SAFETY-CRITICAL MACHINE LEARNING: DEVELOPMENT AND TESTING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Aman Sinha

August 2020

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(John Duchi) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Mykel Kochenderfer)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Marco Pavone)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dorsa Sadigh)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Russ Tedrake)

Approved for the Stanford University Committee on Graduate Studies

Abstract

As machine-learning systems begin deployment in safety-critical domains such as medical imaging and autonomous driving, model failure is increasingly costly. In such applications, it is dangerous to deploy models whose robustness and failure modes we do not understand or cannot certify. This thesis presents techniques to both train and test safety-critical machine-learning systems. In the first part of this work, we employ the lens of *distributional robustness* to develop safety-critical models. Instead of just performing well on a nominal training set, distributionally robust models are designed to perform well on uncertainty sets around the data-generating distribution. In regimes with limited, bounded uncertainty sets (*e.g.* adversarial perturbations to images), we train certifiably robust models with negligible losses in computational or statistical efficiency. In regimes with high uncertainty, we learn how to balance safety and model performance using synthetic data. We employ the latter technique in the domain of autonomous racing, demonstrating safe yet competitive autonomous racing algorithms on real 1/10th-scale vehicles.

In the second part of the thesis, we frame testing safety-critical models through the lens of *risk*. In contrast to formal verification and other traditional software-testing techniques, we present a "risk-based framework," where the goal is to calculate the probability of failure under a base distribution of environment behavior. For safety-critical algorithms, this probability is small and the resulting technical challenge is a rare-event simulation problem. We develop a novel, provably efficient rare-event simulation method that combines exploration, exploitation, and optimization techniques to efficiently find failure modes and estimate their rate of occurrence. We apply this technique as a tool for rapid sensitivity analysis and model comparison in a variety of applications, showcasing its usefulness in efficiently testing safety-critical autonomous systems.

Acknowledgements

This thesis is the product of six wonderful years at Stanford. Sometimes, one builds up a dream so much that reality cannot quite match the expectation. In contrast, my time at Stanford has not only met but surpassed all of my hopes and expectations for graduate school. A large part of this has been due to the amazing colleagues and mentors I have collaborated with during my time here.

First, I would like to thank my advisor, John Duchi. John and I both joined Stanford at the same time—me entering as a naive, wide-eyed graduate student and John returning to his alma mater as a newly minted professor. After a somewhat fortuitous meeting over burritos, John took a chance on me and accepted my request to try out doing some research together. I couldn't have asked for a better advisor, colleague, mentor, and friend who has guided me as I explored my intellectual interests and who has helped me give purpose to my ambitions. John has a great mix of theoretical rigor with a fun, laid-back attitude. This duality is perhaps best illustrated by the fact that many of our meetings have involved throwing a frisbee, doing planks, or watching Youtube videos while figuring out how to solve subtle problems in proofs.

I would also like to thank my "surrogate co-advisor" Russ Tedrake. Due to administrative reasons, Russ is "only" a member of my reading committee, but this title belies the major impact he has had on shaping the nature of my research. I met Russ while interning for his team at Toyota Research Institute (TRI) halfway through my PhD. I can trace the inflection point of my research to my first meeting with him in Cambridge, when he provided some early feedback on my ideas and encouraged me to meet one of his visiting students, Matthew O'Kelly. I have been lucky to collaborate with both of them ever since. Like John, Russ has a striking duality of traits: he is a giant in robotics and knows everything it takes to get systems to actually work in real life, yet, at the same time, he has a humble, patient, and magnanimous personality. Russ always makes time for his students (even his

"surrogate" students like me) despite the fact that he is one of the busiest people I know. Together, John and Russ have provided the perfect balance of guidance that has enabled our research to push the boundaries of safety-critical machine learning. They are also great role models in their ability to excel in their professions while maintaining their commitment to their families.

Many thanks to the other members of my committee: Mykel Kochenderfer, Marco Pavone, Dorsa Sadigh, and Art Owen. Collaborating with them and their students has been invaluable to my work over the past few years; they have been excellent sounding boards for new questions, and they have been especially helpful in providing different perspectives on similar research questions. Art's class on Monte Carlo methods was one of the most influential classes I took at Stanford regarding my research career; I am glad he was able to chair the committee of my oral defense.

In addition to meeting and working with amazing faculty during my PhD, I've had the privilege of meeting new collaborators as well as keeping in touch with old ones. In some cases the old and new worlds collided fortuitously: Justin Norden and Matt O'Kelly have been amazing colleagues and friends, and I am fortunate that I get to collaborate with both of them on our continuing endeavors. Additionally, Hilal Asi, Vivek Bagaria, Yair Carmon, Karan Chadha, Steve Drapcho, Matt Fanelli, Daniel Levy, Tiras Lin, Horia Mania, Hongseok Namkoong, Maxim Rabinovich, Ludwig Schmidt, Vatsal Sharan, Jacob Steinhardt, John Subosits, Josh Wortzel, Steve Yadlowsky, Jesse Zhang, Billy Zheng, Jason Zhu, and many others have made my experience at Stanford richer and more fulfilling than I could have imagined.

During my busy time at Stanford, I squeezed in two internships at TRI and Quantifind, and I would like to thank Jon DeCastro and John Stockton for their guidance. These research experiences gave me a taste of doing research within a real-world setting, and those lessons have been invaluable to my personal growth. Additionally, my collaborations with Christy Tomkins-Lane, Matthew Smuck, Robin Sun, and others in the Wearable Health Lab made me appreciate the ability of machine learning to actually *help* people, whether in medical applications or beyond.

I am grateful to the Stanford Office of the Vice Provost for Graduate Education for offering me a Stanford Graduate Fellowship and the Hewlett family for their sponsorship of my fellowship. I would also like to thank the Fannie and John Hertz Foundation for offering me the "freedom to innovate" and warmly welcoming me into an inspiring community of

scholars. Interacting with the community of Hertz Fellows has led to a variety of connections, opportunities, and ideas that I hadn't ever considered as part of my path.

Finally, I would like to thank my family for their unconditional love, support, and guidance. I consider myself lucky to be part of a large and close-knit extended family. Due to the many role models I have around me, I grew up understanding the critical role of education in creating a life of value. In particular, I am indebted to my father Neeraj, who cultivated my love of math, science, and engineering from an early age, my mother Vandana, who taught me the importance of creativity in my pursuits, and my sister Aashna, who has helped me appreciate the importance of the humanities in contextualizing my technical pursuits. For a few years, I had the surprising opportunity of being able to "talk shop" with my Dada (grandfather) as he embarked on a second PhD in his 80s just for fun. Although he is no longer with us, my memories of his sharp wit, unrelenting commitment to his family, and overall joie de vivre carry on as a guiding light.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> *There are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns—the ones we don't know we don't know.*
>
> — Donald Rumsfeld

Over the past two decades, machine-learning (ML) methods have accelerated progress across various domains such as automated image recognition [249], machine translation [178], and robotic systems [5, 146]. This success has been largely due to the advent of low-cost and high-throughput data-collection methods across nearly every field of human endeavor, a phenomenon that led to dubbing the previous decade as the "Age of Big Data" [181, 229]. Due to the availability of such large, widely-accessible datasets, the standard paradigm for measuring model performance has been average-case performance over test data. New advancements have largely followed the model of increasing model complexity and attaining larger datasets [280].

The widespread adoption of ML across all domains of scientific inquiry belies the reality that ML models do not yet achieve the level of reliability expected of modern software. First of all, they are sensitive to data perturbations. Indeed, perturbations ranging from unstructured noise to structured distributional shifts and adversarial attacks [26, 116] can catastrophically degrade performance [213, 170, 202]. Second, proving model performance over various operating domains is challenging if not intractable for modern architectures [180, 152], so constructively analyzing failure modes from first principles is difficult. In other words, modern ML systems are brittle and there is no straightforward way to analyze this

sensitivity other than testing average performance over existing data. Fortunately, average performance over non-perturbed data has so far been appropriate for the aforementioned successful applications of ML. We are accustomed to the rare mistranslation of a webpage or misclassification of a relative in a set of family photos; even when these failures occur, they are somewhat benign.

Now, however, ML systems are beginning to make high-stakes decisions—as components of autonomous vehicles [122], banking and stock market infrastructure [254], and medical devices [98]—where average test performance is no longer sufficient to assess model quality. Indeed, failures in such safety-critical ML systems are dangerous and costly. As such, there is a growing need for increased rigor regarding model sensitivity to uncertainties. Specifically, engineers need to be able to deploy systems that are certifiably robust, meaning that the models perform provably reliably under specified models of uncertainty. Conversely, regulators and other stakeholders need tools to impartially assess sensitivities of deployed models, even those that claim to be certifiably robust; given a model, they need to be able to find otherwise unknown modes of failure before deployment. Essentially, better training methods and testing methods are necessary to raise the bar of ML methods to the standards of model governance that we expect of modern software. Only then can such methods be ready for safety-critical applications.

This thesis develops training and testing techniques for safety-critical ML models. For training, we consider the lens of *distributional robustness*, wherein we explicitly model uncertainties to the data-generating distribution and build models that have performance guarantees over the entire uncertainty set. This lens formalizes the notion of being robust to *known unknowns*, uncertainties that we can model at training time. As we show in Section 1.1 below, building distributionally robust models takes the form of minimax optimization problems.

For testing, we consider the lens of *risk*, where the goal is to not only discover failure modes for a model but also their rate of occurrence. This lens formalizes the notion of understanding model sensitivities to *unknown unknowns*—that is, we would like to discover failure modes for a model and understand its weaknesses before deployment. Under this framework, certificates of model operation take the form of probabilities—we say a model is ready for deployment when its probability of failure is below some acceptable threshold. For safety-critical applications, this probability is rare, and accurately determining it takes the form of a rare-event simulation problem (see Section 1.1 below).

1.1 Governing equations

The two lenses that frame this thesis—distributional robustness and risk—naturally lend themselves to the overarching governing equations that will guide the technical approaches of the following chapters. Here we provide an overview of these equations as well as contextual background for their development.

1.1.1 Distributional robustness

The classical stochastic optimization problem central to machine learning takes the form

$$\underset{\theta \in \Theta}{\text{minimize}} \, \mathbb{E}_{P_0}[\ell(\theta; Z)] \tag{1.1}$$

over a parameter $\theta \in \Theta$, where $Z \sim P_0$, P_0 is a distribution on a space $\mathcal{Z}$, and ℓ is a loss function. In the most common instantiation of this problem—supervised learning—we have $Z = (X, Y)$ for feature vectors X and labels Y; this setting is the focus of most of Chapter 2. However, with only slight modifications to this equation, we also encompass the domain of reinforcement learning, which we cover in Section 2.5.3 and Chapter 3. There are a variety of classic texts on non-robust formulations of supervised and reinforcement learning (*e.g.* [120, 274]).

As a first try towards robustifying problem (1.1), one can consider an uncertainty set around every input z. The community on robust optimization considers worst-case problems of this form

$$\underset{\theta \in \Theta}{\text{minimize}} \sup_{u \in \mathcal{U}} \ell(\theta; z + u) \tag{1.2}$$

for some uncertainty set $\mathcal{U}$ [235, 27, 304]. Classical approaches to this deterministic form of robustness tend to be overly conservative and require specially structured losses ℓ and uncertainty sets $\mathcal{U}$ to solve efficiently.

Distributional robustness is a relaxation of the overly conservative classical approach to robust optimization. Instead of considering uncertainties over every datapoint z, we consider an uncertainty set over the distribution P_0:

$$\underset{\theta \in \Theta}{\text{minimize}} \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[\ell(\theta; Z)]. \tag{1.3}$$

Importantly, this formulation subsumes classical robust optimization, as we could define $\mathcal{P}$

as a collection of uncertainties $\mathcal{U}$ around the datapoints z. However, the formulation (1.3) is much more flexible and allows more expressive classes of uncertainty sets. For example, one can consider uncertainty sets based on distances between distributions. In particular, two popular uncertainty sets depend on the Wasserstein distance [93, 257, 40], which is the focus of Chapter 2, or f-divergences [28, 34, 173, 199, 89, 207, 208], which is the focus of Chapter 3. As we will show in the sequel, the choice of uncertainty set depends on the types of perturbations one wishes to model and guard against when training an ML model.

As with classical robust optimization, the choice of $\mathcal{P}$ also affects the tractability of the optimization problem (1.3). This has led several authors to constrain the models over which they apply distributionally robust techniques (*e.g.* Namkoong and Duchi [208] consider convex losses ℓ and Blanchet et al. [40] consider convex losses ℓ as well as a constrained set of Wasserstein costs). In contrast, we use the application domain as our guide and proceed in the reverse direction. In both Chapters 2 and 3, we do not constrain the models and, furthermore, we guide the parametrization of $\mathcal{P}$ by the need of the specific application. We then proceed by making various relaxations and approximations to convert otherwise intractable problems into efficient optimization problems that still have meaningful certificates of robustness.

1.1.2 Risk

The problem of testing safety-critical ML algorithms necessitates moving beyond the standard metric of measuring average performance over a nominal test set. In particular, safety-critical algorithms have a small failure rate (otherwise they would not be suitable for the safety-critical application), so the number of examples in this nominal test set is low and yields little information regarding the ways in which the model is sensitive to tails in the data-generating distribution P_0. As such, several communities have adapted existing tools for testing safety-critical software to modern ML systems.

The first of these tools is formal verification (*e.g.* [163, 65, 6]). Such methods attempt to prove the "correctness" of an algorithm. Let $x \in \mathcal{X}$ be an input to the system and $g : \mathcal{X} \to \{0, 1\}$ a binary specification function such that $g(x) = 0$ corresponds to "incorrect" behavior. Then, the goal of verification is to prove that the set $\mathcal{G} := \{x : g(x) = 0\}$ is empty. The main problems with this approach are threefold. First, constructing the specification function g can be extremely difficult as notions of correctness are often ill-defined when interacting with stochastic human agents in unstructured environments. Unfortunately, ruling out

scenarios where the algorithm is not at fault is subject to logical inconsistency, combinatorial growth in specification complexity, and subjective assignment of fault. Second, even if a specification is given, it is extremely difficult if not impossible to formally verify modern safety-critical ML models due to their size and complexity [152]. Verification of individual subcomponents, while potentially tractable, does not necessarily translate to overall safety when all subsystems operate together. Finally, verification requires rewriting the model in a formal language, which is difficult for modern ML systems.

Numerous communities have thus moved beyond verification to more tractable approaches that generally fall under the framework of falsification [71]. Classical falsification (*e.g.* [94, 80, 9, 312, 85, 231]) attempts to find *any* failures. That is, classical falsification attempts to solve the problem

$$\underset{x \in \mathcal{X}}{\text{minimize}}\, g(x) \tag{1.4}$$

To make this approach more amenable to black-box optimization techniques, we can also consider continuous metrics $g : \mathcal{X} \to [0, 1]$. However, the mere existence of failures is trivial to demonstrate in unstructured environments [260], and falsification does not exploit a generative model to prioritize high-likelihood failures over lower-likelihood ones in its search process. Probabilistic falsification (*e.g.* [175, 164]), on the other hand, searches for the most probable failure under a generative model. That is, considering a data-generating distribution P_0 as above and an associated likelihood function $\rho_0(x) : \mathcal{X} \to [0, \infty)$, probabilistic falsification solves the problem

$$\underset{x \in X}{\text{maximize}}\, \rho_0(x)(1 - g(x)) \tag{1.5}$$

This approach is more relevant to safety-critical ML systems that operate in unstructured environments, because it prioritizes higher-likelihood failures.

Our approach builds upon probabilistic falsification. Instead of looking for just the most probable failure, we attempt to get coverage of all likely failures, thereby building a comprehensive view of the *risk* (likelihood and severity) of failure for the ML model. To do so, we solve the problem of finding the overall probability of failure. Again, we let P_0 denote the data-generating distribution and we let $\mathcal{X}$ be a space of inputs to the system. Because we will tend to think of dynamic settings (*e.g.* the ML model is a controller for a self-driving car or a medical device) rather than noninteractive supervised-learning settings, it is useful to think of the random variable $X \sim P_0$ as defining a realization of a simulation.

For an objective function $f : \mathcal{X} \to \mathbb{R}$ that measures "safety"—so that low values of $f(x)$ correspond to dangerous scenarios—our goal is to evaluate the probability of a dangerous event

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma) \tag{1.6}$$

for some threshold γ. We call this overall approach to evaluating safety the "risk-based framework." Importantly, solving problem (1.6) requires three main components:

(a) Simulator: Because evaluating a safety-critical algorithm in the real world is slow and dangerous, we require the ability to simulate performance in a virtual world at scale. Chapter 4 presents a scalable simulation engine developed around the use case of autonomous highway driving.

(b) Generative models: The virtual world requires realistic agents and conditions that match the real world. Generative models provide distributions for low-level components (*e.g.* weather patterns and road conditions for an autonomous driving simulation) as well as high-level components (*e.g.* behaviors of other drivers and/or pedestrians in an autonomous driving simulator). Chapter 4 presents methods to build generative models for high-level components using imitation learning on real-world datasets.

(c) Search algorithm: With a simulator and generative models, one can simulate an infinite number of scenarios. However, we would like to focus on the rare tails of the distribution P_0 that induce low values for f. Since these tails are *a priori* unknown, we employ a search algorithm to solve this *rare-event simulation* problem [13]. Chapter 4 presents a naive search algorithm—the cross-entropy method [247]—in the context of autonomous driving. Chapter 5 presents a novel technique that has guarantees for statistical and computational efficiency. We apply this technique to a variety of simulators with given generative models.

