diffuse color, specular color, specular highlight, metalness, and roughness of car body material, as shown in Figure 3-6B. The mesh is configured at the beginning and remains the same throughout an entire episode. We place the mesh based on the relative transform between the corresponding agent and the egocentric viewpoint for rendering. For lighting, we cast ambient light based on the average color of the scene and directional light from the egocentric viewpoint infinitely far away. For postprocessing in image space, we run image harmonization (Sofiiuk et al., 2021) with the rendered images and the foreground mask.

Multi-agent Objectives and Tasks

VISTA enables end-to-end policy learning with multi-agent data-driven simulation. We showcase these capabilities on two autonomous driving tasks, *car following* and *overtaking*, focusing on the latter as it is more challenging. Both tasks involve two agents in the scene, the ego car and the front car. The front car is randomly initialized at some distance along the forward direction of the ego car with random speed, lateral shift, and heading with respect to the road curvature. The objective of car following is to keep track of the front car while the front car may perform lane changes. The ego car is initialized to the random speed of the front car so that the ego car never loses the front car in its viewpoint and has sufficient information to perform tracking. In overtaking, the ego speed is randomly set to be faster than the front car, enabling an overtaking action. The goal of the ego car is to pass the front car without collision while performing lane stable maneuver. The front car of both tasks use pure pursuit controller to trace out trajectories automatically generated based on road curvature, while the ego car commands steering.

3.3 Learning within Simulation

In this section, we discuss different techniques to effectively learn an optimal policy for decision making and control. We aim to learn an autonomous policy, f, parameterized by $\boldsymbol{w}$ which estimates an optimal action, $\hat{a}_t$, at time t from the current state, s_t:

$$\hat{a}_t = f(s_t; \boldsymbol{w}). \tag{3.11}$$

There are many considerations to take into account when learning a policy from data that effect the ultimate policy attained. Policy learning algorithms can be offline (learned from a fixed pre-collected dataset) or online (learned as the dataset is built and jointly guides the collection process).

When considering dynamic decision making scenarios like autonomous control, it is also critical to consider the effect of compounding errors on the policy. If a model incurs a minor error on every output its overall average error rate may naively appear to be very small. However, because the errors have a direct impact on the future states that the agent sees, the errors quickly compound and cause the inputs to rapidly diverge from the original training distribution. In practice, this phenomenon causes a severe degradation of performance when offline (open-loop) policies are deployed into reality (with closed-loop feedback) and is critical in designing an effective policy learning algorithm.

3.3.1 Supervised Learning

The simplest method for policy learning follows a supervised learning framework. We assume we have access to a dataset of n observed state-action pairs $(s_t, a_t)_{i=1}^{n}$ from expert demonstration. For example, a human driver can collect a stream of image-control pairs while driving and exhibiting safe (optimal) behavior on the road. In supervised learning, the agent outputs a *deterministic* action by minimizing the empirical error,

$$L(\boldsymbol{w}) = \sum_{i=1}^{n} (f(s_t; \boldsymbol{w}) - a_t)^2. \tag{3.12}$$

Our empirical error $L(\boldsymbol{w})$ can be minimized through stochastic gradient descent (SGD) or other similar non-convex optimization techniques:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}). \tag{3.13}$$

However, because the agent's predicted action has a direct impact on future input state's that are observed during execution, this violates a crucial assumption made by supervised learning - which is that the inputs to the system are independent and identically distributed (i.i.d.). Intuitively this is realized by a compounding of errors, where as soon as the policy makes a mistake, it begins to encounter observations that are out-of-distribution from the expert training data, which in turn feedback to cause errors of greater and greater magnitude.

For example, in autonomous driving because our expert training data will only demonstrate how to drive while in the center of the lane, the policy will not be able to recover from any off-orientation perturbations (*i.e.*, slightly translated off-center, or rotated with respect to the curvature of the road). Any small errors in the nominal policy will cause the vehicle to shift slightly into an off-orientation, out-of-distribution position on the road where the policy will know know how to recover from.

Thus, it is necessary to also train our policies to exhibit reliable behavior not only when in the nominal data regimes of deployment, but also during these challenging off-orientation edge-cases situations. In efforts to mitigate this issue with supervised learning, it is desirable to collect the necessary off-orientation recovery data. However, for safety-critical scenarios such as robotics, one cannot collect this edge-case data (*e.g.* near-crash scenarios, in driving) in order to sufficiently train our model to recover from these situations. Thus, simulation becomes critical to generate the necessary edge-case data which still maintaining a high level of safety during training.

3.3.2 Guided Policy Learning

With ***VISTA*** we are able to simulate and generate novel viewpoints in arbitrary environments and multi-agent interactions. Now we discuss a method for taking a data generation technique such as ***VISTA*** and building full datasets amenable for policy learning while overcoming the fundamental challenges associated with vanilla supervised learning. Guided policy learning (GPL) is an algorithm for bridging this gap and injecting privileged information into the data generation process so we can learn how to sufficiently recover from off-orientation position during deployment.

The fundamental insight of GPL is to combine our original supervised learning dataset together with a privileged controller which will label novel viewpoints that we generate prior to training. Specifically, using ***VISTA*** we have a method for generating the sensory data (camera, LiDAR, events) from novel viewpoints and near crash positions; however, we need to associate this sensory data (inputs) with their corresponding action (outputs) which are needed for training. Our privileged controller provides these labels which closes the loop in our data generation procedure.

Note that the privileged controller, since it is running prior to training and within simulation, has access to substantially more information than our final deployment-time policy (which only observes raw sensory data). On the other hand, the privileged controller may have complete information on the state of the vehicle to augment its labelling process and produce the highest quality labels for training. For example, we can provide full state information of the agent (lateral and longitudinal displacement, curvature of the road, location and speed of other agents in the scene, etc). With this full (privileged) view of the world, we can generate high quality labels for training our policy.

Algorithm 1 describes data generation in ***VISTA*** and downstream GPL. First, the simulator is reset with random initialization (e.g., off-center position and heading). The simulator is stepped with control commands from the privileged controller and iteratively generates sensor measurements and optimal control labels for supervising policy learning. While there are no restrictions on the optimal controller used for the privileged agent, we use a pure-pursuit controller in our experiments to smoothly guide the agent back to the center position and heading of the road within some look-ahead distance. We use a shuffled buffer to approximately ensure *i.i.d.* training samples. Before adding data into the buffer, rejection sampling is used to balance the label distributions (Amini et al., 2018b, 2019a,b).

Finally, we introduce a branching step that locally branches out from stepping the simulator with privileged control (e.g., turning right instead of left as instructed). This is extremely important to policy learning with event cameras since event data captures *changes in the scene* and thus conflates the vehicle's ego-motion with its own control. This means that a policy can accurately correlate control to the motion of the scene instead of attending to the road. While not presenting an obvious issue during open-loop evaluation (Maqueda et al., 2018), these policies will fail catastrophically on a closed-loop test. However, by branching with arbitrary control, we effectively disentangle ego-motion and scene information in event patterns.

GPL provides a systematic approach to training a policy on a sequential decision making task by combining an expert demonstration dataset with a simulator and privileged controller to label novel expert trajectories for recovery. However, one

Algorithm 1 Data generation and training in ***VISTA***

```
for $k \leftarrow 1$ to $N$ do
    while ! buffer.full() do
        if VISTA.done() then
            VISTA.reset()
        end if
        $x \leftarrow$ VISTA.readSensorBuffer()
        if useBranching then
            VISTA.randomStep()
            $y \leftarrow$ privilegedController(VISTA.getState())
            revertState(VISTA, privilegedController)
        else
            $y \leftarrow$ privilegedController(VISTA.getState())
        end if
        VISTA.step($y$)
        if ! rejectSample($x, y$) then
            buffer.add($x, y$)
        end if
    end while
    buffer.shuffle()
    trainModel(buffer.next())
end for
```

disadvantage of GPL is the original requirement of expert demonstrations and in some cases defining a high performance privileged controller. In several situations it is desirable to learn only by providing high level reward signals and learning the controller that can achieve as much reward as possible. This eliminates the need to do any collection of data or manually define a privileged controller.

3.3.3 Reinforcement Learning

In the reinforcement learning (RL) setting, the agent has no explicit feedback of the expert human actuated command, a_t. Instead, it receives a reward r_t for every consecutive action that does not result in an intervention and can evaluate the return, R_t, as the discounted, accumulated reward,

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{3.14}$$

where $\gamma \in (0, 1]$ is a discounting factor. In other words, the return that the agent receives at time t is a discounted distance traveled between t and the time when the vehicle requires an intervention. As opposed to in supervised learning, the agent optimizes a *stochastic* policy over the space of all possible actions: $\pi(a|s_t; \boldsymbol{w})$. Since the steering control of autonomous vehicles is a continuous variable, we parameterize the output probability distribution at time t as a Gaussian, (μ_t, σ_t^2). Therefore, the policy gradient, $\nabla_{\boldsymbol{w}}\pi(a|s_t; \boldsymbol{w})$, of the agent can be computed analytically:

$$\nabla_{\boldsymbol{w}}\pi(a|s_t; \boldsymbol{w}) = \pi(a|s_t; \boldsymbol{w})\, \nabla_{\boldsymbol{w}} \log\left(\pi(a|s_t; \boldsymbol{w})\right). \tag{3.15}$$

Thus, the weights $\boldsymbol{w}$ are updating in the direction $\nabla_{\boldsymbol{w}} \log\left(\pi(a|s_t; \boldsymbol{w})\right) \cdot R_t$ during training (Williams, 1992; Sutton et al., 2000).

We train RL agents in various simulated environments, where they only receive rewards based on how far they can drive without intervention. Compared to supervised learning, where agents learn to simply imitate the behavior of the human driver, RL in simulation allows agents to learn suitable actions which maximize their total reward in that particular situation. Thus, the agent has no knowledge of how the human drove in that situation. Using only the feedback from interventions in simulation, the agent learns to optimize its own policy and thus to drive longer distances (Algorithm 2).

While there is no limitation for RL algorithms to be applied in our simulator, we adopt proximal policy optimization (PPO) (Schulman et al., 2017) for its simplicity and ubiquity across robot learning tasks, from land (Guan et al., 2020) and aerial (Bøhn et al., 2019) vehicles to humanoid walking (Schulman et al., 2017). PPO is an on-policy algorithm that maximizes the following objective

$$\mathop{\mathbb{E}}_{s,a\sim\pi_k}\left[\min\left(\frac{\pi_{k-1}(a|s)}{\pi_k(a|s)}A^{\pi_k}(a|s), \mathrm{clip}\left(\frac{\pi_{k-1}(a|s)}{\pi_k(a|s)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_k}(a|s)\right)\right], \tag{3.16}$$

where $\pi(a|s)$ is the policy's action distribution given observation s, $A^{\pi_k}(a|s)$ is advantage function which estimates how good an action is, and ϵ is a hyperparameter.

Algorithm 2 Policy Gradient (PG) training in **VISTA**

```
Initialize $\boldsymbol{\theta}$                                    ▷ NN weights
Initialize $D \leftarrow 0$                                         ▷ Single episode distance
while $D < 10$km do
    $s_t \leftarrow$ VISTA.reset()
    while VISTA.done = False do
        $a_t \sim \pi(s_t; \boldsymbol{\theta})$                    ▷ Sample action
        $s_{t+1} \leftarrow$ VISTA.step($a_t$)                      ▷ Update state
        $r_t \leftarrow 0.0$ if VISTA.done else $1.0$               ▷ Reward
    end while
    $D \leftarrow$ VISTA.episode_distance
    $R_t \leftarrow \sum_{k=1}^{T} \gamma^k r_{t+k}$                ▷ Discounted return
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t; \boldsymbol{\theta})\, R_t$    ▷ Update
end while
return $\boldsymbol{\theta}$
```

The intuition is to maximize task performance, measured by the advantage function, while making sure the new policy π_k does not deviate too much from the old policy π_{k-1} by bounding the ratio π_{k-1}/π_k to a small interval, ϵ. In our case, the policy takes observation s as images and outputs action a as steering angle.

Multi-agent Reward Objectives

We can formulate our end-to-end policy learning as a RL problem given its generalizability across a wide variety of tasks. Setting up a RL environment requires definitions of environment dynamics, terminal conditions, and the reward function. The simulator defines how the scene dynamically evolves after receiving an agent's action. Due to the nature of the simulator rendering, a default set of terminal conditions is exceeding a threshold maximum translation or rotation, as discussed in (Amini et al., 2020a, 2021). Another terminal condition occurs when there is collision among agents, determined by the overlap of polygons with shape as vehicle dimension exceeding certain threshold. Besides, given how a new observation is synthesized based on a local transform with respect to the data in the simulator and all autonomous driving tasks involving lane following, it is natural to define the reward function based on the rotational, lateral, and longitudinal components of vehicle pose with respect to the data (center line). Namely we have:

$$\begin{bmatrix} q_{lat} \\ q_{long} \\ q_{rot} \end{bmatrix} = \begin{bmatrix} \cos\theta_s & -\sin\theta_s & 0 \\ \sin\theta_s & \cos\theta_s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t - x_s \\ y_t - y_s \\ \theta_t - \theta_s \end{bmatrix}, \tag{3.17}$$

where (x_t, y_t, θ_t) and (x_s, y_s, θ_s) are the poses of the virtual agent and corresponding human reference respectively.

With this notation, the two aforementioned terminal conditions can be written as $\mathbb{1}_{q_{lat}>Z_{lat}}$ and $\mathbb{1}_{q_{rot}>Z_{rot}}$, where Z_* is the threshold that triggers termination. Thus, we can formally define a lane following reward as

$$R_{lane} = 1 - \left(\frac{q_{lat}}{Z_{lat}}\right)^2. \tag{3.18}$$

For car following, we can simply adapt the lane reward by changing the center line to the trajectory traced out by the front car. In overtaking, additional to lane reward, we define a pass reward based on comparing the distances traced out by both cars,

$$R_{pass} = \mathbb{1}\left[\int \sqrt{\dot{x}_e(t) + \dot{y}_e(t)} - \int \sqrt{\dot{x}_f(t) + \dot{y}_f(t)} \geq Z_{pass}\right], \tag{3.19}$$

where subscripts $*_e$ and $*_f$ denote ego and front car respectively, and Z_{pass} is hyper-parameter.

Finally, to provide more learning signal for collision avoidance, we dilate the polygon of the ego car and compute its overlap with other agents,

$$R_{collision} = -\frac{|\text{Dilate}(P_{ego}) \cap P_{other}|}{|P_{ego}|}, \tag{3.20}$$

where P denotes a vehicle's polygons. In both tasks, we add a comfort reward that is computed as the negative second derivative of steering $R_{comfort} = -\ddot{\delta}$ to reduce jittering.

3.3.4 Sim-to-real Transfer Techniques

We train imitation learning models with real-world data and multi-camera viewpoint augmentation and refer to these models as **IMIT-AUG** (Figure 3-7A). Alternatively, we can synthesize new viewpoints using a simulation engine.

Because ***VISTA*** is a fully data-driven simulation engine, its resulting simulated views are naturally high fidelity and photorealistic. On the other hand, model-based simulation presents a large amount of modularity benefits but suffers from poor photorealism and semantic mismatch. While tremendous effort has been placed into making model-based environments (Figure 3-7B) as photorealistic as possible, a simulation gap still exists. For example, as opposed to the data-driven ***VISTA*** simulator, the autonomous driving simulator CARLA (Dosovitskiy et al., 2017) is model-based, like many other autonomous driving simulators. The CARLA simulator can be used to evaluate the performance of end-to-end models using sim-to-real transfer learning techniques. We found that end-to-end models trained solely in CARLA were unable to transfer to the real world. Therefore, we evaluated the following two techniques for bridging the sim-to-real gap in CARLA.

Domain Randomization

We test the effect of domain randomization (DR) (Tobin et al., 2017) on learning within model-based simulation. DR attempts to expose the learning agent to many different random variations of the environment, thus increasing its robustness in the real-world. In our experiments, we randomized various properties throughout the environment (Figure 3-7C), including the sun position, weather, and hue of each of the semantic classes (i.e. road, lanes, buildings, etc). Like **IMIT-AUG** we also train DR models with viewpoint augmentation and thus refer to these models as **DR-AUG**.

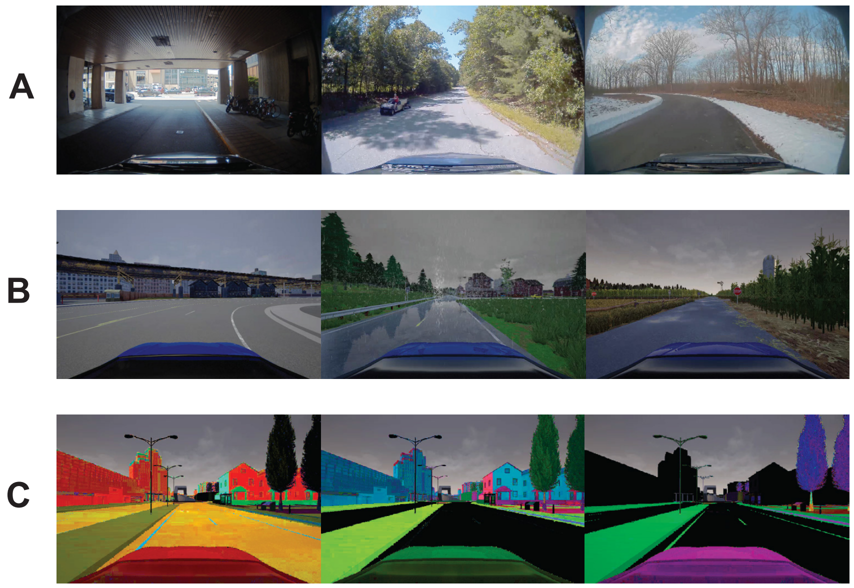


Figure 3-7: **Training images from various comparison methods.** Samples drawn from the real-world (**IMIT-AUG**; A) and the model-based simulator CARLA (B-C). Domain randomization (**DR-AUG**; C) illustrates transformations of a single location for comparison.

Domain Adaptation

We evaluate a model that is trained with both simulated and real images to learn shared control. Since the latent space between the two domains is shared (Bewley et al., 2018), the model can output a control from real images during deployment even though it was only trained with simulated control labels during training. Again, viewpoint augmentation is used when training our sim-to-real baseline, **S2R-AUG**.

3.4 Experimental Setup

Learned controllers were deployed directly onboard a full-scale vehicle (2020 Lexus RX 450H or 2015 Toyota Prius V) retrofitted for full autonomous control (Naser et al., 2017). The primary perception sensors for control are a LI-AR0231-GMSL camera (120 degree field-of-view), operating at 30Hz, a 10Hz Velodyne VLS-128 LiDAR sensor, and a Prophesee Gen3 event-based camera. Data is serialized with h264 encoding with a resolution of 1920×1208. At inference time, images are scaled down approximately $3\times$ fold for performance. Also onboard are inertial measurement units (IMUs), wheel encoders, and centimeter-level accurate OxTS global positioning system (d-GPS) for evaluation. An NVIDIA Drive PX2 and NVIDIA 2080Ti GPU for accelerated computing. To standardize all model trials on the test-track, a constant desired speed of the vehicle was set at 20 kph, while the model commanded steering.

3.4.1 Data Collection

We collect data from multiple sensors (RGB camera, LiDAR, event camera) with vehicle speed 30-60 kph in a wide variety of environments, including different time of day (daytime/night), weather conditions (sun/rain), and road types (urban/rural). The entire dataset contains roughly 3 hours of driving data. RGB images, LiDAR point cloud, event data, and curvature feedback are used for ***VISTA*** simulation, policy learning, and evaluation. GPS data is only used for evaluation.

3.4.2 Test Environment

The model's generalization performance was evaluated on previously unseen roads. That is, the real-world training set contained none of the same areas as the testing track (spanning over 3km) where the model was evaluated.

Agents were evaluated on all roads in the test environment. The track presents a difficult rural test environment, as it does not have any clearly defined road boundaries or lanes. Cracks, where vegetation frequently grows onto the road, as well as strong shadows cast from surrounding trees, cause classical road detection algorithms to fail.

3.5 Results

3.5.1 Offline Lane-stable Policy Learning

In this section, we present results on learning end-to-end control of autonomous vehicles entirely within ***VISTA***, under different weather conditions, times of day, and road types. Each environment collected for this experiment consisted of, on average, one hour of driving data from that scenario.

We started by learning end-to-end policies in different times of day (Figure 3-8A). As expected, we found that agents learned more quickly during the day than at night, where there was often limited visibility of lane markers and other road cues. Next, we considered changes in the weather conditions. Environments were considered "rainy" when there was enough water to coat the road sufficiently for reflections to appear or when falling rain drops were visible in the images. Comparing dry with rainy weather learning, we found only minor differences between their optimization rates (Figure 3-8B). This was especially surprising considering the visibility challenges for humans due to large reflections from puddles as well as raindrops covering the camera lens during driving. Finally, we evaluated different road types by comparing learning on highways and rural roads (Figure 3-8C). Since highway driving has a

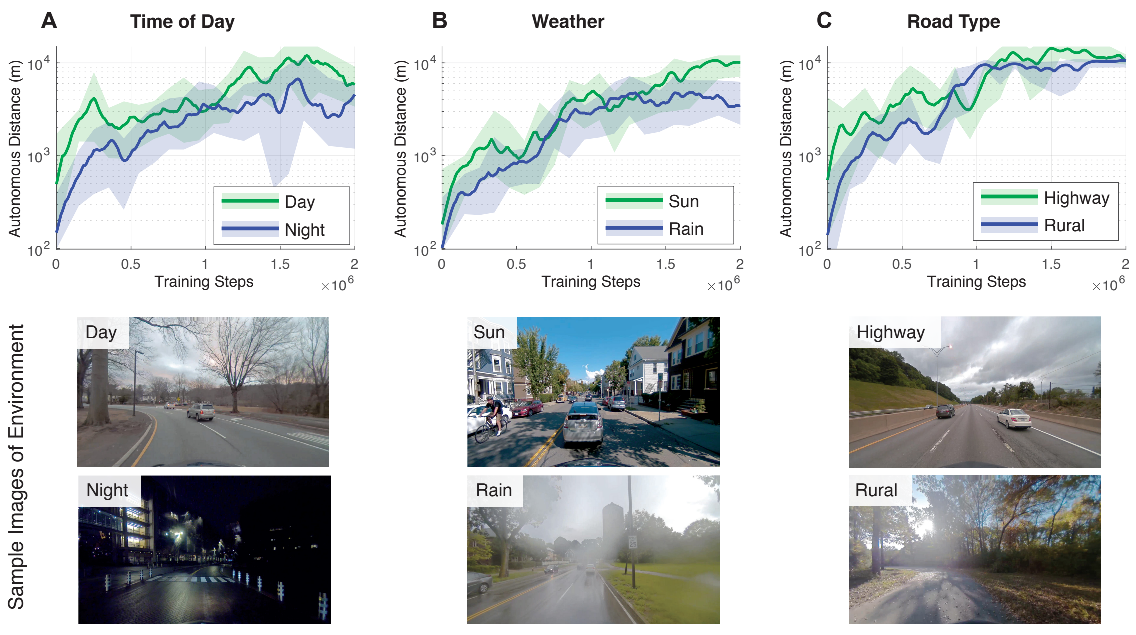


Figure 3-8: **Reinforcement learning in simulation.** Autonomous vehicles placed in the simulator with no prior knowledge of human driving or road semantics demonstrate the ability to learn and optimize their own driving policy under various different environment types. Scenarios range from different times of day (A), to weather condition (B), and road types (C).

tighter distribution of likely steering control commands (*i.e.*, the car is traveling primarily in a nearly straight trajectory), the agent quickly learns to do well in this environment compared to the rural roads, which often have much sharper and more frequent turns. Additionally, many of the rural roads in our database lacked lane markers, thus making the beginning of learning harder since this is a key visual feature for autonomous navigation.

In our experiments, learned agents iteratively explore and observe their surroundings (e.g. trees, cars, pedestrians, etc.) from novel viewpoints. On average, the learning agent converges to autonomously drive 10km without crashing within 1.5 million training iterations. Thus, when randomly placed in new locations with similar features to data encountered during training, the agent is able to use its learned policy to navigate. While demonstration of learning in simulation is critical for development of autonomous vehicle controllers, we also evaluated the learned policies directly on-board our full-scale autonomous vehicle to test translation and generalization to the real world.

3.5.2 Online Evaluation in the Real World

Next, we considered models trained in ***VISTA*** against baseline models, deployed them into the real world, and evaluated their performance. First, we note that models trained solely in CARLA did not transfer, and that training with data viewpoint augmentation (Bojarski et al., 2016) strictly improved performance of the baselines. Thus, we compare against baselines with augmentation. Each model is trained 3 times and tested individually on every road on the test track. At the end of a road, the vehicle is restarted at the beginning of the next road segment. The test driver intervenes when the vehicle exits its lane. The mean trajectory of the three trials are shown in Figure 3-9A, with intervention locations drawn as red points. Road boundaries are plotted in black for scale of deviations.

IMIT-AUG yielded highest performance out of the three baselines, as it was trained directly with real-world data from the human driver. Of the two models trained with only CARLA control labels, **S2R-AUG** outperformed **DR-AUG** requiring an intervention every 700m compared to 220m. Even though **S2R-AUG** only saw control labels from simulation, it received both simulated and real perception. Thus, the model learned to effectively transfer some of the details from simulation into the real-world images allowing it to become more stable than purely randomizing away certain properties of the simulated environment (*i.e.* **DR-AUG**). ***VISTA*** exhibited the best performance of all the considered models and never required any interventions throughout the trials (totaling $>$ 10km of autonomous driving). The variance across trials is visualized in Figure 3-9B-C (line color in (B) indicates variance at that location). For each baseline, the variance tended to spike at locations that resulted in interventions, while the variance of ***VISTA*** was highest in ambiguous situations such as approaching an intersection, or wider roads with multiple possible correct control outputs.

We also initiated the vehicle from off-orientation positions with significant lateral and rotational offsets to evaluate robustness to recover from these near-crash scenarios (Figure 3-10). A successful recovery is indicated if the vehicle is able to successfully maneuver and drive back to the center of its lane within 5 seconds. We observed that agents trained in ***VISTA*** were able to recover from these off-orientation positions on real and previously un-encountered roads, and also significantly outperformed models trained with imitation learning on real world data (**IMIT**) or in CARLA with domain transfer (**DR-AUG** and **S2R-AUG**). On average, ***VISTA*** successfully recovered over $2\times$ more frequently than the next best, **IMIT-AUG**. The performance of **IMIT-AUG** improved with translational offsets, but was still significantly outperformed by ***VISTA*** models trained in simulation by approximately 30%. All models showed greater robustness to recovering from translations than rotations since rotations required significantly more aggressive control to recover with a much smaller room of error. In summary, deployment results for all models are shown in Table 3.1.

3.5.3 Augmenting Control Learning with Novel Viewpoints

Figure 3-11 demonstrates real-world policy deployment of IL and GPL policies. For each policy, we run the vehicle autonomously (controlled by the policy) for 3 trials

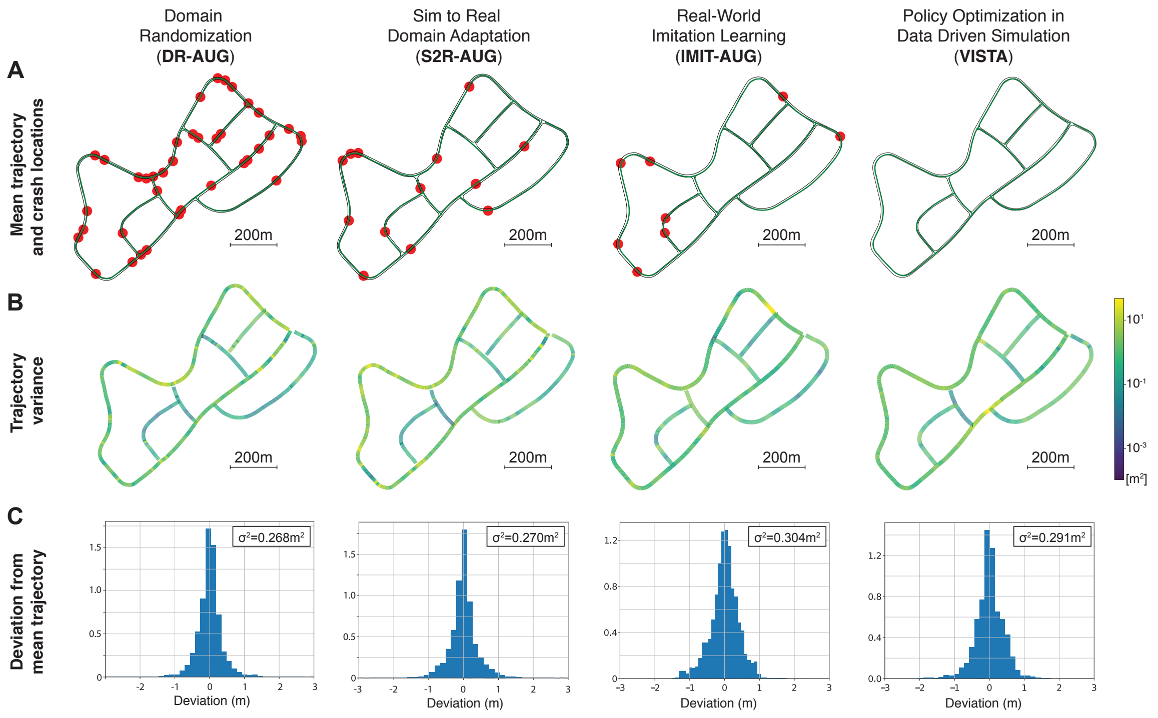


Figure 3-9: **Evaluation of end-to-end autonomous driving.** Comparison of simulated domain randomization (Tobin et al., 2017) and adaptation (Bewley et al., 2018) as well as real-world imitation learning (Bojarski et al., 2016) to learning within ***VISTA*** (left-to-right). Each model is tested 3 times at fixed speeds on every road on the test track (A), with interventions marked as red dots. The variance between runs (B) and the distribution of deviations from the mean trajectory (C) illustrate model consistency.

Table 3.1: **Real-world performance comparison.** Each row depicts a different performance metric evaluated on our test track. Bold cells in a single row represent the best performers for that metric, within statistical significance.

		DR-AUG (Tobin et al. (Tobin et al., 2017))	**S2R-AUG** (Bewley et al. (Bewley et al., 2018))	**IMIT-AUG** (Bojarski et al. (Bojarski et al., 2016))	**VISTA** (Ours)	**HUMAN** (Gold Std.)
Lane Following	# of Interventions	13.6 ± 2.62	4.33 ± 0.47	3.00 ± 0.81	$\mathbf{0.0 \pm 0.0}$	0.0 ± 0.0
	Dev. from mean [m]	$\mathbf{0.26 \pm 0.03}$	$\mathbf{0.31 \pm 0.06}$	$\mathbf{0.30 \pm 0.04}$	$\mathbf{0.29 \pm 0.05}$	0.22 ± 0.01
Near Crash Recovery (rate)	Trans. R (+1.5m)	0.57 ± 0.03	0.6 ± 0.05	0.71 ± 0.03	$\mathbf{1.0 \pm 0.0}$	1.0 ± 0.0
	Trans. L (+1.5m)	0.51 ± 0.08	0.51 ± 0.08	0.67 ± 0.09	$\mathbf{0.97 \pm 0.03}$	1.0 ± 0.0
	Yaw CW (+30°)	0.35 ± 0.06	0.31 ± 0.11	0.44 ± 0.06	$\mathbf{0.91 \pm 0.06}$	1.0 ± 0.0
	Yaw CCW (−30°)	0.37 ± 0.03	0.33 ± 0.05	0.37 ± 0.03	$\mathbf{0.93 \pm 0.05}$	1.0 ± 0.0

in the outerloop of the test track (total distance of all trials is 45km). Figure 3-11 shows interventions throughout multiple trials. Figure 3-12 shows percentage of deviation from center smaller than a range of thresholds, where larger area below the line means more stable lane keeping maneuvers. The performance of GPL policies for RGB and LiDAR are significantly better than IL policies. This is highly aligned with

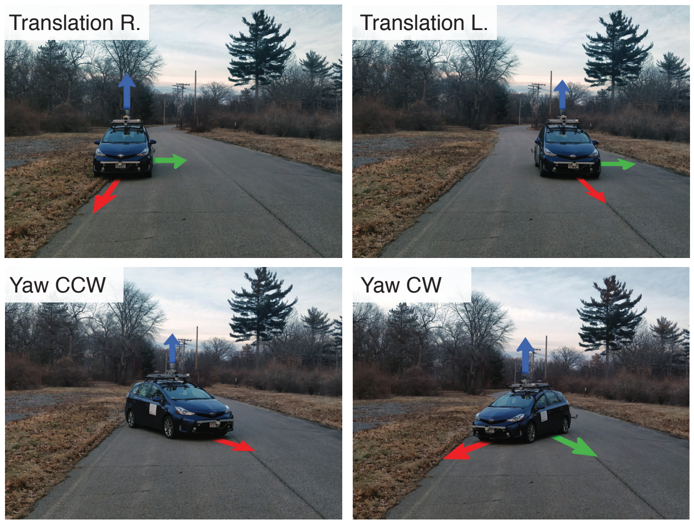


Figure 3-10: **Robustness analysis.** We test robustness to recover from near crash positions, including strong translations (top) and rotations (bottom). Each model and starting orientation is repeated at 15 locations on the test track. A recovery is successful if the car recovers within 5 seconds.

our observation from closed-loop testing in ***VISTA***, further motivating its efficacy for policy evaluation, considering the time and safety costs of real-world testing.

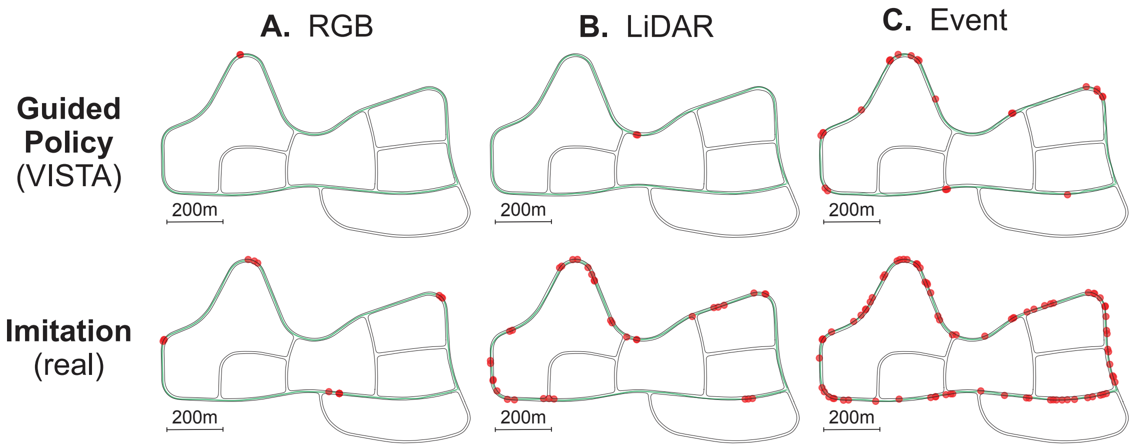


Figure 3-11: **Real-world deployment interventions.** Policy trajectories (n=3; 45km total) and crash locations (red dots) for RGB (A), LiDAR (B), and Event (C) sensors.

For event cameras, while the GPL policy also exhibits superior performance in terms of number of interventions, it deviates more from the center compared to IL policy and suffers from much frequent intervention compared to RGB and LiDAR

GPL policies. This is due to the fact that event cameras can only see the component of the road boundary non-parallel to the vehicle's ego motion, while RGB and LiDAR sensors provide sufficient information at every step for lane following. Such properties highlight the potential utility of fusing event sensing with RGB and LiDAR for greater benefits. We observed a swirling maneuver along straight roads with event policies (8x more jittery than RGB measured by squared second derivative of curvature). However, we found that our method of branched learning for preventing ego-motion conflating the scene understanding (Algorithm 1) is highly effective in real-world tests to reduce the number of interventions: 28.0 ± 3.6 (IL), 18.0 ± 1.0 (GPL), 6.5 ± 3.8 (GPL with branching). To further highlight the efficacy of GPL policies, we conduct a robustness test by initializing the car with $\pm 30°$ rotation and $\pm 2m$ translation from the lane center and measure the success rate of recovery, as shown in Table 3.2.

Recovery Rate	**RGB**	**LiDAR**	**Event**
IL	0.27 ± 0.13	0.16 ± 0.11	0.00 ± 0.00
GPL	$\mathbf{0.90 \pm 0.13}$	$\mathbf{0.83 \pm 0.22}$	$\mathbf{0.90 \pm 0.15}$

Table 3.2: **Robustness test.** GPL policies trained in ***VISTA*** significantly outperform real-world IL at recovering from edge cases.

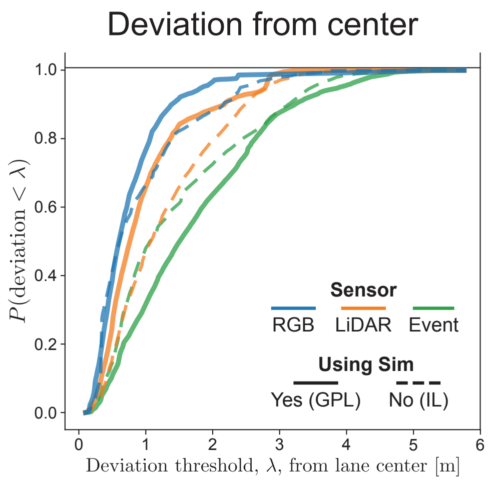


Figure 3-12: **Real-world deployment deviations with *VISTA*.** Cumulative distribution of deviations from the center shows the benefit of training in ***VISTA***.

3.5.4 Multi-agent Deployment Evaluation

To further verify the performance of end-to-end policies learned from ***VISTA***, we deploy the learned model onto our full-scale autonomous vehicle and evaluate performance in multi-agent obstacle avoidance and overtaking scenarios. We carry out 63 experimental trials with over 2 hours autonomously. In Table 3.3, we show overall performance for different multi-agent scenarios. The front car being dynamic poses a more challenging task in comparison to being static and results in more interventions. In comparison to offline tests, minimal clearance, maximal deviation, and yaw are larger, since the agent was placed in a more challenging setting by initializing at either extreme side of the road and starting the ego car immediately behind it. We also find the intervention frequency and minimal clearance to be biased towards overtaking from the right side. Our hypothesis is that the yaw difference between the camera and vehicle slightly drifts along time between the real car at test time and the camera calibration for the simulator. Even 1 degree of drift can cause ∼9cm lateral shift for objects 5m in the front of the camera.

Scenario		**Average Intervention ↓**	**Minimum Clearance ↑**	**Maximum Deviation ↓**	**Maximum Yaw ↓**
Front	Static	0 / 33 (0.00)	1.092	1.341	0.783
Car	Dynamic	3 / 30 (0.10)	1.063	1.453	0.332
Overtake	Left	0 / 32 (0.00)	1.410	1.961	0.573
From	Right	3 / 31 (0.09)	0.735	0.809	0.563

Table 3.3: **Closed-loop testing with multi-agent interactions.** Results of online (real-car) active tests in multi-agent object avoidance and overtaking scenarios.

We perform clearance analysis in Figure 3-13. We show the histogram of steps with clearance less than 2m across all episodes. The peak is at 1m, which is good for safe maneuvering. Observing the local maximum at very small clearance, we evaluate the normalized cumulative trials that recall different clearance thresholds. The small initial slope indicates that most low clearance steps are contributed by a very small proportion of trials. Besides, the largest slope occurs roughly close to 0.8m, indicating that most trials have clearance at such distance.

In Figure 3-14, we demonstrate qualitative results of multi-agent overtaking, with

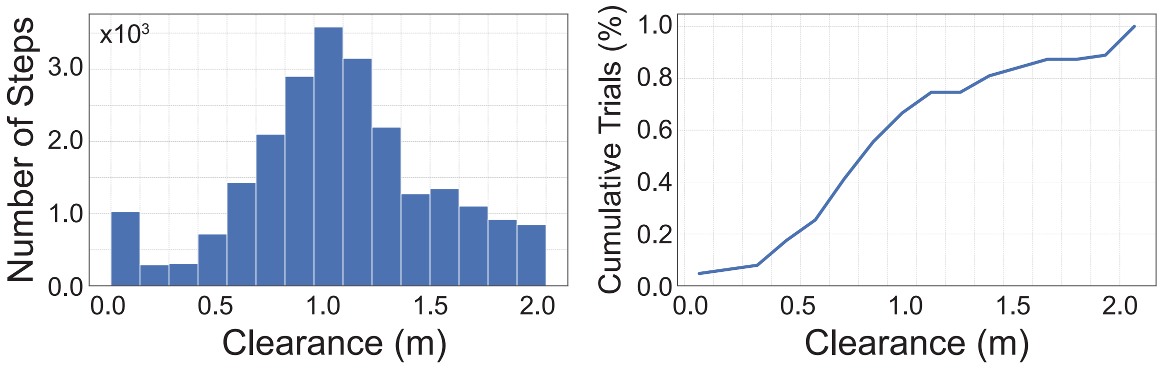


Figure 3-13: **Clearance analysis in real-world tests.** Histogram of clearance across *steps* (left). Recall of normalized cumulative *trials* at clearance (right).

two trials showing successful overtaking and one trial involving intervention. The first trial (first row) shows how the end-to-end learned policy can avoid the moving front car and recover to the lane. In the second row, we further showcase the capability of our policy to make a challenging turn, where the road curves toward the right while the ego car can only overtake from the left. In the last row, we conduct a case study involving intervention. At the first two timesteps $t1$ and $t2$, the policy did manage to avoid the front car, while at $t3$, it cut back too early since the front car is lost in the view and the speed of the ego car is too slow for the recurrent model to memorize there was a car some time ago in the left. Augmenting more cameras on the side of the autonomous vehicles may help address this issue.