1.2 Outline and organization

This thesis is composed of two parts. In the first part, we focus on methods to develop safety-critical machine-learning algorithms. In Chapter 2, we consider building models that are reliable against small uncertainty sets $\mathcal{P}$. In particular, we consider perturbing the

underlying data distribution in a Wasserstein ball, and we provide a training procedure that provably achieves moderate levels of robustness with little computational or statistical cost relative to empirical risk minimization for smooth losses. Indeed, our procedure augments model parameter updates with efficient data perturbations, and our resulting statistical certificates of robustness are efficiently computable. This chapter is based upon joint work with Hongseok Namkoong, Riccardo Volpi, and John Duchi [265].

In Chapter 3, we focus on developing safety-critical models in the high-uncertainty regime, where $\mathcal{P}$ is unbounded or simply unknown. In such regimes, balancing safety with performance is critical because conservativeness degrades model performance whereas aggressiveness is dangerous. Our overall approach first learns an operational proxy for $\mathcal{P}$ that enables a tractable distributionally robust optimization problem. Because we place such a high emphasis on safety, we learn $\mathcal{P}$ without gathering any external data; we use only synthetic data. For this chapter, we use the concrete domain of autonomous racing as a case study for the method. In particular, we develop a novel method for synthesizing an uncertainty set, which in this application manifests itself as population of good racing policies, using self-play techniques. We then use this uncertainty set within a distributionally robust bandit optimization procedure that adaptively adjusts safety based on uncertainty in opponents' behaviors. We quantify the tradeoff between robustness and performance of this two-stage approach, and we experimentally demonstrate it on real 1/10th-scale racecars. This chapter is based on joint work with Matthew O'Kelly, Hongrui Zheng, Rahul Mangharam, John Duchi, and Russ Tedrake [267].

In the second part of the thesis, we focus on testing safety-critical models utilizing the lens of risk. Chapter 4 presents the risk-based framework through the case-study of autonomous driving. Autonomous vehicle (AV) technology still lacks techniques for rigorous and scalable testing. Real-world testing is dangerous, and, due to the rare nature of accidents, slow and expensive. We implement a full simulation system capable of testing an entire AV system, and we employ this system to evaluate the probability of an accident under a base distribution of traffic behavior. Specifically, we demonstrate our framework on a highway scenario. This chapter is based on joint work with Matthew O'Kelly, Hongseok Namkoong, John Duchi, and Russ Tedrake [218].

Finally, Chapter 5 expands upon the risk-based framework by developing a novel rare-event simulation technique. Our method—neural bridge sampling—combines exploration, exploitation, and optimization techniques to find failure modes and estimate their rate of

occurrence. We rigorously analyze the computational and statistical performance of neural bridge sampling, providing theoretical and empirical evidence of its superiority over other state-of-the-art techniques. We demonstrate its usefulness as a tool for model comparison and rapid sensitivity analysis—both essential components to testing safety-critical algorithms—on a variety of application domains. This chapter is based on joint work with Matthew O'Kelly, John Duchi, and Russ Tedrake [266].

Part I

Developing Safety-Critical ML Systems

Chapter 2

Certifiable robustness against adversarial attacks

Take things always by their smooth handle.

— Thomas Jefferson

In this chapter, we study how to efficiently certify robustness to uncertainty sets $\mathcal{P}$ that are sufficiently small. Our motivation comes from the domain of image classification, where imperceptible perturbations to pixels—so-called "adversarial attacks" or "adversarial examples"[276, 116]—force neural-network classifiers to fail. Indeed, neural networks are vulnerable to adversarial examples and researchers have proposed many heuristic attack and defense mechanisms. We address this problem through the principled lens of distributionally robust optimization, which guarantees performance under adversarial input perturbations. By considering a Lagrangian penalty formulation of perturbing the underlying data distribution in a Wasserstein ball, we provide a training procedure that augments model parameter updates with worst-case perturbations of training data. For smooth losses, our procedure provably achieves moderate levels of robustness with little computational or statistical cost relative to empirical risk minimization. Furthermore, our statistical guarantees allow us to efficiently certify robustness for the population loss. For imperceptible perturbations, our method matches or outperforms heuristic approaches.

2.1 Introduction

Consider the classical stochastic optimization problem, in which we minimize an expected loss $\mathbb{E}_{P_0}[\ell(\theta; Z)]$ over a parameter $\theta \in \Theta$, where $Z \sim P_0$, P_0 is a distribution on a space $\mathcal{Z}$, and ℓ is a loss function. In many systems, robustness to changes in the data-generating distribution P_0 is desirable, whether they be from covariate shifts, changes in the underlying domain [26], or adversarial attacks [116, 170]. As deep networks become prevalent in modern performance-critical systems—prominent examples include perception systems for self-driving cars, and automated detection of tumors—model failure is increasingly costly. In these situations, it is irresponsible to deploy models whose robustness and failure modes we do not understand or cannot certify.

Recent work shows that neural networks are vulnerable to adversarial examples; seemingly imperceptible perturbations to data can lead to misbehavior of the model, such as misclassification of the output [116, 213, 170, 202]. Consequently, researchers have proposed adversarial attack and defense mechanisms [225, 226, 227, 245, 57, 125, 188, 283]. These works provide an initial foundation for adversarial training, but it is challenging to rigorously identify the classes of attacks against which they can defend (or if they exist). Alternative approaches that provide formal verification of deep networks [139, 152] are NP-hard in general; they require prohibitive computational expense even on small networks. Recently, researchers have proposed convex relaxations of the NP-hard verification problem with some success [162, 233], though they may be difficult to scale to large networks. Our work is situated between these agendas: we develop efficient procedures with rigorous guarantees for *small to moderate* amounts of robustness.

We take the perspective of distributionally robust optimization and provide an adversarial training procedure with provable guarantees on its computational and statistical performance. Postulating a class $\mathcal{P}$ of distributions around the data-generating distribution P_0, we consider

$$\underset{\theta \in \Theta}{\text{minimize}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[\ell(\theta; Z)]. \tag{2.1}$$

The choice of $\mathcal{P}$ influences robustness guarantees and computability; we develop robustness sets $\mathcal{P}$ with computationally efficient relaxations that apply even when the loss ℓ is non-convex. We provide an adversarial training procedure that, for smooth ℓ, enjoys convergence guarantees similar to non-robust approaches while *certifying* performance even for the worst-case population loss $\sup_{P \in \mathcal{P}} \mathbb{E}_P[\ell(\theta; Z)]$. On a simple implementation in Tensorflow, our

method takes 5–10× as long as stochastic gradient methods for empirical risk minimization (ERM), matching runtimes for other adversarial training procedures [116, 170, 188]. We show that our procedure—which learns to protect against adversarial perturbations in the training dataset—generalizes, allowing us to train a model that prevents attacks to the test dataset.

We briefly overview our approach. Let $c : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+ \cup \{\infty\}$, where $c(z, z_0)$ is the "cost" for an adversary to perturb z_0 to z (we typically use $c(z, z_0) = \|z - z_0\|_p^2$ with $p \geq 1$). We consider the robustness region $\mathcal{P} = \{P : W_c(P, P_0) \leq \rho\}$, a ρ-neighborhood of the distribution P_0 under the Wasserstein metric $W_c(\cdot, \cdot)$ (see Section 2.2 for a formal definition). For deep networks and other complex models, this formulation of problem (2.1) is intractable with arbitrary ρ. Instead, we consider its Lagrangian relaxation for a fixed penalty parameter $\gamma \geq 0$, resulting in the reformulation

$$\underset{\theta \in \Theta}{\text{minimize}} \left\{ F(\theta) := \sup_P \{\mathbb{E}_P[\ell(\theta; Z)] - \gamma W_c(P, P_0)\} = \mathbb{E}_{P_0}[\phi_\gamma(\theta; Z)] \right\} \tag{2.2a}$$

$$\text{where} \quad \phi_\gamma(\theta; z_0) := \sup_{z \in \mathcal{Z}} \{\ell(\theta; z) - \gamma c(z, z_0)\}. \tag{2.2b}$$

(See Proposition 2.1 for a rigorous statement of these equalities.) Here, we replaced the usual loss $\ell(\theta; Z)$ by the robust surrogate $\phi_\gamma(\theta; Z)$; this surrogate (2.2b) allows adversarial perturbations of the data z, modulated by the penalty γ. As P_0 is unknown, we solve the penalty problem (2.2) with P_0 replaced by the empirical distribution $\widehat{P}_n$; we refer to this as the penalty problem below.

The key feature of the penalty problem (2.2) is that moderate levels of robustness—in particular, defense against imperceptible adversarial perturbations—are achievable at essentially no computational or statistical cost for *smooth losses* ℓ. Specifically, for large enough penalty γ (by duality, small enough robustness ρ), the function $z \mapsto \ell(\theta; z) - \gamma c(z, z_0)$ in the robust surrogate (2.2b) is strongly concave and hence easy to optimize if $\ell(\theta, z)$ is smooth in z. Consequently, stochastic gradient methods applied to problem (2.2) have similar convergence guarantees as for non-robust methods (ERM). In Section 2.3, we provide a *certificate of robustness* for any ρ; we give an efficiently computable data-dependent upper bound on the worst-case loss $\sup_{P: W_c(P,P_0) \leq \rho} \mathbb{E}_P[\ell(\theta; Z)]$. That is, the worst-case performance of the output of our principled adversarial training procedure is guaranteed to be no worse than this certificate. Our bound is tight when $\rho = \widehat{\rho}_n$, the achieved robustness for the empirical objective. These results suggest advantages of networks with smooth

activations rather than ReLUs. In Section 2.4, we substantiate our optimization guarantees (Section 2.2) and certificate of robustness (Section 2.3) by providing concrete bounds on the smoothness levels of neural networks. We experimentally verify our results in Section 2.5 and show that we match or achieve state-of-the-art performance on a variety of adversarial attacks.

Robust optimization and adversarial training The standard robust-optimization approach minimizes worst-case losses of the form $\sup_{u \in \mathcal{U}} \ell(\theta; z + u)$ for some uncertainty set $\mathcal{U}$ [235, 27, 304]. Unfortunately, this approach is intractable except for specially structured losses, such as the composition of a linear and simple convex function [27, 304, 305]. Nevertheless, this robust approach underlies recent advances in adversarial training [276, 116, 226, 188], which heuristically perturb data during a stochastic optimization procedure.

One such heuristic uses a locally linearized loss function (proposed with $p = \infty$ as the "fast gradient sign method"[116]):

$$\Delta_{x_i}(\theta) := \operatorname*{argmax}_{\|\eta\|_p \le \epsilon} \{\nabla_x \ell(\theta; (x_i, y_i))^T \eta\} \quad \text{and perturb} \quad x_i \to x_i + \Delta_{x_i}(\theta). \tag{2.3}$$

One form of adversarial training trains on the losses $\ell(\theta; (x_i + \Delta_{x_i}(\theta), y_i))$ [116, 170], while others perform iterated variants [226, 57, 188, 283]. Madry et al. [188] observe that these procedures attempt to optimize the objective $\mathbb{E}_{P_0}[\sup_{\|u\|_p \le \epsilon} \ell(\theta; Z + u)]$, a constrained version of the penalty problem (2.2). This notion of robustness is typically intractable: the inner supremum is generally non-concave in u, so it is unclear whether model-fitting with these techniques converges, and there are possibly worst-case perturbations these techniques do not find. Indeed, it is NP-hard to find worst-case perturbations when deep networks use ReLU activations, suggesting difficulties for fast and iterated heuristics (see Lemma 2.2 in Section 2.2.1). Smoothness, which can be obtained in standard deep architectures with exponential linear units (ELUs)[69], allows us to find Lagrangian worst-case perturbations with low computational cost.

Distributionally robust optimization To situate the current work, we review some of the substantial body of work on robustness and learning. The choice of $\mathcal{P}$ in the robust objective (2.1) affects both the richness of the uncertainty set we wish to consider as well as the tractability of the resulting optimization problem. Previous approaches to distributional

robustness have considered finite-dimensional parametrizations for $\mathcal{P}$, such as constraint sets for moments, support, or directional deviations [64, 78, 114], as well as non-parametric distances for probability measures such as f-divergences [28, 34, 173, 199, 89, 207], and Wasserstein distances [93, 257, 40, 104, 41, 105, 167]. In constrast to f-divergences (*e.g.* χ^2- or Kullback-Leibler divergences) which are effective when the support of the distribution P_0 is fixed, a Wasserstein ball around P_0 includes distributions Q with different support and allows (in a sense) robustness to unseen data.

Many authors have studied tractable classes of uncertainty sets $\mathcal{P}$ and losses ℓ. For example, Ben-Tal et al. [28], Lam [172] and Namkoong and Duchi [208] use convex optimization approaches for f-divergence balls. For worst-case regions $\mathcal{P}$ formed by Wasserstein balls, Esfahani and Kuhn [93], Shafieezadeh-Abadeh et al. [257], Blanchet et al. [40], and Kuhn et al. [167] propose tractable approaches for solving the saddle-point problem (2.1), such as converting it into a regularized ERM problem; but this is possible only for a limited class of convex losses ℓ and costs c. As we are interested in machine learning applications, we treat a larger class of losses and costs and provide direct solution methods for a Lagrangian relaxation of the saddle-point problem (2.1).

One natural application is domain adaptation; Lee and Raginsky [174] provide guarantees similar to ours for the empirical minimizer of the robust saddle-point problem (2.1) and give specialized bounds for domain adaptation problems. Their bounds rely on concentration of the empirical distribution to its population counterpart in Wasserstein distance, which may be prohibitively slow even in moderate dimensional problems. In contrast, our statistical guarantees have the usual dependence on the dimension based on covering numbers, and we develop efficient optimization procedures for our distributionally robust approach. Through an extensive set of experiments, we provide empirical evidence that our algorithm defends against imperceptible adversarial perturbations.

2.2 Proposed approach

Our approach is based on the following simple insight: assume that the function $z \mapsto \ell(\theta; z)$ is smooth, meaning there is some L for which $\nabla_z \ell(\theta; \cdot)$ is L-Lipschitz. Then for any $c : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+ \cup \{\infty\}$ 1-strongly convex in its first argument, a Taylor expansion yields

$$\ell(\theta; z') - \gamma c(z', z_0) \leq \ell(\theta; z) - \gamma c(z, z_0) + \langle \nabla_z (\ell(\theta; z) - \gamma c(z, z_0)), z' - z \rangle + \frac{L - \gamma}{2} \left\| z - z' \right\|_2^2 . \quad (2.4)$$

For $\gamma \geq L$ this is the first-order condition for $(\gamma - L)$-strong concavity of $z \mapsto (\ell(\theta; z) - \gamma c(z, z_0))$. Thus, whenever the loss is smooth enough in z and the penalty γ is large enough (corresponding to less robustness), computing the surrogate (2.2b) is a strongly-concave optimization problem.

We leverage the insight (2.4) to show that as long as we do not require *too much* robustness, this strong concavity approach (2.4) provides a computationally efficient and principled approach for robust optimization problems (2.1). Our starting point is a duality result for the minimax problem (2.1) and its Lagrangian relaxation for Wasserstein-based uncertainty sets, which makes the connections between distributional robustness and the "lazy" surrogate (2.2b) clear. We then show (Section 2.2.1) how stochastic gradient descent methods can efficiently find minimizers (in the convex case) or approximate stationary points (when ℓ is non-convex) for our relaxed robust problems.

Wasserstein robustness and duality Wasserstein distances define a notion of closeness between distributions. Let $\mathcal{Z} \subset \mathbb{R}^m$, and let $(\mathcal{Z}, \mathcal{A}, P_0)$ be a probability space. Let the transportation cost $c : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$ be nonnegative, lower semi-continuous, and satisfy $c(z, z) = 0$. For example, for a differentiable convex $h : \mathcal{Z} \to \mathbb{R}$, the Bregman divergence $c(z, z_0) = h(z) - h(z_0) - \langle \nabla h(z_0), z - z_0 \rangle$ satisfies these conditions. For probability measures P and Q supported on $\mathcal{Z}$, let $\Pi(P, Q)$ denote their couplings, meaning measures M on $\mathcal{Z}^2$ with $M(A, \mathcal{Z}) = P(A)$ and $M(\mathcal{Z}, A) = Q(A)$. The Wasserstein distance between P and Q is

$$W_c(P, Q) := \inf_{M \in \Pi(P,Q)} \mathbb{E}_M[c(Z, Z')].$$

For $\rho \geq 0$ and distribution P_0, we let $\mathcal{P} = \{P : W_c(P, P_0) \leq \rho\}$, considering the Wasserstein form of the robust problem (2.1) and its Lagrangian relaxation (2.2) with $\gamma \geq 0$. The following duality result [39, 104] gives the equality (2.2) for the relaxation and an analogous result for the problem (2.1). We give an alternative proof in Appendix A.2.1 for convex, continuous cost functions.

Proposition 2.1. *Let $\ell : \Theta \times \mathcal{Z} \to \mathbb{R}$ and $c : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+$ be continuous. Let $\phi_\gamma(\theta; z_0) = \sup_{z \in \mathcal{Z}} \{\ell(\theta; z) - \gamma c(z, z_0)\}$ be the robust surrogate (2.2b). For any distribution Q and any $\rho > 0$,*

$$\sup_{P : W_c(P,Q) \leq \rho} \mathbb{E}_P[\ell(\theta; Z)] = \inf_{\gamma \geq 0} \left\{\gamma \rho + \mathbb{E}_Q[\phi_\gamma(\theta; Z)]\right\}, \tag{2.5}$$

and for any $\gamma \geq 0$, we have

$$\sup_P \{\mathbb{E}_P[\ell(\theta; Z)] - \gamma W_c(P, Q)\} = \mathbb{E}_Q[\phi_\gamma(\theta; Z)]. \tag{2.6}$$

Leveraging the insight (2.4), we give up the requirement that we wish a prescribed amount ρ of robustness (solving the worst-case problem (2.1) for $\mathcal{P} = \{P : W_c(P, P_0) \leq \rho\}$) and focus instead on the Lagrangian penalty problem (2.2) and its empirical counterpart

$$\underset{\theta \in \Theta}{\text{minimize}} \left\{ F_n(\theta) := \sup_P \left\{ \mathbb{E}[\ell(\theta; Z)] - \gamma W_c(P, \widehat{P}_n) \right\} = \mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta; Z)] \right\}. \tag{2.7}$$

2.2.1 Optimizing the robust loss by stochastic gradient descent

We now develop stochastic gradient-type methods for the relaxed robust problem (2.7), making clear the computational benefits of relaxing the strict robustness requirements of formulation (2.5). We begin with assumptions we require, which quantify the amount of robustness we can provide.

Assumption 2.1. *The function $c : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+$ is continuous. For each $z_0 \in \mathcal{Z}$, $c(\cdot, z_0)$ is 1-strongly convex with respect to the norm $\|\cdot\|$.*

To guarantee that the robust surrogate (2.2b) is tractably computable, we also require a few smoothness assumptions. Let $\|\cdot\|_*$ be the dual norm to $\|\cdot\|$; we abuse notation by using the same norm $\|\cdot\|$ on Θ and $\mathcal{Z}$, though the specific norm is clear from context.

Assumption 2.2. *The loss $\ell : \Theta \times \mathcal{Z} \to \mathbb{R}$ satisfies the Lipschitzian smoothness conditions*

$$\left\|\nabla_\theta \ell(\theta; z) - \nabla_\theta \ell(\theta'; z)\right\|_* \leq L_{\theta\theta} \left\|\theta - \theta'\right\|, \quad \left\|\nabla_z \ell(\theta; z) - \nabla_z \ell(\theta; z')\right\|_* \leq L_{\mathsf{zz}} \left\|z - z'\right\|,$$
$$\left\|\nabla_\theta \ell(\theta; z) - \nabla_\theta \ell(\theta; z')\right\|_* \leq L_{\theta\mathsf{z}} \left\|z - z'\right\|, \quad \left\|\nabla_z \ell(\theta; z) - \nabla_z \ell(\theta'; z)\right\|_* \leq L_{\mathsf{z}\theta} \left\|\theta - \theta'\right\|.$$

These properties guarantee both (i) the well-behavedness of the robust surrogate ϕ_γ and (ii) its efficient computability. Making point (i) precise, Lemma 2.1 shows that if γ is large enough and Assumption 2.2 holds, the surrogate ϕ_γ is still smooth. Throughout, we assume $\Theta \subseteq \mathbb{R}^d$.

Lemma 2.1. *Let $f : \Theta \times \mathcal{Z} \to \mathbb{R}$ be differentiable and λ-strongly concave in z with respect to the norm $\|\cdot\|$, and define $\bar{f}(\theta) = \sup_{z \in \mathcal{Z}} f(\theta, z)$. Let $\mathsf{g}_\theta(\theta, z) = \nabla_\theta f(\theta, z)$ and $\mathsf{g}_\mathsf{z}(\theta, z) = \nabla_z f(\theta, z)$, and assume g_θ and g_z satisfy Assumption 2.2 with $\ell(\theta; z)$ replaced*

Algorithm 2.1 Distributionally robust optimization with adversarial training

INPUT: Sampling distribution P_0, constraint sets Θ and $\mathcal{Z}$, stepsize sequence $\{\alpha_t > 0\}_{t=0}^{T-1}$
for $t = 0, \ldots, T-1$ **do**
  Sample $z^t \sim P_0$ and find an ϵ-approximate maximizer $\widehat{z}^t$ of $\ell(\theta^t; z) - \gamma c(z, z^t)$
  $\theta^{t+1} \leftarrow \mathsf{Proj}_\Theta(\theta^t - \alpha_t \nabla_\theta \ell(\theta^t; \widehat{z}^t))$

with $f(\theta, z)$. Then $\bar{f}$ is differentiable, and letting $z^\star(\theta) = \operatorname{argmax}_{z \in \mathcal{Z}} f(\theta, z)$, we have $\nabla \bar{f}(\theta) = \mathsf{g}_\theta(\theta, z^\star(\theta))$. Moreover,

$$\|z^\star(\theta_1) - z^\star(\theta_2)\| \le \frac{L_{\mathsf{z}\theta}}{\lambda} \|\theta_1 - \theta_2\| \quad \textit{and} \quad \left\|\nabla \bar{f}(\theta) - \nabla \bar{f}(\theta')\right\|_\star \le \left(L_{\theta\theta} + \frac{L_{\theta\mathsf{z}} L_{\mathsf{z}\theta}}{\lambda}\right) \left\|\theta - \theta'\right\|.$$

See Section A.2.2 for the proof. Fix $z_0 \in \mathcal{Z}$ and focus on the ℓ_2-norm case where $c(z, z_0)$ satisfies Assumption 2.1 with $\|\cdot\|_2$. Noting that $f(\theta, z) := \ell(\theta, z) - \gamma c(z, z_0)$ is $(\gamma - L_{\mathsf{zz}})$-strongly concave from the insight (2.4) (with $L := L_{\mathsf{zz}}$), let us apply Lemma 2.1. Under Assumptions 2.1, 2.2, $\phi_\gamma(\cdot; z_0)$ then has $L = L_{\theta\theta} + \frac{L_{\theta\mathsf{z}} L_{\mathsf{z}\theta}}{[\gamma - L_{\mathsf{zz}}]_+}$-Lipschitz gradients, and

$$\nabla_\theta \phi_\gamma(\theta; z_0) = \nabla_\theta \ell(\theta; z^\star(z_0, \theta)) \quad \text{where} \quad z^\star(z_0, \theta) = \operatorname*{argmax}_{z \in \mathcal{Z}} \{\ell(\theta; z) - \gamma c(z, z_0)\}.$$

This motivates Algorithm 2.1, a stochastic-gradient approach for the penalty problem (2.7). The benefits of Lagrangian relaxation become clear here: for $\ell(\theta; z)$ smooth in z and γ large enough, gradient ascent on $\ell(\theta^t; z) - \gamma c(z, z^t)$ in z converges linearly and we can compute (approximate) $\widehat{z}^t$ efficiently (we initialize our inner gradient ascent iterations with the sampled natural example z^t).

Convergence properties of Algorithm 2.1 depend on the loss ℓ. When ℓ is convex in θ and γ is large enough that $z \mapsto (\ell(\theta; z) - \gamma c(z, z_0))$ is concave for all $(\theta, z_0) \in \Theta \times \mathcal{Z}$, we have a stochastic monotone variational inequality, which is efficiently solvable [145, 66] with convergence rate $1/\sqrt{T}$. When the loss ℓ is nonconvex in θ, the following theorem guarantees convergence to a stationary point of problem (2.7) at the same rate when $\gamma \ge L_{\mathsf{zz}}$. Recall that $F(\theta) = \mathbb{E}_{P_0}[\phi_\gamma(\theta; Z)]$ is the robust surrogate objective for the Lagrangian relaxation (2.2).

Theorem 2.1 (Convergence of Nonconvex SGD)**.** *Let Assumptions 2.1 and 2.2 hold with the ℓ_2-norm and let $\Theta = \mathbb{R}^d$. Let $\Delta_F \ge F(\theta^0) - \inf_\theta F(\theta)$. Assume $\mathbb{E}[\|\nabla F(\theta) - \nabla_\theta \phi_\gamma(\theta, Z)\|_2^2] \le \sigma^2$ and take constant stepsizes $\alpha = \sqrt{\frac{\Delta_F}{L_\phi T \sigma^2}}$ where $L_\phi := L_{\theta\theta} + \frac{L_{\theta\mathsf{z}} L_{\mathsf{z}\theta}}{\gamma - L_{\mathsf{zz}}}$. For $T \ge \frac{L_\phi \Delta_F}{\sigma^2}$,*

Algorithm 2.1 satisfies

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\left\|\nabla F(\theta^t)\right\|_2^2\right] - \frac{4L_{\theta z}^2}{\gamma - L_{zz}}\epsilon \leq 4\sigma\sqrt{\frac{L_\phi \Delta_F}{T}}.$$

See Section A.2.3 for the proof. We make a few remarks. First, the condition $\mathbb{E}[\|\nabla F(\theta) - \nabla_\theta \phi_\gamma(\theta, Z)\|_2^2] \leq \sigma^2$ holds (to within a constant factor) whenever $\|\nabla_\theta \ell(\theta, z)\|_2 \leq \sigma$ for all θ, z. Theorem 2.1 shows that the stochastic gradient method achieves the rates of convergence on the penalty problem (2.7) achievable in standard smooth non-convex optimization [110]. The accuracy parameter ϵ has a *fixed* effect on optimization accuracy, independent of T: approximate maximization has limited effects.

Key to the convergence guarantee of Theorem 2.1 is that the loss ℓ is smooth in z: the inner supremum (2.2b) is NP-hard to compute for non-smooth deep networks (see Lemma 2.2 in Section 2.2.1 below for a proof of this for networks with ReLUs). The smoothness of ℓ is essential so that a penalized version $\ell(\theta, z) - \gamma c(z, z_0)$ is concave in z (which can be approximately verified by computing Hessians $\nabla_{zz}^2 \ell(\theta, z)$ for each training datapoint), allowing computation and our coming certificates of optimality. Replacing ReLUs with sigmoids or ELUs [69] allows us to apply Theorem 2.1, making distributionally robust optimization tractable for deep learning.

Our distributionally robust framework (2.2) is general enough to consider adversarial perturbations to an arbitrary subset of coordinates in Z. For example, it is appropriate in certain applications to hedge against adversarial perturbations to a small fixed region of an image [52]. By modifying the cost function $c(z, z')$ to take value ∞ outside this small region, our general formulation covers such variants. In Section 2.2.2, we illustrate this modification for supervised-learning scenarios, where we adversarially perturb feature vectors of datapoints but not their labels.

Finding worst-case perturbations with ReLUs is NP-hard

To emphasize the importance of smoothness in efficiently finding solutions to the inner supremem (2.2b), we show that computing worst-case perturbations $\sup_{u \in \mathcal{U}} \ell(\theta; z+u)$ is NP-hard for a large class of feedforward neural networks with (non-smooth) ReLU activations. This result is essentially due to Katz et al. [152]. In the following, we use polynomial time to mean polynomial growth with respect to m, the dimension of the inputs z.

An optimization problem is *NPO* (NP-Optimization) if (i) the dimensionality of the

solution grows polynomially, (ii) the language $\{u \in \mathcal{U}\}$ can be recognized in polynomial time (i.e. a deterministic algorithm can decide in polynomial time whether $u \in \mathcal{U}$), and (iii) ℓ can be evaluated in polynomial time. We restrict analysis to feedforward neural networks with ReLU activations such that the corresponding worst-case perturbation problem is NPO.[1] Furthermore, we impose separable structure on $\mathcal{U}$, that is, $\mathcal{U} := \{v \leq u \leq w\}$ for some $v < w \in \mathbb{R}^m$. See Section A.2.4 for the proof of the following result.

Lemma 2.2. *Consider feedforward neural networks with ReLUs and let* $\mathcal{U} := \{v \leq u \leq w\}$*, where* $v < w$ *such that the optimization problem* $\max_{u \in \mathcal{U}} \ell(\theta; z + u)$ *is NPO. Then there exists* θ *such that this optimization problem is also NP-hard.*

2.2.2 Supervised learning

In supervised learning settings such as classification, it is often natural to only consider adversarial perturbations to the feature vectors (covariates). In this section, we give an adaptation of the results in Section 2.2 to such scenarios. Let $Z = (X, Y) \in \mathcal{X} \times \mathbb{R}$ where $X \in \mathcal{X}$ is a feature vector[2] and $Y \in \mathbb{R}$ is a label. In classification settings, we have $Y \in \{1, \ldots, K\}$. We consider an adversary that can only perturb the feature vector X [116], which can be easily represented in our robust formulation (2.2) by defining the Wasserstein cost function $c : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+ \cup \{\infty\}$ as follows: for $z = (x, y)$ and $z' = (x', y')$ define the covariate-shift cost function as

$$c(z, z') := c_x(x, x') + \infty \cdot \mathbf{1}\left\{y \neq y'\right\} \tag{2.8}$$

where $c_x : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is the transportation cost for the feature vector X. As before, we assume that c_x is nonnegative, continuous, convex in its first argument and satisfies $c_x(x, x) = 0$.

Under the cost function (2.8), the robust surrogate loss in the penalty problem (2.2) and its empirical counterpart (2.7) become

$$\phi_\gamma(\theta; (x_0, y_0)) = \sup_{x \in \mathcal{X}} \left\{\ell(\theta; (x, y_0)) - \gamma c_x(x, x_0)\right\}.$$

Similarly as in Section 2.2.1, we require the following two assumptions that guarantee

[1] Note that $z, u \in \mathbb{R}^m$, so trivially the dimensionality of the solution grows polynomially.
[2] We assume that $\mathcal{X}$ is a subset of normed vector space.

efficient computability of the robust surrogate ϕ_γ.

Assumption 2.3. *The function $c_x : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is continuous. For each $x_0 \in \mathcal{X}$, $c_x(\cdot, x_0)$ is 1-strongly convex with respect to the norm $\|\cdot\|$.*

Let $\|\cdot\|_*$ be the dual norm to $\|\cdot\|$; we again abuse notation by using the same norm $\|\cdot\|$ on Θ and $\mathcal{X}$, though the specific norm is clear from context.

Assumption 2.4. *The loss $\ell : \Theta \times \mathcal{Z} \to \mathbb{R}$ satisfies the Lipschitzian smoothness conditions*

$$\|\nabla_\theta \ell(\theta; (x, y)) - \nabla_\theta \ell(\theta'; (x, y))\|_* \le L_{\theta\theta} \|\theta - \theta'\|, \quad \|\nabla_x \ell(\theta; (x, y)) - \nabla_x \ell(\theta; (x', y))\|_* \le L_{\mathsf{x}\mathsf{x}} \|x - x'\|,$$
$$\|\nabla_\theta \ell(\theta; (x, y)) - \nabla_\theta \ell(\theta; (x', y))\|_* \le L_{\theta\mathsf{x}} \|x - x'\|, \quad \|\nabla_x \ell(\theta; (x, y)) - \nabla_x \ell(\theta'; (x, y))\|_* \le L_{\mathsf{x}\theta} \|\theta - \theta'\|.$$

Under Assumptions 2.3 and 2.4, an analogue of Lemma 2.1 still holds. The proof of the following result is nearly identical to that of Lemma 2.1; we state the full result for completeness.

Lemma 2.3. *Let $f : \Theta \times \mathcal{X} \to \mathbb{R}$ be differentiable and λ-strongly concave in x with respect to the norm $\|\cdot\|$, and define $\bar{f}(\theta) = \sup_{x \in \mathcal{X}} f(\theta, x)$. Let $\mathsf{g}_\theta(\theta, x) = \nabla_\theta f(\theta, x)$ and $\mathsf{g}_\mathsf{x}(\theta, x) = \nabla_x f(\theta, x)$, and assume g_θ and g_x satisfy the Lipschitz conditions of Assumption 2.2. Then $\bar{f}$ is differentiable, and letting $x^\star(\theta) = \operatorname{argmax}_{x \in \mathcal{X}} f(\theta, x)$, we have $\nabla \bar{f}(\theta) = \mathsf{g}_\theta(\theta, x^\star(\theta))$. Moreover,*

$$\|x^\star(\theta_1) - x^\star(\theta_2)\| \le \frac{L_{\mathsf{x}\theta}}{\lambda} \|\theta_1 - \theta_2\| \quad \text{and} \quad \left\|\nabla \bar{f}(\theta) - \nabla \bar{f}(\theta')\right\|_\star \le \left(L_{\theta\theta} + \frac{L_{\theta\mathsf{x}} L_{\mathsf{x}\theta}}{\lambda}\right) \left\|\theta - \theta'\right\|.$$

From Lemma 2.3, we immediately get an analogue to Theorem 2.1 for the cost function (2.8).

Corollary 2.1 (Convergence of Nonconvex SGD)**.** *Let Assumptions 2.3 and 2.4 hold with the ℓ_2-norm and let $\Theta = \mathbb{R}^d$. Let $\Delta_F \ge F(\theta^0) - \inf_\theta F(\theta)$. Assume $\mathbb{E}[\|\nabla F(\theta) - \nabla_\theta \phi_\gamma(\theta, Z)\|_2^2] \le \sigma^2$, and take constant stepsizes $\alpha = \sqrt{\frac{2\Delta_F}{L\sigma^2 T}}$ where $L = L_{\theta\theta} + \frac{L_{\theta\mathsf{x}} L_{\mathsf{x}\theta}}{\gamma - L_{\mathsf{x}\mathsf{x}}}$. Then Algorithm 2.1 satisfies*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\|\nabla F(\theta^t)\right\|_2^2\right] - \frac{2L_{\theta\mathsf{x}}^2}{\gamma - L_{\mathsf{x}\mathsf{x}}} \epsilon \le \sigma \sqrt{8 \frac{L \Delta_F}{T}}.$$

The proof of Corollary 2.1 is identical to that of Theorem 2.1, but applies Lemma 2.3 instead of Lemma 2.1.