3.6 Discussion

Simulation has emerged as a potential solution for training and evaluating autonomous systems on challenging situations that are often difficult to collect in the real world. However, successfully transferring learned policies from model-based simulation into the real-world has been a long-standing field in robot learning. In this chapter, we present ***VISTA***, an end-to-end data-driven simulator for training autonomous vehicles for deployment into the real-world. ***VISTA*** supports multimodal sensor synthesis, including 2D RGB cameras, 3D LiDAR, and event-based cameras, and multi-agent rendering for mobile agents. ***VISTA*** is entirely data-driven and can synthesize high-fidelity sensor measurement sufficient for policy learning and evaluation.

VISTA supports training agents anywhere within the feasible band of trajectories that can be synthesized from data collected by a human driver on a single trajectory. We showcase the sim-to-real ability by directly deploying policies learned in ***VISTA*** on a full-scale autonomous vehicle for each sensor and demonstrate consistent results between closed-loop evaluation in simulation and real-world tests as well as improved robustness in recovery from near-crash scenarios. We evaluate learned policies across several multi-agent tasks with increasing levels of complexity and conduct extensive empirical analyses in these multi-agent scenarios within simulation as well as the real world.

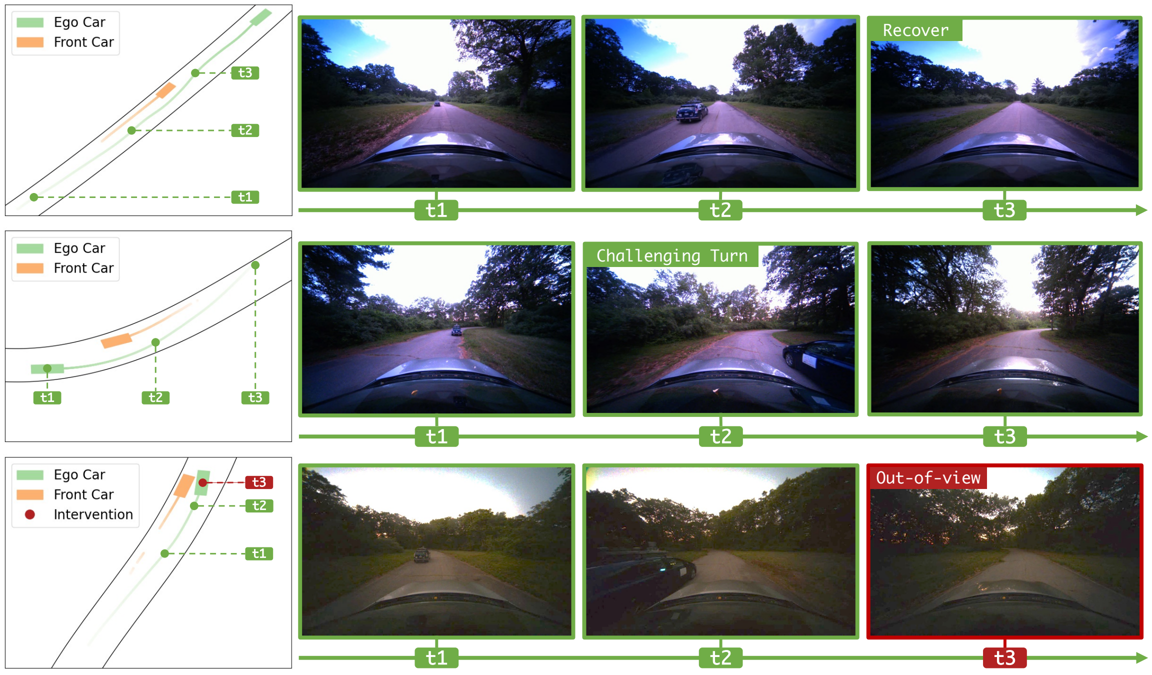


Figure 3-14: **Qualitative real-world inspection.** Rows correspond to individual trials from different locations on on the test track. GPS tracking data is visualized (left) along with timepoints tagged. For each timepoint, the corresponding image view is provided (right) along with a semantic description.

3.7 Scope and Limitations

VISTA is an open-source, data-driven platform for the generation of synthetic environments for training and evaluation of autonomous vehicle controllers. The modularity and scalability of ***VISTA*** provides a generalizable platform for simulation of embodied agent perception and closed-loop control learning directly from real-world data.

The work described in this chapter opens several opportunities for future work. Potential future directions include, but are not limited to, more complicated tasks that involve situational awareness from multi-agent interaction; edge-case generation by evaluation with adversaries (Lee et al., 2019b); and learning more complex policies with ambiguities such as point-to-point navigation (Codevilla et al., 2017; Amini et al., 2019a; Hawke et al., 2020). This work opens up a new avenue for leveraging high-fidelity data-driven simulation in robotics and represents a major step towards enabling the direct, real world deployment of end-to-end learning of autonomous vehicle controllers. Further, we believe the public release of ***VISTA*** will create new research opportunities for perception and control of autonomous vehicles.

To achieve robust data-driven autonomy and decision making, the fidelity of data we use to train and evaluate our systems is of critical importance. ***VISTA*** focuses on this key challenge of the pipeline. By synthesizing edge cases and challenging scenarios directly from real-world data, we bypass model-based sim-to-real limitations while achieving a significant improvement of robustness of the learned models during handling of these scenarios. However, it is insufficient to solely consider the data aspect of this problem and ignore other parts of the pipeline. Namely, how do the models, the inductive biases that we build into their architectures, as well as the learning processes themselves affect decision making ability in the wild? Autonomous decision making is a fundamentally dynamic problem, as agents live within responsive environments which continuously evolve as a function of the decisions they take and behaviors they execute. This requires the ability to build rich and expressive representations from such complex and dynamic data as well as new optimization approaches for learning models capable of building these representations.

In Chapter 4, we begin to explore this through the lens of developing liquid time constant neural units, expressive and causal neural models for continuous-time modeling, and scaling their learning ability to high complexity autonomous control tasks in Chapter 5.

Part II

Model

Chapter 4

Expressive Neuron Models for Continuous-time Decision Making

4.1 Introduction

The safety-critical nature of end-to-end control places the demand that learned control models are able to gracefully handle rare events, have *interpretable* dynamics, and generalize to unforeseen and uncertain scenarios. Chapter 3 described a data-centric method to begin to address these challenges: ***VISTA***, a data-driven engine for generation of synthetic, photorealistic driving environments. We additionally described how control policies could be trained end-to-end in ***VISTA***, entirely in simulation, and demonstrated their translation to real world settings.

The successful end-to-end neural network (NN) autonomous-control approaches to lane-keeping described in Chapter 3, and more generally (Bojarski et al., 2016; Xu et al., 2017; Amini et al., 2018a; Fridman et al., 2019) (Figure 4-1), rely solely on deep convolutional neural network architectures (LeCun et al., 1990), steering a vehicle at a time t, based on the most recent camera frame (Amini et al., 2019a) (Figure 4-2a). While such feedforward models can properly drive the vehicle in case of ideal input-data, they often fail if the data is noisy. This is because they do not exploit the temporal nature of the task, enabling them to filter out transient disturbances. As a result, temporary corruptions of the input stream (*i.e.*, sudden sunlight, as illustrated

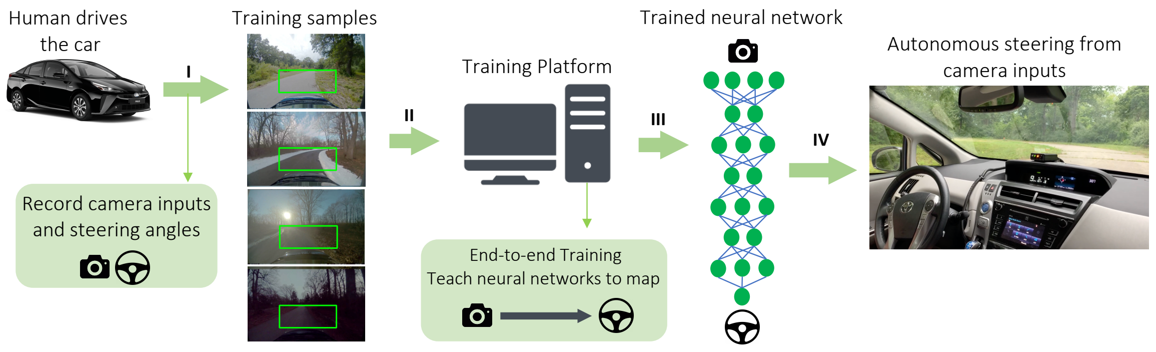

Human drives the car
I
Record camera inputs and steering angles
Training samples
II
Training Platform
End-to-end Training
Teach neural networks to map
III
Trained neural network
IV
Autonomous steering from camera inputs

Figure 4-1: **End-to-end driving.** The process starts by collecting a considerable amount of human driving experiences, in a car that is equipped with camera/s and in-car computing units. The diverse set of training samples are then edited (the green boxes) and are labeled by their corresponding steering angle. An end-to-end training algorithm trains and validates an artificial neural network agent, in a supervised learning fashion to directly turn camera inputs into steering decisions. The obtained network is then deployed on the high-performance computing units mounted inside the car to drive the car autonomously in real unseen environments.

in Figure 4-2a), lead to unstable predictions.

On the contrary, recurrent neural networks (RNNs) (Hochreiter, 1991; Bengio et al., 1994), are a class of artificial neural networks that take into account past observations at a current output-decision through a feedback mechanism. This "memory" is retained within an internal state of the RNN, commonly referred to as the "hidden state". Because autonomous control is fundamentally a *dynamic* process, these architectures hold the potential to define more robust end-to-end controllers (Figure 4-2b). In the discrete time setting, RNNs are trained over finite-length labeled training sequences by the backpropagation algorithm (Rumelhart et al., 1986) applied to their unfolded feed-forward representation (Bengio et al., 1994) (Figure 4-2c,d). These architectures have been extended to the continuous time setting, wherein the RNN hidden state is defined in continuous-time according to ordinary differential equations (ODEs).

One such derivative architecture is the neural ODE. The state of a neural ODE, $\mathbf{x}(t) \in \mathbb{R}^D$, is defined by the solution of this equation (Chen et al., 2018a): $d\mathbf{x}(t)/dt = f(\mathbf{x}(t), t, \theta)$, where the function f is a neural network parameterized by weights θ. One can then compute the state using a numerical ODE solver and train the network by

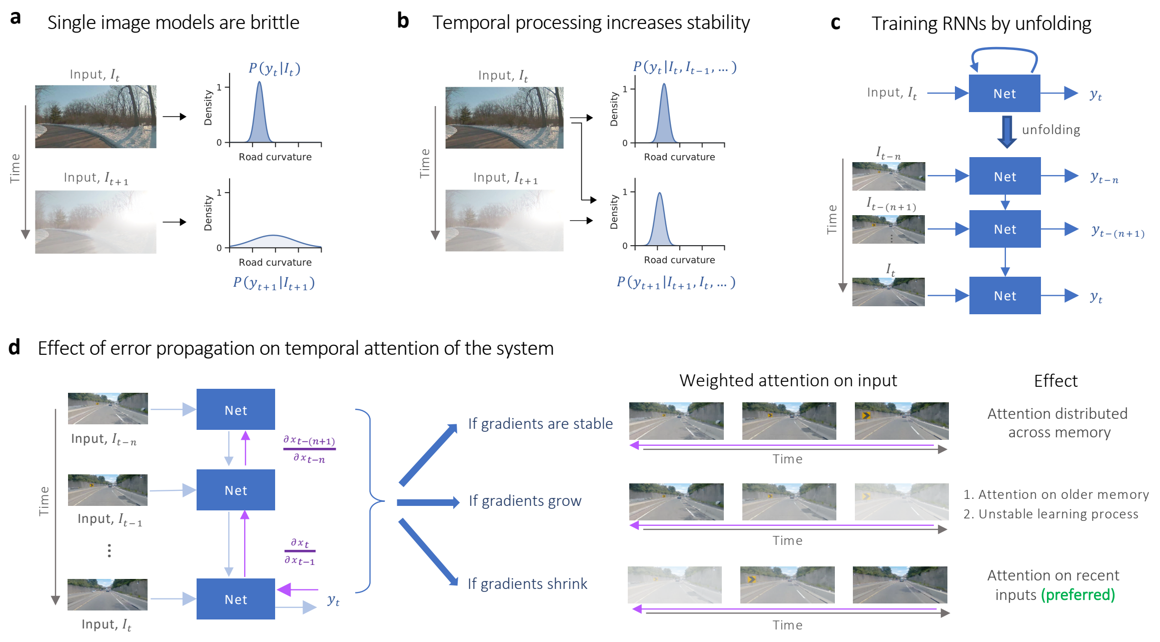

a Single image models are brittle
Input, I_t
$P(y_t|I_t)$
Density
Road curvature
Input, I_{t+1}
$P(y_{t+1}|I_{t+1})$
Time
b Temporal processing increases stability
$P(y_t|I_t, I_{t-1}, ...)$
$P(y_{t+1}|I_{t+1}, I_t, ...)$
c Training RNNs by unfolding
Net
unfolding
y_t
I_{t-n}
$I_{t-(n+1)}$
y_{t-n}
$y_{t-(n+1)}$
d Effect of error propagation on temporal attention of the system
Input, I_{t-n}
Input, I_{t-1}
$\frac{\partial x_{t-(n+1)}}{\partial x_{t-n}}$
$\frac{\partial x_t}{\partial x_{t-1}}$
If gradients are stable
If gradients grow
If gradients shrink
Weighted attention on input
Effect
Attention distributed across memory
1. Attention on older memory
2. Unstable learning process
Attention on recent inputs (preferred)

Figure 4-2: **Recurrent network modules are essential for the lane-keeping tasks. a,** A feedforward CNN network computes its output, $P(y_t|I_t)$ by relying solely on the current observation, I_t. Consequently, inputs that are corrupted by transient perturbations (bottom), will result in high output variance, and faulty decisions **b,** An RNN has access to past observations at a current driving step, enabling it to filter out transient corruptions that are present in the input stream **c,** Training RNNs by unrolling their state in time **d,** Then, applying back-propagation through time in an unfolded RNN. Purple derivatives indicate the dependency of the loss function's derivative with-respect-to an RNN's state-weights to the evolution of the RNN's state, $x(t)$ in time. Blurred images depict weaker attention of the RNN, when computing a current decision.

performing reverse-mode automatic differentiation (Rumelhart et al., 1986), either by gradient descent through the solver (Lechner et al., 2019) or by considering the solver as a black box (Chen et al., 2018a; Dupont et al., 2019; Gholami et al., 2019) and applying the adjoint method (Pontryagin, 2018).

Alternatively, rather than defining the derivatives of the hidden state directly by a neural network f, one can determine a more stable continuous-time recurrent neural network (CT-RNN) by the following equation (Funahashi and Nakamura, 1993): $\frac{d\mathbf{x}(t)}{dt} = -\frac{\mathbf{x}(t)}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)$, in which the term $-\frac{\mathbf{x}(t)}{\tau}$ assists the autonomous system to reach an equilibrium state with a time-constant τ. $\mathbf{x}(t)$ is the hidden state, $\mathbf{I}(t)$ is the input, t represents time, and f is parameterized by θ.

To achieve more expressive representation learning in the continuous-time setting for autonomous control, we propose an alternative formulation: the liquid time constant (LTC) neuron. In this chapter, we discuss the LTC neuron model, characterize its features, and demonstrate its benefits in time series modeling.

4.2 Liquid Time-constant (LTC) Neurons

First we formulate the state equation for the LTC neuron. Let the hidden state flow of a network be declared by a system of linear ODEs of the form: $d\mathbf{x}(t)/dt = -\mathbf{x}(t)/\tau + \mathbf{S}(t)$. Let $\mathbf{S}(t) \in \mathbb{R}^M$ represent the following nonlinearity determined by $\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$, with parameters θ and A. Then, by plugging in $\mathbf{S}$ into the hidden state equation, we obtain:

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A. \tag{4.1}$$

Equation 4.1 manifests a novel time-continuous unit instance, which we term the liquid time constant (LTC) neuron. In this section, we discuss the following properties of LTC neurons:

1. **The Liquid Time Constant:** In LTC neurons, a neural network f not only determines the derivative of the hidden state $\mathbf{x}(t)$, but also serves as an input-

dependent varying time-constant for the learning system:

$$\tau_{sys} = \frac{\tau}{1 + \tau f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)}$$

This property enables single elements of the hidden state to identify specialized dynamical systems for input features arriving at each time-point. In Section 4.2.1, we introduce an ODE solver for implementing LTCs.

2. **Reverse-Mode Automatic Differentiation:** LTCs realize differentiable computational graphs and can be trained via gradient-based optimization. We use a vanilla backpropagation through time (BPTT) algorithm to optimize LTCs and motivate this choice in Section 4.2.2.

3. **Bounded Dynamics and Stability:** In Section 4.2.3, we show that the state and the time-constant of LTCs are bounded to a finite range. This property assures the stability of the output dynamics and is desirable when inputs to the system relentlessly increase.

4. **Expressivity:** In Section 4.2.4, we analyze the approximation capability, universality, and expressivity of LTCs. Specifically we consider the trajectory length (Raghu et al., 2017) of network activations in a latent trajectory representation and benchmark to other time-continuous models.

5. **Causal Structure:** In Section 4.3, we show how LTCs form a causal structure, specifically as a dynamic causal model, and discuss the real-world implications of this for learning control policies for autonomous flight.

4.2.1 LTCs Forward Pass by a Fused ODE Solver

Here we provide an approach for the forward pass through LTC neurons, which are defined by the state equation provided in Equation 4.1. Solving Equation 4.1 analytically is non-trivial due to the nonlinearity of the LTC semantics. The state of the system of ODEs, however, at any time point T, can be computed by a numerical

ODE solver that simulates the system starting from a trajectory $x(0)$, to $x(T)$. An ODE solver breaks down the continuous simulation interval $[0, T]$ to a temporal discretization, $[t_0, t_1, \ldots t_n]$. As a result, a solver's step involves only the update of the neuronal states from t_i to t_{i+1}.

The ODE of LTCs realizes a system of stiff equations (Press et al., 2007). This type of ODE requires an exponential number of discretization steps when simulated with a Runge-Kutta (RK) based integrator. Consequently, ODE solvers based on RK, such as Dormand–Prince (default in torchdiffeq (Chen et al., 2018a)), are not suitable for LTCs. Therefore, we design a new ODE solver that fuses the explicit and implicit Euler methods. Our discretization method results in greater stability and numerically unrolls a given dynamical system of the form $dx/dt = f(x)$ by:

$$x(t_{i+1}) = x(t_i) + \Delta t f(x(t_i), x(t_{i+1})). \tag{4.2}$$

In particular, we replace only the $x(t_i)$ that occur linearly in f by $x(t_{i+1})$. As a result, Equation 4.2 can be solved for $x(t_{i+1})$ symbolically. Applying the fused solver to the LTC representation and solving it for $\mathbf{x}(t + \Delta t)$, we get:

$$\mathbf{x}(t + \Delta t) = \frac{\mathbf{x}(t) \;+\; \Delta t f(\mathbf{x}(t), \mathbf{I}(t), t, \theta) A}{1 + \Delta t\big(1/\tau + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\big)}. \tag{4.3}$$

Equation 4.3 computes one update state for an LTC network.

Correspondingly, Algorithm 3 shows how to implement an LTC network, given a parameter space θ. f is assumed to have an arbitrary activation function (*e.g.* for a *tanh* nonlinearity $f = \tanh(\gamma_r \mathbf{x} + \gamma \mathbf{I} + \mu)$). The computational complexity of the algorithm for an input sequence of length T is $O(L \times T)$, where L is the number of discretization steps. Intuitively, a dense version of an LTC network with N neurons and a dense version of a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network with N cells would be of the same complexity.

Algorithm 3 LTC update by fused ODE Solver

```
Parameters: $\theta = \{$
    $\tau^{(N\times 1)}$ = time-constant
    $\gamma^{(M\times N)}$ = weights
    $\gamma_r^{(N\times N)}$ = recurrent weights
    $\mu^{(N\times 1)}$ = biases
    $A^{(N\times 1)}$ = bias vector
$\}$
Let: $L$ = # of unfolding steps, $\Delta t$ = step size, $N$ = # of
neurons
Inputs: $M$-dimensional Input $\mathbf{I}(t)$ of length $T$, $\mathbf{x}(0)$
Output: Next LTC neural state $\mathbf{x}_{t+\Delta t}$
Function: FusedStep($\mathbf{x}(t)$, $\mathbf{I}(t)$, $\Delta t$, $\theta$)
$\mathbf{x}(t+\Delta t)^{(N\times T)} = \frac{\mathbf{x}(t)\ +\ \Delta t f(\mathbf{x}(t),\mathbf{I}(t),t,\theta)\odot A}{1+\Delta t\big(1/\tau+f(\mathbf{x}(t),\mathbf{I}(t),t,\theta)\big)}$
▷ $f(.)$, and all divisions are applied element-wise.
▷ $\odot$ is the Hadamard product.
end Function
$\mathbf{x}_{t+\Delta t} = \mathbf{x}(t)$
for $i = 1 \dots L$ do
    $\mathbf{x}_{t+\Delta t}$ = FusedStep($\mathbf{x}(t)$, $\mathbf{I}(t)$, $\Delta t$, $\theta$)
end for
return: $\mathbf{x}_{t+\Delta t}$
```

4.2.2 Training LTC Networks by BPTT

Neural ODEs were suggested to be trained by a constant memory cost for each layer in a neural network f by applying the adjoint sensitivity method to perform reverse-mode automatic differentiation (Chen et al., 2018a). The adjoint method, however, comes with numerical errors when running in reverse mode. This phenomenon happens because the adjoint method forgets the forward-time computational trajectories, which was repeatedly denoted by the community (Gholami et al., 2019; Zhuang et al., 2020).

On the contrary, direct backpropagation through time (BPTT) trades memory for accurate recovery of the forward-pass during the reverse mode integration (Zhuang et al., 2020). Thus, we set out to design a vanilla BPTT algorithm to maintain a highly accurate backward-pass integration through the solver. For this purpose, a

	Vanilla BPTT	**Adjoint**
Time	$O(L \times T \times 2)$	$O((L_f + L_b) \times T)$
Memory	$O(L \times T)$	$\mathbf{O(1)}$
Depth	$O(L)$	$O(L_b)$
FWD acc	High	High
BWD acc	**High**	Low

Table 4.1: **Complexity of the vanilla BPTT compared to the adjoint method**, for a single layer neural network f. Note: L = number of discretization steps, L_f = L during forward-pass. L_b = L during backward-pass. T = length of sequence, Depth = computational graph depth.

given ODE solver's output (a vector of neural states), can be recursively folded to build an RNN and then apply Algorithm 4 to train the system. Algorithm 4 uses a vanilla stochastic gradient descent (SGD). One can substitute this with a more performant variant of the SGD, such as Adam (Kingma and Ba, 2014), which we use in our experiments.

Algorithm 4 Training LTC by BPTT

```
Parameter: Loss func $L(\theta)$, initial param $\theta_0$, learning rate $\alpha$, Output w = $W_{out}$, and bias = $b_{out}$
Inputs: Dataset of traces $[I(t), y(t)]$ of length $T$, RNNcell = $f(I, x)$
for $i = 1 \dots$ number of training steps do
    $(I_b, y_b)$ = Sample training batch,    $x := x_{t_0} \sim p(x_{t_0})$
    for $j = 1 \dots T$ do
        $x = f(I(t), x)$,   $\hat{y}(t) = W_{out}.x + b_{out}$,   $L_{total} = \sum_{j=1}^{T} L(y_j(t), \hat{y}_j(t))$,   $\nabla L(\theta) = \frac{\partial L_{tot}}{\partial \theta}$
        $\theta = \theta - \alpha \nabla L(\theta)$
    end for
end for
return: $\mathbf{x}_{t+\Delta t}$
```

Table 4.1 summarizes the complexity of our vanilla BPTT algorithm compared to an adjoint method. We achieve a high degree of accuracy on both forward and backward integration trajectories, with similar computational complexity, at large memory costs.

4.2.3 Bounds on τ and the Neural State of LTCs

LTCs are represented by an ODE which varies its time constant based on inputs. It is therefore important to see if LTCs stay stable for unbounded arriving inputs (Hasani et al., 2019; Lechner et al., 2020b). The time constant and state of LTC neurons are bounded to a finite range, as described in Theorems 1 and 2, respectively.

Theorem 1. *Let x_i denote the state of a neuron i within an LTC network identified by Equation 4.1, and let neuron i receive M incoming connections. Then, the time-constant of the neuron, τ_{sys_i}, is bounded to the following range:*

$$\tau_i/(1+\tau_i W_i) \leq \tau_{sys_i} \leq \tau_i. \tag{4.4}$$

A stable varying time-constant significantly enhances the expressivity of this form of time-continuous RNNs, as we discover more formally in Section 4.2.4.

Theorem 2. *Let x_i denote the state of a neuron i within an LTC, identified by Equation 4.1, and let neuron i receive M incoming connections. Then, the hidden state of any neuron i, on a finite interval $Int \in [0, T]$, is bounded as follows:*

$$min(0, A_i^{min}) \leq x_i(t) \leq max(0, A_i^{max}), \tag{4.5}$$

Theorem 2 illustrates a desired property of LTCs, namely *state stability* which guarantees that the outputs of LTCs never explode even if their inputs grow to infinity.

4.2.4 On the Expressive Power of LTCs

Understanding the impact of a NN's structural properties on their computable functions is known as the expressivity problem. The very early attempts on measuring expressivity of NNs include theoretical studies based on functional analysis. They show that NNs with three layers can approximate any finite set of continuous mapping with any precision. This is known as the *universal approximation theorem* (Hornik et al., 1989; Funahashi, 1989; Cybenko, 1989). Universality was extended to standard

RNNs (Funahashi, 1989) and even continuous-time RNNs (Funahashi and Nakamura, 1993). By careful considerations, we can also show that LTCs are also universal approximators.

Theorem 3. *Let $\boldsymbol{x} \in \mathbb{R}^n$, $S \subset \mathbb{R}^n$ and $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ be an autonomous ODE with $F : S \rightarrow \mathbb{R}^n$ a C^1-mapping on S. Let D denote a compact subset of S and assume that the simulation of the system is bounded in the interval $I = [0,\ T]$. Then, for a positive ϵ, there exist an LTC network with N hidden units, n output units, and an output internal state $\mathbf{u}(t)$, described by Equation 4.1, such that for any rollout $\{\boldsymbol{x}(t)|t \in I\}$ of the system with initial value $x(0) \in D$, and a proper network initialization,*

$$max_{t\in I}|\boldsymbol{x}(t) - \boldsymbol{u}(t)| < \epsilon \tag{4.6}$$

The proof defines an n-dimensional dynamical system and place it into a higher dimensional system. The second system is an LTC. The fundamental difference of the proof of LTC universality to that of CT-RNNs (Funahashi and Nakamura, 1993) lies in the distinction of the semantics of both systems where the LTC network contains a nonlinear input-dependent term in its time-constant module which makes parts of the proof non-trivial.

The universal approximation theorem broadly explores the expressive power of a neural network. However, the theorem does not yield a concrete measure on where the separation is between different neural network architectures. Therefore, a more rigorous measure of expressivity is demanded to compare models, specifically those networks specialized in spatiotemporal data processing, such as LTCs. The advances made on defining measures for the expressivity of static deep learning models (Pascanu et al., 2013b; Montufar et al., 2014; Eldan and Shamir, 2016; Poole et al., 2016; Raghu et al., 2017) could help measure the expressivity of time-continuous models, both theoretically and quantitatively, which we explore in the next section.

Measuring Expressivity by Trajectory Length

A measure of expressivity has to take into account what degrees of complexity a learning system can compute, given the network's capacity (depth, width, type, and weights configuration). A unifying expressivity measure of static deep networks is the *trajectory length* introduced in (Raghu et al., 2017). In this context, one evaluates how a deep model transforms a given input trajectory (e.g., a circular 2-dimensional input) into a more complex pattern, progressively.

We can then perform principle component analysis (PCA) over the obtained network's activations. Subsequently, we measure the length of the output trajectory in a 2-dimensional latent space to uncover its relative complexity (Figure 4-3). The trajectory length is defined as the *arc length* of a given trajectory $I(t)$, (*e.g.* a circle in 2D space) (Raghu et al., 2017): $l(I(t)) = \int_t \|dI(t)/dt\| \, dt$. By establishing a lower-bound for the growth of the trajectory length, one can set a barrier between networks of shallow and deep architectures, regardless of any assumptions on the network's weight configuration (Raghu et al., 2017), unlike many other measures of expressivity (Pascanu et al., 2013b; Montufar et al., 2014; Serra et al., 2017; Gabrié et al., 2018; Hanin and Rolnick, 2018, 2019; Lee et al., 2019a).

We set out to extend the trajectory-space analysis of static networks to time-continuous (TC) models, and to lower-bound the trajectory length to compare models' expressivity. To this end, we designed instances of Neural ODEs, CT-RNNs, and LTCs with shared f. The networks were initialized by weights $\sim \mathcal{N}(0, \sigma_w^2/k)$, and biases $\sim \mathcal{N}(0, \sigma_b^2)$. We then perform forward-pass simulations by using different types of ODE solvers, for arbitrary weight profiles, while exposing the networks to a circular input trajectory $I(t) = \{I_1(t) = \sin(t), I_2(t) = \cos(t)\}$, for $t \in [0, 2\pi]$. By looking at the first two principle components (with an average variance-explained of over 80%) of hidden layers' activations, we observed consistently more complex trajectories for LTCs.

Figure 4-4 gives a glimpse of our empirical observations. All networks are implemented by the Dormand-Prince explicit Runge-Kutta(4,5) solver (Dormand and

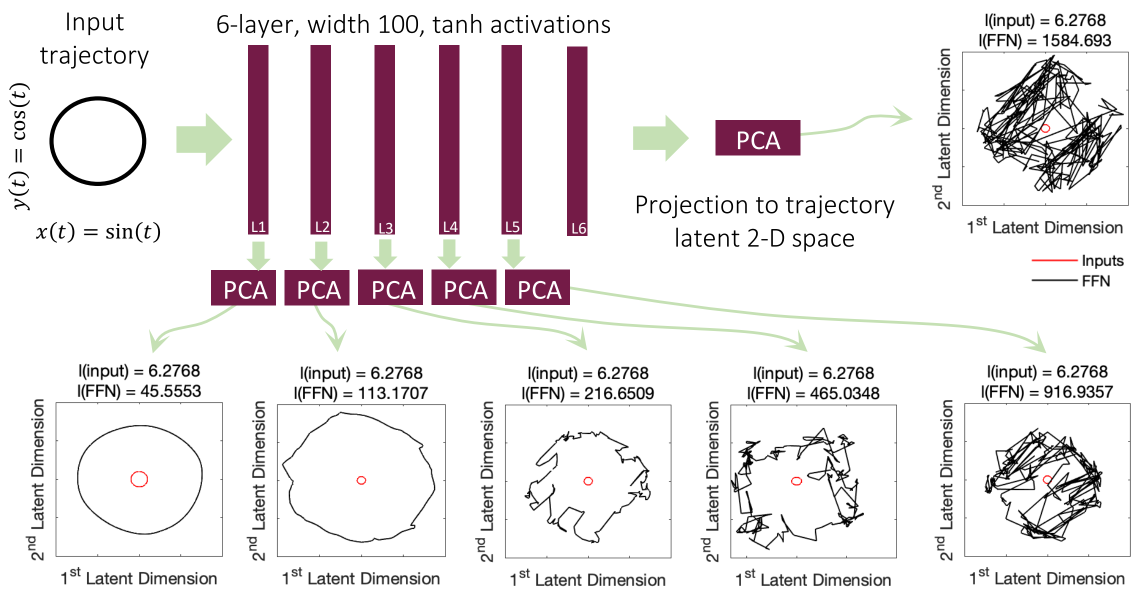


Figure 4-3: **LTC trajectory length as a measure of model expressivity.** Trajectory's latent space becomes more complex as the input passes through hidden layers.

Prince, 1980) with a variable step size. We observe the following:

1. Exponential growth of the trajectory length of Neural ODEs and CT-RNNs with `Hard-tanh` and `ReLU` activations (Figure 4-4A) and unchanged shape of their latent space regardless of their weight profile.

2. LTCs show a slower growth-rate of the trajectory length when designed by `Hard-tanh` and `ReLU`, with the compromise of realizing great levels of complexity (Figure 4-4A, C, and D).

3. Apart from multi-layer time-continuous models built by `Hard-tanh` and `ReLU` activations, in all cases, we observed a longer and a more complex latent space behavior for the LTC networks (Figure 4-4B-D).

4. Unlike static deep networks (Figure 4-3), we witnessed that the trajectory length does not grow by depth in multi-layer continuous-time networks realized by `tanh` and `sigmoid` (Figure 4-4E).

Conclusively, we observed that the trajectory length in TC models varies by a model's activations, the variance of the weight and bias distributions, the network

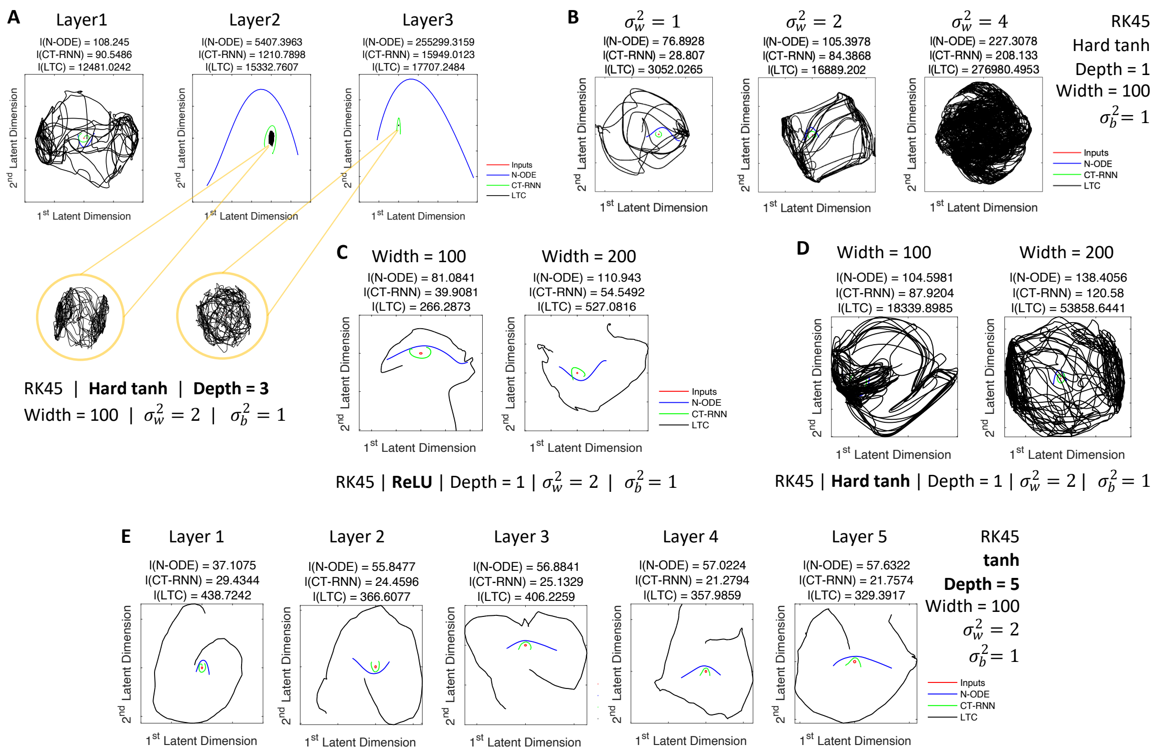


Figure 4-4: **Trajectory length deformation** A) in network layers with `Hard-tanh` activations, B) as a function of the weight distribution scaling factor, C) as a function of network width (`ReLU`), D) as a function of width (`Hard-tanh`), and E)in network layers with `logistic-sigmoid` activations.

width, and network depth. We presented this more systematically in Figure 4-5. Specifically, we observe:

1. Trajectory length is reluctant to the choice of ODE solver (Figure 4-5A).

2. Trajectory length grows linearly with a network's width (Figure 4-5B). We note the logarithmic growth of the curves in the log-scale Y-axis.

3. The growth is considerably faster as the variance of the weight distribution grows (Figure 4-5C).

4. Trajectory length is reluctant to the layer identifier (Figure 4-5D).

Finally, across Figure 4-4 and Figure 4-5, we note that activation functions diversify the complex patterns explored by the TC system, where `ReLU` and `Hard-tanh`

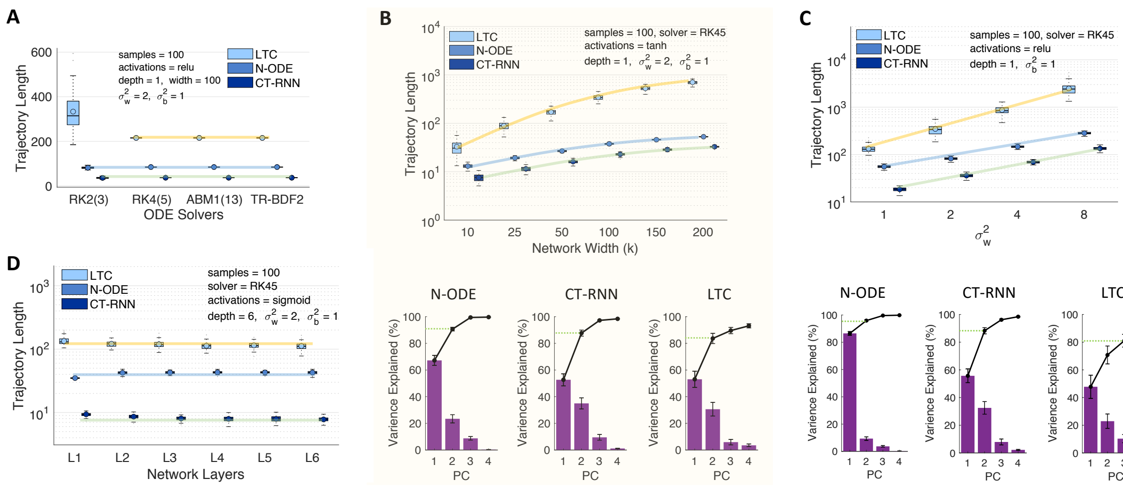


Figure 4-5: **Dependencies of the trajectory length measure.** A) trajectory length vs different solvers (variable-step solvers). RK2(3): Bogacki-Shampine Runge-Kutta (2,3) (Bogacki and Shampine, 1989). RK4(5): Dormand-Prince explicit RK (4,5) (Dormand and Prince, 1980). ABM1(13): Adams-Bashforth-Moulton (Shampine, 1975). TR-BDF2: implicit RK solver with 1st stage trapezoidal rule and a 2nd stage backward differentiation (Hosea and Shampine, 1996). B) Top: trajectory length vs network width. Bottom: Variance-explained of principal components (purple bars) and their cumulative values (solid black line). C) Trajectory length vs variance of the weights distribution. D) Trajectory length vs layers.

networks demonstrate higher degrees of complexity for LTCs. A key reason is the presence of recurrent links between each layer's cells.

Theoretical Bounds on the Trajectory Length

To formulate bounds on the trajectory length growth of CT networks, we first define the computational depth L. For one hidden layer of f in a CT network, L is the average number of integration steps by the solver for each incoming input sample. Note that for an f with n layers we define the total depth as $n \times L$. These observations allow us to formulate lower bounds on the trajectory length growth of CT networks.

Theorem 4. Trajectory Length Growth Bounds for Neural ODEs and CT-RNNs. *Let $dx/dt = f_{n,k}(\boldsymbol{x}(t), \boldsymbol{I}(t), \theta)$ with $\theta = \{W, b\}$, represent a Neural ODE and $\frac{d\boldsymbol{x}(t)}{dt} = -\frac{\boldsymbol{x}(t)}{\tau} + f_{n,k}(\boldsymbol{x}(t), \boldsymbol{I}(t), \theta)$ with $\theta = \{W, b, \tau\}$ a CT-RNN. f is randomly weighted with* `Hard-tanh` *activations. Let $\boldsymbol{I}(t)$ be a 2D input trajectory, with its progressive points (i.e. $I(t + \delta t)$) having a perpendicular component to $\boldsymbol{I}(t)$ for all δt, with $L =$ number* *of solver-steps. Then, by defining the projection of the first two principle components' scores of the hidden states over each other, as the 2D* latent trajectory space *of a layer d, $z^{(d)}(\boldsymbol{I}(t)) = z^{(d)}(t)$, for Neural ODE and CT-RNNs respectively, we have:*

$$\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^{d\times L} l(I(t)), \tag{4.7}$$

$$\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{(\sigma_w-\sigma_b)\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^{d\times L} l(I(t)). \tag{4.8}$$

Next, we define a lower bound for the LTC networks based on trajectory length.