2.3 Certificate of robustness and generalization

From results in the previous section, Algorithm 2.1 provably learns to protect against adversarial perturbations of the form (2.7) on the training dataset. Now we show that such procedures generalize, allowing us to prevent attacks on the test set. Our subsequent results hold uniformly over the space of parameters $\theta \in \Theta$, including θ_{WRM}, the output of the stochastic gradient descent procedure in Section 2.2.1. Our first main result, presented in Section 2.3.1, gives a data-dependent upper bound on the population worst-case objective $\sup_{P:W_c(P,P_0)\le\rho} \mathbb{E}_P[\ell(\theta;Z)]$ for any arbitrary level of robustness ρ; this bound is optimal for $\rho = \widehat{\rho}_n$, the level of robustness achieved for the empirical distribution by solving (2.7). Our bound is efficiently computable and hence *certifies* a level of robustness for the worst-case population objective. Second, we show in Section 2.3.2 that adversarial perturbations on the training set (in a sense) generalize: solving the empirical penalty problem (2.7) guarantees a similar level of robustness as directly solving its population counterpart (2.2).

2.3.1 Certificate of robustness

Our main result in this section is a data-dependent upper bound for the worst-case population objective: $\sup_{P:W_c(P,P_0)\le\rho} \mathbb{E}_P[\ell(\theta;Z)] \le \gamma\rho + \mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta;Z)] + O(1/\sqrt{n})$ for all $\theta \in \Theta$, with high probability. To make this rigorous, fix $\gamma > 0$, and consider the worst-case perturbation, typically called the *transportation map* or Monge map [290],

$$T_\gamma(\theta; z_0) := \operatorname*{argmax}_{z\in\mathcal{Z}} \{\ell(\theta;z) - \gamma c(z, z_0)\}. \tag{2.9}$$

Under our assumptions, T_γ is easily computable when $\gamma \ge L_{\mathtt{zz}}$. Letting δ_z denote the point mass at z, Proposition 2.1 shows the empirical maximizers of the Lagrangian formulation (2.6) are attained by

$$\begin{aligned} P_n^*(\theta) &:= \operatorname*{argmax}_P \left\{\mathbb{E}_P[\ell(\theta;Z)] - \gamma W_c(P, \widehat{P}_n)\right\} = \frac{1}{n}\sum_{i=1}^n \delta_{T_\gamma(\theta,Z_i)} \quad \text{and} \\ \widehat{\rho}_n(\theta) &:= W_c(P_n^*(\theta), \widehat{P}_n) = \mathbb{E}_{\widehat{P}_n}[c(T_\gamma(\theta;Z), Z)]. \end{aligned} \tag{2.10}$$

Our results imply, in particular, that the empirical worst-case loss $\mathbb{E}_{P_n^*}[\ell(\theta;Z)]$ gives a *certificate of robustness* to (population) Wasserstein perturbations up to level $\widehat{\rho}_n$. $\mathbb{E}_{P_n^*(\theta)}[\ell(\theta;Z)]$ is efficiently computable via (2.10), providing a data-dependent guarantee for the worst-case

population loss.

Our bound relies on the usual covering numbers for the model class $\{\ell(\theta;\cdot) : \theta \in \Theta\}$ as the notion of complexity [*e.g.* 288], so, despite the infinite-dimensional problem (2.7), we retain the same uniform convergence guarantees typical of empirical risk minimization. Recall that for a set V, a collection $v_1, \dots, v_N$ is an *ϵ-cover* of V in norm $\|\cdot\|$ if for each $v \in \mathcal{V}$, there exists v_i such that $\|v - v_i\| \le \epsilon$. The *covering number* of V with respect to $\|\cdot\|$ is

$$N(V, \epsilon, \|\cdot\|) := \inf\{N \in \mathbb{N} \mid \text{there is an } \epsilon\text{-cover of } V \text{ with respect to } \|\cdot\|\}.$$

For $\mathcal{F} := \{\ell(\theta,\cdot) : \theta \in \Theta\}$ equipped with the $L^\infty(\mathcal{X})$ norm $\|f\|_{L^\infty(\mathcal{X})} := \sup_{X\in\mathcal{X}} |f(X)|$, we state our results in terms of $\|\cdot\|_{L^\infty(\mathcal{X})}$-covering numbers of $\mathcal{F}$. To ease notation, we let

$$\epsilon_n(t) := \gamma b_1 \sqrt{\frac{M_\ell}{n}} \int_0^1 \sqrt{\log N(\mathcal{F}, M_\ell\epsilon, \|\cdot\|_{L^\infty(\mathcal{X})})}\, d\epsilon + b_2 M_\ell \sqrt{\frac{t}{n}}$$

where b_1, b_2 are numerical constants.

We are now ready to state the main result of this section. We first show from the duality result (2.6) that we can provide an upper bound for the worst-case population performance for any level of robustness ρ. For $\rho = \widehat{\rho}_n(\theta)$ and $\theta = \theta_{\mathrm{WRM}}$, this certificate is (in a sense) tight as we see below.

Theorem 2.2. *Assume $|\ell(\theta; z)| \le M_\ell$ for all $\theta \in \Theta$ and $z \in \mathcal{Z}$. Then, for a fixed $t > 0$ and numerical constants $b_1, b_2 > 0$, with probability at least $1 - e^{-t}$, simultaneously for all $\theta \in \Theta$, $\rho \ge 0$, $\gamma \ge 0$,*

$$\sup_{P: W_c(P, P_0) \le \rho} \mathbb{E}_P[\ell(\theta; Z)] \le \gamma\rho + \mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta; Z)] + \epsilon_n(t). \tag{2.11}$$

In particular, if $\rho = \widehat{\rho}_n(\theta)$ then with probability at least $1 - e^{-t}$, for all $\theta \in \Theta$

$$\begin{aligned}\sup_{P: W_c(P, P_0) \le \widehat{\rho}_n(\theta)} \mathbb{E}_P[\ell(\theta; Z)] &\le \gamma\widehat{\rho}_n(\theta) + \mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta; Z)] + \epsilon_n(t) \\ &= \sup_{P: W_c(P, \widehat{P}_n) \le \widehat{\rho}_n(\theta)} \mathbb{E}_P[\ell(\theta; Z)] + \epsilon_n(t).\end{aligned} \tag{2.12}$$

See Section A.2.5 for its proof. We now give a concrete variant for Lipschitz functions. When Θ is finite-dimensional ($\Theta \subset \mathbb{R}^d$), Theorem 2.2 provides a robustness guarantee scaling linearly with d despite the infinite-dimensional Wasserstein penalty. Assuming there exist

$\theta_0 \in \Theta$, $M_{\theta_0} < \infty$ such that $|\ell(\theta_0; z)| \le M_{\theta_0}$ for all $z \in \mathcal{Z}$, we have the following corollary (see proof in Section A.2.6).

Corollary 2.2. *Let $\ell(\cdot; X)$ be L-Lipschitz with respect to some norm $\|\cdot\|$ for all $X \in \mathcal{X}$. Assume that $\Theta \subset \mathbb{R}^d$ satisfies $\operatorname{diam}(\Theta) = \sup_{\theta,\theta' \in \Theta} \|\theta - \theta'\| < \infty$. Then, the bounds (2.11) and (2.12) hold with*

$$\epsilon_n(t) = b_1 \sqrt{\frac{d(L \operatorname{diam}(\Theta) + M_{\theta_0})}{n}} + b_2 (L \operatorname{diam}(\Theta) + M_{\theta_0}) \sqrt{\frac{t}{n}} \tag{2.13}$$

for some numerical constants $b_1, b_2 > 0$.

A key consequence of the bound (2.11) is that $\gamma\rho + \mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta; Z)]$ *certifies* robustness for the worst-case population objective for any ρ and θ. For a given θ, this certificate is tightest at the achieved level of robustness $\widehat{\rho}_n(\theta)$, as noted in the refined bound (2.12) which follows from the duality result

$$\underbrace{\mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta; Z)]}_{\text{surrogate loss}} + \underbrace{\gamma\widehat{\rho}_n(\theta)}_{\text{robustness}} = \sup_{P: W_c(P, \widehat{P}_n) \le \widehat{\rho}_n(\theta)} \mathbb{E}_P[\ell(\theta; Z)] = \mathbb{E}_{P_n^*(\theta)}[\ell(\theta; Z)]. \tag{2.14}$$

(See Section A.2.5 for a proof of these equalities.) We expect θ_{WRM}, the output of Algorithm 2.1, to be close to the minimizer of the surrogate loss $\mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta; Z)]$ and therefore have the best guarantees. Most importantly, the certificate (2.14) is easy to compute via expression (2.10): as noted in Section 2.2.1, the mappings $T(\theta, Z_i)$ are efficiently computable for large enough γ, and $\widehat{\rho}_n = \mathbb{E}_{\widehat{P}_n}[c(T(\theta, Z), Z)]$.

The bounds (2.11)–(2.13) may be too large—because of their dependence on covering numbers and dimension—for practical use in security-critical applications. With that said, the strong duality result, Proposition 2.1, still applies to any distribution. Given a collection of test examples Z_i^{test}, we may interrogate possible losses under perturbations for the *test* examples by noting that, if $\widehat{P}_{\mathrm{test}}$ denotes the empirical distribution on the test set (say, with putative assigned labels), then

$$\frac{1}{n_{\mathrm{test}}} \sum_{i=1}^{n} \sup_{z: c(z, Z_i^{\mathrm{test}}) \le \rho} \{\ell(\theta; z)\} \le \sup_{P: W_c(P, \widehat{P}_{\mathrm{test}}) \le \rho} \mathbb{E}_P[\ell(\theta; Z)] \le \gamma\rho + \mathbb{E}_{\widehat{P}_{\mathrm{test}}}[\phi_\gamma(\theta; Z)] \tag{2.15}$$

for all $\gamma, \rho \ge 0$. Whenever γ is large enough (so that this is tight for small ρ), we may efficiently compute the Monge-map (2.9) and the test loss (2.15) to guarantee bounds on

the sensitivity of a parameter θ to a particular sample and predicted labeling based on the sample.

2.3.2 Generalization of adversarial examples

We can also show that the level of robustness on the training set generalizes. Our starting point is Lemma 2.1, which shows that $T_\gamma(\cdot; z)$ is smooth under Assumptions 2.1 and 2.2:

$$\|T_\gamma(\theta_1; z) - T_\gamma(\theta_2; z)\| \le \frac{L_{\mathsf{z}\theta}}{[\gamma - L_{\mathsf{zz}}]_+} \|\theta_1 - \theta_2\| \tag{2.16}$$

for all θ_1, θ_2, where L_{zz} was the Lipschitz constant of $\nabla_z \ell(\theta; z)$. We provide explicit bounds on the smoothness constant L_{zz} for neural networks with smooth activation functions in Section 2.4. Leveraging this smoothness, we now show that $\widehat{\rho}_n(\theta) = \mathbb{E}_{\widehat{P}_n}[c(T_\gamma(\theta; Z), Z)]$, the level of robustness achieved for the empirical problem, concentrates uniformly around its population counterpart.

Theorem 2.3. *Let $\mathcal{Z} \subset \{z \in \mathbb{R}^m : \|z\| \le M_{\mathsf{z}}\}$ so that $\|Z\| \le M_{\mathsf{z}}$ almost surely and assume either that (i) $c(\cdot, \cdot)$ is L_{c}-Lipschitz over $\mathcal{Z}$ with respect to the norm $\|\cdot\|$ in each argument, or (ii) that $\ell(\theta, z) \in [0, M_\ell]$ and $z \mapsto \ell(\theta, z)$ is γL_{c}-Lipschitz for all $\theta \in \Theta$.*

If Assumptions 2.1 and 2.2 hold, then with probability at least $1 - e^{-t}$,

$$\sup_{\theta \in \Theta} |\mathbb{E}_{\widehat{P}_n}[c(T_\gamma(\theta; Z), Z)] - \mathbb{E}_{P_0}[c(T_\gamma(\theta; Z), Z)]| \le 4B \sqrt{\frac{1}{n}\left(t + \log N\left(\Theta, \frac{[\gamma - L_{\mathsf{zz}}]_+ t}{4 L_{\mathsf{c}} L_{\mathsf{z}\theta}}, \|\cdot\|\right)\right)}. \tag{2.17}$$

where $B = L_{\mathsf{c}} M_{\mathsf{z}}$ under assumption (i) and $B = M_\ell / \gamma$ under assumption (ii).

See Section A.2.7 for the proof. For $\Theta \subset \mathbb{R}^d$, we have $\log N(\Theta, \epsilon, \|\cdot\|) \le d \log(1 + \frac{\operatorname{diam}(\Theta)}{\epsilon})$ so that the bound (2.18) gives the usual $\sqrt{d/n}$ generalization rate for the distance between adversarial perturbations and natural examples. Another consequence of Theorem 2.3 is that $\widehat{\rho}_n(\theta_{\mathrm{WRM}})$ in the certificate (2.12) is positive as long as the loss ℓ is not completely invariant to data. To see this, note from the optimality conditions for $T_\gamma(\theta; Z)$ that $\mathbb{E}_{P_0}[c(T_\gamma(\theta; Z), Z)] = 0$ iff $\nabla_z \ell(\theta; z) = 0$ almost surely, and hence for large enough n, we have $\widehat{\rho}_n(\theta) > 0$ by the bound (2.18).

An analogous result to Theorem 2.3 holds in the supervised-learning setting with the

modified cost function (2.8). Abusing notation, redefine the transport map with the modified cost function (2.8)

$$T_\gamma(\theta; (x_0, y_0)) := \operatorname*{argmax}_{x \in \mathcal{X}} \{\ell(\theta; (x, y_0)) - \gamma c_x(x, x_0)\}.$$

Corollary 2.3. *Let $\mathcal{Z} \subset \{z \in \mathbb{R}^m : \|z\| \le M_{\mathsf{z}}\}$ so that $\|Z\| \le M_{\mathsf{z}}$ almost surely and assume either that (i) $c_x(\cdot, \cdot)$ is L_{c}-Lipschitz over $\mathcal{X}$ with respect to the norm $\|\cdot\|$ in each argument, or (ii) that $\ell(\theta, z) \in [0, M_\ell]$ and $x \mapsto \ell(\theta, (x, y))$ is γL_{c}-Lipschitz for all $\theta \in \Theta$. If Assumptions 2.3 and 2.4 hold, then with probability at least $1 - e^{-t}$,*

$$\sup_{\theta \in \Theta} |\mathbb{E}_{\widehat{P}_n}[c(T_\gamma(\theta; Z), Z)] - \mathbb{E}_{P_0}[c(T_\gamma(\theta; Z), Z)]| \le 4D\sqrt{\frac{1}{n}\left(t + \log N\left(\Theta, \frac{[\gamma - L_{\mathsf{xx}}]_+ t}{4L_{\mathsf{c}} L_{\mathsf{x}\theta}}, \|\cdot\|\right)\right)}. \tag{2.18}$$

where $B = L_{\mathsf{c}} M_{\mathsf{z}}$ under assumption (i) and $B = M_\ell/\gamma$ under assumption (ii).

The proof of Corollary 2.3 is identical to that of Theorem 2.3, but it applies Lemma 2.3 instead of Lemma 2.1.

2.4 Bounds on smoothness of neural networks

In this section, we give upper bounds on the Lipschitz constant L_{xx} of the loss $x \mapsto \ell(\theta; (x, y))$. We focus on the supervised-learning setting presented in Section 2.2.2 for ease of notation. Since our optimization and generalization guarantees apply only for $\gamma \ge L_{\mathsf{xx}}$, our below examples serve as concrete numbers at which we can provide theoretical guarantees on adversarial training. We first provide bounds on deep neural networks with smooth activation functions, and apply them in the classification setting. Our bounds are based on worst-case matrix norm bounds that can be prohibitively loose for deep architectures but often provide reasonable estimates in moderate scales. Due to the conservative nature of the bound, choosing γ larger than this value—so that theoretical results in previous sections apply—may not yield appreciable adversarial robustness in practice. In Section 2.5, we provide empirical discussion of the bounds, and further provide some negative results even in the MNIST setting in Section A.1.6.

Before we present our examples, we first set some notation. For a parameter θ and a feature vector $x \in \mathcal{X} \subset \mathbb{R}^p$, we denote a deep network with L layers by $x \mapsto F_L(\theta; x)$ which maps inputs x to an output $y = F(\theta; X) \in \mathbb{R}^K$. In the classification setting, K denotes the

number of classes, and $F_{L,k}(\theta; x)$ is the k-th element of $F_L(\theta; x)$; the final loss is computed using the softmax loss

$$\ell(\theta; (x, y)) = -\log p_y(\theta; x) \quad \text{where} \quad p_y(\theta; x) := \frac{\exp(F_{L,y}(\theta; x))}{\sum_{k=1}^{K} \exp(F_{L,k}(\theta; x))} \tag{2.19}$$

where $p_y(\theta; x)$ is the softmax probability for class $y \in \{1, \dots, K\}$.

We let $\theta = (\theta_1, \dots, \theta_L)$, where $\theta_l \in \mathbb{R}^{d_{l,I} \times d_{l-1,O}}$ is the weight matrix at the l-th layer of the deep network, where $d_{0,O} = p$ and $d_{L,O} = K$. We denote a nonlinear operation function after the l-th linear layer $\sigma_l : \mathbb{R}^{d_{l,I}} \to \mathbb{R}^{d_{l,O}}$, so that output after the operation is given by

$$F_l(\theta; x) := \sigma_l(\theta_l \cdot \sigma_{l-1}(\theta_{l-1} \cdots \sigma_1(\theta_1 \cdot x) \cdots)). \tag{2.20}$$

For convolutional neural networks, our notation σ_l includes both pooling operations and activation functions. We also denote the Jacobian of $x \mapsto F_l(\theta; x)$ as $J_x F_l(\theta; x)$.

To proceed with providing an upper bound on the Lipschitz constant of a neural network, we assume a given level of smoothness for each respective layer of the network.

Assumption 2.5. *For all $l = 1, \dots, L$, $\sigma_l : \mathbb{R}^{d_{l,I}} \to \mathbb{R}^{d_{l,O}}$ is L_l^0-Lipschitz w.r.t. the ℓ_2-norm $\|\cdot\|_2$, and its Jacobian $J\sigma_l : \mathbb{R}^{d_{l,I}} \to \mathbb{R}^{d_{l,O} \times d_{l,I}}$ is L_l^1-Lipschitz w.r.t. $(\|\cdot\|_{\text{op}}, \|\cdot\|_2)$, where $\|\cdot\|_{\text{op}}$ is the ℓ_2-operator (spectral) norm. Furthermore, $L_l^0 > 0$ for all l.*

The last part of the assumption that $L_l^0 > 0$ for all layers in the network assures that the network is not degenerate, as $L^0 = 0$ defines a layer whose output is constant and independent of the input. We now provide a few examples of layer-wise Lipschitz constants for smooth operations common in convolutional neural networks.

Example 2.1 (Average pooling)**.** *Let $\sigma : \mathbb{R}^{d_I} \to \mathbb{R}^{d_O}$ be given by $\sigma(x)_j = \frac{1}{|I_j|} \sum_{a \in I_j} x_a$, where $I_j \subseteq \{1, \dots, d_I\}$ with $\min_j |I_j| \geq m_l$ and $\max_j |I_j| \leq m_u$. Further, let N be the maximum number of times an index appears in I_j for $j = 1, \dots, d_O$ so that $N := \max_{1 \leq a \leq d_I} \sum_{j=1}^{d_O} \mathbf{1}\{a \in I_j\} \leq d_O$. Then, σ satisfies Assumption 2.5 with $L^0 = \frac{1}{m_l}\sqrt{N m_u}$ and $L^1 = 0$. To see this, note that*

$$|\sigma(x)_j - \sigma(x')_j| = \left| \frac{1}{|I_j|} \sum_{a \in I_j} (x_a - x'_a) \right| \leq \frac{1}{m_l} \left| \sum_{a \in I_j} (x_a - x'_a) \right| \leq \frac{\sqrt{m_u}}{m_l} \left\| x_{I_j} - x'_{I_j} \right\|_2,$$

where $x_{I_j} = \sum_{a \in I_j} x_a \mathbf{e}_a$. *Then,*

$$\left\|\sigma(x) - \sigma(x')\right\|_2 = \left(\sum_{j=1}^{d_O} (\sigma(x)_j - \sigma(x')_j)^2\right)^{1/2} \leq \frac{\sqrt{m_u}}{m_l} \left(\sum_{j=1}^{d_O} \left\|x_{I_j} - x'_{I_j}\right\|_2^2\right)^{1/2} \leq \frac{\sqrt{N m_u}}{m_l} \left\|x - x'\right\|_2 .$$

Since σ *is linear, we have* $L^1 = 0$.

Example 2.2 (Sigmoid)**.** *Let* $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ *be the elementwise sigmoid activation* $\sigma(x)_j = \frac{1}{1+e^{-x_j}}$. *Then,* σ *satisfies Assumption 2.5 with* $L^0 = 1/4$ *and* $L^1 = 1/10$. *To see this, note that* $J_x\sigma(x)$, *the Jacobian of* $x \mapsto \sigma(x)$ *is the diagonal matrix with* i*-th diagonal entry* $\sigma'(x)_i := \frac{d}{dx_i}\sigma(x)_i = \sigma(x)_i(1 - \sigma(x)_i)$. L_0 *is given by the spectral norm of this matrix, which is* $\max_i \sigma'(x)_i \leq \max_i \sup_x \sigma'(x)_i = 1/4$. *For* L^1, *we first define* $\sigma''(x)_i := \frac{d^2}{dx_i^2}\sigma(x)_i = \sigma(x)_i(1 - \sigma(x)_i)(1 - 2\sigma(x)_i)$ *and note that* $\sup_x \sigma''(x)_i \leq -\frac{(1-\sqrt{3})(2-\sqrt{3})}{(3-\sqrt{3})^3} \leq \frac{1}{10}$. *Then, we have*

$$\begin{aligned}\left\|J_x\sigma(x) - J_x\sigma(x')\right\|_{\text{op}} &= \max_i |\sigma'(x)_i - \sigma'(x')_i| \leq \max_i \sup_{x,x'} |\sigma'(x)_i - \sigma'(x')_i| \\ &\leq \max_i \sup_y |\sigma''(y)_i| \left\|x - x'\right\|_\infty \leq \frac{1}{10} \left\|x - x'\right\|_2 .\end{aligned}$$

Example 2.3 (ELU)**.** *Let* $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ *be the ELU activation with scale parameter* $\alpha = 1$*:*

$$\sigma(x)_j = x_j \mathbf{1}\left\{x_j \geq 0\right\} + (e^{x_j} - 1)\mathbf{1}\left\{x_j < 0\right\} .$$

Then, σ *satisfies Assumption 2.5 with* $L^0 = L^1 = 1$.

The following proposition bounds the smoothness of the map $x \mapsto F_l(\theta; x)$. To ease notation, define

$$\alpha_l(\theta) := \prod_{j=1}^{l} L_j^0 \left\|\theta_j\right\|_{\text{op}} \qquad \text{and} \qquad \beta_l(\theta) := \alpha_l(\theta) \sum_{j=1}^{l} \left\{\frac{L_j^1}{(L_j^0)^2} \alpha_j(\theta)\right\} . \tag{2.21}$$

Assumption 2.5 guarantees $\beta_l(\theta)$ is well-defined, as we consider non-degenerate networks with $L_j^0 > 0$.

Proposition 2.2. *Let Assumption 2.5 hold. For all* $l = 1, \ldots, L$, $F_l(\theta; \cdot) : \mathbb{R}^{d_{0,I}} \to \mathbb{R}^{d_{l,O}}$ *is* $\alpha_l(\theta)$*-Lipschitz w.r.t. the* ℓ_2*-norm* $\left\|\cdot\right\|_2$, *and its Jacobian* $J_x F_l(\theta; \cdot) : \mathbb{R}^{d_{0,I}} \to \mathbb{R}^{d_{l,O} \times d_{0,I}}$ *is* $\beta_l(\theta)$*-Lipschitz w.r.t.* $(\left\|\cdot\right\|_{\text{op}}, \left\|\cdot\right\|_2)$, *where* $\left\|\cdot\right\|_{\text{op}}$ *is the* ℓ_2*-operator (spectral) norm.*

See Section A.2.8 for the proof. We can use these bounds to derive our desired bound on the Lipschitz constant of the final output of the network (2.19). We defer its proof to Section A.2.9.

Corollary 2.4. *Let Assumption 2.5 hold. For the softmax loss defined by expression (2.19), $x \mapsto \nabla_x \ell(\theta; (x, y))$ is $\left(\sqrt{2}\beta_L(\theta) + \alpha_L(\theta)^2\right)$-Lipschitz w.r.t. $(\|\cdot\|_{\text{op}}, \|\cdot\|_2)$.*

2.5 Experiments

Our technique for distributionally robust optimization with adversarial training extends beyond supervised learning. To that end, we present empirical evaluations on supervised and reinforcement learning tasks where we compare performance with empirical risk minimization (ERM) and, where appropriate, models trained with the fast-gradient method (2.3) (FGM) [116], its iterated variant (IFGM) [170], and the projected-gradient method (PGM) [188]. PGM augments stochastic gradient steps for the parameter θ with projected gradient ascent over $x \mapsto \ell(\theta; x, y)$, iterating (for data point x_i, y_i)

$$\Delta x_i^{t+1}(\theta) := \operatorname*{argmax}_{\|\eta\|_p \le \epsilon} \{\nabla_x \ell(\theta; x_i^t, y_i)^T \eta\} \quad \text{and} \quad x_i^{t+1} := \Pi_{\mathcal{B}_{\epsilon,p}(x_i^t)} \left\{x_i^t + \alpha_t \Delta x_i^t(\theta)\right\} \tag{2.22}$$

for $t = 1, \ldots, T_{\text{adv}}$, where Π denotes projection onto $\mathcal{B}_{\epsilon,p}(x_i) := \{x : \|x - x_i\|_p \le \epsilon\}$.

The adversarial training literature (*e.g.* Goodfellow et al. [116]) usually considers $\|\cdot\|_\infty$-norm attacks, which allow imperceptible perturbations to all input features. Since in most scenarios it is reasonable to defend against weaker adversaries that instead perturb influential features more, we consider training against $\|\cdot\|_2$-norm attacks. Namely, we use the squared Euclidean cost for the feature vectors $c_x(x, x') := \|x - x'\|_2^2$ and define the overall cost as the covariate-shift adversary (2.8) for WRM (Algorithm 2.1), and we use $p = 2$ for FGM, IFGM, PGM training in all experiments; we still test against adversarial perturbations with respect to the norms $p = 2, \infty$. We use $T_{\text{adv}} = 15$ iterations for all iterative methods (IFGM, PGM, and WRM) in training and attacks.

In Section 2.5.1, we visualize differences between our approach and ad-hoc methods to illustrate the benefits of certified robustness. In Section 2.5.2 we consider supervised learning problems where we adversarially perturb the test data for the MNIST and Stanford Dogs [155] datasets. Finally, we consider a reinforcement learning problem in Section 2.5.3, where the Markov decision process used for training differs from that for testing.

WRM enjoys the theoretical guarantees of Sections 2.2 and 2.3 for large γ, but for small γ (large adversarial budgets), WRM becomes a heuristic like other methods. In Appendix A.1.4, we compare WRM with other methods on attacks with large adversarial budgets. In Appendix A.1.5, we further compare WRM—which is trained to defend against $\|\cdot\|_2$-adversaries—with other heuristics trained to defend against $\|\cdot\|_\infty$-adversaries. WRM matches or outperforms other heuristics against imperceptible attacks, while it underperforms for attacks with large adversarial budgets.

2.5.1 Visualizing the benefits of certified robustness

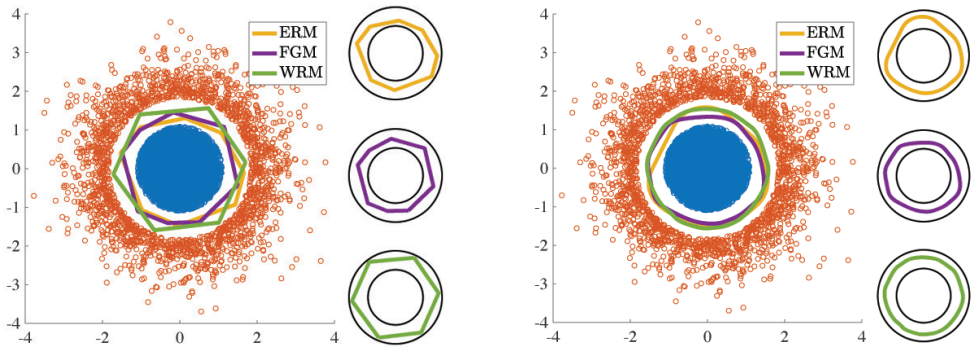
For our first experiment, we generate synthetic data $Z = (X, Y) \sim P_0$ by $X_i \overset{\text{i.i.d.}}{\sim} \mathsf{N}(0_2, I_2)$ with labels $Y_i = \text{sign}(\|x\|_2 - \sqrt{2})$, where $X \in \mathbb{R}^2$ and I_2 is the identity matrix in $\mathbb{R}^2$. Furthermore, to create a wide margin separating the classes, we remove data with $\|X\|_2 \in (\sqrt{2}/1.3, 1.3\sqrt{2})$. We train a small neural network with 2 hidden layers of size 4 and 2 and either all ReLU or all ELU activations between layers, comparing our approach (WRM) with ERM and the 2-norm FGM. For our approach we use $\gamma = 2$, and to make fair comparisons with FGM we use

$$\epsilon^2 = \widehat{\rho}_n(\theta_{\text{WRM}}) = W_c(P_n^*(\theta_{\text{WRM}}), \widehat{P}_n) = \mathbb{E}_{\widehat{P}_n}[c(T(\theta_{\text{WRM}}, Z), Z)], \tag{2.23}$$

for the fast-gradient perturbation magnitude ϵ, where θ_{WRM} is the output of Algorithm 2.1.[3]

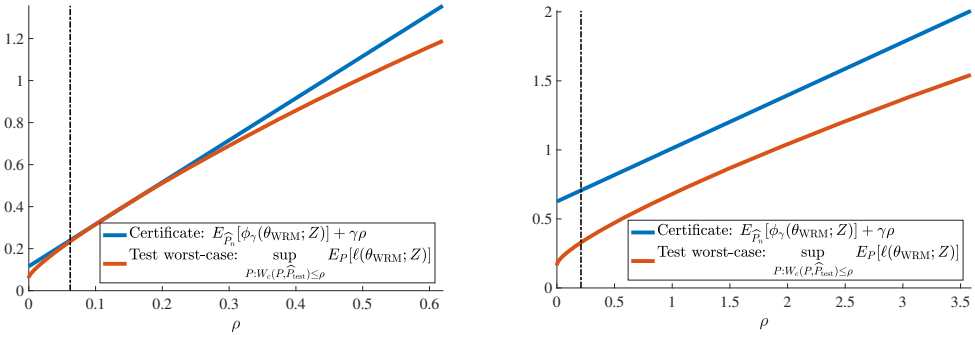
Figure 2.1 illustrates the classification boundaries for the three training procedures over the ReLU-activated (Figure 2.1(a)) and ELU-activated (Figure 2.1(b)) models. Since 70% of the data are of the blue class ($\|X\|_2 \le \sqrt{2}/1.3$), distributional robustness favors pushing the classification boundary outwards; intuitively, adversarial examples will come from pushing blue points outwards across the boundary. ERM and FGM suffer from sensitivities to various regions of the data, as evidenced by the lack of symmetry in their classification boundaries. For both activations, WRM pushes the classification boundaries further outwards than ERM or FGM. However, WRM with ReLUs still suffers from sensitivities (*e.g.* radial asymmetry in the classification surface) due to the lack of robustness guarantees. WRM with ELUs provides a certified level of robustness, yielding an axisymmetric classification boundary that hedges against adversarial perturbations in all directions.

[3]For ELU activations with scale parameter 1, $\gamma = 2$ makes problem (2.2b) strongly concave over the training data. ReLUs have no guarantees, but we use 15 gradient steps with stepsize $1/\sqrt{t}$ for both activations.



(a) ReLU model (b) ELU model

Figure 2.1: Experimental results on synthetic data. Training data are shown in blue and red. Classification boundaries are shown in yellow, purple, and green for ERM, FGM, and WRM respectively. The boundaries are shown with the training data as well as separately with the true class boundaries.



(a) Synthetic data (b) MNIST

Figure 2.2: Empirical comparison between certificate of robustness (2.11) (blue) and test worst-case performance (red) for experiments with (a) synthetic data and (b) MNIST. We omit the certificate's error term $\epsilon_n(t)$. The vertical bar indicates the achieved level of robustness on the training set $\widehat{\rho}_n(\theta_{\text{WRM}})$.

Recall that our *certificates of robustness* on the worst-case performance given in Theorem 2.2 applies for any level of robustness ρ. In Figure 2.2(a), we plot our certificate (2.11) against the out-of-sample (test) worst-case performance $\sup_{P: W_c(P, P_0) \le \rho} \mathbb{E}_P[\ell(\theta; Z)]$ for WRM with ELUs. Since the worst-case loss is hard to evaluate directly, we solve its Lagrangian relaxation (2.6) for different values of γ_{adv}. For each γ_{adv}, we consider the distance to adversarial examples in the test dataset

$$\widehat{\rho}_{\text{test}}(\theta) := \mathbb{E}_{\widehat{P}_{\text{test}}}[c(T_{\gamma_{\text{adv}}}(\theta, Z), Z)], \tag{2.24}$$

where $\widehat{P}_{\text{test}}$ is the test distribution, $c(z, z') := \|x - x'\|_2^2 + \infty \cdot \mathbf{1}\{y \neq y'\}$ as before, and $T_{\gamma_{\text{adv}}}(\theta, Z) = \text{argmax}_z\{\ell(\theta; z) - \gamma_{\text{adv}} c(z, Z)\}$ is the adversarial perturbation of Z (Monge map) for the model θ. The worst-case losses on the test dataset are then given by

$$\mathbb{E}_{\widehat{P}_{\text{test}}}[\phi_{\gamma_{\text{adv}}}(\theta_{\text{WRM}}; Z)] + \gamma_{\text{adv}} \widehat{\rho}_{\text{test}}(\theta_{\text{WRM}}) = \sup_{P: W_c(P, P_{\text{test}}) \leq \widehat{\rho}_{\text{test}}(\theta_{\text{WRM}})} \mathbb{E}_P[\ell(\theta_{\text{WRM}}; Z)].$$

As anticipated, our certificate is almost tight near the achieved level of robustness $\widehat{\rho}_n(\theta_{\text{WRM}})$ for WRM (2.10) and provides a performance guarantee even for other values of ρ.