Theorem 5. Growth Rate of LTC Trajectory Length. *Let Equation 4.1 determine an LTC with $\theta = \{W, b, \tau, A\}$. With the same conditions on f and $I(t)$, as in Theorem 4, we have:*

$$\begin{aligned}\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\Bigg(\Big(&\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\Big)^{d\times L}\times \\ &\Big(\sigma_w + \frac{\left\|z^{(d)}\right\|}{\min(\delta t, L)}\Big)\Bigg) l(I(t)).\end{aligned} \tag{4.9}$$

As expected, the bound for the Neural ODEs is very similar to that of an n layer static deep network with the exception of the exponential dependencies to the number of solver-steps, L. The bound for CT-RNNs suggests their shorter trajectory length compared to neural ODEs, according to the base of the exponent. This results consistently matches our experiments presented in Figure 4-4 and Figure 4-5.

Figure 4-4B and Figure 4-5C show a faster-than-linear growth for LTC's trajectory length as a function of weight distribution variance. This is confirmed by LTC's lower bound shown in Equation 4.9. The lower bound of LTCs also depicts the linear growth of the trajectory length with the width, k, which validates the results presented in Figure 4-5B. **V)** Given the computational depth of the models L in Table 4.2 for `Hard-tanh` activations, the computed lower bound for neural ODEs, CT-RNNs, and LTCs justify a longer trajectory length of LTC networks in subsequent validation experiments.

	Computational Depth		
Activations	Neural ODE	CT-RNN	**LTC**
`tanh`	0.56 ± 0.016	4.13 ± 2.19	9.19 ± 2.92
`sigmoid`	0.56 ± 0.00	5.33 ± 3.76	7.00 ± 5.36
`ReLU`	1.29 ± 0.10	4.31 ± 2.05	56.9 ± 9.03
`Hard-tanh`	0.61 ± 0.02	4.05 ± 2.17	81.01 ± 10.05

Table 4.2: **Computational depth of models.** Note: # of tries = 100, input samples' $\Delta t = 0.01$, $T = 100$ sequence length. # of layers = 1, width = 100, $\sigma_w^2 = 2$, $\sigma_b^2 = 1$.

4.3 Causal Modeling with LTCs

Here we discuss the causal structure of LTCs, beginning by describing the concepts necessary to show that LTCs form causal models. In a structural causal model (SCM), given a set of observable random variables $X_1, X_2, \ldots, X_n$, as vertices of a directed acyclic graph (DAG), we can compute each variable from the following assignment (Schölkopf, 2019):

$$X_i := f_i(\mathbf{PA}_i, U_i), \quad i = 1, \ldots, n. \tag{4.10}$$

Here, f_i is a deterministic function of the parents of the event, X_i, in the graph ($\mathbf{PA}_i$) and of the stochastic variable, U_i. One can intuitively think of the causal structure framework as a function estimation problem rather than in terms of probability distributions (Spirtes et al., 2000). Direct causation is implied by direct edges in the graph through the assignment described in Equation 4.10. The stochastic variables $U_1, \ldots, U_n$ ensure that a joint distribution $P(X_i|\mathbf{PA}_i)$ is constructed as a general objective (Pearl, 2014; Schölkopf, 2019).

The SCM framework enables us to explore through the known physical mechanisms and functions to build flexible probabilistic models with *interventions*, replacing the epistemic probabilities, $P(X_i, \mathbf{PA}_i)$ (Pearl, 2009; Schölkopf, 2019). *Interventions* can be formalized by the SCM framework as operations that alter a subset of properties of Equation 4.10. For instance, modifying U_i, or replacing f_i (and as a result X_i)

(Karimi et al., 2020), constitutes an intervention. Moreover, by assuming joint independence of U_is, we can construct *causal* conditionals known as causal (disentangled) factorization, as follows (Schölkopf, 2019):

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|\mathbf{PA}_i). \tag{4.11}$$

Equation 4.11 stands for the causal mechanisms by which we can model all statistical dependencies of given observables. Accordingly, causal learning involves the identification of the causal conditionals, the function f_i, and the distribution of U_is in assignment Equation 4.10.

4.3.1 Modeling Causal Structures

Physical dynamics can be modeled by a set of differential equations (DEs). DEs allow us to predict the future evolution of a dynamical system and describe its behavior as a result of interventions. Their coupled time-evolution enables us to define averaging mechanisms for computing statistical dependencies (Peters et al., 2017). A system of differential equations enhances our understanding of the underlying physical phenomenon, explains its behavior, and dissects its causal structure.

For instance, consider the following system of DEs: $\frac{d\mathbf{x}}{dt} = g(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d$, with initial values at x_0, where g is a nonlinear function. The Picard-Lindelöf theorem (Nevanlinna, 1989) states that a differential equation of the form above would have a unique solution as long as g is Lipschitz. Therefore, if we unroll the system to infinitesimal differentials using the explicit Euler method, we get: $\mathbf{x}(t + \delta t) = \mathbf{x}(t) + dtf(\mathbf{x})$. This representation under the uniqueness condition shows that the near future events of x are predicted using its past information, thus forming a causal structure.

Thus, a DE system is a causal structure that allows us to process the effect of interventions on the system. On the other side of the spectrum of causal modeling (Peters et al., 2017), pure statistical models allow us to learn structures from data with little insight about causation and associations between epiphenomena. Since causality aims to bridge this gap, we propose to construct causal models with continuous-time neural networks.

We first explain how a continuous-time model identified by a standalone neural ODE with state $\frac{d\mathbf{x}}{dt} = f(\mathbf{x}(t), t, \theta)$, where the function f is a neural network parameterized by weights θ, cannot satisfy the causal structure properties even under Lipschitzness of its network (Peters et al., 2017). We then show that the class of liquid time-constant (LTC) networks can achieve causal modeling.

4.3.2 Dynamic Causal Models

Neural ODEs are not causal models.

Let f be the nonlinearity of a continuous-time neural network. Then the learning system defined by the following neural ODE state equation, $\frac{d\mathbf{x}}{dt} = f(\mathbf{x}(t), t, \theta)$, where the function f is a neural network parameterized by weights θ, cannot account for interventions (change of the environment conditions), and therefore does not form a causal structure even if f is Lipschitz-continuous.

To describe this in detail, we unfold the ODE by infinitesimal differentials as follows:

$$\mathbf{x}(t + \delta t) = \mathbf{x}(t) + dt f(\mathbf{x}, \theta). \tag{4.12}$$

If f is Lipschitz continuous, based on Picard-Lindelöf's existence theorem, the trajectories of this ODE system are unique and thus invertible. This means that, at least locally, the future events of the system can be predicted by its past values. Since the transformation is invertible, this setting is also true if we run the ODE backward in time (*i.e.* during the training process).

When the ODE system is trained by maximum likelihood estimation, given an initial weight distribution, the statistical dependencies between the system's variables might emerge from data. The resulting statistical model can predict in i.i.d. setting and learn from data, but it cannot predict under distribution shift or implicit/explicit interventions. Furthermore, the system cannot answer counterfactual questions (Mooij et al., 2013).[1] Although Neural ODEs in their generic representation

[1] A counterfactual question describe a causal relationship of the form: "If X had not occurred, Y

cannot form causal models, their other forms can help design a causal model.

LTCs are dynamic causal models.

LTCs described by Equation 4.1 resemble the representation of a simple Dynamic Causal Model (DCM) (Friston et al., 2003) with a bilinear Taylor approximation (Penny et al., 2005). DCMs aim to extracting the causal architecture of a given dynamical systems. DCMs represent the dynamics of the hidden nodes of a graphical model by ODEs, and, unlike Bayesian neural networks, allow for feedback connectivity structures (Friston et al., 2003). A simple DCM can be designed by a second-order approximation (bilinear) of a given system such as $d\mathbf{x}/dt = F(\mathbf{x}(t), \mathbf{I}(t), \theta)$, as follows (Friston et al., 2003):

$$d\mathbf{x}/dt = (A + \mathbf{I}(t)B)\mathbf{x}(t) + C\mathbf{I}(t) \tag{4.13}$$

$$A = \frac{\partial F}{\partial \mathbf{x}(t)}\Big|_{I=0}, \quad B = \frac{\partial^2 F}{\partial \mathbf{x}(t)\partial \mathbf{I}(t)}, \quad C = \frac{\partial F}{\partial \mathbf{I}(t)}\Big|_{x=0}, \tag{4.14}$$

where $\mathbf{I}(t)$ is the inputs to the system, and $\mathbf{x}(t)$ is the nodes' hidden states.

A fundamental property of a DCM representation is its ability to capture both internal and external causes on the dynamical system (interventions). Matrix A stands for a fixed internal coupling of the system. Matrix B controls the impact of the inputs on the coupling sensitivity among the network's nodes (controlling internal interventions). Matrix C embodies the external inputs' influence on the state of the system (controlling external interventions).

DCMs can be extended to a universal framework for causal function approximation by neural networks through the LTC neural representation (Hasani et al., 2020a):

Proposition 1. *Let f be the nonlinearity of a given LTC network identified by Equation 4.1. Then the learning system identified by Equation 4.1 can account for internal and external interventions by its weight parameters $\theta = \{W_r^{(D\times D)}, W^{(D\times m)}, b^{(D\times 1)}\}$, for D LTC cells, and input size m and $A^{(D\times 1)}$, and therefore, forms a dynamical causal model, if f is Lipschitz-continuous.*

would not have occurred (Molnar, 2020)"

Proof. To prove this, we need to show two properties of LTC networks: 1) uniqueness of their solution, and 2) existence of intervention coefficients.

Uniqueness. By using the Picard-Lindelöf theorem (Nevanlinna, 1989), it was previously shown that for an $F : \mathbb{R}^D \to \mathbb{R}^D$, bounded C^1-mapping, the differential equation:

$$\dot{x} = -(1/\tau + F(x))x + AF(x), \tag{4.15}$$

has a unique solution on $[0, \infty)$.

Intervention coefficients. Let f be a Lipschitz-continuous activation function such as tanh, then $f(\mathbf{x}(t), \mathbf{I}(t), t, \theta) = tanh(W_r\mathbf{x} + W\mathbf{I} + b)$. If we set $x = 0$, in Equation 4.1, then the external intervention coefficients C in Equation 4.13 for LTCs can be obtained by:

$$\frac{\partial F}{\partial \mathbf{I}}\bigg|_{x=0} = W(1 - f^2) \odot A \tag{4.16}$$

The corresponding internal intervention coefficients B of Equation 4.13, for LTC networks becomes:

$$\frac{\partial^2 F}{\partial \mathbf{x}(t) \partial \mathbf{I}(t)} = W(f^2 - 1) \odot \big[2W_r f \odot (A - x) + 1\big] \tag{4.17}$$

This shows that by manipulating matrices W, W_r, and A one can control internal and external interventions to an LTC system which gives the statement of the proposition.

□

Proposition 1 shows that LTCs are causal models in their forward pass, and by manipulation of their parameters, one can gain insights into the underlying systems. We next show that the LTC backward pass also gives rise to a causal model. A core finding is that LTCs trained from demonstration via reverse-mode automatic differentiation (Rumelhart et al., 1986) can give rise to a causal model.

4.3.3 Training LTCs to Yield Causal Models

The uniqueness of the solution of LTCs allows any ODE solver to reproduce a forward pass trajectory with backward computations from the end point of the forward pass. This property enables us to use the backpropagation algorithm, either by using the adjoint sensitivity (Pontryagin, 2018) method or backpropagation through time (BPTT), to train an LTC system. Formally, for a forward pass trajectory of the system states $\mathbf{x}(t)$ of an LTC network from t_0 to t_n, we can compute a scalar value loss, L at t_n by:

$$L(\mathbf{x}(t_n)) = L\Big(\mathbf{x}(t_0) + \int_{t_0}^{t_n} \frac{d\mathbf{x}}{dt} dt\Big). \tag{4.18}$$

Suppose we use the adjoint method for training the network, then the augmented state is computed by $a(t) = \frac{\partial L}{\partial \mathbf{x}(t)}$, whose kinetics are determined by $\frac{da}{dt} = -a^T(t)\frac{\partial LTC_{rhs}}{\partial \mathbf{x(t)}}$ (Chen et al., 2018a). To compute the loss gradients with respect to the parameters, we run the adjoint dynamics backwards from gradients $\frac{dL}{d\mathbf{x}(t_n)}$ and states $\mathbf{x}(t_n)$, while solving a third integral in reverse as follows: $\frac{dL}{d\theta} = -\int_{t_n}^{t_0} a^T(t)\frac{\partial LTC_{rhs}}{\partial \theta} dt$. All integrals can be solved in reverse-mode by a call to the ODE solver.

Based on Proposition 1, for an LTC network at each training iteration, the internal and external interventions not only help the system learn statistical dependencies from data but also facilitate the learning of the causal mapping. These results bridge the gap between pure physical models and causal structural models to obtain better learning systems.

4.4 Results

In this section, we perform extensive experimental evaluations to assess the performance of LTCs in time-series modeling and dynamic control tasks. First, we assess performance on a series of time-series benchmarking tasks. Second, we evaluate our method in the context of visual-control learning of aerial drones over a series of complex tasks, ranging from short- and long-term navigation, to chasing static and dynamic objects through photorealistic environments. Through these experiments we aim to validate our theoretical results on the performance and properties of LTC models.

4.4.1 Time-series Benchmarking

We evaluate the performance of LTCs against state-of-the-art RNN and neural ODE models in a series of time-series benchmarks.

Time Series Predictions

We evaluated the performance of LTCs realized by the proposed fused ODE solver against the state-of-the-art discretized RNNs, LSTMs (Hochreiter and Schmidhuber, 1997), CT-RNNs (ODE-RNNs) (Funahashi and Nakamura, 1993; Rubanova et al., 2019), continuous-time gated recurrent units (CT-GRUs) (Mozer et al., 2017), and neural ODEs (constructed by a 4^{th} order Runge-Kutta solver as suggested in (Chen et al., 2018a)), in a series of diverse real-life supervised learning tasks. The results are summarized in Table 4.3. We observed between 5% to 70% performance improvement achieved by LTCs compared to other RNN models in four out of seven tasks and comparable performance in the other three (Table 4.3).

Dataset	Metric	LSTM	CT-RNN	Neural ODE	CT-GRU	LTC (ours)
Gesture	(accuracy)	64.57% ± 0.59	59.01% ± 1.22	46.97% ± 3.03	68.31% ± 1.78	**69.55% ± 1.13**
Occupancy	(accuracy)	93.18% ± 1.66	94.54% ± 0.54	90.15% ± 1.71	91.44% ± 1.67	**94.63% ± 0.17**
Activity recognition	(accuracy)	95.85% ± 0.29	95.73% ± 0.47	**97.26%** ± 0.10	96.16% ± 0.39	95.67% ± 0.575
Sequential MNIST	(accuracy)	**98.41%** ± 0.12	96.73% ± 0.19	97.61% ± 0.14	98.27% ± 0.14	97.57% ± 0.18
Traffic	(squared error)	0.169 ± 0.004	0.224 ± 0.008	1.512 ± 0.179	0.389 ± 0.076	**0.099** ± 0.0095
Power	(squared-error)	0.628 ± 0.003	0.742 ± 0.005	1.254 ± 0.149	**0.586** ± 0.003	0.642 ± 0.021
Ozone	(F1-score)	0.284 ± 0.025	0.236 ± 0.011	0.168 ± 0.006	0.260 ± 0.024	**0.302** ± 0.0155

Table 4.3: **Time series prediction with LTCs.** Mean and standard deviation, n=5 trials

Human Activity Dataset

We next used the "Human Activity" dataset described in (Rubanova et al., 2019) in two distinct frameworks. The dataset consists of 6554 sequences of human activities (*e.g.* lying, walking, sitting), with a period of 211 ms. We designed two experimental frameworks to evaluate models' performance.

In the first setting, the baselines are the models described before, and the input representations are unchanged (details in Appendix). LTCs outperform all models and in particular CT-RNNs and neural ODEs with a large margin (Table 4.4. Note that the CT-RNN architecture is equivalent to the ODE-RNN described in (Rubanova et al., 2019), with the difference of having a state damping factor τ.

Algorithm	**Accuracy**
LSTM	83.59%± 0.40
CT-RNN	81.54%± 0.33
Latent ODE	76.48%± 0.56
CT-GRU	85.27%± 0.39
LTC (ours)	**85.48%± 0.40**

Table 4.4: **Person activity performance with LTCs, 1st setting**, n=5.

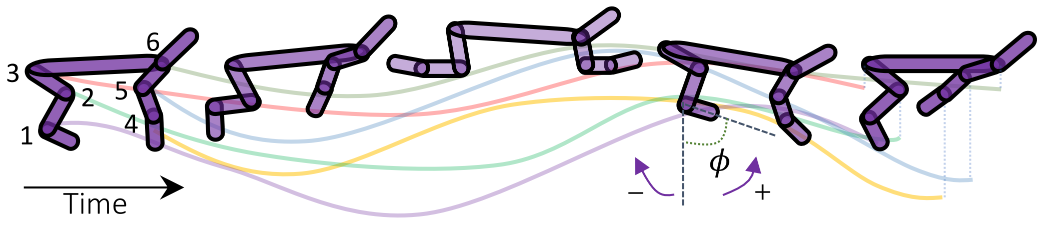
Algorithm	**Accuracy**
RNN Δ_t *	0.797± 0.003
RNN-Decay*	0.800± 0.010
RNN GRU-D*	0.806± 0.007
RNN-VAE*	0.343± 0.040
Latent ODE (D enc.)*	0.835± 0.010
ODE-RNN *	0.829 ± 0.016
Latent ODE(C enc.)*	0.846 ± 0.013
LTC (ours)	**0.882 ± 0.005**

Table 4.5: **Person activity performance, 2nd setting**, n=5 trials. Accuracy for algorithms indicated by $*$, are taken directly from (Rubanova et al., 2019) with: RNN Δ_t = classic RNN + input delays, D-enc. = RNN encoder C-enc = ODE encoder. RNN-Decay = RNN with exponential decay on hidden states (Mozer et al., 2017). GRU-D = gated recurrent unit + exponential decay + input imputation (Che et al., 2018).

In the second setting, we carefully set up the experiment to match the modifications made by (Rubanova et al., 2019) to obtain a fair comparison between LTCs and a more diverse set of RNN variants discussed in (Rubanova et al., 2019). LTCs show superior performance with a high margin compared to other models. The results are summarized in Table 4.5.

Half-Cheetah Kinematic Modeling

We then evaluated how well continuous-time models can capture physical dynamics. To perform this, we collected 25 rollouts of a pre-trained controller for the



17 input observations | 6 control outputs | ϕ = joint angle

Figure 4-6: **Half-cheetah** physics simulation

Algorithm	**MSE**
LSTM	2.500± 0.140
CT-RNN	2.838± 0.112
Neural ODE	3.805 ± 0.313
CT-GRU	3.014± 0.134
LTC (ours)	**2.308± 0.015**

Table 4.6: **Half-cheetah dynamical sequence modeling** with LTCs, n=5 trials.

HalfCheetah-v2 gym environment (Brockman et al., 2016), generated by the MuJoCo physics engine (Todorov et al., 2012b). The task is then to fit the observation space time-series in an autoregressive fashion (Figure 4-6). To increase the difficulty, we overwrite 5% of the actions by random actions. The test results are presented in Table 4.6 and demonstrate the superior performance of LTCs compared to other models.

4.4.2 Causal Navigation in Autonomous Flight

Here we show how continuous-time LTC models can be designed as causal structures to perform more interpretable real-world autonomous control. We consider a simple drone navigation task in which the objective is for the drone to navigate from position A to a target position B while simultaneously avoiding obstacles. We designed photorealistic visual navigation tasks of aerial drones with varying memory horizons including (1) navigating to a static target, (2) chasing a moving target, and (3) hiking with guide markers (Figure 4-7).

In this setting, the agent is asked to infer high-level control signals directly from

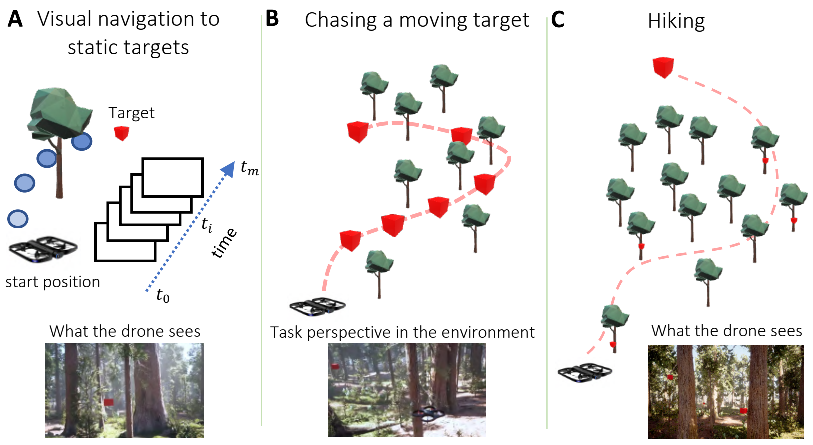

A Visual navigation to static targets
Target
start position
t_m
t_i
time
t_0
What the drone sees
B Chasing a moving target
Task perspective in the environment
C Hiking
What the drone sees

Figure 4-7: **Visual drone navigation tasks.** A) Navigation to a static target, B) Chasing a moving target, C) Hiking with a set of markers in the environment

visual inputs to reach the target goal. Even the simplest vision-based deep model can accomplish this task in passive open-loop settings (Lechner et al., 2020a). However, completion in a closed-loop environment while inferring the true causal structure of the task is significantly more challenging. For instance, where should the agent attend to for taking the next action? With what mechanisms does the system infer the objective of the task? And accordingly, how robust are the decisions of the learned policy? We conduct experiments to begin to answer the aforementioned questions.

Experimental setup

We designed tasks in Microsoft's AirSim (Madaan et al., 2020) and Unreal Engine. To create data for imitation learning, we use greedy path search with a Euclidean heuristic over unoccupied voxels to obtain knot points for cubic spline interpolation, which is then followed via a pure pursuit controller. This strategy is modified for each specific task.

Baselines. We evaluate NCP networks (Lechner et al., 2020a) against a set of baseline models. This includes ODE-RNNs (Rubanova et al., 2019) which are the recurrent network version of neural ODEs (Chen et al., 2018a), long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997), and CT-GRU networks (Mozer et al., 2017), which are the continuous equivalent of GRUs (Chung et al., 2014). These baselines are chosen to validate our theoretical results. Critically, we note that not all continuous-time (CT) models are causal models. Similarly, while discretized RNN models, such as LSTMs, perform well on tasks with long-term dependencies, they are not causal models.

In Section 4.3, we showed that sparse neural networks built based on LTC neurons (Hasani et al., 2020a) (*i.e.*, LTC networks) are dynamical causal models. Therefore, they can learn the true causal structure of a given task. In our experiments, camera images are perceived by convolutional layers and are fed into a downstream network that acts as a controller. For a fair comparison, the number of trainable parameters of all models is within the same range, and they are all trained by Adam optimizer (Kingma and Ba, 2014) with a cosine similarity loss.

Navigation to Static Target with Occlusion

In this task, the drone navigates to a target marker that is less than 25 meters away and visible to it. We place a red cube on a random unoccupied voxel in the environment to function as the target marker. We constrain the target marker to appear in the drone's viewing frustum for most cases. Occlusions create temporal dependencies, and thus there may be occlusion of the target upon random repositioning of the drone in the environment.

Table 4.7 shows that in both neighborhood and forest environments, all agents learn the task with a reasonable validation loss in a passive imitation learning setting. However, once these agents are deployed with closed-loop control, we observed that the success rates for LSTM and ODE-RNNs drop to only 24% and 18% of 50 attempted runs. CT-GRU managed to complete this task in 40% of the runs, whereas NCPs completed the tasks in 48% of the runs.

Table 4.7: **Validation on short-term navigation.** Cosine similarity loss [-1, 1] (smaller is better), n=5.

Algorithms	**Environments**	
	RedWood Forest	Neighborhood
LSTM	-0.823 ± 0.006	-0.838 ± 0.019
ODE-RNN	-0.815 ± 0.043	-0.855 ± 0.019
CT-GRU	-0.855 ± 0.001	**-0.877** ± **0.002**
LTC network (ours)	**-0.859** ± **0.035**	-0.855 ± 0.008

To understand these results better, we used the Visual-Backprop algorithm (Bojarski et al., 2018) to compute the saliency maps of the learned features in the input space. Saliency maps would show us where the attention of the network was when taking the next navigation decision. As shown in Figure 4-8, we observe that the LTC network has learned to attend to the static target within its field of view to make a future decision. This attention profile was not present in the saliency maps of the other agents. For example, LSTM agents are sensitive to lighting conditions compared to the CT models. This experiment supports our theoretical results on the ability of LTC-based models and suggests that they indeed learn causal representations.

Chasing a Moving Target

In this task, the drone follows a target marker along a smooth spline path. Using a generate and test method, we create a path for the drone to follow by using a random walk with momentum to find knot points for fitting a spline.

Table 4.8 shows that all agents were able to learn to follow their targets in a passive open-loop case. However, similar to the previous experiment, we witnessed that not all models can successfully complete the task in a closed-loop setting where interventions play a big role. LTC networks were 78% successful at completing their task, while LSTM in 66%, ODE-RNNs in 52%, and CT-GRU in 38% were successful. Once again, we looked into the attention maps of the models, illustrated in Figure 4-9. We see

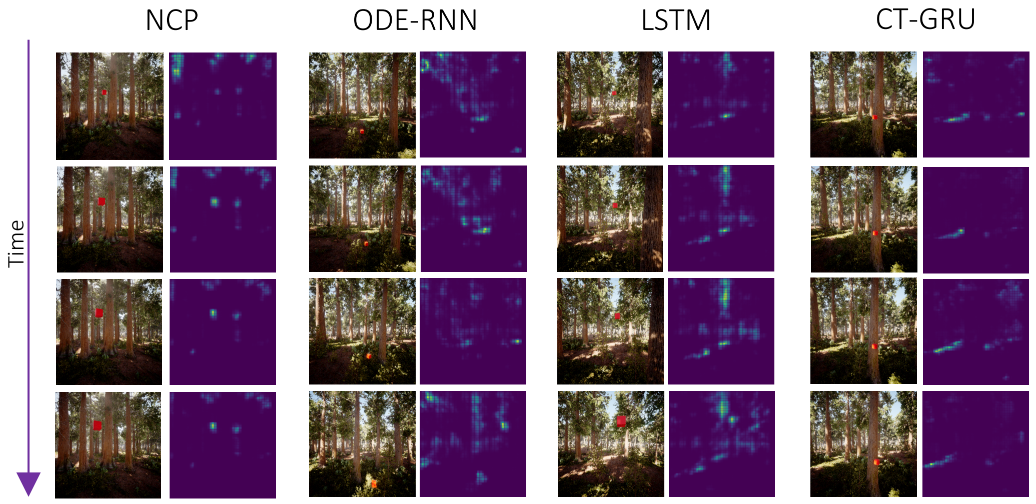


Figure 4-8: **Navigation to a static target in closed-loop environments.** LTC-networks (NCPs) are the only models that can capture the causal structure of the tasks directly from visual data.

that CT-GRU networks did not learn to attend to the target they follow. LSTMs again show sensitivity to lighting conditions. ODE-RNNs keep a close distance to the target, but they occasionally lose the target. In contrast, LTC networks have learned to attend to the target and follow them as they move in the environment.

Table 4.8: **Chasing objects.** Validation performance with co-sine similarity loss (smaller is better), n=5.

Algorithms	**Environments**	
	RedWood Forest	Neighborhood
LSTM	-0.943 ± 0.028	-0.947 ± 0.008
ODE-RNN	**-0.967 ± 0.009**	-0.953 ± 0.011
CT-GRU	-0.958 ± 0.017	**-0.979 ± 0.003**
LTC networks (ours)	-0.936 ± 0.022	**-0.975 ± 0.012**

Hiking Through an Environment

In this task, the drone follows multiple target markers which are placed on the surface of obstacles within the environment (Figure 4-7C). This task is significantly more

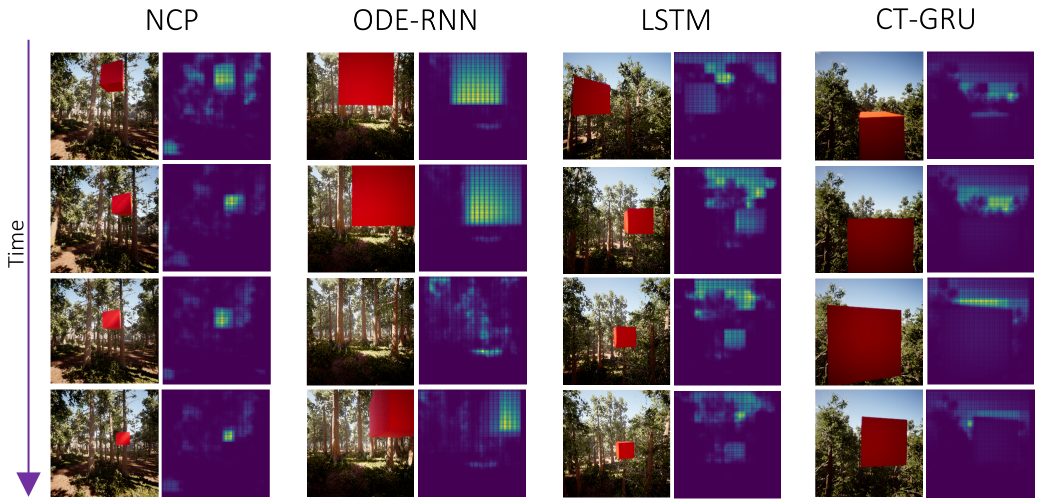


Figure 4-9: **Chasing a moving target in closed-loop environments.** LTC networks (left, NCPs) are the only models that can capture the causal structure of the tasks directly from visual data.

complex than the previous tasks, especially when agents are deployed directly in the environment. This is because the agents have to learn to follow a much longer time-horizon task, by visual cues, in a hierarchical fashion.

Table 4.9: **Hiking.** Validation performance with cosine similarity loss (smaller is better), n=5.

Algorithms	**Environments**	
	Redwood Forest	Neighborhood
LSTM	-0.273 $\pm$ 0.388	**-0.781** $\pm$ **0.030**
ODE-RNN	**-0.896** $\pm$ **0.026**	-0.710 $\pm$ 0.003
CT-GRU	-0.359 $\pm$ 0.073	-0.725 $\pm$ 0.086
LTC network (ours)	-0.676 $\pm$ 0.192	-0.711 $\pm$ 0.013

Interestingly, we see most agents learn a reasonable degree of validation loss during the learning process as depicted by Table 4.9. Even ODE-RNNs realize excellent performance in the passive setting. However, when deployed in the environment, none of the models other than the LTC network could perform the task completely in 50 runs. LTC networks could perform 30% of runs successfully, which we in part assign to their causal structure.

In Table 4, we summarize the success rate of CNNs in all tasks when deployed in

Table 4.10: **Closed-loop evaluation** of trained policies on various navigation and interaction tasks. Agents and policies are reinitialized randomly at the beginning of each trial (n=50). Values correspond to success rates (higher is better).

	Static Target					**Chasing**				**Hiking**
Model	**Clear**	**Fog**	**Light Rain**	**Heavy Rain**	**Occlusion**	**Clear**	**Fog**	**Light Rain**	**Heavy Rain**	**Clear**
LSTM	24%	22%	22%	4%	20%	66%	**62%**	56%	44%	2%
ODE-RNN	18%	10%	18%	2%	24%	52%	42%	62%	44%	4%
CT-GRU	40%	8%	**60%**	32%	28%	38%	36%	48%	42%	0%
LTC network (ours)	**48%**	**40%**	52%	**60%**	**32%**	**78%**	52%	**84%**	**54%**	**30%**

closed-loop. As expected, we observe that when temporal dependencies in the tasks appear (*i.e.* the Occlusion or Hiking tasks), as well as when the input images are highly perturbed (*i.e.* heavy rain or fog), the performance of CNNs drastically decreases. This observation validates that having a memory component is very beneficial across all these tasks.

4.5 Discussion

This chapter introduced liquid time-constant (LTC) neurons and networks. We showed that they could be implemented by arbitrary variable and fixed step ODE solvers and be trained by backpropagation through time. We demonstrated their bounded and stable dynamics, superior expressivity, and superseding performance in supervised learning time-series prediction tasks, compared to standard and modern deep learning models.

Furthermore, we provided theoretical evidence for the capability of different representations of continuous-time neural networks in learning causal structures. We then performed a set of experiments to confirm the effectiveness of continuous-time causal models in high-dimensional and visual drone navigation tasks with different memory-horizons compared to other methods. We conclude that the class of liquid time-constant networks has the great potential of learning causal structures in closed-loop reasoning tasks where other advanced RNN models cannot perform well.

4.6 Scope and Limitations

We have demonstrated that a special presentation of continuous-time neural models, based om liquid time-constant (LTC) neurons, can realize dynamic causal models and thereby significantly enhance the expressivity of decision making in an example control setting of autonomous drone flight. Despite this promise, notable limitations remain, and the theoretical foundations and first experiments described here open several avenues for future investigation.

First, similar to many variants of time-continuous models, LTCs express the vanishing gradient phenomenon (Pascanu et al., 2013a; Lechner and Hasani, 2020) when trained by gradient descent. Although the model shows promise on a variety of time-series prediction tasks, they would not be the obvious choice for learning long-term dependencies in their current format. Second, neural ODEs are remarkably fast compared to more sophisticated models such as LTCs. Nonetheless, they lack expressivity. Our proposed neural model, in its current format, significantly enhances the expressive power of TC models at the expense of elevated time and memory complexity which must be investigated in the future. This is a significant consideration for online autonomy and decision-making applications in which time and memory efficiency is a significant consideration.

Given the causal properties of LTCs, they hold great promise for the control of robots in continuous-time observation and action spaces where their causal structures can help improve robustness and reasoning (Lechner et al., 2020a). While here we perform preliminary experiments in simulated autonomous flight, the ability of LTC-based neural models for real-world autonomous control has yet to be established. Achieving full-scale autonomous control onboard physical platforms, such as vehicles or drones, will require significant scaling of LTC-based neural models, as well as systematic verification of their abilities to enhance the robustness and expressivity of learned control policies. Chapter 5 directly tackles these challenges. There, we describe an LTC-based neural control model, termed a Neural Circuit Policy (NCP), for robust, auditable, and expressive autonomous vehicle control.

Chapter 5

Scaling Neural Circuit Policies for Auditable Autonomy

5.1 Introduction

To achieve performant and robust autonomous control, an agent must learn a coherent representation of its world from multidimensional sensory information; establish an auditable, causal mapping of learned sensory representations to control policies; and utilize these representations and models to generalize well in unseen scenarios. Surprisingly, animals as small as the nematode *Caenorhabditis elegans* have mastered such an ability, in order to perform locomotion (Kato et al., 2015), motor control (Stephens et al., 2008), and navigation (Gray et al., 2005), through their near-optimal nervous system structure (Yan et al., 2017; Cook et al., 2019) and their harmonious neural information processing mechanisms (Kaplan et al., 2019).

Inspired by these structures and functional abilities, we set out to design a brain-inspired intelligent agent that learns to control an autonomous vehicle directly from its camera inputs (*i.e.* end-to-end learning to control (Lecun et al., 2004; Bojarski et al., 2016)). In complex real world scenarios, for instance autonomous driving, such neural computation inspiration (LeCun et al., 2015; Hassabis et al., 2017) can lead to more expressive artificial intelligence agents: models that are simultaneously accurate and auditable (Rudin, 2019). In Chapter 4, we presented liquid time-constant (LTC) neurons, a new continuous-time processing unit with expressive and provably causal structure, and used simulation experiments to explore the potential of these models for control learning. In this chapter, we scale this approach to achieve autonomous vehicle control onboard a full-scale physical platform while preserving the model's robustness, causality, and auditability properties–an outcome uniquely enabled by LTCs.

Specifically, we formulate neural models termed Neural Circuit Policies (NCPs), which implement LTC neurons within a neural architecture based on the circuit diagram of *C. elegans* (Yan et al., 2017; Cook et al., 2019). We demonstrate that NCPs meet key representation learning challenges of end-to-end control. First, the safety-critical nature of the task demands auditable and interpretable *dynamics* of the controllers. Additionally, it is desirable that agents learn the true *causal structure* (Pearl, 2009; Peters et al., 2017) between the observed driving-scenes and their corresponding optimal-steering commands (the specific task of the agent). Ideally, for a lane-keeping task, we wish that the agent implicitly learns to attend to the road's horizon when taking a current steering decision, while maintaining an attractive performance. Finally, within the processing pipeline of the high-dimensional data-stream input, the agent has to incorporate a *short-term memory mechanism* capturing temporal dependencies. We demonstrate that for the autonomous vehicle lane keeping task, very small NCP networks–with a control compartment consisting of only 19 neurons–in combination with compact convolutional neural networks (CNNs) (LeCun et al., 1990) for feature extraction, achieved superior performance and robustness in learning how to steer a vehicle directly from high-dimensional visual inputs.

5.2 Designing Neural Circuit Policies

To address the representation learning challenges and the complexity of autonomous lane keeping, we design a new end-to-end learning system that perceives the inputs by a set of convolutional layers (LeCun et al., 1989), to capture image structures, and performs control by a novel RNN structure, termed a *neural-circuit policy* (NCP).

The network structure of NCPs is inspired by the wiring diagram of the *C. elegans* nematode (Wicks et al., 1996). Many neural circuits within the nematode's nervous system are constructed by a distinct 4-layer hierarchical network topology (Figure 5-1). They receive environmental observations through sensory neurons. These are passed on to interneurons and command neurons which generate an output decision. Finally, this decision is passed to the motor neurons to actuate its muscles. The wiring diagram of *C. elegans* achieves a sparsity of around 90% (Yan et al., 2017), with feedforward connections from sensors to intermediate neurons, highly recurrent connections among interneurons and command neurons, and feedforward connections from command neurons to motor neurons. This specific topology was shown to have attractive computational advantages, such as efficient distributed control, requiring a small number of neurons (Yan et al., 2017), hierarchical temporal dynamics (Kaplan et al., 2019), robot-learning capabilities (Lechner et al., 2019), and maximal information propagation in sparse flow networks (Hasani et al., 2020b).

5.2.1 The Neural Model

The neural dynamics of NCPs is given by continuous-time ordinary differential equations (ODEs), originally developed to capture the dynamics of the nervous system of small species, such as *C. elegans* (Hasani et al., 2020a) (Figure 5-1a). At their core, NCPs possess a nonlinear time-varying synaptic transmission mechanism that improves their expressive power in modeling time series data, relative to their deep learning counterparts. The foundational neural building blocks of NCPs are liquid time-constant (LTC) neurons (Hasani et al., 2020a), each with state dynamics $x_i(t)$, represented as follows, when connected through an input synapse to a neuron j:

$$\dot{x}_i = -\big(\frac{1}{\tau_i} + \frac{w_{ij}}{C_{m_i}}\sigma_i(x_j)\big)x_i + \big(\frac{x_{leak_i}}{\tau_i} + \frac{w_{ij}}{C_{m_i}}\sigma_i(x_j)E_{ij}\big), \tag{5.1}$$

where $\tau_i = C_{m_i}/g_{l_i}$ is the time-constant of the neuron i with a leakage conductance of g_{l_i}, w_ij is a synaptic weight from neuron i to j, C_{m_i} is the membrane capacitance, $\sigma_i(x_j(t)) = 1/(1 + e^{-\gamma_{ij}(x_j - \mu_{ij})})$, x_{leak_i} is the resting potential and E_{ij} is a reversal

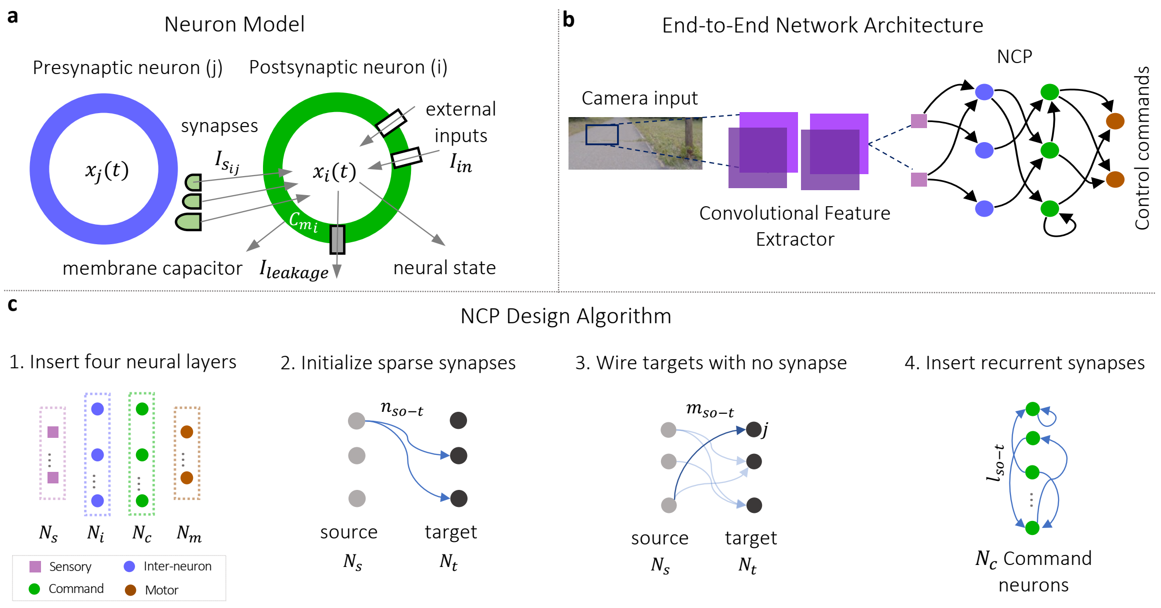


Figure 5-1: **Designing NCP networks with the LTC neural model.** **a** Representation of the neural state, $x_i(t)$, of a postsynaptic LTC neuron i receiving input currents from presynaptic neuron, j. The neural state is determined by the aggregation of the inflows/outflows to/from the cell. Synaptic currents are set by an input dependent nonlinearity f which is a function of the presynaptic neural state, $x_j(t)$ and its synaptic parameters (See Methods for further details). **b** Representation of an end-to-end NCP network. The network perceives the camera inputs that are transformed by a set of convolutional layers to a latent representation which is exploited by the designed NCP (based on the steps described in c) to produce control actions. **c** NCP design procedure based on rules 1 to 4.

synaptic potential that defines the polarity of the synapse. An LTC neuron's overall coupling sensitivity (*i.e.* its time-constant) is defined by:

$$\tau_{system_i} = \frac{1}{\frac{1}{\tau_i} + \frac{w_{ij}}{C_{m_i}}\sigma_i(x_j)}. \tag{5.2}$$

This variable time-constant determines the reaction speed of a neuron during a decision-making process, and holds true for the LTC neurons in the autonomous driving NCP.

Numerical Implementation of the NCP Network

To learn the parameters of an NCP circuit, we transform it into a differentiable representation. After modeling the circuit as a system of ordinary differential equations of LTC neurons, we employ a numerical ODE solver to obtain its computable form. A solving method suitable for our purpose has to comply with the following three constraints.

First, the solver is applied to a real-time system that puts a hard limit on worst-case executing time. Hence, the solver uses a fixed step-size (Press et al., 2007). Secondly, the ODE model of an NCP is stiff (Press et al., 2007; Hasani et al., 2020a). Consequently, to avoid numerical instabilities, we adopt a semi-implicit method (Press et al., 2007). Lastly, during the training phase, we compute partial derivatives by backpropagating through the solver. Similar to the stability arguments in the forward path, we need to monitor the error magnitude in the backward phase. In particular, a suitable solving method must not result in an exploding or vanishing gradient. To comply with these constraints, we employed a simple Euler approach. As a result, in summary, for each neuron, we adopt a semi-implicit Euler approach with a fixed step-size of the form:

$$x_i(t+\Delta) := \frac{x_i(t)C_{m_i}/\Delta + g_{l_i}x_{leak_i}\sum_{j\in I_{in}} w_{ij}\ \sigma_i(x_j(t))E_{ij}}{C_{m_i}/\Delta + g_{l_i} + \sum_{j\in I_{in}} w_{ij}\ \sigma_i(x_j(t))}. \tag{5.3}$$

The set I_{in} represents the set of neurons that are presynaptic to neuron i. This equation was derived from the basic Euler formula (Press et al., 2007):

$$x(t+\Delta) := x(t) + \Delta \dot{f}\Big(x(t+\tau), u(t+1)\Big) \tag{5.4}$$

by setting $\tau = \Delta$ for all $x(t+\tau)$ that appear linear in $\dot{f}$, and setting $\tau = 0$ for all other occurrences. Note that the well-known explicit (forward) Euler method can be obtained from Equation 5.4 by setting $\tau = 0$. Likewise, the implicit (backward) Euler method is realized by Equation 5.4 by setting $\tau = \Delta$ and solving the resulting non-linear equation for $x(t+\Delta)$. RNNs usually process their incoming input stream at a fixed sampling frequency (*e.g.*, 30 Hz in the described end-to-end driving tasks). To achieve a decent precision–a computation complexity trade-off–we simulated the ODE at a frequency six times higher than the input sampling rate; jat os. we packed 6 ODE solver steps into one RNN step. In both the training and testing phases, we initialized states of the ODE/RNN by zeros.

5.2.2 Wiring Design Principles

The architecture of an NCP network is determined by the design principles introduced in rules 1-4, corresponding to the steps presented in Figure 5-1c, as follows:

1. Insert four neural layers — N_s sensory neurons, N_i interneurons, N_c command neurons, and N_m motor neurons (Figure 5-1c-1).

2. Between every two consecutive layers — $\forall$ source neuron, insert n_{so-t} synapses ($n_{so-t} \leq N_t$), with synaptic polarity $\sim Bernoulli(p_2)$, to n_{so-t} target neurons, randomly selected $\sim Binomial(n_{so-t}, p_1)$ (Figure 5-1c-2).

3. Between every two consecutive layers — $\forall$ target neuron j with no synapse, insert m_{so-t} synapses ($m_{so-t} \leq \frac{1}{N_t}\sum_{i=1,\ i\neq j}^{N_t} L_{t_i}$), where L_{t_i} = the number of synapses to target neuron i, with synaptic polarity $\sim Bernoulli(p_2)$, from m_{so-t} source neurons, randomly selected from $\sim Binomial(m_{so-t}, p_3)$ (Figure 5-1c-3).

4. Recurrent connections of command neurons — $\forall$ command neuron, insert l_{so-t} synapses ($l_{so-t} \leq N_c$), with synaptic polarity $\sim Bernoulli(p_2)$, to l_{so-t} target

command neurons, randomly selected from $\sim Binomial(l_{so-t}, p_4)$ (Figure 5-1c-4).

Applying the NCP design principles above results in very compact and sparse networks of LTC neurons. The learning system corresponding to the lane-keeping task consists of the convolutional feature extraction front end stacked with the NCP network control head (Figure 5-1b). This system is trained in an end-to-end, supervised-learning fashion. Given a designed NCP network, we apply a semi-implicit ODE-solver in order to obtain a numerically accurate and stable solution of the system (Hasani et al., 2020a). We then recursively fold the ODE-solver call into an RNN cell and prepare the system's training pipeline. From the gradient propagation perspective, our approach gives rise to a vanishing gradient phenomenon, which, as described in Figure 4-2d, is the preferable setting for learning a real-world autonomous vehicle (AV) control.

5.3 Experimental Methods

To build and train the autonomous driving NCP, a large-scale selection of labeled training data was collected by recording the observations and actions of a human driver. End-to-end driving is a feedback control problem, where the control actuated by the agent proprioceptively affects future observations. However, during the supervised training phase, this feedback mechanism is utterly disregarded.

All data used to train networks was collected on a Toyota Prius 2015 V retrofitted with perception sensors (a forward-facing Leopard Imaging LI-AR0231-GMSL camera), inertial measurement unit (Xsens MTi 100-series IMU), GPS, and drive-by-wire steering (Naser et al., 2017). All data logging was done directly on an NVIDIA Drive PX2, the in-car high-performance computing platform. The IMU was used to record rotation of the vehicle's rigid body frame and thus compute the curvature of the vehicle's traversed path. Specifically, given a yaw rate γ_t $(rads/sec)$, and the speed of the vehicle, v_t (m/sec), we compute the curvature of the path as:

$$y_t = \frac{1}{r_t} = \frac{\gamma_t}{v_t} \tag{5.5}$$

where r_t is the radius of the traversing circle.

Ultimately, for the networks learned in this paper, we consider the problem of directly learning a control command from the human traversed road curvature (y_t) instead of the steering wheel angle (α_t). This is because α_t is a nonlinear function of both y_t and v_t and depends on the tire slip angle, road surface, weather conditions, and vehicle dynamics. Hence, simply learning the steering wheel angle (*i.e. what* the human commanded) is not sufficient for autonomous navigation. Instead, we require knowledge of the traversed road curvature (*i.e. where* the human drove). We can compute the steering wheel angle online by using a bicycle model approximation to control the car at the inference time:

$$\alpha_t = K \arctan(L\, y_t) \tag{5.6}$$

where K is the steering ratio (*i.e.* the ratio between steering and tire angle), and L is the vehicle length.

5.3.1 Offline Data Collection

For passive evaluation experiments, we collected approximately five hours of driving data throughout diverse regions of the Boston metropolitan area during dry, wet, and snowy weather conditions on the highway, local, and residential roads. We processed the data by removing ambiguous segments such as lane switches, crossings, and congestion. We split the data into ten non-overlapping sets of equally sized chunks for the cross-testing procedure. We trained a model on the union of the remaining nine sets for each of the ten sets, then evaluated the performance of the model on the withhold test set. The number of training epochs was optimized based on a validation set, which we separated from the union of the nine sets before training.

Table 5.1: **Results of the passive lane-keeping 10-fold cross-testing evaluation.** As the squared errors on the test set are determined by large outliers we reported squared and absolute error. Sparse LSTM models are trained with projected gradient descent to enforce a 95% sparsity level. All tested NCP architectures are composed of 19 neurons.

Model	**Training square error**	**Test squared error**
CNN	1.41 ± 0.30	4.28 ± 4.63
Vanilla RNN	0.14 ± 0.05	3.39 ± 4.39
CT-GRU	0.19 ± 0.05	3.63 ± 4.61
CT-RNN (19 units)	0.44 ± 0.14	3.62 ± 4.35
CT-RNN (64 units)	0.23 ± 0.09	3.43 ± 4.55
Sparse CT-RNN (19 units)	0.77 ± 0.35	4.03 ± 4.80
Sparse CT-RNN (64 units)	0.40 ± 0.43	3.72 ± 4.71
GRU	1.25 ± 1.02	5.06 ± 6.64
LSTM (64 units)	0.19 ± 0.05	**3.17** ± 3.85
LSTM (19 units)	0.16 ± 0.06	3.38 ± 4.48
Sparse LSTM (19 units)	1.05 ± 0.57	3.68 ± 5.21
Sparse LSTM (64 units)	0.29 ± 0.14	3.25 ± 3.93
NCP	0.43 ± 0.26	**3.22** ± 3.92
NCP (randomly wired)	2.12 ± 2.93	5.19 ± 5.43
NCP (fully-connected)	2.41 ± 3.44	5.18 ± 4.19

We observed that such a train-test discrepancy led to situations where a trained neural network model that performs exceptionally well on the labeled sequences in an offline testing environment (Table 5.1) fails to steer the car safely in a real testing case. Modern RNNs are particularly vulnerable to these scenarios, as their decision-making process heavily relies on past observations. Hence, to properly assess performance, we chose the architectures that worked well during offline testing and evaluated them actively on a real car.

5.3.2 Training Procedure

Here we describe the training procedure of the models. If not stated otherwise, this description applies to the passive and active test scenarios. We formulated the end-to-end autonomous driving control problem as a regression task. Hence, we adopted the squared error as the training loss function. As recordings of curves and turns are underrepresented in the training data, we multiplied a weighting factor to the loss value of each sample. This weighting factor $w(y) := \exp(|y|\alpha)$ depends on the target curvature y exponentially and thus assigns a higher priority to samples containing road-curves and turns. As the test track is located in a forest area where trees cast shadows with variable profiles on the road, we implemented a shadow augmentation data technique during training. In essence, we draw a semi-transparent black or white line over each training image. The location, orientation, and width of lines are randomly sampled from uniform distributions.

We trained all models, except the feedforward CNN, on sub-sequences of 16 time-steps, which correspond to 0.53 real-time seconds. The neural state of standard CT-RNN and LSTM implementations are unbounded, which may lead to instabilities during closed-loop testing, as they are only trained on finite sequences. To avoid the internal states of the controller to grow indefinitely, *i.e.*, a phenomenon known in control theory as wind-up (Bohn and Atherton, 1995), we apply a clipping operation to the states of the CT-RNN and LSTM to keep the values within the range $[-5, 5]$. We used *Adam* (Kingma and Ba, 2014) as the optimization algorithm in all experiments.

The convolutional layers' architecture for all RNN models are listed in Table 5.2. After the last convolutional layer, we applied four per-channel linear layers to obtain $8 \times 4 = 32$ latent features serving as sensory inputs to the RNN control compartment. We empirically tuned the learning rates and the convolutional layers' hyperparameters and evaluated them on the passive dataset. We observed that the NCP model took advantage of a lower learning rate for the convolutional layers and a higher learning rate for the RNN compartment. We apply a per-image whitening filter to the images before feeding them into the networks.

To perform a fair comparison, we equipped all RNN models with the same convolutional head that reduces the dimensionality of the input image to a more compact latent representation to be fed into the RNN compartments (Table 5.2). We trained and evaluated the networks with the following architectures: 64-neurons LSTM, 64-neurons continuous-time (CT) RNN, and19-neurons NCP. Moreover, we compared these recurrent agents to the state-of-the-art feedforward CNN model for autonomous

Table 5.2: **Convolutional head.** Specifications of the convolutional head used for extracting driving features.

Layer	Filters	Kernel size	Strides
1	24	5	2
2	36	5	2
3	48	3	2
4	64	3	1
5	8	3	1

vehicle control (Bojarski et al., 2016).

5.3.3 Closed-loop Evaluations

We conducted the active driving experiments on a private road system. To prepare the models, we collected approximately 94 minutes of data by maneuvering the vehicle through the test track. We split the data into a training and a validation set based on a 3:1 ratio. The number of training epochs was selected based on the lowest error on the validation set achieved during training. We tested each trained model 5 times around the test track, without input perturbations, and two times while the input was disrupted by a zero-mean Gaussian distribution with variances $0.1, 0.2$, and 0.3 (two times per perturbation magnitude).

Each evaluation consists of driving the car, in the counterclockwise direction, around one cycle of the outermost path of the track.

We started an evaluation by placing the vehicle at a designated initial location, accelerating the car up to a constant speed of 4.47 m/s, and delegating the steering system's control to the neural network. Every time the vehicle was maneuvered off the road, we manually steered the car back on track and reported an intervention (*i.e.* crash).

5.3.4 Saliency Maps

Saliency maps are interpretation methods to visualize the inner workings of a trained neural network by highlighting parts of the input image that contributed most to the decision of a network. We employed saliency maps to analyze qualitatively what our networks have learned to attend to. In particular, we were interested in how layers that are common to all tested architectures evolve differently during training. Consequently, we narrow our analysis to the convolutional layers in the feature extraction head at the beginning of the network. We additionally adopted VisualBackProp (Bojarski et al., 2018), which has been developed deliberately for autonomous driving research, to compute the saliency maps presented. This method leverages the property of the ReLU activation in that that the value of each neuron in the feature map is either positive or zero.

Algorithm 5 Compute Saliency Map

```
**Inputs:** Convolutional feature maps $h_1, h_2, \dots h_N$
$s$ := average-over-channel-dimension($h_N$)
**for** i **from** N-1 **to** 1 **do**
    $z$ := average-over-channel-dimension($h_i$)
    $s$ := $z \odot$ deconvolution-to-size-of($s$,$z$)
**end for**
**return** $s$.
```

In Algorithm 5, $\odot$ represents the element-wise multiplication, and deconvolution to-size-of function scales the first argument to the dimension of the second argument by applying a deconvolution operation.

Structural Similarity Index

Structural Similarity Index (SSIM) is a method to compare quality of given images (Wang et al., 2004). It is computed as the product of three comparison criteria, the luminance l, contrast c, and structure s for given images x and y, as follows (Wang et al., 2004):

$$SSIM(x,y) = [l(x,y)]^{\alpha}.[c(x,y)]^{\beta}.[s(x,y)]^{\gamma}, \tag{5.7}$$

where:

$$l = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \quad c = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \quad s = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \tag{5.8}$$

Here, C_1,C_2, and C_3 are the regularisation constants, μ_x and μ_y are the means of x and y, σ_x and σ_y are standard deviations, and σ_{xy} is the cross-covariance of x and y. We computed the SSIM for pairs of saliency maps at each time-frame (overall 200 frames), between a noise-free version as reference and a perturbed one resulted from input noise injections. We set the exponents $\alpha = \beta = \gamma = 1$, the regularisation components $C_1 = (0.01L)^2$, $C_2 = (0.03L)^2$, and $C_3 = C_2/2$, with $L = 255$ corresponding to the dynamics range of the input image values.

5.4 Results

5.4.1 Compactness

A full-stack NCP network is 63 times smaller than the CNN network that established the state-of-the-art of end-to-end driving (Bojarski et al., 2016). Its control-network is 970 times sparser than that of LSTM and 241 times than CT-RNN. An NCP's RNN compartment possesses 233 times smaller trainable parameter-space than that of LSTM, and 59 times lower than CT-RNN. Interestingly, the performance achieved by such a compact neural representation is superior to that of other models in multiple aspects of an ideal autonomous mobile robot controller, discussed as follows.

5.4.2 Robustness to Perturbations

As outlined in Figure 4-2, an ideal model should incorporate temporal information to allow the filtering of any form of perturbation. To demonstrate this, we simply used an additive zero-mean Gaussian noise, because such noise was not present in the date used for the training process, and it requires a minimal assumption on the form, shape, and severity of the perturbation signal. Compared to all learning systems under-test, NCPs are significantly more robust in avoiding crashes (going off road) which are caused by raising the pixel-wise input perturbations (Figure 5-2a). The reason for their noise resiliency is that their continuous-time model serves as a filter (Equation 5.1). Note that the primary objective of this experiment was to identify

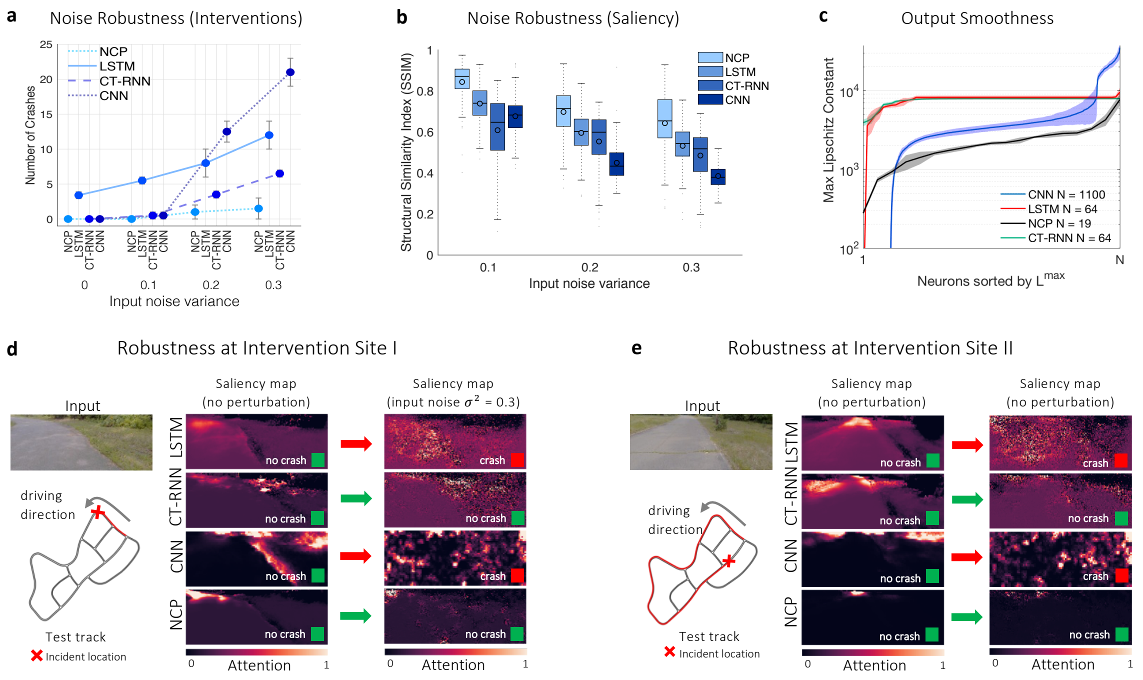


Figure 5-2: **Robustness analysis.** **a** Number of crashes (steering commands with tendency to drive the vehicle off-road) for four RNN types, as the input noise variance increases, in active driving test (n=3) **b** Variation of the structural sensitivity index of the saliency maps for four RNN topologies as the input noise variance increases (see Methods for computational details), A higher value of SSIM is preferable (n=3) **c** Maximum Lipschitz constant (an indicator of smoothness and stable dynamics) of the activity of every single neuron (sorted by the amplitude of the max Lipschitz constant on the horizontal axis) (n=5), Lower values are preferable. **de** Example of the saliency maps before a crash event caused by LSTM and CNN, in the presence of input perturbations, and how it was handled by CT-RNN and NCP. **ee** A second example of a crash incident caused by LSTM and CNN. **Videos** of the driving performance with no input perturbations: NCP, LSTM, CT-RNN and CNN; and with a $\sigma^2 = 0.3$ input perturbation: NCP, LSTM, CT-RNN and CNN.

how differently each model relies on its memory for making a prediction.

We quantified the influence of the input perturbations on the attention maps (the learned causal structure), by computing their structural similarity index (SSIM), represented in Figure 5-2b. SSIM indicates how much the structure of the attention maps gets distorted when the incoming inputs are perturbed. The closer SSIM is to 1, the less distorted is the attention. Thus the network can handle input noise more robustly. The closer the SSIM index is to 0, the more distorted the network's attention is, which results in the increased uncertainty of the network when making a correct driving decision. Under different levels of input perturbations, NCPs consistently maintain a higher SSIM compared to the other models, therefore, reducing its output decision's uncertainty (Figure 5-2b). Figure 5-2d-e depict examples of crash incidents that happened at the locations shown on the map, when the inputs to the networks were heavily perturbed by an input noise. These figures also illustrate how the attention of each intact network is disrupted by the input noise and caused LSTM and CNN networks to drive the vehicle off-road.

5.4.3 Smoothness and Stability

We quantitatively measured the maximum steepness of the neural dynamics derivative (maximum local Lipschitz constant) for all neurons in the system. The limitation on the speed of change of a function can be computed by the Lipschitz continuity criteria ($|f(t_2) - f(t_1)| \leq L|t_2 - t_1|$). The smaller the Lipschitz constant (L), the smoother the transitions of function f are.

Algorithm 6 Compute Maximum Lipschitz Constants

```
Inputs: $X^{(N\times T)}$ $N$ = number of neurons, $T$ = length of the test episode
for n from 1 to N do
    for $t$ from 1 to $T-1$ do
        $L(n,t) = \frac{dX}{dt} \approx X(n,t+1) - X(n,t)/\Delta t$, $\Delta t = 1$
    end for
end for
$L_{max}^{(N\times 1)} = \max(L^{(N,T)}$
Return $L_{max_{sorted}} = \text{sort}(L_{max})$.
```

We compute the maximum Lipschitz constants for neural state activity of RNNs $X(t)$, for an episode of active testing for all RNN types (Algorithm 6) reported in Figure 5-2c. We observed that the local decision-making process in NCPs is remarkably smoother than those of other network types.

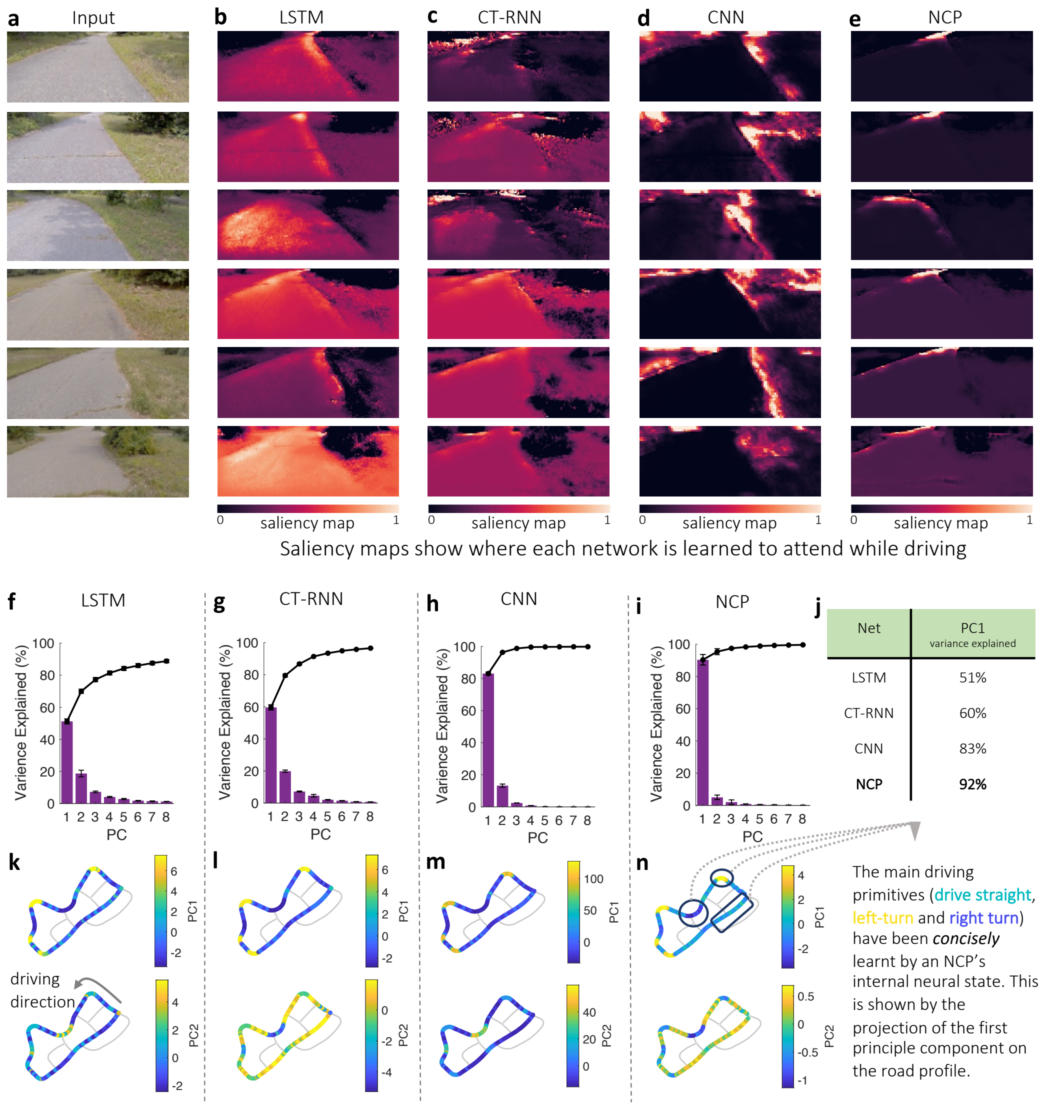
5.4.4 Interpretability and Attention Analysis

Interpretation is the process of providing explanations to humans. Although no formal definition for interpretability exists (Molnar, 2020), we define a model to be more interpretable, if its causal mapping between input observations and output decisions, as well as its global hidden-state dynamics, are more comprehensible to humans (Hasani, 2020).

In this regard, we conduct quantitative interpretability analysis by explaining the attention maps of the convolutional layers, and by computing the global network dynamics of the recurrent network compartment of the models. We then explain cell-level contributions through visualization techniques. For the driving task, we find that there is a close relationship between the geometry of the environment, the specific driving task, and the network nodes responsible for the required behavior, This is a consequence of defining the function of each neuron by differential equations. Accordingly, we experimented with the learned lane-keeping networks to measure their interpretability in three distinct ways.

Given an input image (Figure 5-3a), (Figure 5-3b-e represent sample attention maps of the convolutional parts of the networks during live testing (Figure 5-3). We have observed that the attention patterns are exclusive to the choice of network architecture (*e.g.* CNN, LSTM, CT-RNN, NCP), and the explanations are invariant to the choice of hyperparameters (*e.g.* network size).

For instance, the convolutional layers in the NCP networks predominantly attend to the road's horizon to make a driving decision Figure 5-3e). This is very desirable in the lane-keeping task. In contrast, a CNN network looks at the roadside to make a driving decision and ignores the road itself (Figure 5-3d). LSTM forced its perception network to learn to attend to the roadside in most scenarios. However, lighting conditions, as well as road profiles, can significantly alter the network's attention portfolio (Figure 5-3b). CT-RNN's attention is inconsistent and is heavily influenced by the variations of the road's lighting conditions (Figure 5-3c). These maps give an intuitive insight into the decision-making process of a task-specific network within a



Net	PC1 variance explained
LSTM	51%
CT-RNN	60%
CNN	83%
NCP	**92%**

The main driving primitives (drive straight, left-turn and right turn) have been *concisely* learnt by an NCP's internal neural state. This is shown by the projection of the first principle component on the road profile.

Figure 5-3: **Global network dynamics.** **a,** A sequence of input camera images during the active testing. b to e present a set of saliency maps computed to obtain the attention of the convolutional layers of the trained networks while driving (See Methods "Saliency maps computation" for details). **b,** LSTM learned to attend to the roadsides in most scenarios; however, lighting conditions significantly affect its attention portfolio. **c,** CT-RNN's attention is inconsistent and is heavily influenced by the variations of the road's lighting conditions. **d,** CNN learned to drive by focusing on the side roads **e,** NCP learned to attend to the road's horizon when taking a driving decision. f to i represent the variance explained by first eight principal components of the activity of all neurons of **f,** LSTM, **g,** CT-RNN, **h,** CNN, and **i,** NCP, (n=5). The black line indicates the cumulative variance explained. **j,** First PC's variance-explained for all models. k to n shows the projection of the first (top) and the second (bottom) PC's score (The score of the n^{th} PC is computed by $PC^{(n)}_{score} = output\ vector \times weight_{PC^{(n)}}$), over the driving trajectory for **k,** LSTM, **l,** CT-RNN, **m,** CNN, and **n,** NCP. (Click on networks' names to watch their driving performance.)

full-stack autonomous driving system. This insight could help in further safety and robustness analysis.

5.4.5 Global Network Dynamics

To measure how concisely the networks learned the primitives of driving (straight roads, handling curves, and road jitters), we performed a principal component analysis (PCA) and reported its variance in Figure 5-3f-i. PCA is conducted over the activations of the hidden neurons (*without* the inclusion of the output signal) of the RNN compartments of the driving networks, collected during live-testing. The analysis demonstrated that the first principal component (PC) of NCP's neural dynamics concisely learned the global driving features (explaining 92%), as shown in Figure 5-3j, while the second PC learned fine-grained decisions. The conciseness was less apparent in networks with LSTM and CT-RNN recurrent compartments, and therefore it is more challenging to associate their behavior to intuitive explanations.

To motivate this phenomenon further, we plotted the first and the second PC scores over the driving trajectory in Figure 5-3k-n. NCP is the only model among the others that allocated distinct PC1 activation regimes to the main driving primitives, while *fine-grained* control decisions have been largely captured by PC2. Other baseline networks require at least two to three PCs to capture the driving profile up to 90%, as shown in Figure 5-3f-i. Consequently, the added value of these results for a more complex autonomous control system is that the global dynamics of a learned agent can be interpreted and used for further improvements on the task-specific networks.

5.4.6 Cell-level Auditability

The neural state (the amplitude of a neuron's output) and the coupling sensitivity (how a neuron adjusts its reaction speed when interacting with the environment) of LTC cells comprising an NCP network (Figure 5-4a), can help to understand how an LTC network's decision is made. Figure 5-4b-d illustrate the activity of five selected neurons from the NCP driving agent, projected over the driving trajectory. The *motor neuron*'s activity illustrates how the inferred motion primitives correspond to various driving situations (Figure 5-4b left). Its coupling sensitivity demonstrates that the neuron tends to become more reactive during turns (setting smoother dynamics) while keeping its reaction speed at a relatively constant-rate during straight motions.

Interneuron 1 learned to activate during left-turns (Figure 5-4c top-left) while adjusting its dynamics to react faster at left-turning events (Figure 5-4c top-right). *Interneuron 2*, on the other hand, learned to rapidly get more active during right-turns (Figure 5-4c bottom). *Command neuron 1* is consistently activated during straight driving with a sensitive reaction speed while it is switched off on left-turns (Figure 5-4d top). *Command neuron 2* is biased at lower membrane potentials and tunes to road jitters when the vehicle drives on a straight path (Figure 5-4d bottom).

This degree of immediate interpretation of dynamics is generalizable to every single cell within an NCP (Figure 5-5). As the number of computational elements of an NCP system is considerably lower than that of state-of-the-art neural networks, such a degree of access to each cell dynamics could potentially be beneficial for designing fault-test and corner-case analysis to improve the safety of the deployed autonomous system.

5.5 Discussion

Neural circuit policies (NCPs) are, to the best of our knowledge, the smallest task-specific neural network agents that can proficiently control a vehicle on previously-unseen roads, while at the same time being robust to input artifacts, taking advantage of causality, and realizing interpretable dynamics. NCPs are beneficially used within full-stack autonomous control frameworks (Figure 5-6. They are designed to improve the performance and transparency of black-box compartments om such complex full-stack autonomous control systems.

A vision-based full-stack autopilot has to incorporate many different tasks for the incoming image-streams (Figure 5-6a). In contrast, current state-of-the-art functional

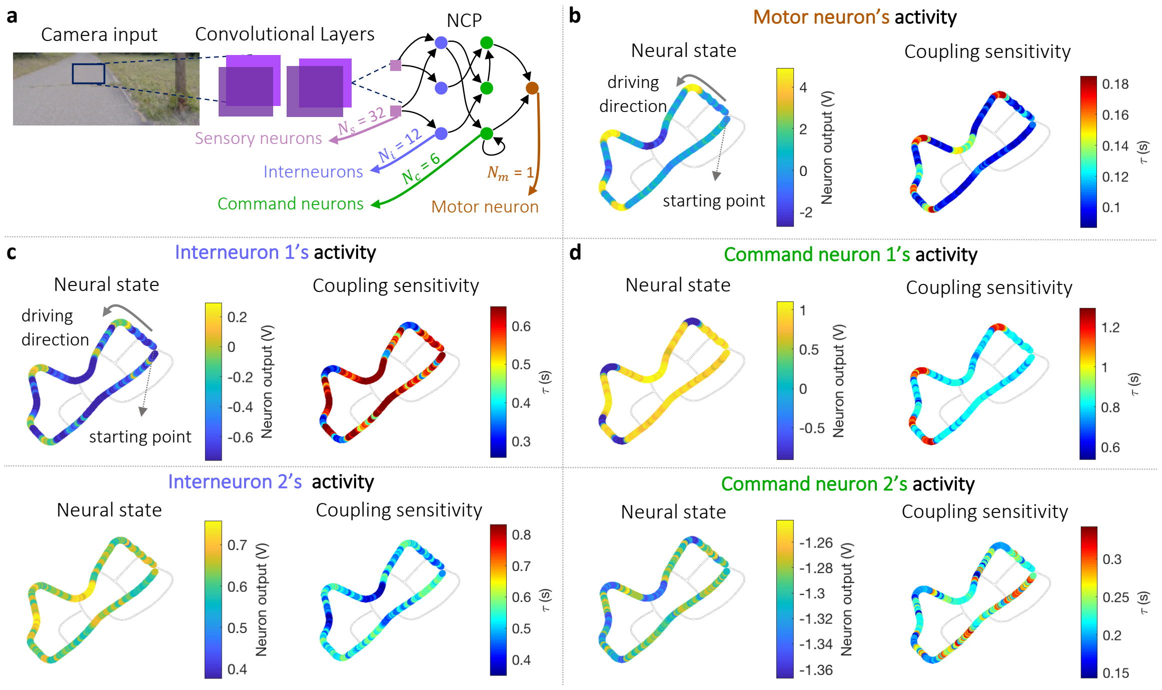


Figure 5-4: **Intuitive comprehension of NCP's cells activity while driving. a,** An NCP network trained end-to-end for autonomous lane-keeping. b to d depict examples of neural activities projected over the road-trajectory on which the car was driven. The neural state (representing the amplitude of a neuron's dynamics) and the coupling sensitivity (representing how a neuron adjusts its reaction speed) are plotted in each subsection. **b,** Neural activity of the motor neuron. **c,** Neural activity of two interneurons 1 and 2. **d,** Neural activity of two command neurons 1 and 2. An immediate explanation of the cell-level dynamics for the NCP network is achieved and extends to every internal element of the network.

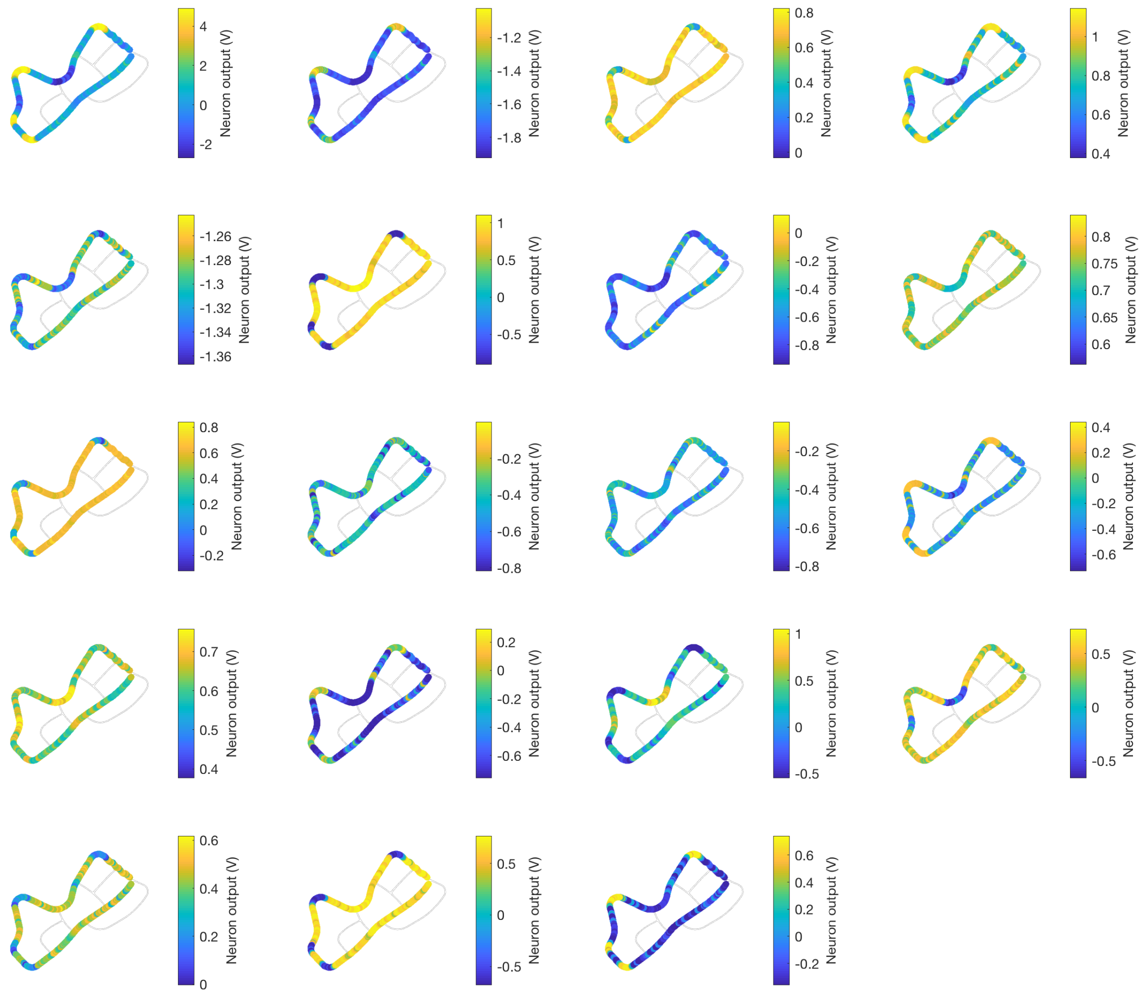


Figure 5-5: **Neural activity of all NCP neurons presented in Fig. 5-4**

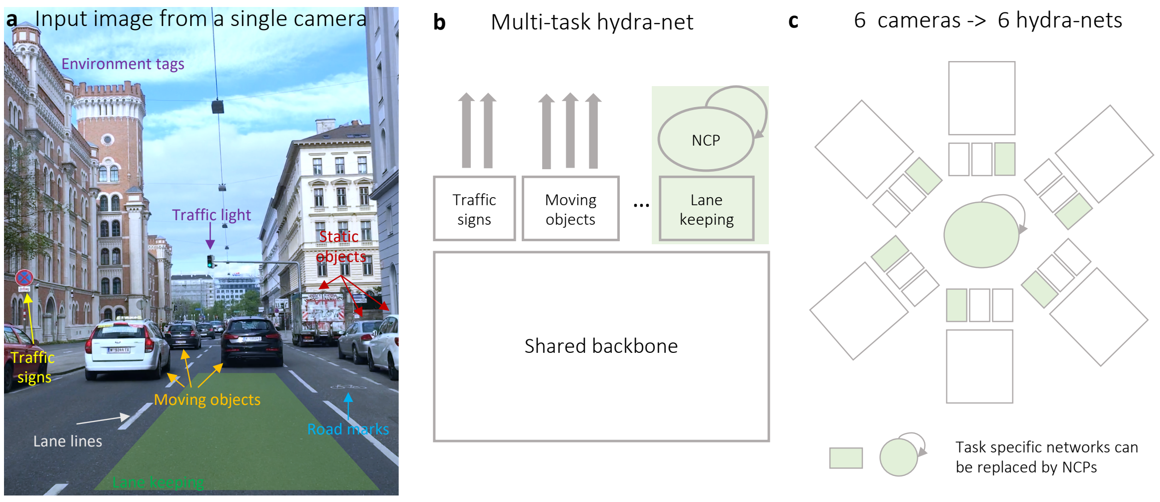


Figure 5-6: **NCPs as task-specific networks within a full-stack autonomous vehicle engine.** **a** Camera input and examples of tasks. **b** An overview of Hydra-nets of Tesla Autopilot (Tesla, 2020) redesigned from Tesla at PyTorch 2019 (Karpathy, 2019). **c** The overall structure of a vision-based full-stack AV system redesigned from Tesla at PyTorch 2019 (Karpathy, 2019).