2.5.2 Learning a more robust classifier

We now consider two real-world supervised-learning benchmarks. In the first, we use our adversarial training method on the MNIST dataset in an unprincipled manner: we use a fixed level γ but without prescribing this level to be such that the robust surrogate (2.2b) is concave. For the more realistic Stanford Dogs dataset [155], we first present results where we ran Algorithm 2.1 on a semantic feature space with γ chosen large enough to satisfy bounds given in Section 2.4. We complement these experiments with results where we ran Algorithm 2.1 in an unprincipled manner on raw pixel perturbations, analogous to our MNIST results.

The MNIST dataset

For the MNIST dataset, we train small neural network classifiers consisting of 8×8, 6×6, and 5×5 convolutional filter layers with ELU activations followed by a fully connected layer and softmax output. We train WRM with $\gamma = 0.04\mathbb{E}_{\widehat{P}_n}[\|X\|_2]$, and for the other methods we choose ϵ as the level of robustness achieved by WRM (2.23).[4] In the figures, we scale the budgets $1/\gamma_{\text{adv}}$ and ϵ_{adv} for the adversary with $C_p := \mathbb{E}_{\widehat{P}_n}[\|X\|_p]$.[5]

In Figure 2.2(b) we illustrate the validity of our certificate of robustness (2.11) for the worst-case test performance for arbitrary level of robustness ρ. We see that our certificate provides a performance guarantee for out-of-sample worst-case performance. In Figure 2.3, we compare our method against different adversarial training techniques; all methods achieve $> 99\%$ test-set accuracy, implying there is little test-time penalty for the robustness levels (ϵ and γ) used for training. It is thus important to distinguish the

[4] For this γ, $\phi_\gamma(\theta_{\text{WRM}}; z)$ is strongly concave for 98% of the training data.

[5] For the standard MNIST dataset, $C_2 := \mathbb{E}_{\widehat{P}_n}\|X\|_2 = 9.21$ and $C_\infty := \mathbb{E}_{\widehat{P}_n}\|X\|_\infty = 1.00$.

methods' abilities to combat attacks. We test performance of the five methods (ERM, FGM, IFGM, PGM, WRM) under PGM attacks (2.22) with respect to 2- and ∞-norms. In Figures 2.3(a) and 2.3(b), all adversarial methods outperform ERM, and WRM offers more robustness even with respect to these PGM attacks. Training with the Euclidean cost still provides robustness to ∞-norm fast gradient attacks. We provide further evidence in Appendix A.1.1.

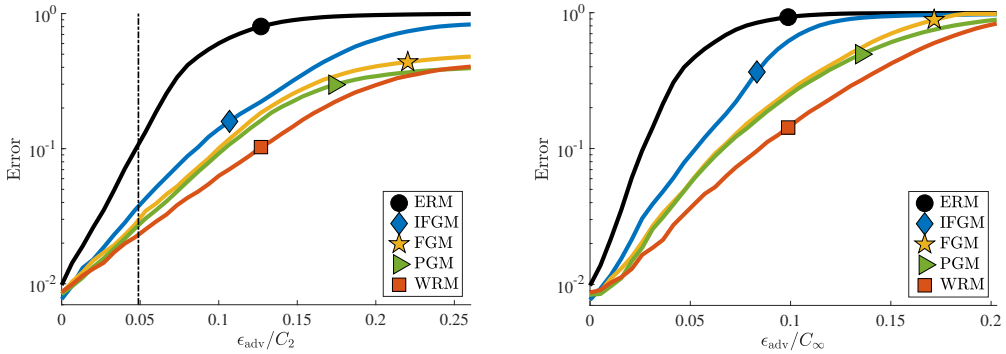
In Figure 2.4(a), we study stability of the loss surface with respect to perturbations to inputs. We note that small values of $\widehat{\rho}_{\text{test}}(\theta)$, the distance to adversarial examples (2.24), correspond to small magnitudes of $\nabla_z \ell(\theta; z)$ in a neighborhood of the nominal input, which ensures stability of the model. Figure 2.4(a) shows that $\widehat{\rho}_{\text{test}}$ differs by orders of magnitude between the training methods (models $\theta = \theta_{\text{ERM}}, \theta_{\text{FGM}}, \theta_{\text{IFGM}}, \theta_{\text{PGM}}, \theta_{\text{WRM}}$); the trend is nearly uniform over all γ_{adv}, with θ_{WRM} being the most stable. Thus, we see that our adversarial-training method defends against gradient-exploiting attacks by reducing the magnitudes of gradients near the nominal input.

Figure 2.4(b) presents qualitative examples that illustrate the utility of our approach. For a single test datapoint, we adversarially perturb the image until the model misclassifies it. We again consider WRM attacks and we decrease γ_{adv} until each model misclassifies the input. The original label is 8, whereas on the adversarial examples IFGM predicts 2, PGM predicts 0, and the other models predict 3. WRM's "misclassifications" appear consistently reasonable to the human eye (see Appendix A.1.2 for examples of other digits); WRM defends against gradient-based exploits by learning a representation that makes gradients point towards inputs of other classes. Together, Figures 2.4(a) and 2.4(b) depict our method's defense mechanisms to gradient-based attacks: creating a stable loss surface by reducing the magnitude of gradients and improving their interpretability.

The Stanford Dogs dataset

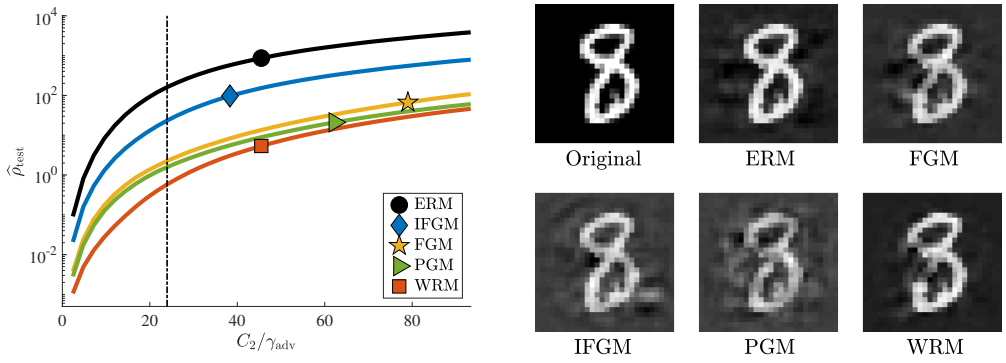
We now test our approach in a more realistic classification scenario, where our goal is to reliably classify images of dogs to one of 120 breeds using the Stanford Dogs dataset [155]. In our experiments, we use 2000 training images resized to 224×224 pixels, and use the ResNet-50 network [124] pre-trained on the ImageNet dataset for initialization.

First, we begin with a principled experiment, where we choose γ based on concrete upper bounds derived in the previous section. Due to the large size of the ResNet model, our concrete upper bounds on the smoothness of this model become vacuous even when



(a) Test error vs. ϵ_{adv} for $\|\cdot\|_2$ attack (b) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$ attack

Figure 2.3: PGM attacks on the MNIST dataset. (a) and (b) show test misclassification error vs. the adversarial perturbation level ϵ_{adv} for the PGM attack with respect to Euclidean and ∞ norms respectively. The vertical bar in (a) indicates the perturbation level used for training the PGM, FGM, and IFGM models as well as the estimated radius $\sqrt{\widehat{\rho}_n(\theta_{\text{WRM}})}$. For MNIST, $C_2 = 9.21$ and $C_\infty = 1.00$.



(a) $\widehat{\rho}_{test}$ vs. $1/\gamma_{\text{adv}}$ (b) Perturbations on a test datapoint

Figure 2.4: Stability of the loss surface. In (a), we show the average distance of the perturbed distribution $\widehat{\rho}_{\text{test}}$ for a given γ_{adv}, an indicator of local stability to inputs for the decision surface. The vertical bar in (a) indicates the γ we use for training WRM. In (b) we visualize the smallest WRM perturbation (largest γ_{adv}) necessary to make a model misclassify a datapoint. More examples are in Appendix A.1.2.

replace ReLU activations with its smooth counterparts. Therefore, we begin this subsection by considering adversarial perturbations in the learned semantic feature space given by the output of the pre-trained ResNet on the large ImageNet dataset. We use the pre-trained network as a fixed feature extractor, and only fine-tune the last few layers following our principled adversarial training procedure Algorithm 2.1, with γ chosen according to our bounds in Section 2.4.

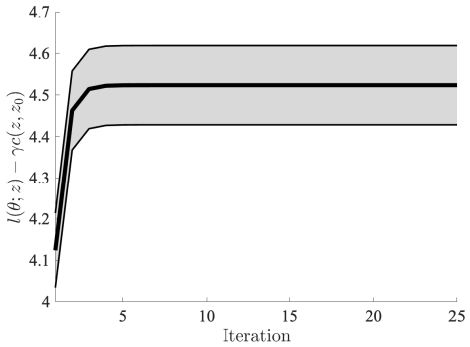
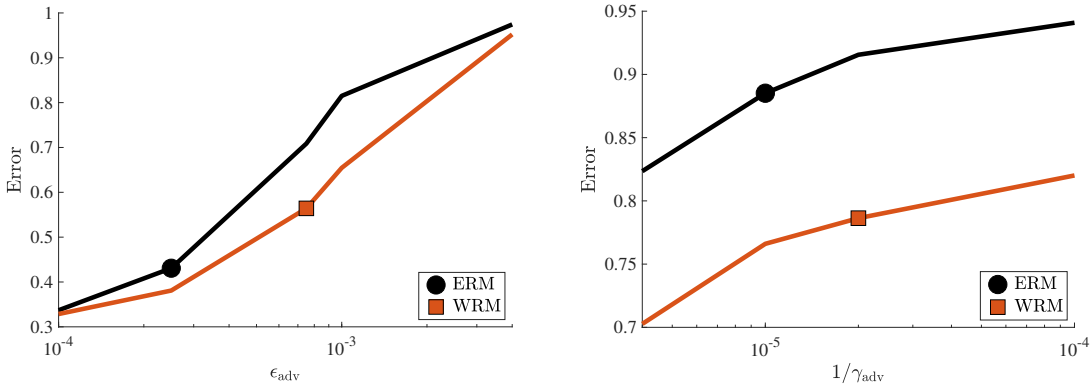


Figure 2.5: Convergence of the inner problem (2.2b). We show the mean and standard error across all datapoints of the loss $l(\theta; z) - \gamma c(z, z_0)$ with respect to iteration. The plot is shown for the the first epoch of training.

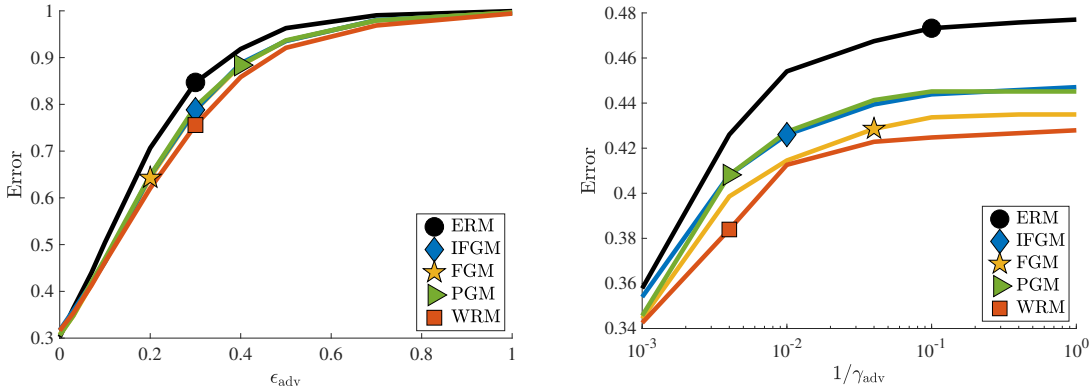
We pass the images through the pre-trained ResNet architecture and extract the $7 \times 7 \times 2048$ semantic features (normalized to $[0, 1]$) before they are fed into the output layer. Instead of passing the features through a global average pooling layer followed by a softmax layer, as usually done when fine-tuning a ResNet on a new dataset, we design a classifier with smooth activation functions (ELUs). Specifically, we use a convolutional layer with 1024 convolutions of kernel size 7×7 followed by an ELU, a fully connected layer with 120 outputs, and finally a softmax layer. For our experiments, we set $\gamma = 10^5$, observing that for this value we satisfy have a smoothness upper bound (via the bound (2.21)) of 822.5. We set the number of iterations $T_{\text{adv}} = 20$ and the step size $\eta = 10.0$. Figure 2.5 shows the convergence of the supremum indicating that we have indeed solved the inner problem (2.2b), which plots averages of the adversarial loss (2.2b) $\phi_\gamma(\theta; z_i)$ during the first epoch of the training procedure. In Figure 2.6, we present classification errors under both PGM and WRM attacks in the semantic feature space as in the last subsection.

We now illustrate the usefulness of our approach even large-scale scenarios where our conservative bounds on smoothness given in Section 2.4 become too loose. We fine-tune the whole ResNet-50 network on the Stanford Dogs dataset with Algorithm 2.1 with perturbations on raw pixels. We use $\gamma = 1.0$, and set the adversarial budget ϵ for the other adversarial training algorithms as the level of robustness achieved, as in the MNIST experiment. Figure 2.7 shows results over PGM (Figure 2.7(a)) and WRM (Figure 2.7(b)) attacks. We see that, as with the MNIST dataset, WRM is competitive against the other baselines over a full range of test perturbations.



(a) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-PGM attack (b) Test error vs. $1/\gamma_{\text{adv}}$ for $\|\cdot\|_2$-WRM attack

Figure 2.6: Semantic-space PGM and WRM attacks on the Stanford Dogs dataset.



(a) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-PGM attack (b) Test error vs. $1/\gamma_{\text{adv}}$ for $\|\cdot\|_2$-WRM attack

Figure 2.7: Pixel-space PGM and WRM attacks on the Stanford Dogs dataset.

2.5.3 Robust Markov decision processes

For our final experiments, we consider distributional robustness in the context of reinforcement learning. In Chapter 3, we will consider a more extensive treatment of reinforcement learning, especially when the uncertainty set $\mathcal{P}$ is large. For now, we consider a small subset of reinforcement learning that is amenable to the principled adversarial training techniques we have developed in this chapter. Namely, we consider distributional robustness in the context of Q-learning [297], a model-free reinforcement learning technique. We consider Markov decision processes (MDPs) $(\mathcal{S}, \mathcal{A}, P_{sa}, r)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, state-action transition probabilities P_{sa}, and rewards $r : \mathcal{S} \to \mathbb{R}$. The goal of a reinforcement-learning agent is to maximize (discounted) cumulative rewards $\sum_t \lambda^t \mathbb{E}[r(s^t)]$ (with discount factor λ);

this is analogous to minimizing $\mathbb{E}_P[\ell(\theta; Z)]$ in supervised learning. Robust MDPs consider an ambiguity set $\mathcal{P}_{sa}$ for state-action transitions. The goal is maximizing the worst-case realization $\inf_{P \in \mathcal{P}_{sa}} \sum_t \lambda^t \mathbb{E}_P[r(s^t)]$, analogous to problem (2.1).

In a standard MDP, Q-learning learns a quality function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ via the iterations

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha_t \left(r(s^t) + \lambda \max_a Q(s^{t+1}, a) - Q(s^t, a^t) \right) \tag{2.25}$$

such that $\operatorname{argmax}_a Q(s, a)$ is (eventually) the optimal action in state s to maximize cumulative reward. When the underlying environment has a continuous state-space and we represent Q with a differentiable function (*e.g.* [200]), we can modify the update (2.25) with an adversarial state perturbation to incorporate distributional robustness. Namely, we draw the nominal state-transition update $\widehat{s}^{t+1} \sim p_{sa}(s^t, a^t)$, and proceed with the update (2.25) using the perturbation

$$s^{t+1} \leftarrow \operatorname*{argmin}_s \left\{ r(s) + \lambda \max_a Q(s, a) + \gamma c(s, \hat{s}^{t+1}) \right\}. \tag{2.26}$$

For large γ, we can again solve problem (2.26) efficiently using gradient descent. This provides robustness to uncertainties in state-action transitions. For tabular Q-learning, where we represent Q only over a discretized covering of the underlying state-space, we can either neglect the second term in the update (2.26) and, after performing the update, round s^{t+1} as usual, or we can perform minimization directly over the discretized covering. In the former case, since the update (2.26) simply modifies the state-action transitions (independent of Q), standard results on convergence for tabular Q-learning (*e.g.* Szepesvári and Littman [277]) apply under these adversarial dynamics.