AV systems (Tesla, 2020) typically share a convolutional backbone network, with many task-specific networks (Karpathy, 2019) upstream (Figure 5-6b). We made sure that NCPs maintain compositionality within full-stack AVs, by enabling an end-to-end training pipeline that can backpropagate errors through the NCPs to the static CNN-based backbone. The resulting task-specific NCPs (*e.g.* for the lane-keeping task) improve many aspects of the contemporary neural control modules in use.

Real-world application domains, such as autonomous driving, are surrounded by environmental artifacts and uncertainty and thus demand robust real-time decision making. Moreover, similar to autonomous driving tasks, many applications–such as avionics, service robots, and medicine–deal with complex, high-dimensional input-output spaces that become safety-critical when deployed in the real world. The success of NCPs in task-specific AVs indicates that tackling the complexity of real-world problems does not necessarily require learning very large neural networks that are hard to comprehend.

5.6 Scope and Limitations

A central goal of artificial intelligence in high-stakes decision-making applications is to design a single algorithm that simultaneously expresses generalizability by learning coherent representations of their world and interpretable explanations of its dynamics. We combine brain-inspired neural computation principles and scalable deep learning architectures to design compact neural controllers for task-specific compartments of a full-stack autonomous vehicle control system. We discover that a single algorithm with 19 control neurons, connecting 32 encapsulated input features to outputs by 253 synapses, learns to map high-dimensional inputs into steering commands. This system shows superior generalizability, interpretability, and robustness compared with orders-of-magnitude larger black-box learning systems. The obtained neural agents enable high-fidelity autonomy for task-specific parts of a complex autonomous system.

The compactness, auditability, and robustness of NCPs opens the door to new, tailored computing architectures for low-power, high-performance neural inference in a variety of applications. Indeed, real-world decision making demands the ability to make those decisions in real time, often under limited computational resources, and guarantees on both the performance and robustness of those decisions. The properties of NCPs makes these architectures attractive candidates for these types of settings, for example in edge computing or embedded biocomputing (*i.e.* from biometric signals). Furthermore, the unique architecture of NCPs raises the possibility of a tailored chip or processor for efficient, sustainable in-memory neural computation using NCPs. Future work to evaluate the performance of NCPs in low-power settings, assess their generalizability to various decision making tasks, and develop new computing and hardware platforms for NCPs will be critical to meet these goals.

More generally, NCPs provide one example of how the design of the model architecture itself can improve robustness and generalizability in learning to control. This chapter describes how we achieve this by developing a learning-based models that is decomposable into more compact neural processing units with improved auditability. When learning a robust decision making system, not only must the base model architecture be expressive and reliable, but the associated learning and optimization algorithms that train such models must also be well-equipped to generalize to challenging scenarios in order to further guarantee robustness. In Chapter 6 we present a step towards this goal and describe an algorithmic approach to improve learning from underrepresented and edge-case scenarios.

Chapter 6

Learning Algorithms to Generalize to Underrepresented Edge-cases and Mitigate Bias

6.1 Introduction

Robots operating in human-centric environments have to perform reliably in unanticipated situations. While deep neural networks (DNNs) offer great promise in enabling robots to learn from humans and their environments (as opposed to hand-coding rules), substantial challenges remain (Schwarting et al., 2018). For example, previous work in autonomous driving has demonstrated the ability to train end-to-end a DNN capable of generating vehicle steering commands directly from car-mounted video camera data with high accuracy so long as sufficient training data is provided (Bojarski et al., 2016). But true autonomous systems should also gracefully handle scenarios with insufficient training data.

A society where autonomous systems are safely and reliably integrated into daily life demands agents that are aware of scenarios for which they are insufficiently trained. To date, many such systems have been trained with datasets that are either biased or contain class imbalances, due to the lack of labeled data. This negatively impacts both the speed and accuracy of training and thus the downstream performance of the autonomous system.

These considerations extend more generally beyond the setting of autonomy and robotics. Indeed, as machine learning (ML) systems are increasingly making decisions in safety-critical and human-centric applications throughout society, it is critical to ensure the development and deployment of fair and unbiased AI systems in order to achieve the long-term acceptance of these algorithms (Miller, 2015; Courtland, 2018). For example, ML and artificial intelligence (AI) are already being used to determine if a human is eligible to receive a loan (Khandani et al., 2010), how long a criminal should spend in prison (Berk et al., 2016), the order in which a person is presented the news (Nalisnick et al., 2016), or even diagnoses and treatments for medical patients (Mazurowski et al., 2008).

While deep learning based systems have been shown to achieve state-of-the-art performance on many of these tasks, it has also been demonstrated that algorithms trained with imbalanced or biased data lead to algorithmic discrimination (Bolukbasi et al., 2016; Caliskan et al., 2017). Even the seemingly simple task of facial recognition (Zafeiriou et al., 2015; Ranjan et al., 2017) has been shown to be subject to extreme amounts of algorithmic bias among select demographics (Buolamwini and Gebru, 2018). For example, (Klare et al., 2012) analyzed the face detection system used by the US law enforcement and discovered significantly lower accuracy among dark women between the age of 18-30 years old. This is especially concerning since these facial recognition systems are often not deployed in isolation but rather as part of a larger surveillance or criminal detection pipeline (Abdullah et al., 2017). Recent benchmarks quantifying discrimination (Kilbertus et al., 2017; Hardt et al., 2016) and even datasets specifically created to evaluate the fairness of these algorithms (Buolamwini and Gebru, 2018) have emerged. However, the problem of severely imbalanced training datasets and the question of how to integrate debiasing capabilities into AI algorithms for autonomous systems and decision-making pipelines still remain largely unsolved.

This chapter presents an algorithmic approach that tackles these challenges by integrating debiasing capabilities directly into a model training process that adapts automatically and without supervision to the shortcomings of the training data. Our approach features an end-to-end deep learning algorithm that simultaneously learns the desired task (*e.g.*, autonomous vehicle control, facial detection) as well as the underlying latent structure of the training data. Learning the latent distributions in an unsupervised manner enables us to uncover underrepresented edge cases and hidden or implicit biases within the training data. We then leverage the learned latent structure to adaptively resample the dataset as training progresses. Our algorithm, which is built on top of a variational autoencoder (VAE), is capable of identifying underrepresented examples in the training dataset and subsequently increases the probability at which the learning algorithm samples these data points.

We first describe the effects of underrepresentation on algorithmic bias and discuss methods for learning the latent structure of a dataset. We then formulate an algorithm for automated debiasing based on learned latent structure. This provides two key methodological contributions:

1. A novel, tunable debiasing algorithm which utilizes learned latent variables to adjust the respective sampling probabilities of individual data points while training; and

2. A semi-supervised model for simultaneously learning a debiased classifier as well as the underlying latent variables governing the given classes.

Next, we demonstrate two concrete applications of this method:

1. Facial detection, in which we show how our algorithm can be used to debias a facial detection system trained on a biased dataset;

2. Autonomous driving, in which we demonstrate how this method can identify underrepresented edge cases to improve training efficiency and predictive performance.

6.2 Uncovering and Mitigating Algorithmic Bias

6.2.1 The Effect of Underrepresentation on Bias

We first consider the problem of binary classification in which we are presented with a set of paired training data samples $\mathcal{D}_{train} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{n}$ consisting of features $\boldsymbol{x} \in \mathbb{R}^m$ and labels $\boldsymbol{y} \in \mathbb{R}^d$. Our goal is to find a functional mapping $f : \boldsymbol{X} \to \boldsymbol{Y}$ parameterized by $\boldsymbol{\theta}$ which minimizes a certain loss $\mathcal{L}(\boldsymbol{\theta})$ over our entire training dataset. In other words, we seek to solve the following optimization problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i(\boldsymbol{\theta}) \tag{6.1}$$

Given a new test example, $(\boldsymbol{x}, \boldsymbol{y})$, our classifier should ideally output $\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ where $\hat{\boldsymbol{y}}$ is "close" to $\boldsymbol{y}$, with the notion of closeness being defined from the original loss function.

Now, assume that each datapoint also has an associated continuous latent vector $z \in \mathbb{R}^k$ which captures the hidden, sensitive features of the sample. We can formalize the notion of a biased classifier as follows:

Definition 6. *A classifier, $f_{\boldsymbol{\theta}}(x)$, is **biased** if its decision changes after being exposed to additional sensitive feature inputs. In other words, a classifier is fair with respect to a set of sensitive latent features, z, if:*

$$f_{\boldsymbol{\theta}}(x) = f_{\boldsymbol{\theta}}(x, z) \tag{6.2}$$

For example, when deciding if an image contains a face or not, the skin color, gender, or even age of the individual are all underlying latent variables and should not impact the classifier's decision.

To ensure fairness of a classifier across these various latent variables, the dataset should contain roughly uniform samples over the sensitive latent space. In other words, the training distribution itself should not be biased to overrepresent a certain category while under-representing others. Note that this is different than claiming that our dataset should be balanced with respect to the classes (*i.e.*, include roughly the same number of faces as non-faces in the dataset). Namely, we are saying that *within* a single class the unobserved latent variables should also be balanced. This would promote the notion that all instances of a single class will be treated fairly by the classifier such that, even if a latent variable was changed to the opposite extreme (*e.g.* skin tone from light to dark), the accuracy of the classifier would not be changed.

Furthermore, given a labeled test set across the space of sensitive latent variables, z, we can measure the bias of the classifier by computing its accuracy across a set of sensitive categories (*e.g.* dark vs. light skinned faces). While the overall accuracy of the classifier is simply the mean of the accuracies for each sensitive class, the bias can be estimated as the variance in accuracies across classes. For example, if a classifier performs equally well no matter the realization of a specific sensitive latent variable (*e.g.* skin tone), it will have a variance in accuracy equal to zero, and thus be called unbiased. On the other hand, if some realizations of the latent variable cause the classifier to perform better or worse, the variance in the accuracies will increase, and thus, so will the overall bias of said classifier.

While it is possible to use a set of human defined sensitive variables to ensure fair representation during training, this requires time-intensive manual annotation of each variable over the entire dataset. Additionally, this approach is subject to potential human bias in the selection of which variables are deemed sensitive or not. Our method addresses this problem by learning the latent variables of the class in an entirely unsupervised manner and proceed then using these learned variables to adaptively resample the dataset while training. In the following subsection, we will outline the architecture used to learn the latent variables.

6.2.2 Learning the Latent Structure of a Dataset

We propose an extension of the variational autoencoder (VAE) network architecture (Kingma and Welling, 2013; Rezende et al., 2014) which we use for both unsupervised learning of the latent training structure and debiasing. We refer to this architecture as a debiasing variational autoencoder (or DB-VAE). An overview of the network

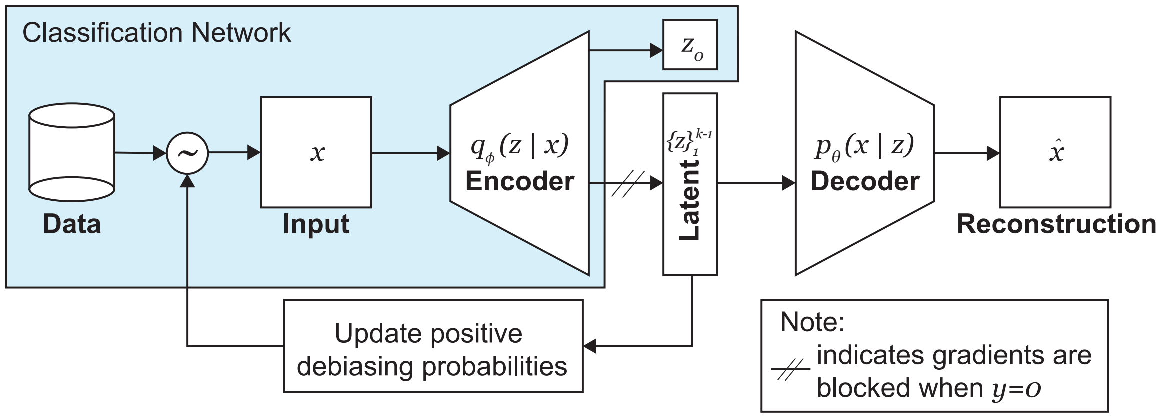


Figure 6-1: **Debiasing Variational Autoencoder.** Architecture of the semi-supervised DB-VAE for binary classification (blue region). The unsupervised latent variables are used to adaptively resample the dataset while training.

architecture is shown in Figure 6-1. At a high level, the goal is to learn the explanatory factors, *i.e.*, the latent variables z, that underlie the data distribution. The encoder portion of the VAE learns an approximation $q_\phi(z|x)$ of the intractable posterior, the true distribution of the latent variables given a data point. However, as opposed to classical VAE architectures, we also introduce d additional output variables corresponding to the dimension of our output $\hat{y}$. With k latent variables and d output variables, the encoder outputs $2k + d$ activations corresponding to $\boldsymbol{\mu} \in \mathbb{R}^k$, $\boldsymbol{\Sigma} = Diag[\sigma^2] \succ 0$, which are used to define the distribution of z, and the d-dimensional output, $\hat{\boldsymbol{y}}$.

Note that, in order to still learn our original supervised learning task we assign and explicitly supervise the d output variables. This, in turn, transforms our traditional VAE model from an entirely unsupervised model to a semi-supervised model, where some latent variables are implicitly learned by trying to reconstruct the input and the others are explicitly supervised for a specific task (e.g. classification). For example, if we originally wanted to train a binary classifier (i.e., $\hat{\boldsymbol{y}} \in \{0, 1\}$), our DB-VAE model would learn a latent encoding of k latent variables (i.e., $\{z_i\}_{i\in\{1,k\}}$) as well as a single variable specifically for classification: $z_0 = \hat{y}$.

A decoder network mirroring the encoder is then used to reconstruct the input back from the latent space by approximating $p_\theta(x|z)$. VAEs utilize reparameterization to differentiate the outputs through a sampling step, where we sample $\epsilon \sim \mathcal{N}(0, (I))$ and compute z:

$$\boldsymbol{z} = \boldsymbol{\mu}(x) + \boldsymbol{\Sigma}^{\frac{1}{2}}(x) \circ \epsilon \tag{6.3}$$

This decoded reconstruction enables unsupervised learning of the latent variables during training, and is thus necessary for automated debiasing of the data corresponding to the learned latent variables.

We train the network end-to-end using backpropagation with a three component loss function comprising of a supervised latent loss, a reconstruction loss, and a latent loss for the unsupervised variables. For comparison, the baseline model used for the desired task would have a similar architecture as the DB-VAE, without the unsupervised latent variables and decoder network. It would be trained according to only the supervised loss function.

The supervised loss in binary classification, for example, could be simply given by the cross-entropy loss:

$$\mathcal{L}_y(y, \hat{y}) = -\big(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\big) \tag{6.4}$$

When training the unsupervised section of the network, the reconstruction loss is given by the L_p norm between the input and the reconstructed output. This is used to train the decoder and is given by:

$$\mathcal{L}_x(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|_p \tag{6.5}$$

where x_i is an input and $\hat{x_i}$ is the decoded reconstruction of that input.

The latent loss is given by the Kullback-Liebler (KL) divergence between the latent variables and a specified distribution (Kingma and Welling, 2013; Rezende et al., 2014). This regularizes the latent space and, for Gaussian distributions, has the form:

$$\mathcal{L}_{KL}(\mu, \sigma) = \frac{1}{2} \sum_{j=0}^{k-1} (\sigma_j + \mu_j^2 - 1 - \log(\sigma_j)) \tag{6.6}$$

Finally, the total loss is simply a weighted combination of these three losses:

$$\mathcal{L}_{TOTAL}(\cdot) = c_1\mathcal{L}_y(y, \hat{y}) + c_2\mathcal{L}_x(x, \hat{x}) + c_3\mathcal{L}_{KL}(\mu, \sigma) \tag{6.7}$$

where c_1, c_2, c_3 are the weighting coefficients to impact the relative importance of each of the individual loss functions. Note that special care needs to be taken when feeding training examples from classes which you do not want to debias. For example, in the facial detection problem, we primarily care about ensuring that our positive dataset of faces is fair and unbiased, and less about debiasing the negative example where there is no face present. For these negative samples, the gradients from the decoder and latent space should be stopped and not backpropogated. This effectively means that, for these classes, we only train the encoder to improve the supervised loss.

6.2.3 Algorithm for Automated Debiasing

In this section, we present the algorithm for adaptive resampling of the training data based on the learned latent structure by our DB-VAE model. By dropping over-represented regions of the latent space according to their frequency of occurrence, we increase the probability of selecting rarer data for training. This is done adaptively as the latent variables themselves are being learned during training. Thus, our debiasing approach accounts for the complete distribution of the underlying features in the training data.

The training dataset is fed through the encoder network, which provides an estimate $\mathcal{Q}(\boldsymbol{z}|X)$ of the latent distribution. We seek to increase the relative frequency of rare data points by increased sampling of under-represented regions of the latent space. To do so, we approximate the distribution of the latent space with a histogram with dimensionality defined by the number of latent variables, k:

$$\hat{\mathcal{Q}}(\boldsymbol{z}|X) \propto \mathcal{Q}(\boldsymbol{z}|X) \tag{6.8}$$

To circumvent the high-dimensionality of the histogram when the latent space becomes increasingly complex, we simplify this approximation further and use independent histograms to approximate the joint distribution, $\hat{\mathcal{Q}}(\boldsymbol{z}|X)$. Specifically, we define an independent histogram, $\hat{\mathcal{Q}}_i(\boldsymbol{z}_i|X)$, for each latent variable $\boldsymbol{z}_i$. Thus, we make the approximation:

$$\hat{\mathcal{Q}}(\boldsymbol{z}|X) \propto \prod_i \hat{\mathcal{Q}}_i(\boldsymbol{z}_i|X) \tag{6.9}$$

This allows us to neatly approximate $\mathcal{Q}(\boldsymbol{z}|X)$ based on the frequency distribution of each of the learned latent variables. Finally, we introduce a single parameter, α, to tune the degree of debiasing introduced during training. We define the probability distribution of selecting a datapoint $\boldsymbol{x}$ as $\mathcal{W}(\boldsymbol{z}(\boldsymbol{x})|X)$, parameterized by the debiasing parameter α:

$$\mathcal{W}(\boldsymbol{z}(\boldsymbol{x})|X) \propto \prod_i \frac{1}{\hat{\mathcal{Q}}_i(\boldsymbol{z}_i(\boldsymbol{x})|X) + \alpha} \tag{6.10}$$

We provide pseudocode for training the DB-VAE in Algorithm 7. At every epoch all inputs x from the original dataset X are propagated through the model to evaluate the corresponding latent variables $z(x)$. The histograms $\hat{Q}_i(z_i(x)|X)$ are updated accordingly. During training, a new batch is drawn by keeping inputs, x, from the original dataset, X, with likelihood $W(z(x)|X)$. Training on the debiased data batch now forces the classifier into a choice of parameters that work better in rare cases without strong deterioration of performance for common training examples. Most importantly, the debiasing is not manually specified beforehand but instead based on *learned* latent variables.

Intuitively, we can think of the parameter α as tuning the degree of debiasing. As $\alpha \to 0$, the subsampled training set will tend towards uniform over the latent variables z. As $\alpha \to \infty$, the subsampled training set will tend towards a random uniform sample of the original training dataset (*i.e.*, no debiasing).

6.3 Results

With this foundation, we next demonstrate the utility of our algorithm for identifying underrepresented regions and mitigating algorithmic balance in two concrete real-

Algorithm 7 Adaptive re-sampling for automated debiasing of the DB-VAE architecture

```
Input: Training data $\{X, Y\}$, batch size $b$
  Initialize weights $\{\phi, \theta\}$
  for each epoch, $E_t$ do
      Sample $z \sim q_\phi(z|X)$
      Update $\hat{\mathcal{Q}}_i(z_i(x)|X)$
      $\mathcal{W}(z(x)|X) \leftarrow \prod_i \frac{1}{\hat{\mathcal{Q}}_i(z_i(x)|X)+\alpha}$
      while $iter < \frac{n}{b}$ do
          Sample $\boldsymbol{x}_{batch} \sim \mathcal{W}(z(x)|X)$
          $L(\phi, \theta) \leftarrow \frac{1}{b}\sum_{i \in \boldsymbol{x}_{batch}} \mathcal{L}_i(\phi, \theta)$
          Update: $[w \leftarrow w - \eta \nabla_{\phi,\theta}\mathcal{L}(\phi, \theta)]_{w \in \{\phi,\theta\}}$
      end while
  end for
```

world applications: (1) facial detection and recognition, and (2) autonomous driving.

6.3.1 Facial Detection and Recognition

Problem Setup

In order to validate our debiasing algorithm on a real-world problem with significant social impact, we attempt to learn a debiased facial detector using potentially biased training data. We demonstrate that our method can identify underrrepresented examples in the training dataset and subsequently increase the probability at which the learning algorithm samples these data points (Figure 6-2) In this section, we define the facial detection problem, describe the datasets used, and outline model training, debiasing, and evaluation procedures.

For the facial detection problem, we are given a set of paired training data samples $\mathcal{D}_{train} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{n}$, where $\boldsymbol{x}^{(i)}$ are the raw pixel values of an image patch and $\boldsymbol{y}^{(i)} \in \{0, 1\}$ are their respective labels, indicating the presence of a face.

Our goal is to ensure that the set of positive examples used to train a facial detection classifier is fair and unbiased. The positive training data may potentially be biased with respect to certain attributes such as skin tone, in that particular instances of those attributes may appear more or less frequently than other instances. Thus, in

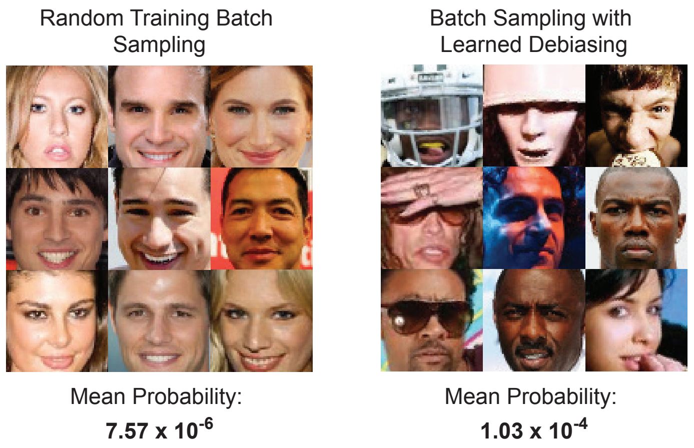

Random Training Batch Sampling
Batch Sampling with Learned Debiasing
Mean Probability: 7.57 x 10^-6
Mean Probability: 1.03 x 10^-4

Figure 6-2: **Batches sampled for training without (left) and with (right) learned debiasing.** The proposed algorithm identifies, in an unsupervised manner, under-represented parts of training data and subsequently increasing their respective sampling probability. The resulting batch (right) from the CelebA dataset shows increased diversity in features such as skin color, illumination, and occlusions.

our experiments, we train a full DB-VAE model to learn the latent structure underlying the positive (face) images and use the adaptive resampling approach outlined in Algorithm 7 to debias the model with respect to facial features. For negative examples (images of non-faces), we only train the encoder portion of our network. We evaluate the performance of our debiased models relative to standard, biased classifiers on the Pilot Parliaments Benchmark (PPB) dataset (Buolamwini and Gebru, 2018) and provide estimates of the precision and bias of each model as performance metrics.

Datasets

We train our classifiers on a dataset of $n = 4 \times 10^5$ images, consisting of 2×10^5 positive (images of faces) and negative (images of non-faces) examples, split 80% and 20% into training and validation sets, respectively. The positive examples were taken from the CelebA dataset (Liu et al., 2015). Aligned CelebA images were cropped to a square based on the annotated face bounding box and resized to 64×64. Negative examples were taken from the ImageNet dataset (Deng et al., 2009), from a wide variety of non-human categories, including but not limited to cars, houses, textures, landscapes, non-human primates, dogs, and tools. Both full images and random crops of the negative examples were resized to 64×64 to improve network generalizability.

After training, we evaluate our debiasing algorithm on the PPB test dataset (Buolamwini and Gebru, 2018), which consists of images of 1270 male and female parliamentarians from three African countries and three European countries. Images are consistent in pose, illumination, and facial expression, and the dataset exhibits parity in both skin tone and gender. The gender of each face is annotated with the sex-based "Male" and "Female" labels. Skin tone annotations are based on the gold standard Fitzpatrick skin type classification system (Fitzpatrick, 1988), with each image labeled as "Lighter" (Fitzpatrick score I-III) or "Darker" (Fitzpatrick score IV-VI).

Learned Latent Structure of Facial Databases

In order to interpret the features which the DB-VAE is debiased against on the CelebA dataset, we analyzed the latent space of the learned by the encoder using pertubation analysis. Beginning with a mean sampled latent vector, we perturb a single latent variable by slowly increasing/decreasing its magnitude. We feed the resulting vector through the decoder to visualize the reconstructed output face. The results of this analysis for five representative latent variables are shown in Figure 6-3. Qualitative and interpretable meanings deduced from each variable are labeled on the left of each row. We observed that the DB-VAE is able to learn facial features such as skin tone, presence of hair, and azimuth, as well as other features such as gender and age (Figure 6-3). This supports the hypothesis that our DB-VAE algorithm is capable of debiasing against such features since the resampling probabilities are directly defined based on the probability distributions of *individual* learned latent variables (Algorithm 7).

Automated Debiasing of Facial Detection Systems

In this section, we explore the output of the debiasing algorithm for facial detection and provide extensive evaluation of our learned models on the Pilot Parliaments Benchmark (PPB) dataset (Buolamwini and Gebru, 2018). We consider the resam-

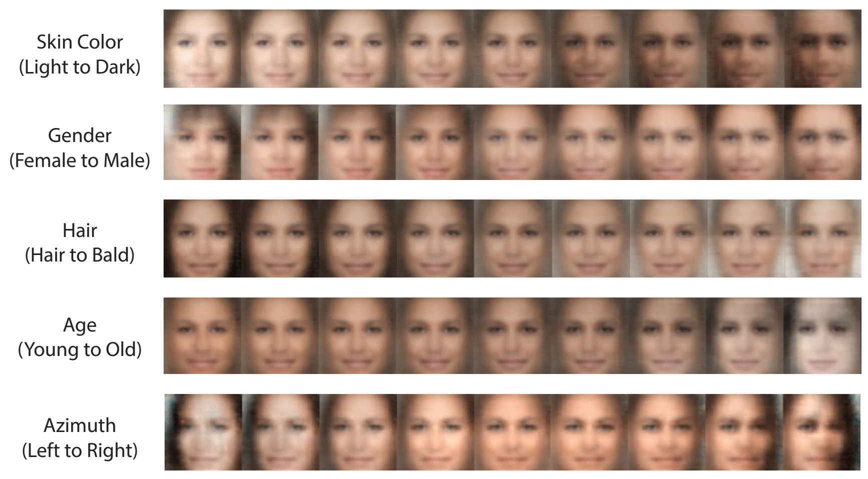


Figure 6-3: **Learned latent variables.** Examples of faces generated by perturbing (left to right) a single latent variable. The DB-VAE is able to learn classical sensitive features such as skin color, gender, hair, age, and even face heading.

pling probabilities that arise from learning a debiased model. These probabilities are defined by the distribution $\mathcal{W}(\boldsymbol{z}(\boldsymbol{x})|X)$. As shown in Figure 6-4A, as the probability of resampling increases, the number of data points within the corresponding bin decreases, suggesting that those images more likely to be resampled are those characterized by 'rare' features.

Indeed, as the probability of resampling increases, the corresponding images become more diverse, as evidenced by the four sample faces from each frequency bin in Figure 6-4A. This observation is further validated by considering the ten faces in the training data with the lowest and highest resampling probabilities (Figure 6-4B,C respectively). The ten faces with the lowest resampling probability appear quite uniform, with consistent skin tone, hair color, forward gaze, and background color. In contrast, the ten faces with the highest resampling probability display rarer features such as headwear or eyewear, tilted gaze, shadowing, and darker skin. Taken together, these results imply that our algorithm identifies and then actively resamples those data points with rarer, more diverse features based on a learned latent representation.

To evaluate the performance of our debiasing approach, we utilized classification accuracy (positive predictive value) as a metric, and tested our models on the PPB dataset. For this evaluation, we extracted patches from each image using sliding

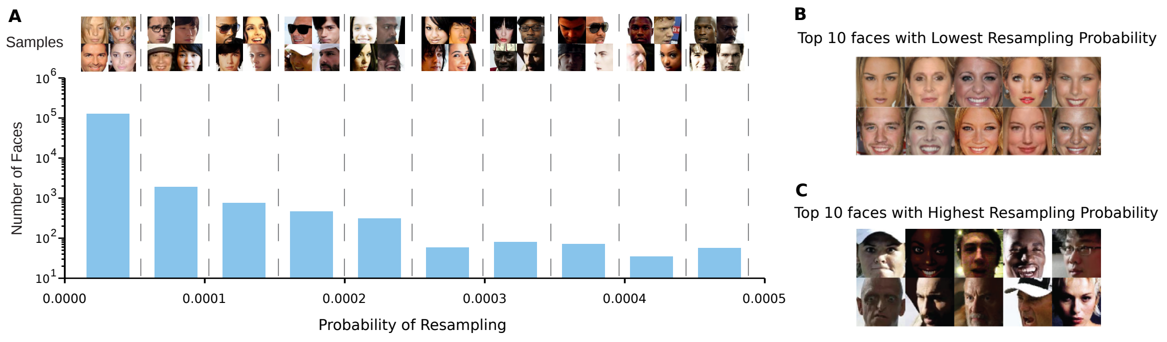


Figure 6-4: **Sampling probabilities over the training dataset.** Histogram over the resampling probabilities showing four sample faces from each bin (**A**). The top ten faces with the lowest (**B**) and highest (**C**) probabilities of being sampled.

windows of varying dimension, and fed these extracted image patches to our trained models. We output a positive match of a face if the classifier identifies a face in any one of the subpatches within the image. We formalize this as follows.

Let $\{\widetilde{\boldsymbol{x}}_{\mathrm{p}}\}$ be the set of extracted patches for a given image $\widetilde{\boldsymbol{x}}$ in the test dataset, where $\widetilde{\boldsymbol{x}}_{\mathrm{p}}$ corresponds to an individual patch. The probability that a given patch contains a face is given by the output of the encoder network:

$$\hat{y}_{\mathrm{p}} = q_{\phi}(y = 1|\widetilde{\boldsymbol{x}}_{\mathrm{p}}). \tag{6.11}$$

The predicted classification for an image, $\widetilde{\boldsymbol{x}}$, in the test dataset is then given by $\hat{y} = \left[\left[(\max\ \hat{y}_{\mathrm{p}}) > \frac{1}{2}\right]\right]$.

To address whether our approach could effectively debias against specific sensitive features, we quantified classification performance on individual demographics. Specifically, we considered skin tone (light/dark) and gender (male/female). We denote $\boldsymbol{\mathcal{A}}$ as the set of classification accuracies of a model on each of the four intersectional classes. We compared the accuracy of models trained with and without debiasing on both individual demographics (race/gender) and the PPB dataset as a whole, and provide results on the effect of the debiasing parameter α on performance (Figure 6-5). Recall that no debiasing corresponds to the limit $\alpha \to \infty$, where we uniformly sample over the *original training set* without learning the latent variables. Conversely, $\alpha \to 0$, corresponds to sampling from a uniform distribution over *the latent* *space*. Error bars (standard error of the mean) are provided to visualize statistical significance of differences between the trained models.

As shown in Figure 6-5, as we increased debiasing power (decrease α) classification accuracy significantly increased on "Dark Male" subjects. This is consistent with the hypothesis that adaptive resampling of rare instances (*e.g.*, dark faces) in the training data results in less algorithmic discrimination. This also suggests that our algorithm can effectively debias for a qualitative feature like skin tone, which has significant social implications for its utility in improving fairness in facial recognition systems.

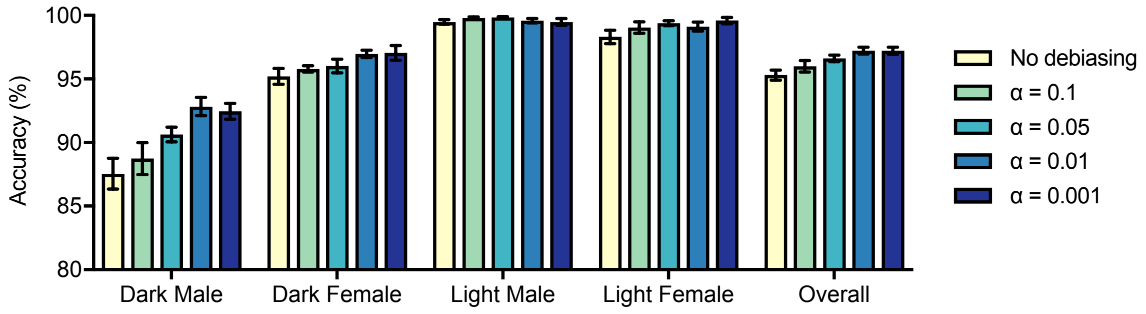


Figure 6-5: **Increased performance and decreased categorical bias with DB-VAE.** Models were evaluated on the PPB dataset to demonstrate increased performance and decreased categorical bias with increased debiasing.

In contrast to the trend observed with dark male subjects, the classification accuracy on "Light Male" faces remained nearly constant for both the biased and debiased models. Additionally, the accuracy on light male subjects was higher than the three other groups, consistent with previous reports (Buolamwini and Gebru, 2018). This suggests that our debiasing algorithm does not significantly sacrifice performance on categories which already have high precision. Importantly, the high, near constant accuracy suggests that an arbitrary classification model trained on the CelebA dataset may be biased towards light male subjects, and further supports the need for approaches that seek to reduce such biases.

Although the DB-VAE models improved accuracy on dark males significantly, they never reached the accuracy of light males. Despite the fact that we debias our training data with respect to latent variables such as skin tone, there are inherently fewer examples of dark males in the dataset. Thus, our model is simply limited by

Table 6.1: **Accuracy and bias on PPB test dataset.**

	$\mathbb{E}[\boldsymbol{\mathcal{A}}]$ (Precision)	$Var[\boldsymbol{\mathcal{A}}]$ (Measure of Bias)
No Debiasing	95.13	28.84
$\alpha = 0.1$	95.84	25.43
$\alpha = 0.05$	96.47	18.08
$\alpha = 0.01$	97.13	9.49
$\alpha = 0.001$	**97.36**	**9.43**

infrequency of these examples. Increasing the overall size of our training dataset may further mitigate this effect.

We summarize the key trends in overall performance with DB-VAE in Table 6.1. As confirmed by Figure 6-5, the overall precision, $\mathbb{E}[\boldsymbol{\mathcal{A}}]$, increased with increased debiasing power (decreasing α). Additionally, we observed a decrease in the variance in accuracy between categories, indicative of decreased bias with greater debiasing. Together, these results suggest effective debiasing with DB-VAE.

6.3.2 Autonomous Driving

Problem Setup

We consider end-to-end autonomous driving, where a steering control command is predicted from only a single input image. We start from the end-to-end model framework. In this framework we observe n training images, $X = \{x_1, \ldots, x_n\}$, which are collections of raw pixels from a front-facing video camera. We aim to build a model that can directly map our input images, X, to output steering commands based on the curvature of the road $Y = \{y_1, \ldots, y_n\}$.

We formulate a DB-VAE model in which one particular latent variable is explicitly supervised to predict steering control, and combined with the remaining latent variables to reconstruct the input image. The model accepts as input a $66 \times 200 \times 3$ RGB image in mini-batches of size $n = 50$. We use a convolutional network encoder, comprised of convolutional and fully connected layers, to compute $Q(z|X)$, the distribution of the latent variables given a data point. The latent space has the encoder outputting $2k$ activations corresponding to $\boldsymbol{\mu} \in \mathbb{R}^k, \boldsymbol{\Sigma} = Diag[\sigma] \succ 0$ used to define the distribution of z. We additionally supervise one of the latent variables to take on the value of the curvature of the vehicle's path. We represent this modified variable as $z_0 = z_{\hat{y}} = \hat{y}$. The DB-VAE is trained using the three-component loss provided in Equation 6.7. We found that $c_1 = 0.033$, $c_2 = 0.1$, $c_3 = 0.001$ yielded a nice trade off in importance between steering control, reconstruction, and KL loss, wherein no one individual loss component overpowers the others during training.

A majority of driving data consists of straight road driving, while turns and curves are vastly underrepresented. End-to-end control network (Bojarski et al., 2016) training often handles this by resampling the training set to place more emphasis on the rarer events (*i.e.* turns). We generalize this notion to the latent space of a VAE model, better exploring the space of both control events and nuisance factors, for not only the steering command but all other underlying features. As with facial detection, by feeding the original training data through the learned model, we estimate the training data distribution $Q(z|X)$ in the latent space. We approximate $Q(z|X)$ as a histogram $\hat{Q}(z|X)$, where z is the output of the Encoder NN corresponding to the input images $x \in X$, and further simplify by utilizing independent histograms $\hat{Q}_i(z_i|X)$ for each latent variable z_i and approximate $\hat{Q}(z|X) \propto \prod_i \hat{Q}_i(z_i|X)$.

Naturally, we would like to train on a higher number of unlikely training examples and drop many samples over-represented in the dataset. We therefore train on a subsampled training set X_{sub} including datapoints x with probability $W(z(x)|X) \propto \prod_i 1/(\hat{Q}_i(z_i(x)|X) + \alpha)$, following Algorithm 7. This procedure yields a subsampled dataset X_{sub}. Training on the subsampled data X_{sub} now forces the model into a choice of parameters that work better in rare cases without strong deterioration of performance for common training examples. Most importantly, the debiasing is not manually specified beforehand but based on *learned* latent variables.

Uncovering Latent Biases in Driving

In this subsection, we analyze the resulting latent space that our DB-VAE model learned. We start by gauging the underlying meaning of each of the latent variables,

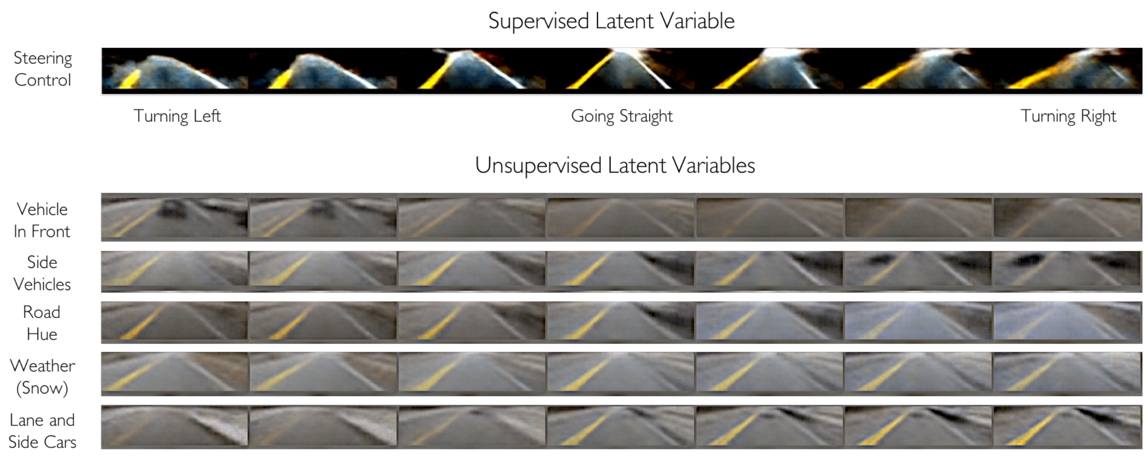


Figure 6-6: **Latent variable perturbation**. A selection of six learned latent variables with associated interpretable descriptors (left labels). Images along the x-axis (from left to right) were generated by linearly perturbing the latent vector encoding along that single latent dimension. While steering command (top) was a supervised latent variable, all others (bottom five) were entirely unsupervised and learned by the model from the dataset.

performing the same style of perturbation experiment as Figure 6-3. The results for an exemplary set of latent variables. for a DB-VAE trained on driving data for the steering wheel control task, is shown in Figure 6-6. By identifying latent variable representations, we can immediately observe what the network sees and explain how the corresponding steering control is derived. We observe that the network is able to generate intuitive representations for lane markings and additional environmental structures, such as other surrounding vehicles and weather, without ever actively being told to do so.

Debiasing End-to-end Driving Models

We evaluate the effect of debiasing during training by training two models, one without debiasing the dataset for inherent latent imbalances and once again now subsampling our dataset to reduce these over-represented (*i.e.* biased) samples. On every epoch we sample only 50% of the dataset for training while the remaining data is discarded. Figure 6-7 illustrates the loss evolution throughout both training schemes. We note that the loss is computed on the original data distribution (and not the

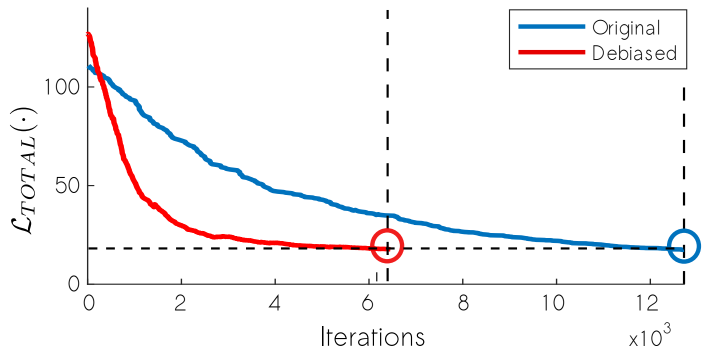


Figure 6-7: **Accelerated training with debiasing**. Comparison of training loss evolution with/without automated debiasing.

subsampled distribution), since we ultimately only care about our performance on the original data. Debiasing the training pipeline allows the model to focus on events that are typically more rare and inherently more difficult to learn from since they occur less frequently. This debiasing procedure results in training that is more data efficient, using only 50% of the data, and also faster than standard training. Figure 6-7 shows a minimum loss of 20 achieved after roughly half as many training iterations compared to training on the original data distribution. These results highlight that our algorithm enables more efficient training and debiasing of end-to-end driving models.

6.4 Discussion

We have developed a novel, tunable debiasing algorithm to adjust the respective sampling probabilities of individual data points while training. By learning the underlying latent variables in an entirely unsupervised manner, we can scale our approach to large datasets and debias for latent features without ever hand labeling them in our training set. We demonstrate results of our algorithm on two real-world applications: mitigating bias in facial detection systems and handling underrepresented examples in autonomous driving.

In facial detection, given a biased training dataset, our debiased models show increased classification accuracy and decreased categorical bias across race and gender, compared to a standard classifier. In autonomous driving, we demonstrate that adaptively sampling the dataset effectively removes overrepresented latent regions and results in 2× training speedups in empirical tests.

Safety-critical applications of AI, such as facial detection and autonomous driving, demand the ability to learn effectively from underrepresented regions as well as guarantees on the fairness of these algorithms. We envision that the proposed approach will serve as a tool to enhance the generalizability, robustness, and fairness of machine learning algorithms for safety-critical decision making and autonomous control.

6.5 Scope and Limitations

Our algorithm provides a generalizable method for automatically uncovering representation disparities in the feature space and mitigating biases that may consequently result. We envision that this method will be widely applicable across predictive tasks and safety-critical applications, including those in autonomous control, as presented here. There are several lines of future work that warrant further investigation.

First, while here we utilize a VAE architecture to learn the latent structure of a dataset, the modularity of our algorithm allows for incorporation of other methods for density estimation. Different density estimation methods may prove more performant or more amenable to different tasks or applications, for example in considering time-series models or learning from discrete feature sets. In addition, future analysis is required to systematically assess the susceptibility of the method to nuisance variation that may be underrepresented in the data and in the latent space. Finally, while here we utilize the learned latent distributions to perform data resampling during training, future efforts could utilize this information to guide data generation for the purpose of training set augmentation.

Fundamentally, the work described in this chapter provides a mechanism for assessing feature representation disparities, identifying rare events, and then adapting the training process itself based on this information. Underrepresented regions and rare events are instances in which we expect the model to be less confident in its prediction; the debiasing algorithm presented here attempts to lessen the uncertainty around these predictions (*i.e.*, increase confidence) by upsampling these instances in training. This concept underscores the importance of grounded methods to identify sources of uncertainty in deep learning models and to ultimately adjust learning, inference, or deployment procedures to handle those uncertainties effectively. Chapter 7 explores this aim systematically, wherein we describe a novel algorithm for calibrated single-shot uncertainty quantification in deep neural networks for regression tasks.

Part III

Deploy

Chapter 7

Probabilistic Modeling and Uncertainty Awareness

7.1 Introduction

Neural networks (NNs) are being deployed in safety critical domains, where calibrated, robust, and efficient measures of uncertainty are crucial for wide-scale adoption. Namely, regression-based NNs are prominent in several domains including computer vision (Godard et al., 2017) to robotics and control (Amini et al., 2019a; Bojarski et al., 2016). Furthermore, precise and calibrated uncertainty estimates are useful for interpreting confidence, capturing domain shift of out-of-distribution (OOD) test samples, and recognizing when the model is likely to fail.

There are two axes of NN uncertainty that can be modeled: (1) uncertainty in the data, called aleatoric uncertainty, and (2) uncertainty in the prediction, called epistemic uncertainty. While metrics of aleatoric uncertainty can be derived directly from data, there exist several approaches for estimating epistemic uncertainty, such as Bayesian NNs, which place probabilistic priors over network weights and use sampling to approximate output variance (Kendall and Gal, 2017). However, Bayesian NNs face several limitations, including the intractability of directly inferring the posterior distribution of the weights given data, the requirement and computational expense of sampling during inference, and the question of how to choose a weight prior.

In contrast, evidential deep learning formulates learning as an evidence acquisition process (Sensoy et al., 2018; Malinin and Gales, 2018). Every training example adds support to a learned higher-order, *evidential* distribution. Sampling from this distribution yields instances of lower-order likelihood functions from which the data was drawn. Instead of placing priors on network weights, as is done in Bayesian NNs, evidential approaches place priors directly over the likelihood function. By training a neural network to output the hyperparameters of the higher-order evidential distribution, a grounded representation of both epistemic and aleatoric uncertainty can then be learned without the need for sampling.

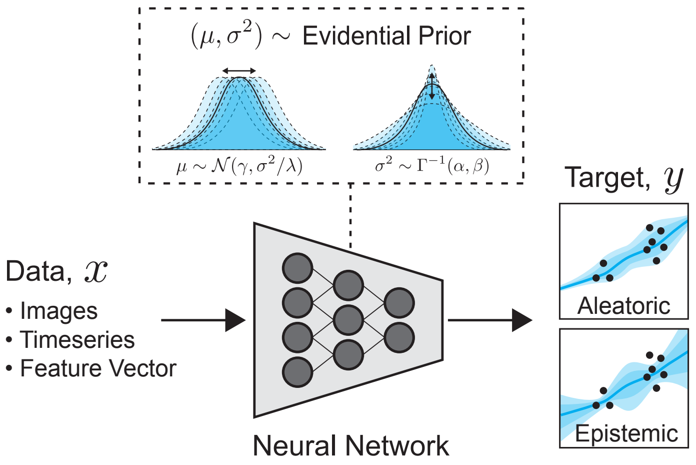
To date, evidential deep learning has been targeted towards discrete classification problems (Sensoy et al., 2018; Malinin and Gales, 2018; Joo et al., 2020) and has either required a well-defined distance measure to a maximally uncertain prior (Sensoy et al., 2018) or relied on training with OOD data to inflate model uncertainty (Malinin and Gales, 2018; Malinin, 2019). In contrast, continuous regression problems present the complexity of lacking a well-defined distance measure to regularize the inferred evidential distribution. Further, pre-defining a reasonable OOD dataset is non-trivial in the majority of applications; thus, methods to obtain calibrated uncertainty on OOD data from only an in-distribution training set are required.

We developed a novel approach that models the uncertainty of regression networks via learned evidential distributions (Figure 7-1). Specifically, we present:

1. A novel and scalable method for learning epistemic and aleatoric uncertainty on regression problems, without sampling during inference or training with out-of-distribution data;

2. Formulation of an evidential regularizer for continuous regression problems, necessary for penalizing incorrect evidence on errors and OOD examples;

3. Evaluation of epistemic uncertainty on benchmark and complex vision regression tasks along with comparisons to state-of-the-art NN uncertainty estimation techniques; and



$(\mu, \sigma^2) \sim$ Evidential Prior
$\mu \sim \mathcal{N}(\gamma, \sigma^2/\lambda)$
$\sigma^2 \sim \Gamma^{-1}(\alpha, \beta)$
Data, x
• Images
• Timeseries
• Feature Vector
Neural Network
Target, y
Aleatoric
Epistemic

Figure 7-1: **Evidential regression** simultaneously learns a continuous target along with aleatoric (data) and epistemic (model) uncertainty. Given an input, the network is trained to predict the parameters of an evidential distribution, which models a higher-order probability distribution over the individual likelihood parameters, (μ, σ^2).

4. Robustness and calibration evaluation on OOD and adversarially perturbed test input data.

7.2 Modelling uncertainties from data

7.2.1 Preliminaries

We consider the following supervised optimization problem. Given a dataset, $\mathcal{D}$, of N paired training examples, $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, we aim to learn a functional mapping f, parameterized by a set of weights, $\boldsymbol{w}$, which approximately solves the following optimization problem:

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}); \quad J(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i(\boldsymbol{w}), \tag{7.1}$$

where $\mathcal{L}_i(\cdot)$ describes a loss function. We focus on deterministic regression problems, which commonly optimize the sum of squared errors, $\mathcal{L}_i(\boldsymbol{w}) = \frac{1}{2}\left\|y_i - f(\boldsymbol{x}_i; \boldsymbol{w})\right\|^2$. In doing so, the model is encouraged to learn the average correct answer for a given input, but does not explicitly model any underlying noise or uncertainty in the data when making its estimation.

7.2.2 Maximum likelihood estimation

One can approach this problem from a maximum likelihood perspective, where we learn model parameters that maximize the likelihood of observing a particular set of training data. In the context of deterministic regression, we assume our targets, y_i, were drawn i.i.d. from a distribution such as a Gaussian with mean and variance parameters $\boldsymbol{\theta} = (\mu, \sigma^2)$. In maximum likelihood estimation (MLE), we aim to learn a model to infer $\boldsymbol{\theta}$ that maximize the likelihood of observing our targets, y, given by $p(y_i|\boldsymbol{\theta})$. This is achieved by minimizing the negative log likelihood loss function:

$$\mathcal{L}_i(\boldsymbol{w}) = -\log p(y_i|\underbrace{\mu, \sigma^2}_{\boldsymbol{\theta}}) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{(y_i-\mu)^2}{2\sigma^2}. \tag{7.2}$$

In learning $\boldsymbol{\theta}$, this likelihood function successfully models the uncertainty in the data, also known as the aleatoric uncertainty. However, our model is oblivious to its predictive epistemic uncertainty (Kendall and Gal, 2017).

In this section, we present a novel approach for estimating the evidence supporting network predictions in regression by directly learning both the aleatoric uncertainty present in the data as well as the model's underlying epistemic uncertainty. We achieve this by placing higher-order prior distributions over the learned parameters governing the distribution from which our observations are drawn.

7.3 Evidential uncertainty for regression

7.3.1 Problem setup

We consider the problem where the observed targets, y_i, are drawn i.i.d. from a Gaussian distribution, as in standard MLE (Section 7.2.2), but now with *unknown mean and variance* (μ, σ^2), which we seek to also probabilistically estimate. We model this by placing a prior distribution on (μ, σ^2). If we assume observations are

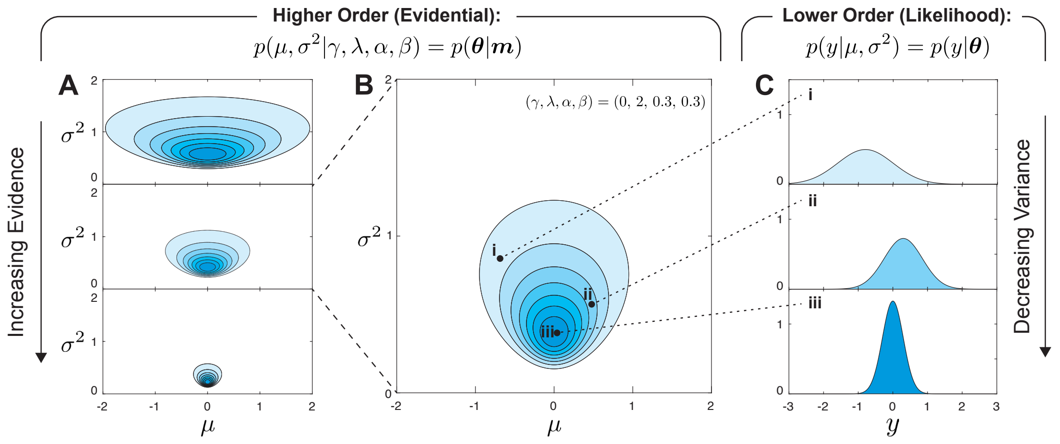


Figure 7-2: **Normal Inverse-Gamma distribution.** Different realizations of our evidential distribution (A) correspond to different levels of confidences in the parameters (e.g. μ, σ^2). Sampling from a single realization of a higher-order evidential distribution (B), yields lower-order likelihoods (C) over the data (e.g. $p(y|\mu,\sigma^2)$). Darker shading indicates higher probability mass. We aim to learn a model that predicts the target, y, from an input, x, with an evidential prior imposed on our likelihood to enable uncertainty estimation.

drawn from a Gaussian, in line with assumptions Section 7.2.2, this leads to placing a Gaussian prior on the unknown mean and an Inverse-Gamma prior on the unknown variance:

$$\begin{gathered}(y_1,\ldots,y_N) \sim \mathcal{N}(\mu,\sigma^2) \\ \mu \sim \mathcal{N}(\gamma, \sigma^2\upsilon^{-1}) \qquad\qquad \sigma^2 \sim \Gamma^{-1}(\alpha,\beta).\end{gathered} \tag{7.3}$$

where $\Gamma(\cdot)$ is the gamma function, $\boldsymbol{m} = (\gamma,\upsilon,\alpha,\beta)$, and $\gamma \in \mathbb{R}$, $\upsilon > 0$, $\alpha > 1$, $\beta > 0$.

Our aim is to estimate a posterior distribution $q(\mu,\sigma^2) = p(\mu,\sigma^2|y_1,\ldots,y_N)$. To obtain an approximation for the true posterior, we assume[1] that the estimated distribution can be factorized (Parisi, 1988) such that $q(\mu,\sigma^2) = q(\mu)\,q(\sigma^2)$. Thus, our approximation takes the form of the Gaussian conjugate prior, the Normal Inverse-

[1]In practice, we can empirically measure how limiting this assumption is by looking at the ability to fit samples from the true posterior (*e.g.*, obtained from Deep Ensembles, *etc.*) to the approximated posterior $q(\mu,\sigma^2)$).

Gamma (`NIG`) distribution:

$$p(\underbrace{\mu, \sigma^2}_{\boldsymbol{\theta}} \,|\, \underbrace{\gamma, \upsilon, \alpha, \beta}_{\boldsymbol{m}}) = \frac{\beta^\alpha \sqrt{\upsilon}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta + \upsilon(\gamma-\mu)^2}{2\sigma^2}\right\}. \tag{7.4}$$

A popular interpretation of the parameters of this conjugate prior distribution is in terms of "virtual-observations" in support of a given property (Jordan, 2009). For example, the mean of a `NIG` distribution can be intuitively interpreted as being estimated from υ virtual-observations with sample mean γ, while its variance is estimated from α virtual-observations with sample mean γ and sum of squared deviations 2β. Following from this interpretation, we define the total evidence, Φ, of our evidential distributions as the sum of all inferred virtual-observations counts: $\Phi = 2\upsilon + \alpha$.

Drawing a sample $\boldsymbol{\theta}_j$ from the `NIG` distribution yields a single instance of our likelihood function, namely $\mathcal{N}(\mu_j, \sigma_j^2)$. Thus, the `NIG` hyperparameters, $(\gamma, \upsilon, \alpha, \beta)$, determine not only the location but also the dispersion concentrations, or uncertainty, associated with our inferred likelihood function. Therefore, we can interpret the `NIG` distribution as the *higher-order*, *evidential* distribution on top of the unknown *lower-order* likelihood distribution from which observations are drawn.

For example, in Figure 7-2A we visualize different evidential `NIG` distributions with varying model parameters. We illustrate that by increasing the evidential parameters (i.e., υ, α) of this distribution, the p.d.f. becomes tightly concentrated about its inferred likelihood function. Considering a single parameter realization of this higher-order distribution (Figure 7-2B), we can subsequently sample many lower-order realizations of our likelihood function, as shown in Figure 7-2C.

Our evidential deep learning algorithm uses neural networks to infer, given an input, the hyperparameters, $\boldsymbol{m}$, of this higher-order, evidential distribution. This approach presents several distinct advantages. First, our method enables simultaneous learning of the desired regression task, along with aleatoric and epistemic uncertainty estimation, by enforcing evidential priors and without leveraging any out-of-distribution data during training. Second, since the evidential prior is a higher-order `NIG` distribution, the maximum likelihood Gaussian can be computed analytically from the expected values of the (μ, σ^2) parameters, without the need for sampling. Third, we can effectively estimate the epistemic uncertainty (i.e., model uncertainty) associated with the network's prediction by simply evaluating the variance of our inferred evidential distribution.

7.3.2 Prediction and uncertainty estimation

The aleatoric uncertainty, also referred to as statistical or data uncertainty, is representative of unknowns that differ each time we run the same experiment. The epistemic (or model) uncertainty, describes the estimated uncertainty in the prediction. Given a `NIG` distribution, we can compute the prediction, aleatoric uncertainty, and epistemic uncertainty as

$$\underbrace{\mathbb{E}[\mu] = \gamma}_{\text{prediction}}, \qquad \underbrace{\mathbb{E}[\sigma^2] = \tfrac{\beta}{\alpha-1}}_{\text{aleatoric}}, \qquad \underbrace{\operatorname{Var}[\mu] = \tfrac{\beta}{\upsilon(\alpha-1)}}_{\text{epistemic}}. \tag{7.5}$$

Note that $\operatorname{Var}[\mu] = \mathbb{E}[\sigma^2]/\upsilon$, which is expected as υ is one of our two evidential virtual-observation counts.

7.3.3 Learning the evidential distribution

Having formalized the use of an evidential distribution to capture both aleatoric and epistemic uncertainty, we next describe our approach for learning a model to output the hyperparameters of this distribution. For clarity, we structure the learning process as a multi-task learning problem, with two distinct parts: (1) acquiring or maximizing model evidence in support of our observations and (2) minimizing evidence or inflating uncertainty when the prediction is wrong. At a high level, we can think of (1) as a way of fitting our data to the evidential model while (2) enforces a prior to remove incorrect evidence and inflate uncertainty.

(1) Maximizing the model fit. From Bayesian probability theory, the "model evidence", or marginal likelihood, is defined as the likelihood of an observation, y_i, given the evidential distribution parameters $\boldsymbol{m}$ and is computed by marginalizing over the likelihood parameters $\boldsymbol{\theta}$:

$$p(y_i|\boldsymbol{m}) = \frac{p(y_i|\boldsymbol{\theta}, \boldsymbol{m})p(\boldsymbol{\theta}|\boldsymbol{m})}{p(\boldsymbol{\theta}|y_i, \boldsymbol{m})} = \int_{\sigma^2=0}^{\infty} \int_{\mu=-\infty}^{\infty} p(y_i|\mu, \sigma^2)p(\mu, \sigma^2|\boldsymbol{m})\, \mathrm{d}\mu\, \mathrm{d}\sigma^2 \qquad (7.6)$$

The model evidence is, in general, not straightforward to evaluate since computing it involves integrating out the dependence on latent model parameters. However, in the case of placing a `NIG` evidential prior on our Gaussian likelihood function an analytical solution does exist:

$$p(y_i|\boldsymbol{m}) = \mathrm{St}\left(y_i; \gamma, \frac{\beta(1+\upsilon)}{\upsilon\, \alpha}, 2\alpha\right). \qquad (7.7)$$

where $\mathrm{St}\left(y; \mu_{\mathrm{St}}, \sigma^2_{\mathrm{St}}, \upsilon_{St}\right)$ is the Student-t distribution evaluated at y with location μ_{St}, scale σ^2_{St}, and υ_{St} degrees of freedom. We denote the loss, $\mathcal{L}_i^{\mathrm{NLL}}(\boldsymbol{w})$, as the negative logarithm of model evidence

$$\mathcal{L}_i^{\mathrm{NLL}}(\boldsymbol{w}) = \tfrac{1}{2}\log\left(\tfrac{\pi}{\upsilon}\right) - \alpha\log(\Omega) + \left(\alpha + \tfrac{1}{2}\right)\log((y_i - \gamma)^2\upsilon + \Omega) + \log\left(\tfrac{\Gamma(\alpha)}{\Gamma(\alpha+\frac{1}{2})}\right) \qquad (7.8)$$

where $\Omega = 2\beta(1+\upsilon)$. This loss provides an objective for training a NN to output parameters of a `NIG` distribution to fit the observations by maximizing the model evidence.

(2) Minimizing evidence on errors. Next, we describe how to regularize training by applying an incorrect evidence penalty (i.e., high uncertainty prior) to try to minimize evidence on incorrect predictions. This has been demonstrated with success in the classification setting where non-misleading evidence is removed from the posterior, and the uncertain prior is set to a uniform Dirichlet (Sensoy et al., 2018). The analogous minimization in the regression setting involves $KL[\,p(\boldsymbol{\theta}|\boldsymbol{m})\,||\,p(\boldsymbol{\theta}|\tilde{\boldsymbol{m}})\,]$, where $\tilde{\boldsymbol{m}}$ are the parameters of the uncertain `NIG` prior with zero evidence (i.e., $\{\alpha, \upsilon\} = 0$). Unfortunately, the KL between any `NIG` and the zero evidence `NIG` prior is undefined. Furthermore, this loss should not be enforced everywhere, but instead specifically where the posterior is "misleading". Past works in classification (Sensoy et al., 2018) accomplish this by using the ground truth likelihoood classification (the one-hot encoded labels) to remove "non-misleading" evidence. However, in regression, it is not possible to penalize evidence everywhere except our single label point estimate, as this space is infinite and unbounded. Thus, these previous approaches for regularizing evidential learning are not applicable.

Formulation and implementation of the evidential loss for regression

To address these challenges in the regression setting, we formulate a novel evidence regularizer, $\mathcal{L}_i^{\mathrm{R}}$, scaled on the error of the i-th prediction,

$$\mathcal{L}_i^{\mathrm{R}}(\boldsymbol{w}) = |y_i - \mathbb{E}[\mu_i]| \cdot \Phi = |y_i - \gamma| \cdot (2\upsilon + \alpha). \tag{7.9}$$

This loss imposes a penalty whenever there is an error in the prediction and scales with the total evidence of our inferred posterior. Conversely, large amounts of predicted evidence will not be penalized as long as the prediction is close to the target. A naïve alternative to directly penalizing evidence would be to soften the zero-evidence prior to instead have ϵ-evidence such that the KL is finite and defined. However, doing so results in hypersensitivity to the selection of ϵ, as it should be small yet $KL \to \infty$ as $\epsilon \to 0$. In the remainder of this chapter, we will demonstrate the added value of our evidential regularizer through ablation analysis (Section 7.4.1), establish the limitations of the soft KL regularizer

The total loss, $\mathcal{L}_i(\boldsymbol{w})$, consists of the two loss terms for maximizing the model fit and regularizing evidence, scaled by a regularization coefficient, λ:

$$\mathcal{L}_i(\boldsymbol{w}) = \mathcal{L}_i^{\mathrm{NLL}}(\boldsymbol{w}) + \lambda\, \mathcal{L}_i^{\mathrm{R}}(\boldsymbol{w}). \tag{7.10}$$

Here, λ trades off uncertainty inflation with model fit. Setting $\lambda = 0$ yields an over-confident estimate while setting λ too high results in over-inflation. In practice, our NN is trained to output the parameters, $\boldsymbol{m}$, of the evidential distribution: $\boldsymbol{m}_i = f(\boldsymbol{x}_i; \boldsymbol{w})$. Since $\boldsymbol{m}$ is composed of 4 parameters, f has 4 output neurons for every target y. We enforce the constraints on $(\upsilon, \alpha, \beta)$ with a `softplus` activation (and

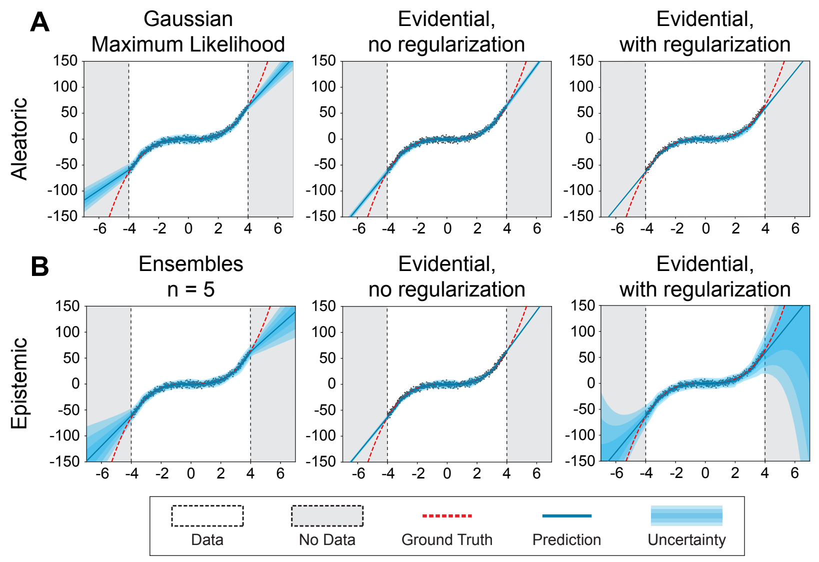


Figure 7-3: **Toy uncertainty estimation.** Aleatoric (A) and epistemic (B) uncertainty estimates on the dataset $y = x^3 + \epsilon$, $\epsilon \sim \mathcal{N}(0, 3)$. Regularized evidential regression (right) enables precise prediction within the training regime and conservative epistemic uncertainty estimates in regions with no training data. Baseline results are also illustrated.

additional $+1$ added to α since $\alpha > 1$). Linear activation is used for $\gamma \in \mathbb{R}$.

7.4 Experiments

7.4.1 Predictive accuracy and uncertainty benchmarking

We first qualitatively compare the performance of our approach against a set of baselines on a one-dimensional cubic regression dataset (Figure 7-3). Following (Hernández-Lobato and Adams, 2015; Lakshminarayanan et al., 2017), we train models on $y = x^3 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 3)$ within ± 4 and test within ± 6. We compare aleatoric (A) and epistemic (B) uncertainty estimation for baseline methods (left), evidence without regularization (middle), and with regularization (right). Gaussian MLE (Nix and Weigend, 1994) and Ensembling (Lakshminarayanan et al., 2017) are used as respective baseline methods. All aleatoric methods (A) accurately capture uncertainty within the training distribution, as expected. Epistemic uncertainty (B) captures uncertainty on OOD data; our proposed evidential method estimates uncertainty

Dataset	**RMSE** Dropout	Ensembles	Evidential	**NLL** Dropout	Ensembles	Evidential	**Inference Speed (ms)** Dropout	Ensemble	Evidential
Boston	**2.97 ± 0.19**	**3.28 ± 1.00**	**3.06 ± 0.16**	**2.46 ± 0.06**	**2.41 ± 0.25**	**2.35 ± 0.06**	3.24	3.35	**0.85**
Concrete	**5.23 ± 0.12**	6.03 ± 0.58	5.85 ± 0.15	3.04 ± 0.02	3.06 ± 0.18	**3.01 ± 0.02**	2.99	3.43	**0.94**
Energy	**1.66 ± 0.04**	2.09 ± 0.29	2.06 ± 0.10	1.99 ± 0.02	**1.38 ± 0.22**	**1.39 ± 0.06**	3.08	3.80	**0.87**
Kin8nm	0.10 ± 0.00	**0.09 ± 0.00**	**0.09 ± 0.00**	-0.95 ± 0.01	-1.20 ± 0.02	**-1.24 ± 0.01**	3.24	3.79	**0.97**
Naval	0.01 ± 0.00	**0.00 ± 0.00**	**0.00 ± 0.00**	-3.80 ± 0.01	**-5.63 ± 0.05**	**-5.73 ± 0.07**	3.31	3.37	**0.84**
Power	**4.02 ± 0.04**	**4.11 ± 0.17**	4.23 ± 0.09	**2.80 ± 0.01**	**2.79 ± 0.04**	**2.81 ± 0.07**	2.93	3.36	**0.85**
Protein	**4.36 ± 0.01**	4.71 ± 0.06	4.64 ± 0.03	2.89 ± 0.00	2.83 ± 0.02	**2.63 ± 0.00**	3.45	3.68	**1.18**
Wine	**0.62 ± 0.01**	**0.64 ± 0.04**	**0.61 ± 0.02**	**0.93 ± 0.01**	**0.94 ± 0.12**	**0.89 ± 0.05**	3.00	3.32	**0.86**
Yacht	**1.11 ± 0.09**	**1.58 ± 0.48**	**1.57 ± 0.56**	1.55 ± 0.03	**1.18 ± 0.21**	**1.03 ± 0.19**	2.99	3.36	**0.87**

Table 7.1: **Benchmark regression tests.** RMSE, negative log-likelihood (NLL), and inference speed for dropout sampling (Gal and Ghahramani, 2016a), model ensembling (Lakshminarayanan et al., 2017), and evidential regression. Top scores for each metric and dataset are bolded (within statistical significance), $n = 5$ for sampling baselines. Evidential models outperform baseline methods for NLL and inference speed on all datasets.

appropriately and grows on OOD data, without dependence on sampling.

Additionally, we compare our approach to baseline methods for NN predictive uncertainty estimation on real world datasets used in (Hernández-Lobato and Adams, 2015; Lakshminarayanan et al., 2017; Gal and Ghahramani, 2016a). We evaluate our proposed evidential regression method against results presented for model ensembles (Lakshminarayanan et al., 2017) and dropout (Gal and Ghahramani, 2016a) based on root mean squared error (RMSE), negative log-likelihood (NLL), and inference speed. Table 7.1 indicates that even though, unlike the competing approaches, the loss function for evidential regression does not explicitly optimize accuracy, it remains competitive with respect to RMSE while being the top performer on all datasets for NLL and speed. To give the two baseline methods maximum advantage, we parallelize their sampled inference ($n = 5$). Dropout requires additional multiplications with the sampled mask, resulting in slightly slower inference compared to ensembles, whereas evidence only requires a single forward pass and network.

7.4.2 Monocular depth estimation

After establishing benchmark comparison results, in this subsection we demonstrate the scalability of our evidential learning approach by extending it to the complex, high-dimensional task of depth estimation. Monocular end-to-end depth estimation is a central problem in computer vision and involves learning a representation of depth directly from an RGB image of the scene. This is a challenging learning task as the target y is very high-dimensional, with predictions at every pixel.

Our training data consists of over 27k RGB-to-depth, $H \times W$, image pairs of indoor scenes (e.g. kitchen, bedroom, etc.) from the NYU Depth v2 dataset (Nathan Silberman and Fergus, 2012). We train a U-Net style NN (Ronneberger et al., 2015) for inference and test on a disjoint test-set of scenes. The final layer outputs a single $H \times W$ activation map in the case of vanilla regression, dropout, and ensembling. Spatial dropout uncertainty sampling (Amini et al., 2018c; Tompson et al., 2015) is used for the dropout implementation. Evidential regression outputs four of these output maps, corresponding to $(\gamma, \upsilon, \alpha, \beta)$, with constraints according to Section 7.3.3.

We evaluate the models in terms of their accuracy and their predictive epistemic uncertainty on unseen test data. Figure 7-4A visualizes the predicted depth, absolute error from ground truth, and predictive entropy across two randomly picked test images. Ideally, a strong epistemic uncertainty measure would capture errors in the prediction (i.e., roughly correspond to where the model is making errors). Compared to the gold-standard methods for epistemic uncertainty estimation in NN, namely dropout sampling and model ensembling, the evidential methods captures the depth errors while providing clear and localized predictions of confidence. In general, dropout drastically underestimates the amount of uncertainty present, while ensembling occasionally overestimates the uncertainty. Figure 7-4B shows how each model performs as pixels with uncertainty greater than certain thresholds are removed. Evidential models exhibit strong performance, as error steadily decreases with increasing

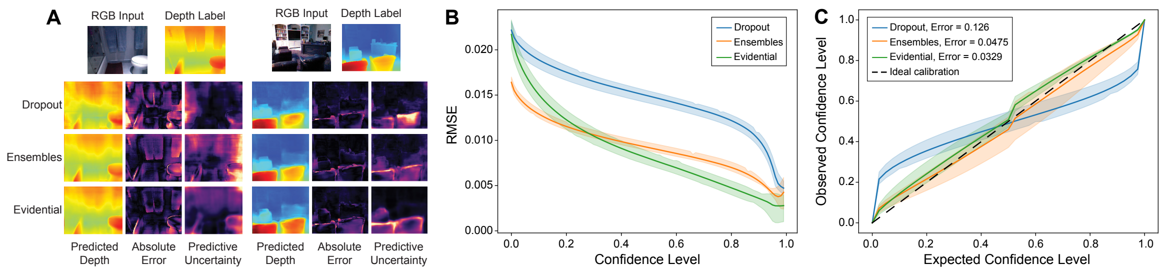


Figure 7-4: **Epistemic uncertainty in depth estimation.** (A) Example pixel-wise depth predictions and uncertainty for each model. (B) Relationship between prediction confidence level and observed error; a strong inverse trend is desired. (C) Model uncertainty calibration (Kuleshov et al., 2018); (ideal: $y = x$). Inset shows calibration errors.

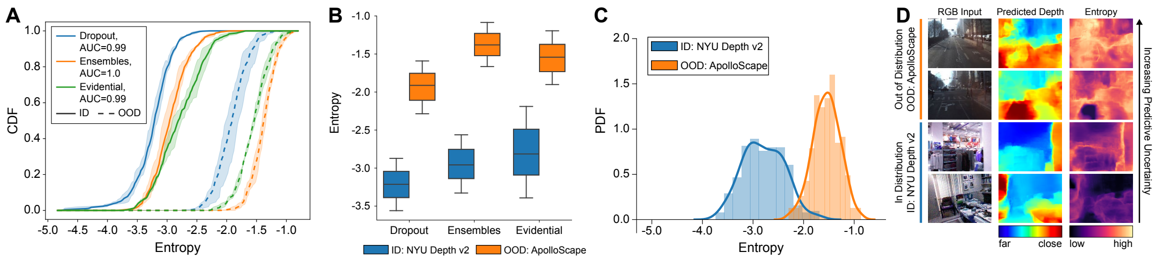


Figure 7-5: **Uncertainty on out-of-distribution (OOD) data.** Evidential models estimate low uncertainty (entropy) on in-distribution (ID) data and inflate uncertainty on OOD data. (A) Cumulative density function (CDF) of ID and OOD entropy for tested methods. OOD detection assessed via AUC-ROC. (B) Uncertainty (entropy) comparisons across methods. (C) Full density histograms of entropy estimated by evidential regression on ID and OOD data, along with sample images (D). All data has not been seen during training.

confidence.

Figure 7-4C additionally evaluates the calibration of our uncertainty estimates. Calibration curves are computed according to (Kuleshov et al., 2018), and ideally follows $y = x$ to represent, for example, that a target falls in a 90% confidence interval approximately 90% of the time. Again, we see that dropout overestimates confidence when considering low confidence scenarios (calibration error: 0.126). Ensembling exhibits better calibration error (0.048) but is still outperformed by the proposed evidential method (0.033).

In addition to epistemic uncertainty experiments, we also evaluate aleatoric uncertainty estimates, with comparisons to Gaussian MLE learning. Since evidential models fit the data to a higher-order Gaussian distribution, it is expected that they can also accurately learn aleatoric uncertainty (as is also shown in (Sensoy et al., 2018; Gurevich and Stuke, 2020)). Therefore, we focus the remainder of the results on evaluating the harder task of epistemic uncertainty estimation in the context of out-of-distribution (OOD) and adversarially perturbed samples.

7.4.3 Out-of distribution testing

A key use of uncertainty estimation is to understand when a model is faced with test samples that fall out-of-distribution (OOD) or when the model's output cannot be trusted. In this subsection, we investigate the ability of evidential models to capture increased epistemic uncertainty on OOD data, by testing on images from ApolloScape (Huang et al., 2018), an OOD dataset of diverse outdoor driving. It is crucial to note here that related methods such as Prior Networks in classification (Malinin and Gales, 2018, 2019) explicitly require OOD data during training to supervise instances of high uncertainty. Our evidential method, like Bayesian NNs, does not have this limitation and sees only in distribution (ID) data during training.

For each method, we feed in the ID and OOD test sets and record the mean predicted entropy for every test image. Figure 7-5A shows the cumulative density function (CDF) of entropy for each of the methods and test sets. A distinct positive shift in the entropy CDFs can be seen for evidential models on OOD data and is competitive across methods. Figure 7-5B summarizes these entropy distributions as interquartile boxplots to again show clear separation in the uncertainty distribution on OOD data. We focus on the distribution from our evidential models in Figure 7-5C and provide sample predictions (ID and OOD) in Figure 7-5D. These results show that evidential models, without training on OOD data, capture increased uncertainty on OOD data on par with epistemic uncertainty estimation baselines.

Robustness to adversarial samples

Next, we consider the extreme case of OOD detection where the inputs are adversarially perturbed to inflict error on the predictions. We compute adversarial perturbations to our test set using the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), with increasing scales, ϵ, of noise. Note that the purpose of this experiment is not to propose a defense for state-of-the-art adversarial attacks, but rather to demonstrate that evidential models accurately capture increased predictive uncertainty on samples which have been adversarially perturbed. Figure 7-6A confirms that the absolute error of all methods increases as adversarial noise is added. We also observe a positive effect of noise on our predictive uncertainty estimates in Figure 7-6B. Furthermore, we observe that the entropy CDF steadily shifts towards higher uncertainties as the noise in the input sample increases (Figure 7-6C).

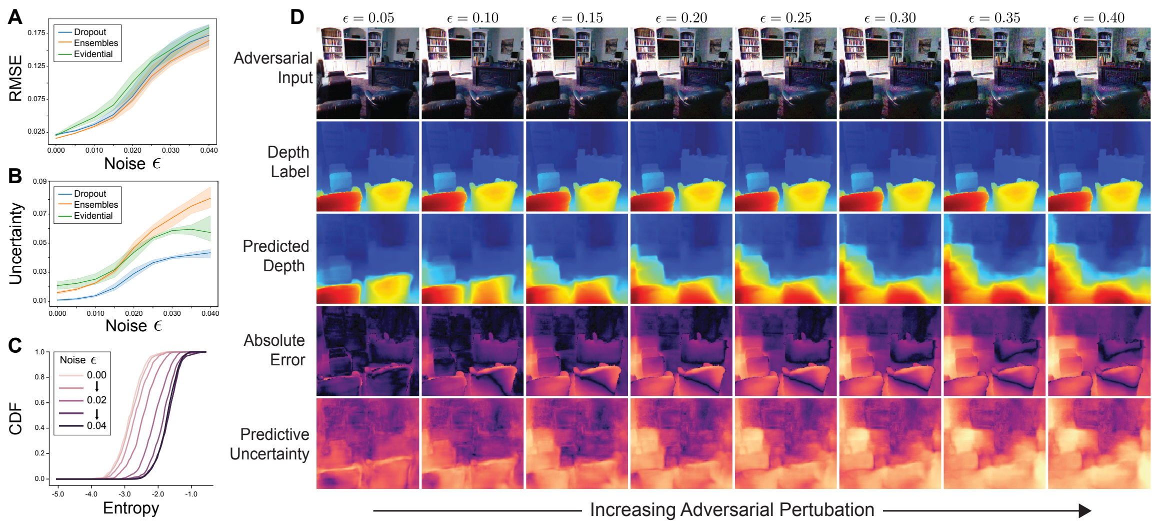


Figure 7-6: **Evidential robustness under adversarial noise.** Relationship between adversarial noise ϵ and predictive error (A) and estimated epistemic uncertainty (B). (C) CDF of entropy estimated by evidential regression under the presence of increasing ϵ. (D) Visualization of the effects of increasing adversarial pertubation on the predictions, error, and uncertainty for evidential regression. Results of sample test-set image are shown.

The robustness of evidential uncertainty against adversarial perturbations is visualized in greater detail in Figure 7-6D, which illustrates the predicted depth, error, and estimated pixel-wise uncertainty as we perturb the input image with greater amounts of noise (left to right). Not only does the predictive uncertainty steadily increase with increasing noise, but the spatial concentrations of uncertainty throughout the image also maintain tight correspondence with the error. These results demonstrate that evidential deep learning for regression can effectively capture uncertainty linked to adversarial perturbation and, furthermore, that these uncertainties are calibrated to the degree of perturbation.