We test our adversarial training procedure in the cart-pole environment, where the goal is to balance a pole on a cart by moving the cart left or right. The environment caps episode lengths to 400 steps and ends the episode prematurely if the pole falls too far from the vertical or the cart translates too far from its origin. We use reward $r(\beta) := e^{-|\beta|}$ for the angle β of the pole from the vertical. We use a tabular representation for Q with 30 discretized states for β and 15 for its time-derivative $\dot{\beta}$ (we perform the update (2.26) without the Q-dependent term). The action space is binary: push the cart left or right with a fixed force. Due to the nonstationary, policy-dependent radius for the Wasserstein ball, an analogous ϵ for the fast-gradient method (or other variants) is not well-defined. Thus, we only compare with an agent trained on the nominal MDP. We test both models

Environment	Regular	Robust
Original	399.7 ± 0.1	400.0 ± 0.0
Easier environments		
Light	400.0 ± 0.0	400.0 ± 0.0
Long	400.0 ± 0.0	400.0 ± 0.0
Soft g	400.0 ± 0.0	400.0 ± 0.0
Harder environments		
Heavy	150.1 ± 4.7	334.0 ± 3.7
Short	245.2 ± 4.8	400.0 ± 0.0
Strong g	189.8 ± 2.3	398.5 ± 0.3

Table 2.1: Episode length over 1000 trials (mean ± standard error)

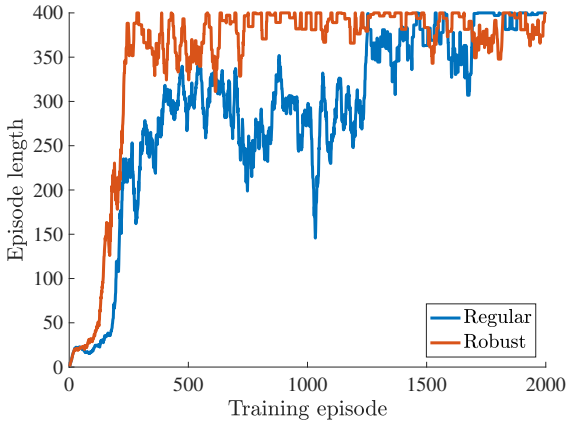


Figure 2.8: Episode lengths during training. The environment caps episodes to 400 steps.

with perturbations to the physical parameters: we shrink/magnify the pole's mass by 2, the pole's length by 2, and the strength of gravity g by 5. The system's dynamics are such that the heavy, short, and strong-gravity cases are more unstable than the original environment, whereas their counterparts are less unstable.

Table 2.1 shows performance of trained models over the original and all perturbed MDPs. Both models perform similarly over easier environments, but the robust model greatly outperforms in harder environments. Interestingly, as shown in Figure 2.8, the robust model also learns more efficiently than the nominal model in the original MDP. We hypothesize that a potential side-effect of robustness is that adversarial perturbations encourage better exploration of the environment.

2.6 Discussion

Explicit distributional robustness of the form (2.5) is intractable except in limited cases. We provide a principled method for efficiently guaranteeing distributional robustness with a simple form of adversarial data perturbation. Using only assumptions about the smoothness of the loss function ℓ, we prove that our method enjoys strong statistical guarantees and fast optimization rates for a large class of problems. The NP-hardness of certifying robustness for ReLU networks, coupled with our empirical success and theoretical certificates for smooth networks in deep learning, suggest that using smooth networks may be preferable if we wish to guarantee robustness. Empirical evaluations indicate that our methods are

in fact robust to perturbations in the data, and they match or outperform less-principled adversarial training techniques. The major benefit of our approach is its simplicity and wide applicability across many models and machine-learning scenarios.

There remain many avenues for future investigation. Our optimization result (Theorem 2.1) applies only for small values of robustness ρ and to a limited class of Wasserstein costs. Our statistical guarantees (Theorems 2.2 and 2.3) use $\|\cdot\|_\infty$-covering numbers as a measure of model complexity, which can become prohibitively large for deep networks. In a learning-theoretic context, where the goal is to provide insight into convergence behavior as well as comfort that a procedure will "work" given enough data, such guarantees are satisfactory, but this may not be enough in security-essential contexts. This problem currently persists for most learning-theoretic guarantees in deep learning, and the recent works of Bartlett et al. [23], Dziugaite and Roy [92], and Neyshabur et al. [212] attempt to mitigate this shortcoming. Replacing our covering number arguments with more intricate notions such as margin-based bounds [23] would extend the scope and usefulness of our theoretical guarantees. Of course, the certificate (2.15) still holds regardless.

More broadly, this work focuses on small-perturbation attacks, and our theoretical guarantees show that it is possible to efficiently build models that provably guard against such attacks. Our method becomes another heuristic for protection against attacks with large adversarial budgets. Indeed, in the large-perturbation regime, efficiently training certifiably secure systems remains an important open question. We believe that conventional $\|\cdot\|_\infty$-defense heuristics developed for image classification do not offer much comfort in the large-perturbation/perceptible-attack setting: $\|\cdot\|_\infty$-attacks with a large budget can render images indiscernible to human eyes, while, for example, $\|\cdot\|_1$-attacks allow a concerted perturbation to critical regions of the image. Certainly $\|\cdot\|_\infty$-attack and defense models have been fruitful in building a foundation for security research in deep learning, but moving beyond them may be necessary for more advances in the large-perturbation regime.

In Chapter 3, we explore a particularly challenging rendition of the large-uncertainty regime where the uncertainty set $\mathcal{P}$ may not even be known and data that could be used to learn it is unavailable. We exploit synthetic data to learn a parametrization of $\mathcal{P}$ that leads to tractable procedures for building distributionally robust models.

Chapter 3

Balancing safety and performance in high-uncertainty regimes

Show me how you drive and I'll show you who you are.

— Dominic Toretto, *Fast & Furious 6*

In the last chapter, we studied how to solve distributionally robust optimization problems for sufficiently small uncertainty sets $\mathcal{P}$. This chapter expands the horizon to consider intractably large and potentially unbounded uncertainty sets $\mathcal{P}$. In such regimes, overestimation of uncertainty and leads to safe yet potentially useless models with low performance, whereas underestimation of safety can be dangerous in safety-critical settings. Our overall approach will be to first learn an operational proxy for $\mathcal{P}$ that allows us to proceed with a tractable distributionally robust optimization problem. We require that the learning process be done safely. Due to the high-uncertainty, this requirement necessitates a learning process that does not collect data or otherwise interact with the environment in a potentially dangerous manner. That is, we use synthetic data to learn $\mathcal{P}$. Although our approach is general, we frame our development within the specific application of autonomous driving—more specifically, the extreme limit of autonomous driving that is autonomous racing.

Balancing performance and safety is crucial to deploying autonomous vehicles in multi-agent environments. In particular, autonomous racing is a domain that penalizes safe but conservative policies, highlighting the need for robust, adaptive strategies. Current approaches either make simplifying assumptions about other agents or lack robust mechanisms

for online adaptation. This work makes algorithmic contributions to both challenges. First, to generate a realistic, diverse set of opponents, we develop a novel method for self-play based on replica-exchange Markov chain Monte Carlo. Second, we propose a distributionally robust bandit optimization procedure that adaptively adjusts risk aversion relative to uncertainty in beliefs about opponents' behaviors. We rigorously quantify the tradeoffs in performance and robustness when approximating these computations in real-time motion-planning, and we demonstrate our methods experimentally on autonomous vehicles that achieve scaled speeds comparable to Formula One racecars.

3.1 Introduction

Current autonomous vehicle (AV) technology still struggles in competitive multi-agent scenarios, such as merging onto a highway, where both maximizing performance (negotiating the merge without delay or hesitation) and maintaining safety (avoiding a crash) are important. The strategic implications of this tradeoff are magnified in racing. During the 2019 Formula One season, the race-winner achieved the fastest lap in only 33% of events [95]. Empirically, the weak correlation between achieving the fastest lap-time and winning suggests that consistent and robust performance is critical to success. In this work, we investigate this intuition in the setting of autonomous racing (AR). In AR, an AV must lap a racetrack in the presence of other agents deploying unknown policies. The agent wins if it completes the race faster than its opponents; a crash automatically results in a loss.

AR is a competitive multi-agent game, a general setting challenging for a number of reasons, especially in robotics applications. First, failures are expensive and dangerous, so learning-based approaches must avoid such behavior or rely on simulation while training. Second, the agents only partially observe their opponent's state, and these observations do not uniquely determine the opponent's behavior. Finally, the agents must make decisions online; the opponent's strategy is a tightly-held secret and cannot be obtained by collecting data before the competition.

Problem: We frame the AR challenge in the context of robust reinforcement learning. We analyze the system as a partially-observed Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, P_{sa}, \mathcal{O}, r, \lambda)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, state-action transition probabilities P_{sa}, observation space $\mathcal{O}$, rewards $r : \mathcal{O} \to \mathbb{R}$, and discount factor λ. Furthermore,

we capture uncertainty in behaviors of other agents through an *ambiguity*[1] set $\mathcal{P}$ for the state-action transitions. Then the AV's objective is

$$\text{maximize} \inf_{P_{sa} \in \mathcal{P}} \sum_t \lambda^t \mathbb{E}[r(o(t))]. \tag{3.1}$$

The obvious price of robustness [33] is that a larger ambiguity set ensures a greater degree of safety while sacrificing performance against a particular opponent. If we knew the opponent's behavior, we would need no ambiguity set; equivalently, the ambiguity set would shrink to the nominal state-action transition distribution. Our goal is to automatically trade between performance and robustness as we play against opponents, which breaks down into two challenges: parametrizing the ambiguity set to allow tractable inference and computing the robust cost efficiently online.

Contributions: This chapter has three contributions: (i) a novel population-based self-play method to parametrize opponent behaviors, (ii) a provably efficient approach to estimate the ambiguity set and the robust cost online, and (iii) a demonstration of these methods on real autonomous vehicles. The name of our approach—FormulaZero—alludes both to the Formula One racing league and the fact that we use self-play (and no demonstrations) to learn competitive behaviors, similar to the approach of AlphaZero [263].

Section 3.1.1 gives context to our learning problem, including connections to classical control techniques. In Section 3.2, we describe the first challenge: learning how to parametrize the ambiguity set $\mathcal{P}$. Rather than directly consider the continuous action space of throttle and steering outputs, we synthesize a library of "prototype" opponent behaviors offline using population-based self-play. When racing against a particular opponent, the agent maintains a belief vector $w(t)$ of the opponent's behavior patterns as a categorical distribution over these prototype behaviors. We then parametrize the ambiguity set as a ball around this nominal belief $w(t)$.

The second challenge, presented in Section 3.3, is an online optimization problem, wherein the agent iteratively updates the ambiguity set (*e.g.* updates $w(t)$) and computes the robust cost of this set. In other words, the agent attempts to learn the opponent's behavior online to maximize its competitive performance. Since this optimization occurs on a moving vehicle with limited computational resources, we provide convergence results that

[1] Ambiguity is a synonym for uncertainty [111]. Formal descriptions in this work use the term ambiguity.

highlight tradeoffs of performance and robustness with respect to these budgets. Finally, Section 3.4 details the practical implications of the theoretical results, emergent properties of the method, and the experimental performance of our approach.

3.1.1 Related work

Reinforcement learning (RL) has achieved unprecedented success on classic two-player games [*e.g.* 263], leading to new approaches in partially-observable games with continuous action spaces [12, 32]. In these works, agents train via self-play using Monte Carlo tree search [53, 274] or population-based methods [143, 144]. The agents optimize expected performance rather than adapt to individual variations in opponent strategy, which can lead to poor performance against particular opponents [20]. In contrast, our method explicitly incorporates adaptivity to opponents.

Robust approaches to RL and control (like this work) explicitly model uncertainty. In RL, this amounts to planning in a robust MDP [214] or a POMDP [147]. Early results Bagnell et al. [17] and Nilim and El Ghaoui [214] describe solutions for robust planning in (PO)MDPs with tabular state/action spaces. Equivalent results in control are analytical formulations applicable to uncertainty in linear time-invariant systems [83, 291, 311]. Recent works [278, 230, 190, 113] describe minimax and adversarial RL frameworks for nonlinear systems and continuous action spaces. Like our approach, these methods fall broadly under the framework of robust optimization. Unlike these works, which consider worst-case planning under a fixed uncertainty distribution, our approach updates the distribution online.

Our approach is designed to adjust the agent's evaluation of short-term plans relative to uncertainty in the opponent's behavior rather than provide worst-case guarantees. Complementary to and compatible with our approach are techniques which provide the latter guarantees, such as robust model predictive control [25]. Extensions of robust control for nonlinear systems and complex uncertainty models are also compatible (*e.g.* Majumdar and Tedrake [189], Althoff and Dolan [7], Gao et al. [106]). In contrast to formal approaches which explicitly guarantee robustness, some authors have proposed multitask or meta-learning approaches (*e.g.* Caruana [58], He et al. [123], Finn et al. [97]) can implicitly learn to play against multiple opponents. However, such techniques do not explicitly model uncertainty or quantify robustness, which we deem necessary in the high-risk, safety-critical regime.

Planning in belief space is closely related to our approach and is well-studied in robotics [see *e.g.* 161]. Specifically in the AV domain, Galceran et al. [102] and Ding and Shen [79] use a Bayesian approach to plan trajectories for AVs in belief space; like this work, both of these approaches characterize the other agent's behavior in the environment categorically. Also similar to this work, Van Den Berg et al. [287] use a sampled set of goals obtained by planning from other agents' perspectives. The main difference in this work from standard belief-space planning formulations is inspired by recent results from distributionally robust optimization (DRO) in supervised-learning settings [28, 208]. These methods reweight training data to reduce the variance of the training loss [208]. While others apply DRO to episodic RL for training *offline* [265, 268], we reweight the belief *online*.

Online methods for control fall under the umbrella of adaptive control [169, 15]. Dean et al. [74] and Agarwal et al. [4] establish regret bounds for adaptive control methods applied to LTI systems, tightening the relationship to online learning. Due to the more general nature of our problem, we draw from the adversarial multi-armed bandit framework of online learning [3, 55, 261].

Our belief state corresponds to a categorical distribution of polices governing an opponent's next action; the goal is to predict which strategy the opponent is using and compute the best response. This approach is similar to game-theoretic methods for AR and AV decision making that use the standard heuristic of iterated best response. Our work is distinct from previous work, which either assumes that all agents act with respect to the same cost function, simplifying the structure of the game [179, 296]; or, without this simplifying assumption, that uses demonstrations to learn possible sets of policies [252, 300]. In constrast, we learn the set of policies without demonstrations and use DRO to robustly score the AV's plans.

We convert the problem of predicting opponent behavior in a continuous action space into an adversarial bandit problem by learning a set of cost functions that characterize a discrete set of policies. As a result, we would like the opponent models to be both near-optimal and diverse. We use determinantal point processes (DPPs) [168] to sample diverse configurations of the parameter space. However, first we must learn a DPP kernel, which requires that we efficiently sample *competitive* cost functions from the larger configuration space. Since we assume no structure to the set of competitive cost functions, we employ a Markov chain Monte Carlo (MCMC) method. Complementary methods include variational-inference (*e.g.* Arenz et al. [10]) and evolutionary (*e.g.* Mouret and Clune [203]) approaches,

which can be challenging to scale up to unstructured, high-dimensional settings of which we have little prior domain knowledge. In our approach, we update the classic simulated tempering method [192] with a novel annealing scheme [159, 59] designed for population diversity. We describe this approach next.

3.2 Offline population synthesis

The goal of offline population synthesis is to generate a diverse set of competitive agent behaviors. Formally, we would like to sample pairs $(x, \theta) \in \mathcal{X} \times \Theta$ that are both diverse as well as achieve small values for a function $f(x, \theta)$. In our AV application, θ parametrizes a neural network used to sample trajectories to follow, x is a weighting of various cost functions that the vehicle uses to select trajectories from the samples, and f is the simulated lap time. With this motivation, we treat the method in more generality assuming (as in our application) that while we can differentiate $f(x, \theta)$ with respect to θ, x represents hyperparameters and admits only function evaluations $f(x, \theta)$ rather than first-order developments. The key challenge is that we do not *a priori* know a metric with which to evaluate diversity (*e.g.*, a kernel for a DPP) nor do we know a base value of f that is deemed acceptable for competitive performance.

We make this problem more tractable via temperature-based Markov chain Monte Carlo (MCMC) and annealing methods [194, 121, 159, 59, 140, 136]. Our goal is to sample from a Boltzmann distribution $g(x, \theta; \beta(t)) \propto e^{-\beta(t) f(x,\theta)}$, where $\beta(t)$ is an inverse "temperature" parameter that grows (or "anneals") with iterations t. When $\beta(t) = 0$, all configurations (x, θ) are equally likely and all MCMC proposals are accepted; as $\beta(t)$ increases, accepted proposals favor smaller f. Unlike standard hyperparameter optimization methods [31, 143] that aim to find a single near-optimal configuration, our goal is to sample a diverse population of (x, θ) achieving small $f(x, \theta)$. As such, our approach—annealed adaptive population tempering (AAdaPT)—maintains a population of configurations and employs high-exploration proposals based on the classic hit-and-run algorithm [269, 24, 182].