7.5 Discussion

We have developed a novel method for learning uncertainty in regression problems by placing evidential priors over the likelihood output. We first formulate these higher-order evidential distributions and formulate derivations of the target prediction, aleatoric uncertainty, and epistemic uncertainty from the moments of the associated evidential distribution. We demonstrate that our algorithm enables combined target prediction together with direct aleatoric and epistemic uncertainty estimation, scales to complex vision tasks, and achieves calibrated uncertainty estimates on OOD data and adversarially-perturbed inputs. A key advantage of our method is its efficiency, as it does not require sampling, as well as its flexibility to operate with existing task datasets, as it does not rely on augmented OOD training data to calibrate uncertainty. The efficiency, scalablity, and calibration of our approach could enable the precise and fast uncertainty estimation required for robust NN deployment in safety-critical prediction domains.

7.6 Scope and Limitations

Our algorithm provides a generalizable uncertainty estimation method widely applicable across regression tasks, including temporal forecasting (Greff et al., 2016), property prediction (Soleimany et al., 2021), and control learning (Amini et al., 2019a; Levine et al., 2016). While evidential deep learning provides key advantages over existing methods for uncertainty quantification in neural models deployed for regression tasks, there are several considerations that motivate opportunities for future work.

First, our method's primary limitations are in tuning the regularization coefficient and in effectively removing non-misleading evidence when calibrating the uncertainty. While dual-optimization formulations (Zhao et al., 2018) could be explored for balancing regularization, we believe further investigation is warranted to discover alternative ways to remove non-misleading evidence. In addition, future analysis using other choices of the variance prior distribution, such as the log-normal or the heavy-tailed log-Cauchy distribution, will be critical to determine the effects of the choice of prior on the estimated likelihood parameters. Finally, while here we have established the foundations of a new method for epistemic uncertainty quantification in NNs deployed for regression tasks, we have yet to demonstrate the utility of the evidential method to guide learning, inform decisions at test time, or create uncertainty-aware models capable of identifying and gracefully handling out-of-distribution or unexpected events at deployment.

Indeed, this last consideration underscores the critical need for systematic uncertainty estimation methods to be placed and evaluated in larger decision and control systems, where predictive errors can pose immediate threats to downstream tasks. This is especially true for systems deployed in safety-critical domains, such as in autonomous vehicle control (Amini et al., 2018b) – these systems must not only be accurate and efficient, but also highly robust. For example, in the control setting, capturing the model's uncertainty associated with each prediction could enable greater robustness by accounting for potential ambiguities in future behavior as well as unexpected out-of-distribution events.

The work described in this chapter provides a foundational algorithm, deep evidential regression, for uncertainty estimation in neural networks. Moving forward, it will be critical to assess the broad utility of this evidential learning algorithm for robust, uncertainty-aware decision making. Autonomous control systems must be able to deal with high amounts of uncertainty in their environment, sensory data, and decision-making. To this end, uncertainty estimation and probabilistic reasoning methods must be systematically synergized with control learning algorithms to increase the robustness of autonomous systems, for example to environmental perturbations, to unexpected sensor failures, or in the face of observational noise. Chapter 8 furthers this aim and presents the development of novel uncertainty-aware decision-making and control algorithms that incorporate metrics of uncertainty or probabilistic reasoning to achieve greater robustness in challenging control scenarios.

Chapter 8

Algorithms for Uncertainty-aware Decision Making

8.1 Introduction

Humans have an innate ability to reason about the high-level structure of their environment even under severe uncertainty and limited observation. They use this ability to relate high-level instructions to concrete control commands, as well as to better localize themselves even without concrete localization information. Inspired by these abilities, we aim to build a learning engine that enables a robot agent to learn how propagate its own uncertainties–both in its sensory system and predictive processes–to resolve ambiguities and achieve more reliable and robust decision making. Specifically, we focus on the problems of navigation, localization, and temporal control within an end-to-end autonomous driving system.

For navigation and localization, coarse grained maps afford us a higher-level of understanding of the environment, both because of their expanded scope, but also due to their distilled nature. This allows for reasoning about the low-level control within a hierarchical framework with long-term goals (Sutton et al., 1999), as well as localizing, preventing drift, and performing loop-closure when possible. We note that unlike much of the work in place recognition and loop closure, we are looking at a higher level of matching, where the vehicle is matching intersection and road patterns to the coarse scale geometry found in the map. This offers the potential to handle different appearances and small variations in the scene structure, or even unknown fine-scale geometry, as long as the overall road network structure matches the expected structures.

While end-to-end control learning (Bojarski et al., 2016) holds promise due to its easily scalable and adaptable nature, it has a limited capability to handle long-term plans, relating to the nature of imitation learning (Codevilla et al., 2017; Shalev-Shwartz et al., 2016). Some recent methods incorporate maps as inputs (Wei et al., 2017; Hecker et al., 2018) to capture longer term action structure, yet they ignore the uncertainty maps inherently allow us to address – uncertainty about the location, and uncertainty about the longer-term plan.

Real-world deployable robotic systems must not only be accurate and efficient, but also highly robust. End-to-end models typically predict instantaneous control and generally suffer from the high sensitivity to perturbations (*e.g.*, noisy sensory inputs). To address this, a plausible solution is to integrate several consecutive frames as input (Xu et al., 2017). However, this is neither efficient in the 3D domain nor effective in closed-loop control settings, since it requires either modeling recurrence (Hochreiter and Schmidhuber, 1997) or applying 4D convolutions (Choy et al., 2019) to process multiple input frames, which is not computationally affordable nor effective at learning causal control representations (Lechner et al., 2020a). Instead, by training a model to additionally predict *future* control and apply odometry-corrected fusion, actuation commands could potentially be stabilized. Explicit modeling of uncertainty is critical because (1) predictions from the past are usually less confident due to the ambiguity of the future; and (2) unexpected out-of-distribution (OOD) events (*e.g.*, sensor failure) can completely invalidate the model's prediction. Modeling and fusing control predictions using uncertainty could provide increased stability overall and resilience on OOD events.

In this chapter, we address these limitations by developing novel uncertainty-aware deployment algorithms for navigation, localization, and temporal control. First, we develop a method for integrating navigational information with raw sensory data into a single end-to-end variational network, and do so in a way that preserves reasoning about uncertainty. This allows the system to not only learn to navigate complex environments entirely from human perception and navigation data, but also understand when localization or mapping is incorrect, and thus correct for the pose. Our model processes coarse grained, unrouted roadmaps along with forward facing camera images to produce a probabilistic estimate of the different possible low-level steering commands which the robot can execute at that instant, thus reasoning about ambiguity and uncertainty in the environment. Second, we develop a model that integrates estimates of epistemic uncertainty directly into the predictive process of the system and thus quantifies how much trust we can place in any predicted decision. We formulate a novel algorithm for integrating single-shot uncertainty estimates back into the controller of the agent to achiever robustness in the face of entirely out-of-distribution events and surprises during deployment.

8.2 A Variational Model for Navigation and Localization

8.2.1 Control Mixture Model

In this section, we formulate the end-to-end navigation problem and describe the model used in our approach. We use a variational neural network, which takes raw camera images, I, and an image of a noisy, unrouted roadmap, M_U, as input. At the output we attempt to learn a full, parametric probability distribution over road curvature or steering (θ_s) to navigate that instant. We use a Gaussian Mixture Model (GMM) with $K > 0$ modes to describe the possible steering control command, and penalize the $L_{1/2}$ norm of the weights to discourage extra components. Empirically, we chose $K = 3$ since it captured the majority of driving situations encountered. Additionally, the model will optionally output a deterministic control command if a routed version of the map is also provided as input. The overall network can be written separately as two functions, representing the probabilistic (unrouted) and

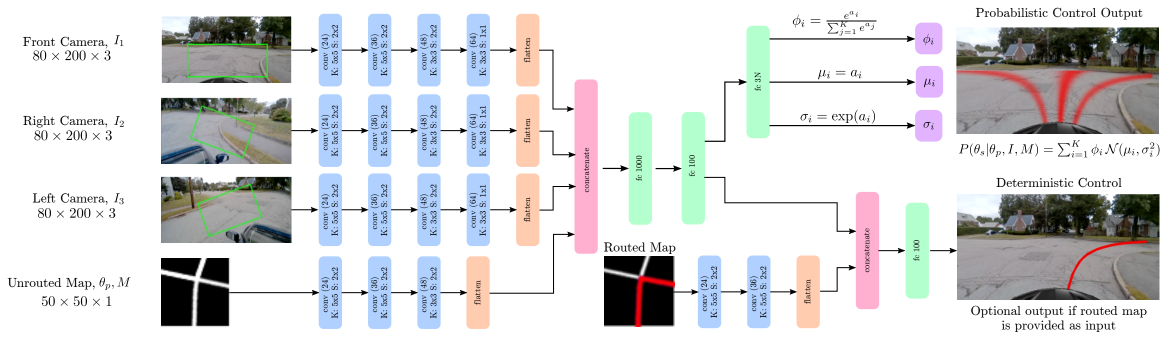


Figure 8-1: **Model architecture overview.** Raw camera images and noisy roadmaps are fed to parallel convolutional pipelines, then merged into fully-connected layers to learn a full parametric Gaussian Mixture Model (GMM) over control. If a routed map is also available, it is merged at the penultimate layer to learn a deterministic control signal for navigation along a provided route. Green rectangles denote the image region provided as input to the network.

deterministic (routed) parts respectively:

$$\{(\phi_i, \mu_i, \sigma_i^2)\}_{i=1}^{K} = f_S(I, M_U, \theta_p), \hat{\theta}_s = f_D(I, M_R, \theta_p),$$

where $\theta_p = [p_x, p_y, p_\alpha]$ is the current pose in the map (position and heading), and $f_S(I, M_U, \theta_p)$, $f_D(I, M_R, \theta_p)$ are network outputs computed by cropping a square region of the relevant map according to θ_p, and feeding it, along with the forward facing images, I, to the network. M_U denotes the *unrouted* map, with only the traversible areas marked, while M_R denotes the *routed* map, containing the desired route highlighted. The deterministic control command is denoted as $\hat{\theta}_s$. We refer to steering command interchangeably as the road curvature: the actual steering angle requires reasoning about road slip and control plant parameters that change between vehicles, making it less suitable for our purpose. Finally, the parameters (i.e. weight, mean, and variance) of the GMM's i-th component are denoted by $(\phi_i, \mu_i, \sigma_i^2)$, which represents the steering control in the absence of a given route.

The overall network structure is given in Figure 8-1. Each camera image is processed by a separate convolutional pipeline similar to the one used in (Bojarski et al., 2016). Similarly, the cropped, non-routed, map patch is fed to a set of convolutional layers before concatenation to the image processing outputs. However, here we use fewer layers for two main reasons. First, the map images contain significantly fewer features and thus don't require a complex feature extraction pipeline. Second, we wish to avoid translational invariance effects often associated with convolutional layers and subsampling, as we are interested in the pose on the map. The output of the convolutional layers is flattened and fed to a set of fully connected layers to produce the parameters of a probability distribution of steering commands, forming f_S. As a second task, we the previous layer output along with a convolutional module processing the routed map, M_R, to output a single deterministic steering command, forming f_D. This network structure allows us to handle both routed and non-routed maps, and later affords localization and driver intent, as well as driving according to high level navigation (*i.e.* turn-by-turn instruction).

We learn the weights of our model using backpropogation with the loss defined as:

$$\mathbb{E}\left\{\begin{array}{c} \mathcal{L}\Big(f_S(I, M, \theta_p), \theta_s\Big) + \|\phi\|_p + \\ \sum_i \psi_S(\sigma_i) + \Big(f_D(I, M, \theta_p) - \theta_s\Big)^2 \end{array}\right\} \tag{8.1}$$

where ψ_S is a per-component penalty on the standard deviation σ_i. We chose a quadratic term in log-σ as the regularization,

$$\psi_S(\sigma) = \|\log \sigma - c\|^2. \tag{8.2}$$

$\mathcal{L}\big(f_S(I, M, \theta_p,), \theta_s\big)$ is the negative log-likelihood of the steering command according to a GMM with parameters $\{(\phi_i, \mu_i, \sigma_i)\}_{i=0}^N$ and

$$P(\theta_s|\theta_p, I, M) = \sum \phi_i \mathcal{N}(\mu_i, \sigma_i^2). \tag{8.3}$$

8.2.2 Localization via End-to-end Networks

The conditional structure of the model enables updates to the posterior belief about the vehicle's pose, based on the relation between the map and the road topology seen from the vehicle. For example, if the network is provided visual input, I, which appears to be taken at a 4 way intersection, we aim to compute $P(\theta_p|I,M)$ over different poses on the map to reason about where this input could have been taken. Note that our network only computes $P(\theta_s|\theta_p,I,M)$, but we are able to estimate our pose given the visual input through double marginalization over θ_s and θ_p. Given a prior belief about the pose, $P(\theta_p)$, we can write the posterior belief after seeing an image, I, as:

$$\begin{aligned} P(\theta_p|I,M) &= \mathbb{E}_{\theta_s} P(\theta_p|\theta_s,I,M) \\ &= \mathbb{E}_{\theta_s}\left[\frac{P(\theta_p,\theta_s|I,M)}{P(\theta_s|I,M)}\right] \\ &= \mathbb{E}_{\theta_s}\left[\frac{P(\theta_p,\theta_s|I,M)}{\mathbb{E}_{\theta_{p'}} P(\theta_s|\theta_{p'},I,M)}\right] \\ &= \mathbb{E}_{\theta_s}\left[\frac{P(\theta_s|\theta_p,I,M)}{\mathbb{E}_{\theta_{p'}} P(\theta_s|\theta_{p'},I,M)} P(\theta_p)\right], \end{aligned} \tag{8.4}$$

where the equalities are due to full probability theorem and Bayes theorem.

The posterior belief can therefore be computed via marginalization over θ_p, θ_s. While marginalization over two random variables is traditionally inconvenient, in two cases of interest, marginalizing over θ_p becomes easily tractable: a) when the pose is highly localized due to previous observations, as in the case of online localization; and b) where the pose is sampled over a discrete road network, as is done in mapmatching algorithms. The algorithm to update the posterior belief is shown in Algorithm 8. Intuitively, the algorithm computes, over all steering angle samples, the probability that a specific pose and images/map explain that steering angle, with the additional loop required to estimate the partition function and normalize the distribution. We note the same algorithm can be used with small modifications within the map-matching framework (Bernstein and Kornhauser, 1998; Newson and Krumm, 2009).

Algorithm 8 Posterior Pose Estimate from Driving Direction

```
Input: $I$, $M$, $p(\theta_p)$
Output: $P(\theta_p|I, M)$
  for $i = 1...N_s$: do
      Sample $\theta_s$
      Compute $P(\theta_s|\theta_p, I, M)$
      for $j = 1...N_p$: do
          Compute $P(\theta_s|\theta_{p'}, I, M)$
          Aggregate $\mathbb{E}_{\theta_{p'}} P(\theta_s|\theta_{p'}, I, M)$
      end for
      Aggregate $\mathbb{E}_{\theta_s}\left[\frac{P(\theta_s|\theta_p,I,M)}{\mathbb{E}_{\theta_{p'}} P(\theta_s|\theta_{p'},I,M)} P(\theta_p)\right]$
  end for
  Output $P(\theta_p|I, M)$ according to Equation 8.4.
```

8.3 Adaptive Decision Making under Predictive Uncertainty

8.3.1 Uncertainty-aware Temporal Control Model

In the previous section, we formulated how we can build an end-to-end model to perform "reactive" control without any high-definition localization input, for instantaneous execution. By training the model to additionally predict control commands at steps into the future, we can ensemble and fuse these predictions once the robot reaches those points, thereby stabilizing control. This section takes this idea even further and intelligently fuses the control predictions according to the model's uncertainty at each point, in order to deal with sudden unexpected events or an ambiguous future environment (*i.e.*, highly uncertain events).

LiDAR sensors provide complementary benefits to vision data from cameras. Specifically LiDAR can provide more accurate distance (depth) information and greater robustness to environmental changes like illumination. Thus to robustly handle sudden unexpected events or ambiguities in the environment, we develop a LiDAR-based end-to-end navigation system capable of achieving efficient and robust control of a full-scale autonomous vehicle using only raw 3D point clouds and coarse-grained GPS maps.

Given a raw LiDAR point cloud I_{L}, and a rendered bird's-eye view image of the noisy, routed roadmap I_{M}, our objective is to learn an end-to-end neural network f_θ to directly predict the control signals that can drive the vehicle as well as the corresponding epistemic (model) uncertainty:

$$\{(x_k, \boldsymbol{e}_k)\}_{k=0}^{K-1} = f_\theta(I_{\mathrm{L}}, I_{\mathrm{M}}), \tag{8.5}$$

where the network outputs K predictions, each pair of x_k and $\boldsymbol{e}_k$ corresponding to a control predictions with future lookahead distance of k [m] from the current frame: x_k is the predicted control value (which can be supervised by the recorded human control y_k), and $\boldsymbol{e_k}$ are the hyperparameters to estimate the uncertainty of this prediction. In principle, we can depend on the current control x_0 alone to drive the vehicle. However, the remaining x_k's with $k > 0$ might also be used to improve the robustness of the model with uncertainty-weighted temporal fusion (using $\boldsymbol{e}_k$'s); learning these additional x_k's also provides the model with a sense of planning and predicting the future.

We illustrate an efficient and robust LiDAR-only end-to-end neural network in Figure 8-2. The following subsections describe our training and deployment methodology in two parts. First, we describe how we learn efficient representations directly from input point cloud data I_{L} and a rough routed map I_{M} (following Amini et al. (2019a)). These perception (LiDAR) and localization (map) features are combined and fed to a fully-connected network to predict our control and uncertainty estimates. Next, in Section 8.3.2, we describe how our network can be optimized to learn its uncertainty and present a novel algorithm for leveraging this uncertainty during deployment to increase the robustness of the autonomous system.

8.3.2 Hybrid Evidential Fusion

Figure 8-2 illustrates that one of our model's output branches directly predicts the control values $\{x_k\}$, which can simply be supervised with the L_1 loss: $\mathcal{L}_{\mathrm{MAE}}(x_k, y_k) =$

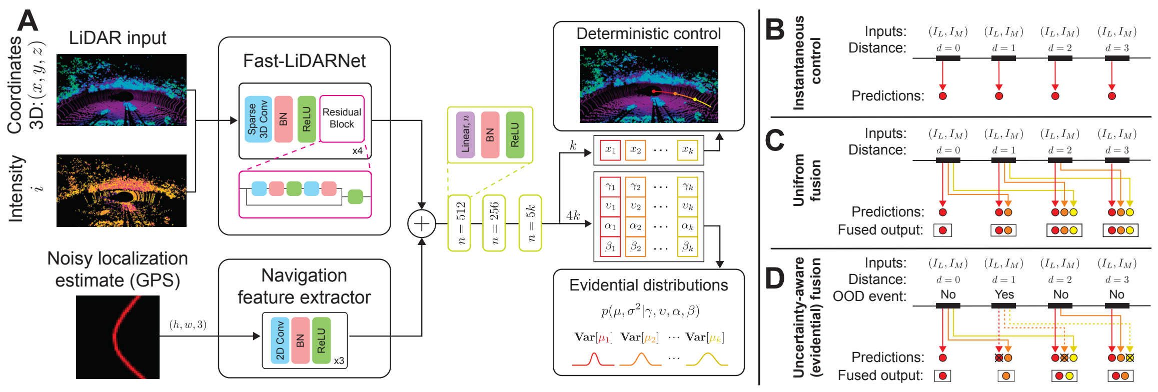


Figure 8-2: **Hybrid Evidential Fusion for Control.** **(A)** Raw LiDAR point clouds (the visualized colors are based on heights and intensities) and noisy roadmaps are fed to the model and navigation feature extractor, and integrated to learn both a deterministic control value as well as the parameters of a higher-order evidential distribution capturing the underlying predictive distribution. Output control predictions can be **(B)** instantaneously executed or **(C)** uniformly fused with odometry-corrected past predictions. **(D)** Uncertainty estimation using our evidential outputs enables intelligently weighting our predictions to increase the robustness, especially on out-of-distribution (OOD) events, or through increased uncertainty on ambiguous future time steps.

$\|x_k - y_k\|_1$. During deployment, we use the predictions from both current and odometry-corrected previous frames to improve stability. For each frame, we keep track of its absolute travelled distance d and denote its corresponding predictions as $\{x_i^{(d)}\}_{i=0}^{K-1}$. As the accuracy of d is important only locally (within the K lookahead distances), we estimate d from odometry based on an Extended Kalman Filter (Julier and Uhlmann, 2004). We can then fuse

$$\mathcal{X}^{(d)} = \left\{ x_0^{(d)}, x_1^{(d-1)}, x_2^{(d-2)}, \ldots, x_i^{(d-i)}, \ldots \right\} \tag{8.6}$$

to obtain the estimated control for the current frame, as they are all estimates of this frame. One straightforward way of fusion is to directly average all predictions in $\mathcal{X}^{(d)}$, including the current instantaneous prediction, as well as previous future predictions. However, this approach neglects the uncertainty of the predictions. This is important because (1) predictions from the past are usually less accurate due to ambiguity; and (2) out-of-distribution (OOD) events (*e.g.*, sudden changes, sensor failures) can make the current prediction extremely important or completely wrong.

To address this issue, our model additionally learns to estimate the epistemic uncertainty of every prediction, rather than relying on the rule-based fusion. This is accomplished by training an additional branch of the network to output an evidential distribution (Amini et al., 2020c; Sensoy et al., 2018) for each prediction. We follow the evidential learning method described in Chapter 7. Evidential distributions aim to capture the evidence (or confidence) associated to any decision that a network makes, thus allowing us to intelligently weigh less-confident predictions by the network less than those which are more confident. We assume that our control labels, y_d, are drawn from an underlying Gaussian Distribution with unknown mean and variance (μ, σ^2), which we seek to probabilistically estimate. As for regression targets, like robotic control problems, we place priors over the likelihood variables to obtain a joint distribution, $p(\mu, \sigma^2 | \gamma, \upsilon, \alpha, \beta)$ with

$$\mu \sim \mathcal{N}(\gamma, \sigma^2 \upsilon^{-1}), \qquad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta). \tag{8.7}$$

Our network is trained to output the hyperparameters defining this distribution, $\boldsymbol{e}_k = (\gamma_k, \upsilon_k, \alpha_k, \beta_k)$, by jointly maximizing model fit ($\mathcal{L}_{\text{NLL}}$) and minimizing evidence on errors ($\mathcal{L}_{\text{R}}$):

$$\begin{aligned} \mathcal{L}_{\text{NLL}}(\boldsymbol{w}_k, y_k) &= \tfrac{1}{2}\log\left(\tfrac{\pi}{\nu_k}\right) - \alpha_k \log(\Omega_k) \\ &+ \left(\alpha_k + \tfrac{1}{2}\right)\log\left((y_k - \gamma_k)^2 \nu_k + \Omega_k\right) + \log\left(\tfrac{\Gamma(\alpha_k)}{\Gamma(\alpha_k + 1/2)}\right) \\ \mathcal{L}_{\text{R}}(\boldsymbol{w}_k, y_k) &= |y_k - \gamma_k| \cdot (2\alpha_k + \nu_k) \end{aligned} \tag{8.8}$$

where $\Omega_k = 2\beta_k(1 + \nu_k)$. We refer the readers back to Chapter 7 and to Amini *et al.* (Amini et al., 2020c) for more details about evidential regression.

Finally, all losses are summed together to compute the total loss:

$$\mathcal{L}(\cdot) = \sum\nolimits_k \left(\alpha \mathcal{L}_{\text{MAE}}(\cdot) + \mathcal{L}_{\text{NLL}}(\cdot) + \mathcal{L}_{\text{R}}(\cdot)\right) \tag{8.9}$$

where $\alpha = 1000$ for weighting scale differences. After optimization, the epistemic uncertainty can be directly computed as $\mathrm{Var}[x_k] = \beta_k/(\nu_k(\alpha_k - 1))$ without sampling (Amini et al., 2020c). During deployment, we can collect the deterministic prediction x_k as well as the corresponding evidential uncertainty $\mathrm{Var}[x_k]$ from previous frames. We can then use the uncertainty to evaluate a confidence-weighted average of our predictions according to the procedure outlined in 9.

Algorithm 9 Uncertainty-aware deployment

```
**Given**: policy $f_\theta$, inputs $(I_\mathrm{L}, I_\mathrm{M})$, and fusion method
$\{(x_k, \gamma_k, \upsilon_k, \alpha_k, \beta_k)\} \leftarrow f_\theta(I_\mathrm{L}, I_\mathrm{M})$          ▷ Inference
for $k \in \{0, 1, \ldots, K-1\}$ do
    $\mathrm{Var}[\mu_k] \leftarrow \beta_k/(\upsilon_k(\alpha_k - 1))$          ▷ Compute uncertainty
    $\lambda_k \leftarrow 1 \,/\, \mathrm{Var}[\mu_k]$          ▷ Compute confidence
    $\Lambda^{(d+k)} \leftarrow \Lambda^{(d+k)} \cup \{\lambda_k\}$          ▷ Store confidence
    $\mathcal{X}^{(d+k)} \leftarrow \mathcal{X}^{(d+k)} \cup \{x_k\}$          ▷ Store prediction
end for
$\Lambda^{(d)} \leftarrow \Lambda^{(d)} / \sum_{\lambda \in \Lambda^{(d)}} \lambda$          ▷ Normalize confidence
switch fusion do
    case none:          ▷ Instantaneous (no fusion)
        return $x_0^{(d)}$
    case uniform:          ▷ Uniform fusion
        return $(\sum_j \mathcal{X}_j^{(d)}) / \|\mathcal{X}^{(d)}\|$
    case evidential:          ▷ Evidential fusion
        return $(\sum_j \mathcal{X}_j^{(d)} \Lambda_j^{(d)}) / \|\mathcal{X}^{(d)}\|$
```

8.4 Results

8.4.1 Reducing Localization Uncertainty in the Wild

We demonstrate how our control mixture model, described in Section 8.2.1, can be used to localize the vehicle based on the observed driving directions using Algorithm 8. We investigate in our experiments the reduction of pose uncertainty, and visualize areas which offer better types of pose localization.

For our first experiments, we began with the pose obtained from the GPS and assumed an initial error in this pose with some uncertainty (Gaussian over the spatial position, heading, or both). We compute the posterior probability of the pose as given by Algorithm 8, and look at the individual uncertainty measures or total entropy of the prior and posterior distributions. If the uncertainty in the posterior distribution is lower than that of the prior distribution, we can conclude that our learned model was able to increase its localization confidence after seeing the visual inputs provided (*i.e.* the camera images). In Figure 8-3 we show results on the training (A) and testing (B) datasets. Note that the roads and intersections in both of these datasets were entirely disjoint; the model was never trained on roads/intersections from the test set.

For the test set, we overlaid the individual GPS points on the map and colored each point according to whether our algorithm increased (blue) or decreased (orange) posterior uncertainty. When looking at uncertainty reduction, it is important to note which degrees of freedom (*i.e.* spatial vs. angular heading) localize better at different areas in the road network. For this reason, we visualize the uncertainty reduction heatmaps four times individually across (1) spatial variance, (2) angular variance, (3) overall pose variance, and (4) overall entropy reduction (Figure 8-3).

While header angle is corrected easily at both straight driving and more complex areas (turns and intersections), spatial degrees of freedom are corrected best at rich map areas and corrected poorly at linear road segments. This is expected and is similar to the aperture problem in computer vision (Marr and Ullman, 1981) – the information in a linear road geometry is not enough to establish 3-degrees-of-freedom (3DOF) localization.

If we focus on areas preceding intersections ($\sim$ 20 meters before), we typically see that the spatial uncertainty (prior uncertainty of $2m$) is reduced right before the intersection, which makes sense given that after passing through the intersection our forward facing visual inputs are not able to capture the intersection behind the vehicle. Looking in the vicinity of intersections, we achieved an average reduction of $0.31 nats$. For the angular uncertainty, with initial uncertainty of $\sigma = 0.8$ radians (45°), we achieved a reduction in the standard deviation σ of 0.2 radians (11°).

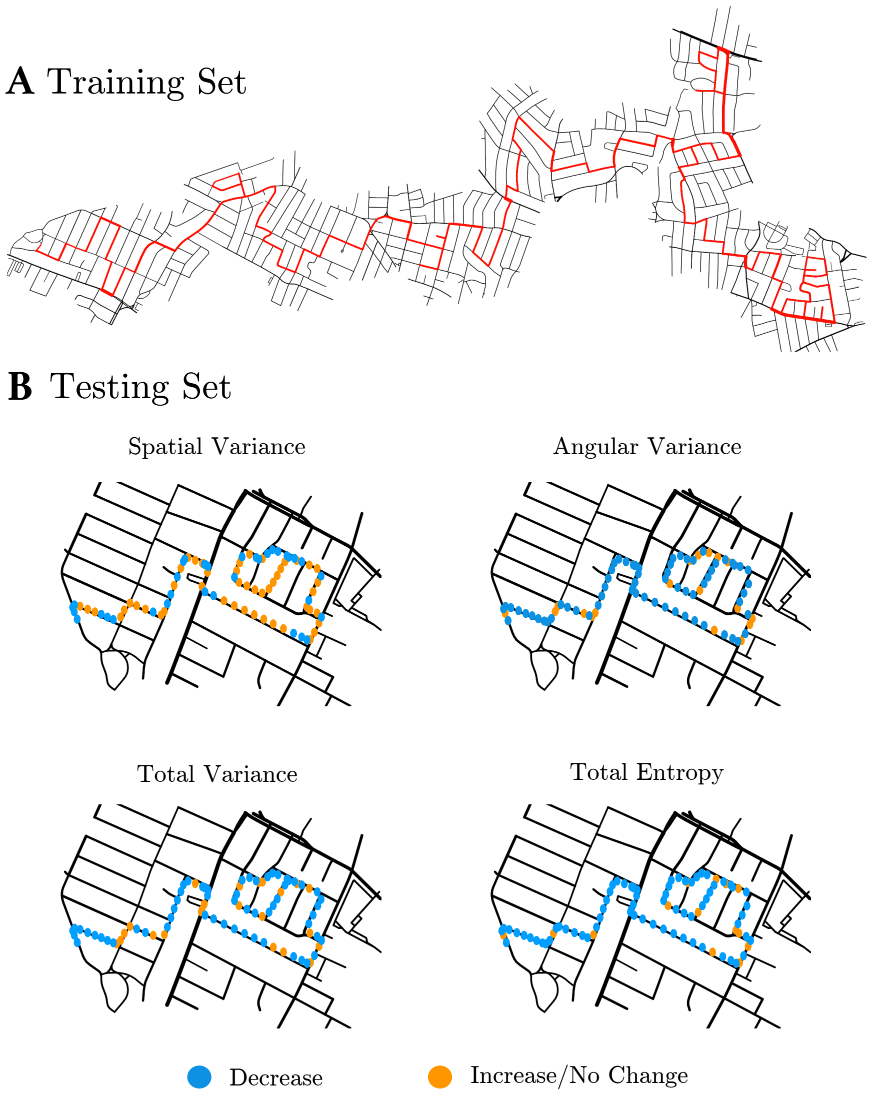


Figure 8-3: **Evaluation of posterior uncertainty improvement.** **(A)** A roadmap of the data used for training with the route driven in red (total distance of 25km). **(B)** A heatmap of how our approach increases/decreases four different types of variance throughout test set route. Points represent individual GPS readings, while the color (orange/blue) denotes the absolute impact (increase/decrease) our algorithm had on its respective variance. Decreasing variance (i.e. increasing confidence) is the desired impact of our algorithm.

We quantify the degree of posterior uncertainty reduction around intersections in Figure 8-4. Specifically, for each of the degrees of uncertainty (spatial, angular, etc) in Figure 8-3, we present the corresponding numerical uncertainty reduction as a function of the prior uncertainty in Figure 8-4. Note that we obtain reduction of both heading and spatial uncertainty for a variety of prior uncertainty values.

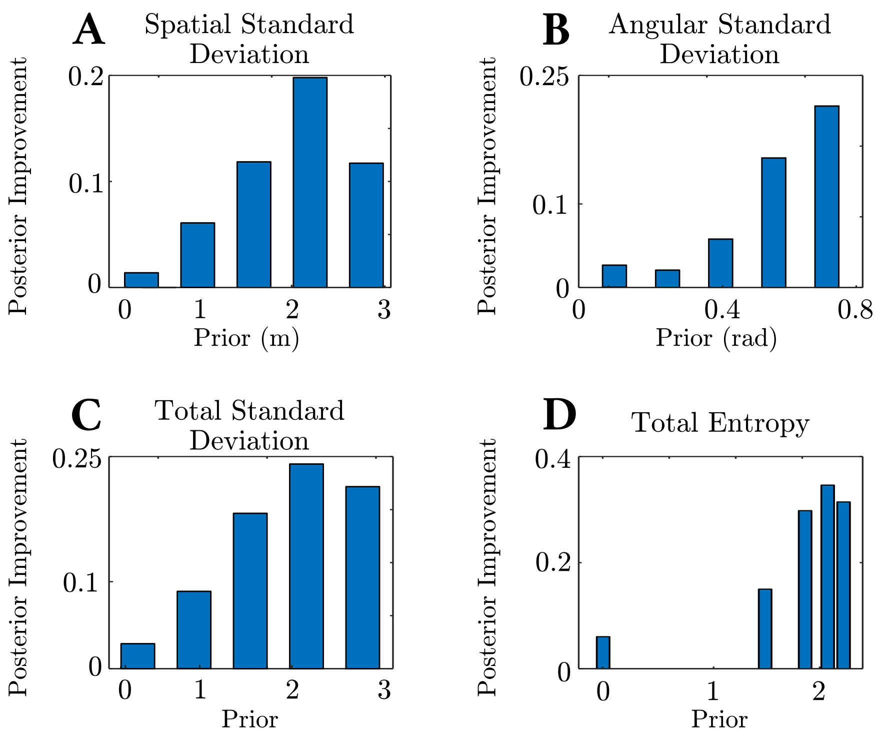


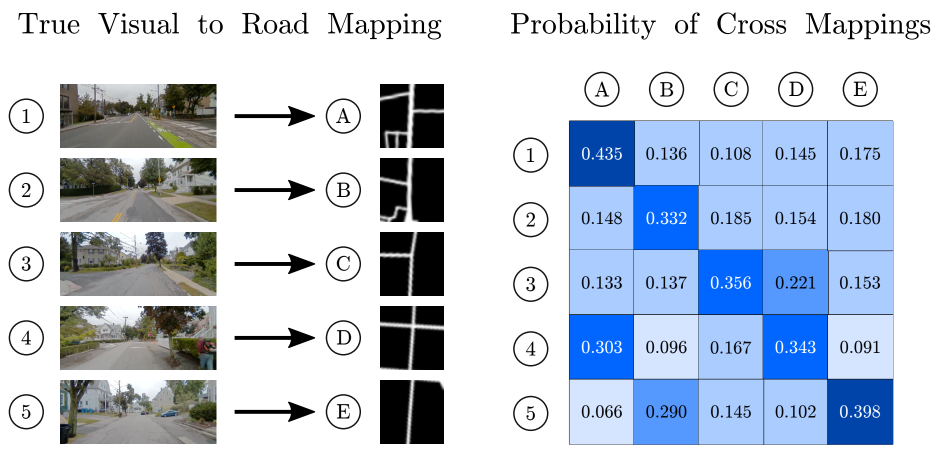
Figure 8-4: **Pose uncertainty reduction at intersections.** The reduction of uncertainty in our estimated posterior across varying levels of added prior uncertainty. We demonstrate improvement in **(A)** spatial $\sigma^2(p_x) + \sigma^2(p_y)$, **(B)** angular: $\sigma^2(p_\alpha)$, **(C)** sum of variance over p_x, p_y, p_α, and **(D)** entropy in p_x, p_y, p_α, Gaussian approximation. Note that we observe a "positive" improvement over all levels of prior uncertainty (averaged over all samples in regions preceding intersections).

Additionally, the averaged improvement over intersection regions is always positive for all prior uncertainty values indicating that, on average, localization does not worsen after using our algorithm.

8.4.2 Coarse Grained Localization

We evaluate our model's ability to distinguish between significantly different locations without any prior on pose. For example, imagine that you are in a location without GPS but still want to perform rough localization given your visual surroundings. We seek to establish correspondences between the map and the visual road area for coarse grained place recognition.

In Figure 8-5 we demonstrate how we can identify and disambiguate a small set of locations, based on the map and the camera images' interpreted steering direc-



	A	B	C	D	E
1	0.435	0.136	0.108	0.145	0.175
2	0.148	0.332	0.185	0.154	0.180
3	0.133	0.137	0.356	0.221	0.153
4	0.303	0.096	0.167	0.343	0.091
5	0.066	0.290	0.145	0.102	0.398

Figure 8-5: **Coarse Localization from Perception.** Five example locations from the test set (image, roadmap pairs). Given images from location i, we compute the network's probability conditioned on map patch from location j in the confusion matrix. Thus, we demonstrate how our system can establish correspondences between its camera and map input and even determine when its map pose has a gross error.

tion. Our results show that we can easily distinguish between places of different road topology or road geometry, in a way that should be invariant to the appearance of the region or environmental conditions. Additionally, the cases where the network struggles to disambiguate various poses is understandable. For example, when trying to determine which map the image from environment 4 was taken, the network selects maps A and D where both have upcoming left and right turns. Likewise, when trying to determine the location of environment 5, maps B and E achieve the highest probabilities. Even though the road does not contain any immediate turns, it contains a large driveway on the left hand side which resembles a possible left turn (thus, justifying the choice of map B). However, the network is able to correctly localize each of these five cases to the correct map location, as indicated by the strong diagonal of the confusion matrix (Figure 8-5).

8.4.3 Uncertainty-aware Deployment

We next evaluated the performance of the LiDAR-only models. To test our system's robustness to out-of-distribution (OOD) events, we also manually trigger sensor failures of LiDAR every 50 meters and evaluated the number of resulting interventions

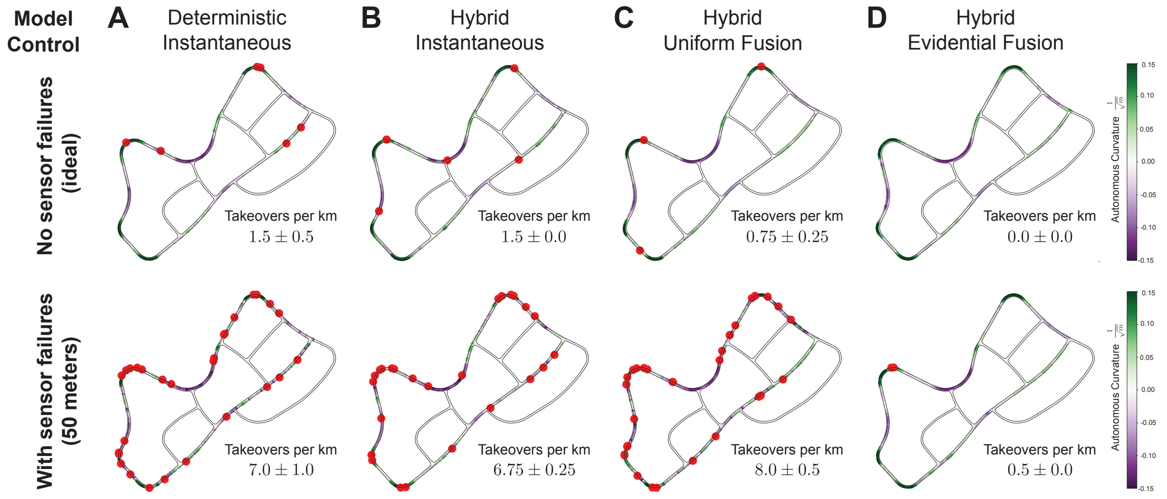


Figure 8-6: **Real-world evaluation of LiDAR-only models.** Performance of learning models (deterministic or hybrid) and fusion strategies (instantaneous, uniform, or evidential) in scenarios without (top) or with (bottom) sensor failures. Interventions are marked by red circles, and vehicle path colored predicted curvature. Trials repeated twice at fixed speeds on the test track.

(Figure 8-6). In comparing our method to baseline model control algorithms, we observed that the model deployed with evidential fusion yields drastically improved performance with and without OOD sensor failures (Figure 8-6). This model is the most effective at not only reliably estimating high uncertainty on the OOD events, but also in ensuring that the associated outputs will not be propagated downstream to the controller after the fusion, as evidenced by the decreased number of resulting interventions relative to the evluated baselines (Figure 8-6D). On the other hand, the model with uniform fusion (Figure 8-6**C**) also obtains increased performance over instantaneous models (Figure 8-6**A**, **B**) without sensor failures, but in the presence of failures propagates the failed response to the controller and amplifies the error by fusing it over multiple time points. Here, deterministic (**A**) and hybrid (**B**) models are compared as an ablation analysis to verify roughly similar performance regardless of output parameterization if not considering uncertainty.

All models in Figure 8-6 are trained with data augmentations strategies. We also test to disable the rotation augmentation in to see its effect (Figure 8-7). Without rotation augmentation, the model is not able to control the vehicle to recover from

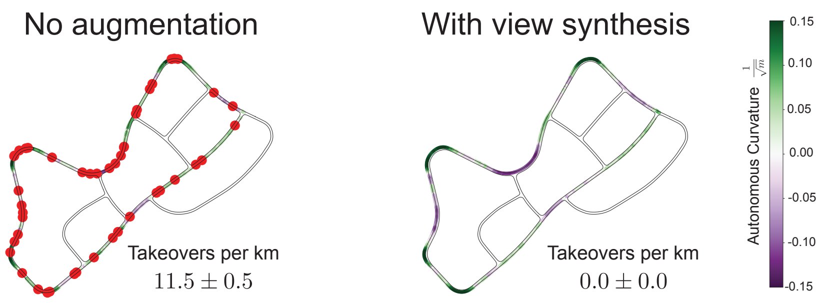


Figure 8-7: **Rotation augmentation** teaches the model to recover from off-center views. When deployed with the model trained without rotation augmentation, the vehicle crashes frequently.

off-center views because they are not encountered during data collection.

To test the recovery robustness, we start the controller in manually distorted orientations on the road and measure the network's ability to recover (Table 8.1). A successful recovery is marked if the vehicle returns to a stable position within 10 seconds. We noticed that both evidential fusion and uniform fusion outperform others. All models have a better performance recovering from CCW than CW since LiDAR is not occluded by the far side of the road on a CCW turn, while the near side is occluded in a CW turn.

Model	CW	CCW	Average
PointNet	0.12 ± 0.17	0.00 ± 0.00	0.06 ± 0.08
Deterministic	0.80 ± 0.13	0.92 ± 0.10	0.86 ± 0.08
Hybrid	0.80 ± 0.13	0.90 ± 0.11	0.85 ± 0.08
Uniform	0.84 ± 0.16	0.94 ± 0.10	0.89 ± 0.07
Evidence	0.86 ± 0.10	0.92 ± 0.10	0.89 ± 0.07

Table 8.1: **Performance of recovering from near-crash positions**. Here, CW denotes the success rate of recovering from clockwise rotation, and CCW denotes counter-clockwise.

8.5 Discussion

In this chapter, we developed a probabilistic variational model for incorporating coarse-grained localization information together with raw perceptual data to directly learn control of autonomous agent. We demonstrate estimation of the likelihood of different possible control commands, as well as localization correction and place recognition based on the map. We formulate a concrete pose estimation algorithm using our learned network to reason about the localization of the robot within the environment and demonstrate reduced uncertainty (greater confidence) in our resulting pose. Our method allows for reasoning about control even under the presence of highly imperfect localization information, and more critically, leveraging priors about control to adaptive improve the agent's noisy localization online, live during deployment.

In addition to increasing uncertainty-awareness of the input sensory data, we also propose an algorithm for principled integration of epistemic uncertainty estimation directly into the predictive model to guide control based on predictive confidences. Our approach leverages deep evidential regression, described in Chapter 7, to achieve single-shot uncertainty estimation. In comparison to other state-of-the-art uncertainty estimators, this means that our method is able to estimate uncertainty with only a single model and single inference pass through that model. Our approach fuses predictions from multiple frames together while taking their uncertainties into consideration. Our framework has been evaluated on a full-scale autonomous vehicle and demonstrates lane-stable as well as navigation capabilities. Our proposed fusion algorithm significantly improves robustness and reduces the number of takeovers in the presence of out-of-distribution events such as sensor failures.

8.6 Scope and Limitations

We have demonstrated the critical importance and potential power of a principled approach towards uncertainty modeling and infused uncertainty-awareness within an autonomous system. Physical systems that operate in real time will inevitably be faced with scenarios they are ill equipped to handle and have not been sufficiently trained for. These events are not discrete or binary events, but rather exist on a continuous spectrum of how much trust we can place into our model's decision making process. Despite the promise of integrating a sense of uncertainty awareness into the decision making system, notable limitations remain. The algorithms and first experiments described in this chapter highlight the importance of this direction and open several avenues for future investigation.

Firstly, the approaches described in this chapter provide algorithms which target improved robustness through uncertainty-awareness at singular stages in the decision making pipeline (*e.g.*, localization, scene understanding, control, *etc.*). In the future, there exists enormous potential to scale these results to online settings and to propagate uncertainty-awareness throughout the decision making process. When dealing with multimodal sensory inputs or modular predictive pipelines, we can now begin to craft algorithms for extracting uncertainties of the features obtained by each of these components instead of exclusively late stage control predictions. There also exists potential for more theoretically-grounded fusion of uncertainties in such a framework. Existing approaches are limited to either aleatoric or epistemic modeling in isolation, but our work in Chapter 7 decouples this information within a single-shot estimation method. However, we still require algorithms for taking such decoupled information and learning how to effectively fuse these together into the decision making system depending on the situation or scenario.

Part IV

Conclusions

Chapter 9

Perspectives and Future Directions

9.1 Summary

In this thesis, we present multiple avenues towards engineering robust autonomy solutions for decision making in the wild. We take a holistic approach to tackling this grand vision by considering all stages of the autonomy lifecycle – from data that the autonomous agents are trained on, to their underlying decision-making models, to the algorithms that ultimately enable deployment in the real world. We achieve this by developing theoretically-grounded technical advancements for new capabilities and realizations of the theory through robust and rigorous experiments.

We contribute a powerful framework for data-driven generation of synthetic environments to produce challenging edge-cases synthetically and demonstrate that such a simulation platform provides a flexible testbed for training and evaluating autonomous agents (Part I). Our results are showcased specifically in the context of autonomous driving; however, the algorithms developed in this section, and the concept of advancing data-driven simulation, are broadly generalizable to other decision-making scenarios. Moreover, our approach can serve as a modular and flexible proving technique for various types of autonomy and decision making settings.

We then move forward in the learning pipeline, from data to the models, to develop new architectures and learning algorithms for improved robustness and generalizability of NNs and learning-based systems (Part II). To this end, we develop compact and expressive neural architectures for auditable autonomy, and demonstrate they have provably causal structures, critically important for dynamic control. We build upon this theme of robustness in the model to design a learning algorithm for identifying and improving learning from underrepresented regions and rare events. We find that these approaches lead to more efficient, performant, and robust learning models in practice, again with a focus on the practical autonomy and robotic control setting.

Finally, we consider what algorithmic changes are necessary to ultimately deploy end-to-end learning for autonomous decision making into the wild (Part III). One of the most significant considerations rests in the assumption that data-driven statistical models require their testing distribution to be equal to their training distribution, as well as more generally being able to handle uncertainty and ambiguity in the real world. To address this challenge, we develop new algorithms for calibrated, single-shot uncertainty quantification in NNs for regression tasks. Critically, we do not simply develop the estimation method, but more importantly extend beyond to consider how these uncertainties are actually useful in deployment. To that end we formulate novel algorithms leverage and flexibily integrate predictive uncertainties into our systems in order to achieve more robust decision making, first through probabilistic reasoning and then through handling data out-of-distribution edge cases.

Our work tackles critical challenges throughout the stages of the autonomy pipeline and provides theoretically-grounded solutions to overcoming outstanding challenges that plague ML within dynamic autonomy and decision making settings. Together, these contributions advance an end-to-end vision of learning to reliable execute decisions in the wild. We envision this provides a generalizable framework for increasing the robustness end-to-end learning for important and safety-critical applications.

9.2 Lessons Learned

Practical Realization of Results

The most important lesson learned throughout my PhD has been on the importance of realizing and evaluating research in the context of a practical, real-world system. While much of the machine learning field is fixated on extremely optimized datasets or clean simulations, the real-world is noisy, unstructured, and messy. Datasets can serve as interesting testbeds to benchmark problems on unified structures; however, they should not serve as the measuring stick to design new methodologies and algorithms. Just as a machine learning algorithm can overfit to its data, I believe we are currently at a stage where the community itself is overfitting its own research objectives tailored to artificial datasets that do not scale or advance practical achievements.

A great deal of effort was spent at all stages of this thesis to not only scale results on-board physical hardware, but also to identify fundamental algorithmic challenges that were exposed through the process of real-world experimentation and evaluation. Almost all of these challenges would have been obscured without explicit deployment into reality. Some of these challenges were hardware and engineering specific (*e.g.*, low-level control interfaces to control autonomous systems). However, the vast majority of these challenges were much more generally *algorithm-centric*, and caused by fundamental limitations of machine learning that are either minimized or overlooked all-together (*e.g.*, generalization, train-test mismatches, *etc.*).

Method-first *vs*. Problem-first Engineering

I have learned that in developing impactful engineering advances, a mixture of method-first combined with problem-first research has the potential to enable the most exciting advances. Throughout this thesis I broadly adopt two mentalities in design research aims. Firstly, a *method-first* mentality where we seek to develop a new foundational method to solve a broad challenge and then identify impactful applications which can benefit from such a method. Or alternatively, a *problem-first* approach where we identify a foundational problem or challenging application area and let this domain guide our design of a computational or algorithmic solution.

This thesis has benefit from both mentalities, frequently complementing each other and often times in a cyclical fashion. The overall vision of this thesis has been driven by the dream of achieving robust learning-based agents that can make decision in the wild. This grand vision put forth a series of fundamental challenges which in turn inspired concrete methodological advances. Other the other hand, we have also been motivated to create concrete methodological advances (*e.g.*, single-shot uncertainty estimation algorithms), which we later applied broadly to impact a variety of broad applications ranging from computer vision (Amini et al., 2020c), medical drug discovery Soleimany et al. (2021), autonomous driving (Liu et al., 2021), and robotic manipulation Amini et al. (2020b). Critically, both approaches have been invaluable to effectively and principally build toward grand visions while both exploring and exploiting technological advances.

Societal Implications of AI

Over the course of my research, I have seen a remarkable evolution of AI solutions rapidly permeating through lab and development environments and into reality, interacting directly with humans on critical decision making tasks. The wide-scale adoption of AI presents the potential for very significant societal impact. Neural networks are increasingly being trained as black-box predictors and being placed in larger decision systems where errors in their predictions can pose immediate threat to downstream tasks. Systematic methods for training robust models and calibrated uncertainty estimators, like those developed in this thesis, are necessary but not sufficient. This is especially true for systems that are deployed in safety critical domains, such for autonomous vehicle control (Lechner et al., 2020a), medical diagnosis (Shen et al., 2019), or in settings with large dataset imbalances and bias such as crime forecasting (Kang and Kang, 2017) and facial recognition (Amini et al., 2019b).

Recent studies have shown that AI systems are severely vulnerable to widespread algorithmic bias, particularly towards instances that are underrepresented in training data. This realization shaped several aspects of this thesis ranging from the development of new auditable neuron models with greater levels of interpretability, to new debiasing algorithms to uncover and mitigate the bias of AI systems. We believe this is of critical importance to advance this field further to achieve significant impact. We must aim to move beyond training deep learning systems to be solely performance optimized – but instead, usher in a new age of jointly performant and ethical agents that are trained using fair, auditable, and debiased learning.

9.3 Future Directions

9.3.1 Dynamically Evolving Environments for Robust Learning

To be effectively deployed into society, autonomous systems must be able to perceive, react, and interact with humans at timescales ranging from short, subtle cues to long-term behavioral patterns. Learning to control within such dynamic environments requires either (1) training agents directly in reality or (2) synthesizing such behaviors offline, training, and then transferring learned knowledge into reality. However, real-world exposure comes with impractical experimental costs and the risk of dangerous interactions, while simulations today typically consist of pre-defined trajectories which inevitably lead to repetitive behaviors that do not translate to reality.

There is enormous potential to create scalable, data-driven, and dynamically evolving environments required to maximize learning-to-control performance. A first aim is to advance data-driven simulation for learning to control by integrating multi-agent behavioral models within synthetic environments to design dynamic and interactive scenarios. We will develop frameworks that enable the agent and environment to iteratively and mutually improve each other: agents learn to control and also inform the design of challenging synthetic scenarios, which will in turn enable generalization of control. Once the agent is deployed into reality, it becomes an embodied extension that will be physically able to collect data from natural interactions in order to feed back into and further improve the data-driven environments in which it was trained. This research will develop a research platform for data-driven simulation that will enable a tight coupling of perception, interaction, and control through an agent's iterative learning and deployment lifecycle.

9.3.2 Generalization and Extrapolation for Trustworthy Decision Making

Humans can facilitate robot learning in complex environments by providing task demonstrations using rich, multimodal sensory inputs. However, current approaches for multimodal learning are not flexible, as they are often limited to a small, predefined set of modalities and are unable to reason when a subset of information is noisy, corrupted, or missing. Further, existing methods rely on a brute force concatenation of learned features, and thus are prohibitively sample inefficient to scale to larger numbers of sensing modalities and task specifications.

In the future, I will build learning-based agents capable of ingesting a fluid sensory stream of information and simultaneously self-reflecting on their own learned representations in order to realize generalizable control in extremely rich sensory environments. We will achieve these capabilities by developing new technologies for (1) learning shared intermediate spaces across sensing modes; (2) building self-attention across sensory features; and (3) propagating learned uncertainty through shared representations to make more intelligent and robust decisions. By learning to associate different sensory modes into a reusable, shared intermediate space, agents could cope with a dynamically changing number of sensors as well as incomplete data. With multiple sources of information being represented in a common embedding space, we will build agents with self-attention across the components of this space to discover the most salient aspects for learning control. Finally, we will create learned reasoning techniques to propagate uncertainty throughout this entire pipeline: from uncertainty-aware neural representations in the shared embedding space, to attention mechanisms that will systematically leverage uncertainty to guide the model's focus away from noisy or corrupt sensing modalities.

9.3.3 Human-friendly and Interpretable Decision Making

A central challenge of learning to control is that the reward signals in the real world are often too delayed, sparse, and ambiguous for complex task specifications — where even massive datasets still produce unstainable models and uninterpretable decisions. Existing end-to-end approaches impose minimal structure or prior task knowledge on the learning model, causing learned policies to suffer from sample inefficiency, lack reasoning abilities when placed in novel scenarios, and appear "artificial" by not reflecting natural human intuition. In order to fully deploy and integrate learning-based control in the wild, agents must be guided towards solutions that not only solve the tasks, but also integrate seamlessly with humans in society, even in the most challenging scenarios.

Future directions include the design of new learning algorithms for guiding models towards control solutions that reflect human priors in order to both (1) facilitate natural interactions with humans and (2) achieve generalization to extreme scenarios. We will explore ways in which priors are structured both in abstract forms from human demonstrations, but also through semantic and geometric instruction to effectively convey intent in higher-level symbolic spaces. Integration of prior information is critical in control, as deployed agents may be faced with situations in the wild that are completely different from anything in their training data, far beyond distributional shifts. While my work in principled uncertainty estimation has enabled reliable identification of such scenarios, combining these uncertainties with intuitive priors and implementing them within the decision process will empower autonomous agents to achieve robust and flexible control, even in completely unknown environments.

Bibliography

N. A. Abdullah, M. J. Saidi, N. H. A. Rahman, C. C. Wen, and I. R. A. Hamid. Face recognition for criminal identification: An implementation of principal component analysis for face recognition. In *AIP Conference Proceedings*, volume 1891, page 020002, 2017.

H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126(9), 2018.

E. Alcalá, V. Puig, J. Quevedo, and U. Rosolia. Autonomous racing using linear parameter varying-model predictive control (LPV-MPC). *Control Engineering Practice*, 95, 2020.

A. Amini, L. Paull, T. Balch, S. Karaman, and D. Rus. Learning steering bounds for parallel autonomous systems. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018a.

A. Amini, W. Schwarting, G. Rosman, B. Araki, S. Karaman, and D. Rus. Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 568–575. IEEE, 2018b.

A. Amini, A. Soleimany, S. Karaman, and D. Rus. Spatial uncertainty sampling for end-to-end control. In *NeurIPS Workshop on Bayesian Deep Learning*, 2018c.

A. Amini, G. Rosman, S. Karaman, and D. Rus. Variational end-to-end navigation and localization. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019a.

A. Amini, A. P. Soleimany, W. Schwarting, S. N. Bhatia, and D. Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019b.

A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150, 2020a.

A. Amini, J. Lipton, and D. Rus. Uncertainty aware texture classification and mapping using soft tactile sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020b.

A. Amini, W. Schwarting, A. Soleimany, and D. Rus. Deep Evidential Regression. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020c.

A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus. VISTA 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2021.

M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

S. Balaban. Deep learning and face recognition: the state of the art. *Biometric and surveillance technology for human and activity identification XII*, 9457:68–75, 2015.

N. Baram, O. Anschel, I. Caspi, and S. Mannor. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pages 390–399. PMLR, 2017.

Y. Bengio, P. Simard, P. Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

R. A. Berk, S. B. Sorenson, and G. Barnes. Forecasting domestic violence: A machine learning approach to help inform arraignment decisions. *Journal of Empirical Legal Studies*, 13(1):94–115, 2016.

D. Bernstein and A. Kornhauser. An introduction to map matching for personal navigation assistants. *tIDE*, 1998.

A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V.-D. Lam, and A. Kendall. Learning to drive from simulation without real world labels. *arXiv preprint arXiv:1812.03823*, 2018. URL http://arxiv.org/abs/1812.03823.

C. M. Bishop. Mixture density networks. In *Tech. Rep. NCRG/94/004, Neural Computing Research Group*. Aston University, 1994.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

P. Bogacki and L. F. Shampine. A 3 (2) pair of Runge-Kutta formulas. *Applied Mathematics Letters*, 2(4):321–325, 1989.

S. Bohez, T. Verbelen, E. De Coninck, B. Vankeirsbilck, P. Simoens, and B. Dhoedt. Sensor fusion for robot control through deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

C. Bohn and D. Atherton. An analysis package comparing PID anti-windup strategies. *IEEE Control Systems Magazine*, 15(2):34–40, 1995.

E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 523–533. IEEE, 2019.

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

M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. J. Ackel, U. Muller, P. Yeres, and K. Zieba. VisualBackProp: Efficient visualization of CNNs for autonomous driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.

T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357, 2016.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford. Learning deployable navigation policies at kilometer scale from a single traversal. *arXiv preprint arXiv:1807.05211*, 2018.

J. Buolamwini and T. Gebru. Gender Shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pages 77–91, 2018.

W. Burgard, O. Brock, and C. Stachniss. *Map-Based Precision Vehicle Localization in Urban Environments.* MIT Press, 2008.

A. Caliskan, J. J. Bryson, and A. Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

F. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, and K. R. Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, pages 3992–4001, 2017.

J. V. Carrau, A. Liniger, X. Zhang, and J. Lygeros. Efficient implementation of Randomized MPC for miniature race cars. In *European Control Conference (ECC)*, 2016.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357, 2002.

Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018a.

W. Chen, Y. Shen, H. Jin, and W. Wang. A variational dirichlet framework for out-of-distribution detection. *arXiv preprint arXiv:1811.07308*, 2018b.

Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun. GeoSim: Photorealistic image simulation with geometry-aware composition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. *Advances in neural information processing systems*, 28, 2015.

C. Choy, J. Gwak, and S. Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun. End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410*, 2017.

F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

S. J. Cook, T. A. Jarrell, C. A. Brittin, Y. Wang, A. E. Bloniarz, M. A. Yakovlev, K. C. Nguyen, L. T.-H. Tang, E. A. Bayer, J. S. Duerr, et al. Whole-animal connectomes of both Caenorhabditis elegans sexes. *Nature*, 571(7763):63–71, 2019.

E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016–2021.

R. Courtland. Bias detectives: the researchers striving to make algorithms fair. *Nature*, Jun 2018. URL https://www.nature.com/articles/d41586-018-05469-3.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. AutoAugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi. RoboTHOR: An open simulation-to-real embodied ai platform. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

S. Desai, I. Durugkar, H. Karnan, G. Warnell, J. Hanna, P. Stone, and A. Sony. An imitation from observation approach to transfer learning with dynamics mismatch. *Advances in Neural Information Processing Systems*, 33, 2020.

J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

E. Dupont, A. Doucet, and Y. W. Teh. Augmented neural ODEs. In *Advances in Neural Information Processing Systems*, pages 3134–3144, 2019.

C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7509–7520, 2019.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940, 2016.

L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *OpenReview*, 2018.

S. S. Farfade, M. J. Saberian, and L.-J. Li. Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 643–650, 2015.

T. B. Fitzpatrick. The validity and practicality of sun-reactive skin types I through VI. *Archives of dermatology*, 124(6):869–871, 1988.

L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, A. Patsekin, J. Kindelsberger, L. Ding, et al. Mit advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation. *IEEE Access*, 7:102021–102038, 2019.

S. A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E. P. Hamilton, and D. Roth. A comparative study of fairness-enhancing interventions in machine learning. *arXiv preprint arXiv:1802.04422*, 2018.

K. J. Friston, L. Harrison, and W. Penny. Dynamic causal modelling. *Neuroimage*, 19(4):1273–1302, 2003.

H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr. Super-human performance in gran turismo sport using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4257–4264, 2021.

K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

K.-i. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

M. Gabrié, A. Manoel, C. Luneau, N. Macris, F. Krzakala, L. Zdeborová, et al. Entropy and mutual information in models of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1821–1831, 2018.

Y. Gal and Z. Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016a.

Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016b.

Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In *Advances in neural information processing systems*, pages 3581–3590, 2017.

E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, 41(6), 2017.

G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza. Event-based, 6-dof camera tracking from photometric depth maps. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2402–2412, 2017.

C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

J. Gast and S. Roth. Lightweight probabilistic deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

D. Gehrig, M. Gehrig, J. Hidalgo-Carrio, and D. Scaramuzza. Video to events: Recycling video datasets for event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020a.

D. Gehrig, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza. Video to events: Recycling video datasets for event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3586–3595, 2020b.

D. Gehrig, M. Rüegg, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza. Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction. *IEEE Robotics and Automation Letters*, 6(2), 2021.

A. Gelman, A. Jakulin, M. G. Pittau, Y.-S. Su, et al. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2(4):1360–1383, 2008.

A. Gelman et al. Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian analysis*, 1(3):515–534, 2006.

A. Gholami, K. Keutzer, and G. Biros. Anode: Unconditionally accurate memory-efficient gradients for NeuralODEs. *arXiv preprint arXiv:1902.10298*, 2019.

I. Gilitschenski, R. Sahoo, W. Schwarting, A. Amini, S. Karaman, and D. Rus. Deep orientation uncertainty learning based on a Bingham loss. In *International Conference on Learning Representations (ICLR)*, 2019.

C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 7, 2017.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

K. Gopalratnam, H. Kautz, and D. S. Weld. Extending continuous time bayesian networks. In *AAAI*, pages 981–986, 2005.

A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

J. M. Gray, J. J. Hill, and C. I. Bargmann. A circuit for navigation in Caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 102(9):3184–3191, 2005.

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo, and K. Li. Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization. *IEEE Transactions on Vehicular Technology*, 69(11):12597–12608, 2020.

W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman. FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

P. Gurevich and H. Stuke. Gradient conjugate priors and multi-layer neural networks. *Artificial Intelligence*, 278:103184, 2020.

D. Hafner, D. Tran, T. Lillicrap, A. Irpan, and J. Davidson. Noise contrastive priors for functional uncertainty. In *Uncertainty in Artificial Intelligence*, pages 905–914. PMLR, 2020.

M. Haklay and P. Weber. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 2008.

B. Hanin and D. Rolnick. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*, pages 571–581, 2018.

B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. *arXiv preprint arXiv:1901.09021*, 2019.

Y. Hanshu, D. Jiawei, T. Vincent, and F. Jiashi. On robustness of neural ordinary differential equations. In *International Conference on Learning Representations*, 2020.

M. Hardt, E. Price, N. Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

D. Harwath and J. R. Glass. Learning word-like units from joint audio-visual analysis. *arXiv preprint arXiv:1701.07481*, 2017.

R. Hasani. *Interpretable Recurrent Neural Networks in Continuous-time Control Environments*. PhD dissertation, Technische Universität Wien, 05 2020.

R. Hasani, A. Amini, M. Lechner, F. Naser, R. Grosu, and D. Rus. Response characterization for auditing cell dynamics in long short-term memory networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu. Liquid Time-constant Networks. In *Proceedings of the Seventeenth AAAI Conference on Artificial Intelligence*, 2020a.

R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu. The natural lottery ticket winner: Reinforcement learning with ordinary neural circuits. In *Proceedings of the 2020 International Conference on Machine Learning. JMLR. org*, 2020b.

D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, and A. Kendall. Urban driving with conditional imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

S. Hecker, D. Dai, and L. Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *European Conference on Computer Vision (ECCV)*, 2018.

J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning (ICML)*, 2015.

J. Hidalgo-Carrió, D. Gehrig, and D. Scaramuzza. Learning monocular dense depth from events. In *2020 International Conference on 3D Vision (3DV)*, pages 534–542, 2020. doi: 10.1109/3DV50981.2020.00063.

J. Ho and S. Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation (NC)*, 1997.

P. Holl, V. Koltun, and N. Thuerey. Learning to control PDEs with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

M. Hosea and L. Shampine. Analysis and implementation of TR-BDF2. *Applied Numerical Mathematics*, 20(1-2):21–37, 1996.

X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang. The ApolloScape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 954–960, 2018.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

J. Jia and A. R. Benson. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems*, pages 9843–9854, 2019.

H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

T. Joo, U. Chung, and M.-G. Seo. Being bayesian about categorical probability. *arXiv preprint arXiv:2002.07965*, 2020.

M. I. Jordan. The exponential family: Conjugate priors, 2009.

S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 2004.

H.-W. Kang and H.-B. Kang. Prediction of crime occurrence from multi-modal data using deep learning. *PloS one*, 12(4):e0176244, 2017.

H. S. Kaplan, O. S. Thula, N. Khoss, and M. Zimmer. Nested neuronal dynamics orchestrate a behavioral hierarchy across timescales. *Neuron*, 2019.

A.-H. Karimi, B. Schölkopf, and I. Valera. Algorithmic recourse: from counterfactual explanations to interventions. *arXiv preprint arXiv:2002.06278*, 2020.

P. Karkus, X. Ma, D. Hsu, L. P. Kaelbling, W. S. Lee, and T. Lozano-Pérez. Differentiable algorithm networks for composable robot learning. *arXiv preprint arXiv:1905.11602*, 2019.

A. Karpathy. PyTorch at Tesla. PyTorch Devcon Conference 19, 2019. URL https://youtu.be/oBklltKXtDE.

S. Kato, H. S. Kaplan, T. Schrödel, S. Skora, T. H. Lindsay, E. Yemini, S. Lockery, and M. Zimmer. Global brain dynamics embed the motor command sequence of caenorhabditis elegans. *Cell*, 163(3):656–669, 2015.

C. Kemp, N. D. Goodman, and J. B. Tenenbaum. Learning to learn causal models. *Cognitive Science*, 34(7):1185–1243, 2010.

A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. *arXiv preprint arXiv:1807.00412*, 2018.

A. E. Khandani, A. J. Kim, and A. W. Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

N. Kilbertus, M. R. Carulla, G. Parascandolo, M. Hardt, D. Janzing, and B. Schölkopf. Avoiding discrimination through causal reasoning. In *Advances in Neural Information Processing Systems*, pages 656–666, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

B. F. Klare, M. J. Burge, J. C. Klontz, R. W. V. Bruegge, and A. K. Jain. Face recognition performance: Role of demographic information. *IEEE Transactions on Information Forensics and Security*, 7(6):1789–1801, 2012.

P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

M. Lechner and R. Hasani. Learning long-term dependencies in irregularly-sampled time series. *arXiv preprint arXiv:2006.04418*, 2020.

M. Lechner, R. Hasani, M. Zimmer, T. A. Henzinger, and R. Grosu. Designing worm-inspired neural networks for interpretable robotic control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 87–94. IEEE, 2019.

M. Lechner, R. Hasani, A. Amini, T. A. Henzinger, D. Rus, and R. Grosu. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2(10): 642–652, 2020a.

M. Lechner, R. Hasani, D. Rus, and R. Grosu. Gershgorin loss stabilizes the recurrent neural network compartment of an end-to-end robot learning scheme. In *2020 International Conference on Robotics and Automation (ICRA)*. IEEE, 2020b.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

Y. Lecun, E. Cosatto, J. Ben, U. Muller, and B. Flepp. DAVE: Autonomous off-road vehicle control using end-to-end learning. *Courant Institute/CBLL, http://www. cs. nyu. edu/yann/research/dave/index. html, Tech. Rep. DARPA-IPTO Final Report*, 2004.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

G.-H. Lee, D. Alvarez-Melis, and T. S. Jaakkola. Towards robust, locally linear deep networks. *arXiv preprint arXiv:1907.03207*, 2019a.

R. Lee, O. J. Mengshoel, and M. J. Kochenderfer. Adaptive stress testing of safety-critical systems. In *Safe, Autonomous and Intelligent Vehicles*, pages 77–95. Springer, 2019b.

J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1991.

S. Levine and V. Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR, 2013.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

W. Li, C. W. Pan, R. Zhang, J. P. Ren, Y. X. Ma, J. Fang, F. L. Yan, Q. C. Geng, X. Y. Huang, H. J. Gong, W. W. Xu, G. P. Wang, D. Manocha, and R. G. Yang. AADS: Augmented autonomous driving simulation using data-driven algorithms. *Science Robotics*, 4(28), 2019.

A. Liniger and J. Lygeros. Real-time control for autonomous racing based on viability theory. *IEEE Transactions on Control Systems Technology*, 27(2), 2019.

A. Liniger and J. Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 28(3), 2020.

G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision*, 12 2015.

Z. Liu, A. Amini, S. Zhu, S. Karaman, S. Han, and D. Rus. Efficient and robust lidar-based end-to-end navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *Transactions on Robotics*, 36(1), 2020.

Y. Lu, H. Guo, and L. Feldkamp. Robust neural learning from unbalanced data samples. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 3, pages 1816–1821. IEEE, 1998.

R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor. Airsim drone racing lab. In *NeurIPS 2019 Competition and Demonstration Track*, pages 177–191. PMLR, 2020.

J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26): eaau4984, 2019.

A. Malinin. *Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment*. PhD thesis, University of Cambridge, 2019.

A. Malinin and M. Gales. Predictive uncertainty estimation via prior networks. In *Conference on Neural Information Processing Systems*, 2018.

A. Malinin and M. Gales. Reverse KL-divergence training of prior networks: Improved uncertainty and adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 14520–14531, 2019.

S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun. LiDARsim: Realistic LiDAR simulation by leveraging the real world. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5419–5427, 2018.

D. Marr and S. Ullman. Directional selectivity and its use in early visual processing. *Proc. R. Soc. Lond. B*, 211(1183):151–180, 1981.

M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21 (2-3):427–436, 2008.

W. Menapace, S. Lathuiliere, S. Tulyakov, A. Siarohin, and E. Ricci. Playable video generation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

C. C. Miller. When algorithms discriminate. *New York Times*, Jul 2015. URL https://www.nytimes.com/2015/07/10/upshot/when-algorithms-discriminate.html.

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

D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

C. Molnar. *Interpretable machine learning*. Lulu. com, 2020.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2002.

G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

J. M. Mooij, D. Janzing, and B. Schölkopf. From ordinary differential equations to structural causal models: the deterministic case. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 440–448, 2013.

A. More. Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv preprint arXiv:1608.06048*, 2016.

M. C. Mozer, D. Kazakov, and R. V. Lindsey. Discrete event, continuous time RNNs. *arXiv preprint arXiv:1710.04110*, 2017.

E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017.

E. Nalisnick, B. Mitra, N. Craswell, and R. Caruana. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84. International World Wide Web Conferences Steering Committee, 2016.

F. Naser, D. Dorhout, S. Proulx, S. D. Pendleton, H. Andersen, W. Schwarting, L. Paull, J. Alonso-Mora, M. H. Ang, S. Karaman, et al. A parallel autonomy research platform. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 933–940. IEEE, 2017.

P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

O. Nevanlinna. Remarks on picard-lindelöf iteration. *BIT Numerical Mathematics*, 29(3):535–562, 1989.

P. Newson and J. Krumm. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM, 2009.

A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

G. H. Nguyen, A. Bouzerdoum, and S. L. Phung. A supervised learning approach for imbalanced data sets. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

U. Nodelman, C. R. Shelton, and D. Koller. In *Proceedings of the Eighteenth International Conference on Uncertainty in Artificial Intelligence*, 2002.

U. Nodelman, C. R. Shelton, and D. Koller. Learning continuous time bayesian networks. In *Proceedings of the Nineteenth International Conference on Uncertainty in Artificial Intelligence*, 2003.

U. D. Nodelman. *Continuous time Bayesian networks.* PhD thesis, Stanford University, 2007.

T. Ort, L. Paull, and D. Rus. Autonomous vehicle navigation in rural environments without detailed prior maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

T. Ort, K. Murthy, R. Banerjee, S. K. Gottipati, D. Bhatt, I. Gilitschenski, L. Paull, and D. Rus. MapLite: Autonomous intersection navigation without a detailed prior map. *IEEE Robotics and Automation Letters (RA-L)*, 2019.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

D. W. Otter, J. R. Medina, and J. K. Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

X. Pan, Y. You, Z. Wang, and C. Lu. Virtual to real reinforcement learning for autonomous driving. In *British Machine Vision Conference*, 2017.

H. Papadopoulos and H. Haralambous. Reliable prediction intervals with regression neural networks. *Neural Networks*, 24(8):842–851, 2011.

F. Paredes-Valles and G. C. H. E. de Croon. Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3446–3455, June 2021.

G. Parisi. *Statistical field theory.* Addison-Wesley, 1988.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013a.

R. Pascanu, G. Montufar, and Y. Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013b.

N. Patel, A. Choromanska, P. Krishnamurthy, and F. Khorrami. Sensor modality fusion with CNNs for UGV autonomous driving in indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

T. Pearce, M. Zaki, A. Brintrup, and A. Neel. Uncertainty in neural networks: Bayesian ensembling. *arXiv preprint arXiv:1810.05546*, 2018.

J. Pearl. *Causality.* Cambridge university press, 2009.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Elsevier, 2014.

J. Pearl et al. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

W. Penny, Z. Ghahramani, and K. Friston. Bilinear dynamical systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457):983–993, 2005.

L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms.* The MIT Press, 2017.

D. A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

L. S. Pontryagin. *Mathematical theory of optimal processes.* Routledge, 2018.

B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 3 edition, 2007.

X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga. Ensemble deep learning for regression and time series forecasting. In *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pages 1–6. IEEE, 2014.

A. Quaglino, M. Gallieri, J. Masci, and J. Koutník. SNODE: Spectral discretization of neural ODEs for system identification. In *International Conference on Learning Representations*, 2020.

M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2847–2854. JMLR. org, 2017.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa. An all-in-one convolutional neural network for face analysis. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pages 17–24. IEEE, 2017.

H. Rebecq, D. Gehrig, and D. Scaramuzza. ESIM: An open event camera simulator. In *Conference on Robot Learning*, pages 969–982. PMLR, 2018.

H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

T. Remez, J. Huang, and M. Brown. Learning to segment via cut-and-paste. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

N. Rhinehart, R. McAllister, and S. Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Y. Rubanova, R. T. Chen, and D. Duvenaud. Latent ODEs for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907*, 2019.

P. K. Rubenstein, S. Bongers, B. Schölkopf, and J. M. Mooij. From deterministic ODEs to dynamic structural causal models. *arXiv preprint arXiv:1608.08028*, 2016.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

F. Ruggeri, F. Faltin, and R. Kenett. Bayesian networks. encyclopedia of statistics in quality and reliability, 2007.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

S. Russell and P. Norvig. *Artificial intelligence: a modern approach.* 2002.

F. Sadeghi and S. Levine. CAD2RL: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

P. Sattigeri, S. C. Hoffman, V. Chenthamarakshan, and K. R. Varshney. Fairness GAN. *arXiv preprint arXiv:1805.09910*, 2018.

M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A platform for embodied ai research. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

S. Schneider, A. Baevski, R. Collobert, and M. Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

B. Schölkopf. Causality for machine learning. *arXiv preprint arXiv:1911.10500*, 2019.

Y. Schroecker and C. Isbell. State aware imitation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2915–2924, 2017.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv prepring:1707.06347*, 2017.

W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50), 2019a.

W. Schwarting, A. Pierson, S. Karaman, and D. Rus. Stochastic dynamic games in belief space. *arxiv preprint: 1909.06963*, 2019b.

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-CAM: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

M. Sensoy, L. Kaplan, and M. Kandemir. Evidential Deep Learning to Quantify Classification Uncertainty. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

T. Serra, C. Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks. *arXiv preprint arXiv:1711.02114*, 2017.

S. Shah, D. Dey, C. Lovett, and A. Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *FSR*, 2017.

S. Shah, D. Dey, C. Lovett, and A. Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

L. F. Shampine. Computer solution of ordinary differential equations. *The Initial Value Problem*, 1975.

L. Shen, L. R. Margolies, J. H. Rothstein, E. Fluder, R. McBride, and W. Sieh. Deep learning to improve breast cancer detection on screening mammography. *Scientific reports*, 9(1):1–12, 2019.

C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield. Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4241–4247. IEEE, 2017.

K. Sofiiuk, P. Popenova, and A. Konushin. Foreground-aware semantic representations for image harmonization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1620–1629, 2021.

A. P. Soleimany, A. Amini, S. Goldman, D. Rus, S. N. Bhatia, and C. W. Coley. Evidential deep learning for guided molecular property prediction and discovery. *ACS central science*, 7(8):1356–1367, 2021.

P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman. *Causation, prediction, and search.* MIT press, 2000.

G. J. Stephens, B. Johnson-Kerner, W. Bialek, and W. S. Ryu. Dimensionality and dynamics in the behavior of C. elegans. *PLoS computational biology*, 4(4):e1000028, 2008.

M. Sun and X. Ma. Adversarial imitation learning from incomplete demonstrations. *arXiv preprint arXiv:1905.12310*, 2019.

R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. *Advances in neural information processing systems*, 26, 2013a.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013b.

Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess. dm_control: Software and tasks for continuous control. *arXiv preprint: 2006.12983*, 2020.

R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

Tesla. Tesla Autopilot, 2020. URL https://www.tesla.com/autopilot.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2012a.

E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012b.

J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015.

T. Tsiligkaridis. Information robust dirichlet networks for predictive uncertainty estimation. *arXiv preprint arXiv:1910.04819*, 2019.

S. Tulyakov, D. Gehrig, S. Georgoulis, J. Erbach, M. Gehrig, Y. Li, and D. Scaramuzza. Time lens: Event-based video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16155–16164, June 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

C. Vorbach, R. Hasani, A. Amini, M. Lechner, and D. Rus. Causal navigation by continuous-time neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun. V2VNet: Vehicle-to-vehicle communication for joint perception and prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

T.-H. Wang, A. Amini, W. Schwarting, I. Gilitschenski, S. Karaman, and D. Rus. Learning interactive driving policies via data-driven simulation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2021.

Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

G. Wei, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian. Intention-Net: Integrating planning and deep learning for goal-directed autonomous navigation. *arXiv preprint arXiv:1710.05627*, 2017.

S. Weichwald, M. E. Jakobsen, P. B. Mogensen, L. Petersen, N. Thams, and G. Varando. Causal structure learning from time series: Large regression coefficients may predict causal links better in practice than small p-values. In *NeurIPS 2019 Competition and Demonstration Track*, pages 27–36. PMLR, 2020.

S. R. Wicks, C. J. Roehrig, and C. H. Rankin. A dynamic network simulation of the nematode tap withdrawal circuit: Predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, 16(12):4017–4031, 1996.

G. Williams, B. Goldfain, P. Drews, J. M. Rehg, and E. A. Theodorou. Autonomous racing with AutoRally vehicles and differential games. *arXiv preprint: 1707.04540*, 2017.

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

R. W. Wolcott and R. M. Eustice. Visual localization within LiDAR maps for automated urban driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827. PMLR, 2019.

B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 4(6):2, 2000.

F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson Env: Real-world perception for embodied agents. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López. Multimodal end-to-end autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2020a.

Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López. Multimodal end-to-end autonomous driving. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, 2020b.

H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li. Depth completion from sparse lidar data with depth-normal constraints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

G. Yan, P. E. Vértes, E. K. Towlson, Y. L. Chew, D. S. Walker, W. R. Schafer, and A.-L. Barabási. Network control principles predict neuron function in the Caenorhabditis elegans connectome. *Nature*, 550(7677):519, 2017.

Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659, 2016.

T. Yu, P. Abbeel, S. Levine, and C. Finn. One-shot hierarchical imitation learning of compound visuomotor tasks. *arXiv preprint arXiv:1810.11043*, 2018.

S. Zafeiriou, C. Zhang, and Z. Zhang. A survey on face detection in the wild: past, present and future. *Computer Vision and Image Understanding*, 138:1–24, 2015.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64 (3):107–115, 2021.

S. Zhao, J. Song, and S. Ermon. The information autoencoding family: A Lagrangian perspective on latent variable generative models. *arXiv preprint arXiv:1806.06514*, 2018.

Y. Zhao, L. Bai, Z. Zhang, and X. Huang. A surface geometry model for lidar depth completion. *IEEE Robotics and Automation Letters*, 6(3):4457–4464, 2021. doi: 10.1109/LRA.2021.3068885.

Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11): 3212–3232, 2019.

Z.-H. Zhou and X.-Y. Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77, 2006.

J. Zhuang, N. Dvornek, X. Li, S. Tatikonda, X. Papademetris, and J. Duncan. Adaptive checkpoint adjoint method for gradient estimation in neural ODE. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR 119, 2020.