3.2.1 AAdaPT

AAdaPT builds upon replica-exchange MCMC, also called parallel tempering, which is a standard approach to maintaining a population of configurations [275, 109]. In parallel tempering, one maintains replicas of the system at L different temperatures $\beta_1 \geq \beta_2 ... \geq \beta_L$

Algorithm 3.1 AAdaPT

input: annealing parameter α, vertical steps V, horizontal exchange steps E, temperature levels L, population size d, initial samples $\{x^{i,j}, \theta^{i,j}\}_{i\in\{1,L\}}^{j\in\{1,D\}}$, iterations T
Evaluate $f(x^{i,j}, \theta^{i,j})$
for $t = 1$ **to** T
  for $j = 1$ **to** L **do** anneal $\beta_{L-j+1}(t)$ (problem (3.2))
  for $k = 1$ **to** V asynchronously, in parallel
    for each population i asynchronously, in parallel
      Sample $\hat{x}^{i,j}$ according to hit-and-run proposal
      Evaluate $f(\hat{x}^{i,j}, \theta^{i,j})$
      Apply MH criteria to update $x^{i,j}$
      Train $\theta^{i,j}$ via SGD
  for $e = 1$ **to** E **do** horizontal swaps (Appendix B.1)

(which are predetermined and fixed), defining the density of the total configuration as $\prod_{i=1}^{L} g(x^i, \theta^i; \beta_i)$. The configurations at each level perform standard MCMC steps (also called "vertical" steps) as well as "horizontal" steps wherein particles are swapped between adjacent temperature levels (see Figure 3.1). Horizontal proposals consist of swapping two configurations in adjacent temperature levels uniformly at random; the proposal is accepted using standard Metropolis-Hastings (MH) criteria [121]. The primary benefit of maintaining parallel configurations is that the configurations at "colder" levels (higher β) can exploit high-exploration moves from "hotter" levels (lower β) which "tunnel" down during horizontal steps [109]. This approach allows for faster mixing times, particularly when parallel MCMC proposals occur concurrently in a distributed computing environment.

Maintaining a population: In AAdaPT (Algorithm 3.1), we maintain a *population* of D configurations at each separate temperature level. Note that this design always maintains D individuals at the highest performance level (highest β). The overall configuration density is $\prod_{i=1}^{L} \prod_{j=1}^{D} g(x^{i,j}, \theta^{i,j}; \beta_i(t))$. Similar to parallel tempering, horizontal proposals are chosen uniformly at random from configurations at adjacent temperatures (see Appendix B.1). We get the same computational benefits of fast mixing in distributed computing environments and a greater ability to exploit high-temperature "tunneling" due to the greater number of possible horizontal exchanges between adjacent temperature levels. The benefit of the horizontal steps is even more pronounced in the RL setting as only vertical steps require new evaluations of f (*e.g.* simulations).

High-exploration vertical proposals: Another benefit of maintaining parallel populations is to improve exploration. We further improve exploration by using hit-and-run proposals [269, 24, 182] for the vertical MCMC chains. Namely, from a current point (x, θ) we sample a uniformly random direction $\hat{u}$ and then choose a point uniformly on the segment $\mathcal{X} \cap (\{x + \mathbb{R} \cdot \hat{u}\} \times \{\theta\})$. This approach has several guarantees for efficient mixing [182, 183, 184]. Note that in our implementation the MCMC steps are only performed on x, while θ updates occur via SGD (see below).

Adaptively annealed temperatures: A downside to parallel tempering is the need to determine the temperature levels β_i beforehand. In AAdaPT. we adaptively update temperatures. Specifically, we anneal the prescribed horizontal acceptance probability of particle exchanges between temperature levels as $\alpha^{t/(L-1)}$ for a fixed hyperparameter $\alpha \in (0, 1)$. Define the empirical acceptance probability of swaps of configurations between levels $i-1$ and i as

$$p_{i-1,i} := \frac{1}{D^2} \sum_{j=1}^{D} \sum_{k=1}^{D} (y_{i-1,i}^{j,k})^{\beta_{i-1}-\beta_i}$$
$$y_{i-1,i}^{j,k} := \min\left(1, e^{f(x^{i-1,j},\theta^{i-1,j}) - f(x^{i,k},\theta^{i,k})}\right).$$

Then, at the beginning of each iteration (in which we perform a series of vertical and horizontal MCMC steps), we update the $\beta_i(t)$ sequentially; we fix $\beta_L(t) := \beta_L = 0$ and for a given β_i, we set β_{i-1} by solving the following convex optimization problem:

$$\underset{\{\beta_{i-1} \geq \beta_i, \quad p_{i-1,i} \leq \alpha^{\frac{t}{(L-1)}}\}}{\text{minimize}} \quad \beta_{i-1}, \tag{3.2}$$

which binary search efficiently solves. This adaptive scheme is crucial in our problem setting, where we *a priori* have no knowledge of appropriate scales for f and, as a result, β. In practice, we find that forcing β_i to monotonically increase in t yields better mixing, so we set $\beta_i(t) = \max(\beta_i(t-1), \hat{\beta}_i(t))$, where $\hat{\beta}_i(t)$ solves problem (3.2).

Evaluating proposals via self-play: We apply AAdaPT to a multi-agent game. It is only possible to evaluate $f(x, \theta)$ in the context of other agents. Since we are interested in the setting where demonstrations from potential opponents are either difficult to obtain or held secret, we iteratively evaluate f via self-play. For each configuration (x, θ), we place

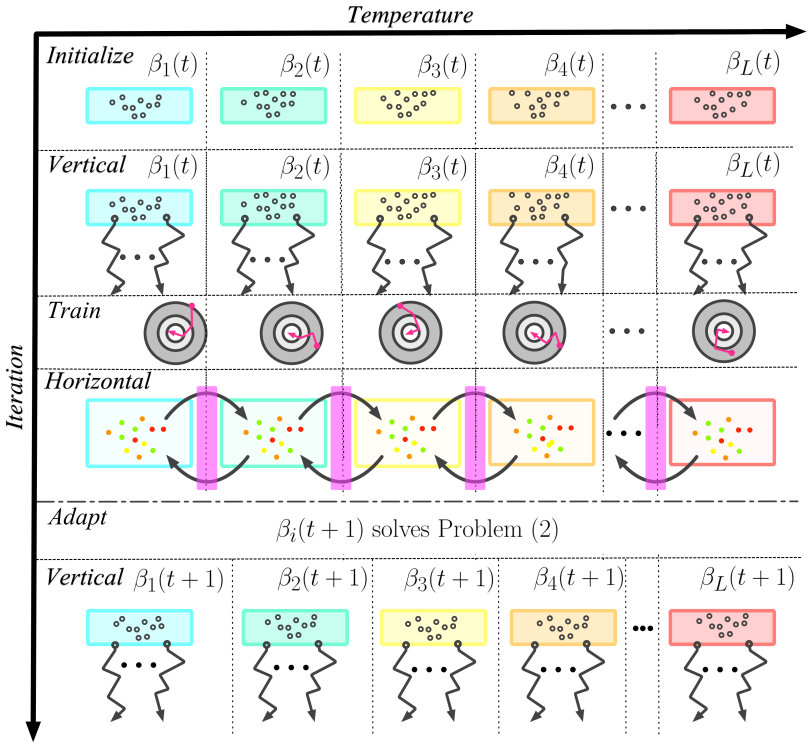


Figure 3.1: Illustration of AAdaPT. Vertical MCMC steps (jagged black arrows) occur in parallel for all $x^{i,j}$, followed by gradient descent steps for trainable parameters $\theta^{i,j}$ (magenta arrows) and horizontal MCMC swaps of configurations between populations (curved black arrows). Then, temperatures $\beta_i(t)$ are updated via problem (3.2).

two vehicles with the same policy in the simulated environment and perform a race (with $f(x, \theta)$ being the lap time of the agent that starts behind the other). Vertical MCMC steps propose new x, which are then accepted according to MH criteria. After a number of vertical iterations, a stochastic gradient descent (SGD) step is applied to θ (which maximizes the likelihood of the trajectories chosen by the agent with cost functions parametrized by x). Following this process, the updated agents in adjacent temperature levels are exchanged via horizontal MCMC steps. Although we choose $f(x, \theta)$ as the laptime, explicit entropic terms can also be included to further encourage diversity within a single vertical chain or across the population.

At the conclusion of AAdaPT, we use the coldest population of D agents at inverse

temperature $\beta_1(T)$ to build a DPP sampler. Specifically, define the matrix H via configurations $x^{1,\cdot}$ at the lowest temperature

$$H_{ab} = \|x^{1,a} - x^{1,b}\|. \tag{3.3}$$

Then we define the DPP kernel K as $K_{ab} = \exp\left(-H_{ab}^2/\sigma^2\right)$ with a scale parameter $\sigma = 0.5$, and we sample $d \leq D$ configurations from this DPP.

3.3 Online learning with computation budgets

Now we exploit the population of d learned prototype behaviors to enable robust performance. The agent's (our) goal is to act robustly against uncertainty in opponent behaviors and adapt online to a given opponent. We parametrize the agent's (stochastic) policy as follows. At each time step, we sample goal states (consisting of pose and velocity) via a generative model $G(\theta)$ parametrized by θ (as in Section 3.2). For a given goal state, we compute the parameters of a cubic spline that reaches the goal by solving a nonconvex trajectory optimization problem [196]; on this proposed trajectory we evaluate a collection of cost functions (such as the maximum curvature or minimum acceleration along the path) weighted by the vector x (recall Section 3.2), similar to Sadat et al. [251] (see Appendix B.4 for a description of all costs). Finally, we choose the sampled goal trajectory with minimum robust cost and perform an action to track this trajectory.

Some of the costs that evaluate the utility of a goal state involve beliefs about the opponent's future trajectory. For a goal p, we rewrite the performance objective at time t with respect to a protoype opponent i as a receding-horizon cost

$$c_i(t; p) := -\sum_{s>t} \lambda^{s-t} \mathbb{E}[r(o(s); p)],$$

where we omit dependence on the agent's cost weights x for convenience. We parametrize the agent's belief of the opponent's behavior as a categorical distribution of beliefs over the prototypes. Specifically, let $w(t) \in \Delta$ be a weight vector at a given time t, where $\Delta := \{a \in \mathbb{R}_+^d \mid a^T \mathbf{1} = 1\}$, and let $P_0(t) := \text{Categorical}(w(t))$. Then $P_0(t)$ is the nominal distribution describing the agent's belief about the opponent. Furthermore, we consider ambiguity sets $\mathcal{P}(t)$ defined by divergence measures on the space of probability measures over Δ. For a convex function ϕ with $\phi(1) = 0$, the ϕ-divergence between distributions

P and Q is $D_\phi(P\|Q) = \int \phi(\frac{dP}{dQ})dQ$.[2] We use sets $\mathcal{P}(t) := \{Q : D_\phi(Q\|P_0)(t) \leq \rho\}$ where $\rho > 0$ is a specified constant. Our implementation employs the χ^2-divergence $\phi(t) = t^2 - 1$.

Having defined the ambiguity set $\mathcal{P}(t)$ and the cost with respect to each prototype opponent, we rewrite the robust performance objective (3.1) to clearly illustrate the optimization problem. Let $C(t;p)$ be a random variable representing the expected cost with respect to the belief of the opponent (and goal state p). Then the robust cost at time t is

$$\sup_{Q\in\mathcal{P}(t)} \mathbb{E}_Q[C(t;p)] = \sup_{q:\sum_i w_i\phi(\frac{q_i}{w_i})\leq\rho} \sum_i q_i c_i(t;p). \tag{3.4}$$

When $\rho = 0$, this is the expected cost under P_0; larger ρ adds robustness. Solving the convex optimization problem (3.4) first requires computing the costs $c_i(t)$. Using $\lambda \geq 0$ for the constraint $D_\phi(Q\|P_0) \leq \rho$, a partial Lagrangian is

$$\mathcal{L}(q,\lambda) = \sum_i q_i c_i(t) - \lambda\left(\sum_i w_i\phi(q_i/w_i) - \rho\right).$$

The corresponding dual function is $v(\lambda) = \sup_{q\in\Delta}\mathcal{L}(q,\lambda)$, and minimizing $v(\lambda)$ via bisection yields the solution to problem (3.4). Maximizing $\mathcal{L}(q,\lambda)$ with respect to q for a given λ requires $O(d)$ time using a variant of median-based search [87] (see Appendix B.2). Thus, computing an ϵ-suboptimal solution uses $O(d\log(1/\epsilon))$ time.

The supremum in the robust cost (3.4) is over belief ambiguity. Thus, our approach generalizes beyond the goal-sampling and trajectory-optimization approach presented at the beginning of this section; it is compatible with any policy that minimizes a cost $c_i(t)$ with respect to a parametrization for opponent i's policy. In this way, it is straightforward to combine our framework with robust model predictive control formulations that have rigorous stability guarantees.

In order to perform competitive actions, the agent updates the ambiguity set $\mathcal{P}(t)$ and computes the robust cost (3.4) on an embedded processor on board the vehicle in real-time (*e.g.* within 100 milliseconds). In the next two subsections, we describe how to perform both operations in the presence of a severely limited computational budget, and we quantitatively analyze the implications of the budget on the robustness/performance tradeoff.

[2]These types of divergences are also commonly written as f-divergences. We use ϕ for notational clarity.

3.3.1 Approximating the robust cost

For a large library of prototypical opponents (large d), computing every c_i in the objective (3.4) is prohibitively expensive. Instead, we consider an empirical approximation of the objective, where we draw N_w indices $J_k \overset{\text{i.i.d.}}{\sim} P_0(t)$ (where $N_w < d$) and consider the weighted sum of these costs c_{j_k}. Specifically, we define the empirical approximation $\mathcal{P}_{N_w} := \{q : D_\phi(q \| \mathbf{1}/N_w) \le \rho\}$ to $\mathcal{P}$ and solve the following empirical version of problem (3.4):

$$\underset{q \in \mathcal{P}_{N_w}}{\text{maximize}} \quad \sum_k q_k c_{j_k}(t; p). \tag{3.5}$$

This optimization problem (3.5) makes manifest the price of robustness in two ways. The first involves the setup of the problem—computing the c_{j_k}. First, we denote the empirical distribution as $\hat{w}(t)$ with $\hat{w}_i(t) = \sum_k^{N_w} \mathbf{1}\{j_k = i\}/N_w$. Even for relatively small N_w/d, $\hat{w}(t)$ concentrates closely around $w(t)$ (see *e.g.* Weissman et al. [299] for a high-probability bound). Thus, when the vehicle's belief about its opponent $w(t)$ is nearly uniform, the j_k values have few repeats. Conversely, when the belief is peaked at a few opponents, the number of unique indices is much smaller than N_w, allowing faster computation of c_{j_k}. The short setup-time enables faster planning or, alternatively, the ability to compute the costs c_{j_k} with longer horizons. Therefore, theoretical performance automatically improves as the vehicle learns about the opponent and the robust evaluation approaches the true cost.

The second way we illustrate the price of robustness is by quantifying the quality of the approximation (3.5) with respect to the number of samples N_w. For shorthand, define the true expected and approximate expected costs for goal p and distributions Q and q respectively as

$$R(Q; p) := \mathbb{E}_Q[C(t; p)], \quad \hat{R}(q; p) := \frac{1}{N_w} \sum_{k=1}^{N_w} q_k c_{j_k}(t; p).$$

Then, we have the following bound:

Proposition 3.1 (Approximation quality)**.** *Suppose $C(t; p) \in [-1, 1]$ for all t, p. Let $A_\rho = \frac{2(\rho+1)}{\sqrt{1+\rho}-1}$ and $B_\rho = \sqrt{8(1+\rho)}$. Then with probability at least $1 - \delta$ over the N_w samples $J_k \overset{\text{i.i.d.}}{\sim} P_0$,*

$$\left| \sup_{q \in \mathcal{P}_{N_w}} \hat{R}(q; p) - \sup_{Q \in \mathcal{P}} R(Q; p) \right| \le 4A_\rho \sqrt{\frac{\log(2N_w)}{N_w}} + B_\rho \sqrt{\frac{\log \frac{2}{\delta}}{N_w}}$$

See Appendix B.2 for the proof. Intuitively, increasing accuracy of the robust cost requires

more samples (larger N_w), which comes at the expense of computation time. Similar to computing the full cost (3.4), ϵ-optimal solutions require $O(N_u \log(1/\epsilon))$ time for $N_u \leq N_w$ unique indices j_k. In our experiments (*cf.* Section 3.4), most of the computation time involves the setup to compute the N_u costs c_{j_k}.

3.3.2 Updating the ambiguity set

To maximize performance against an opponent, the agent updates the ambiguity set $\mathcal{P}$ as the race progresses. Since we consider ϕ-divergence balls of fixed size ρ, this update involves only the nominal belief vector $w(t)$. As with computation of the robust cost, this update must occur efficiently due to time and computational constraints.

For a given sequence of observations of the opponent $o^H_{\text{opp}}(t) := \{o_{\text{opp}}(t), o_{\text{opp}}(t-1), ..., o_{\text{opp}}(t-h+1)\}$ over a horizon h, we define the likelihood of this sequence coming from the i^{th} prototype opponent as

$$l^h_i(t) = \log d\mathbb{P}\left(o^h_{\text{opp}}(t) | G(\theta^{1,i})\right), \tag{3.6}$$

where $G(\theta^{1,i})$ is a generative model of goal states for the i^{th} prototype opponent. Letting $\bar{l}$ be a uniform upper bound on $l^h_i(t)$, we define the losses $L_i(t) := 1 - l^h_i(t)/\bar{l}$.

If we had enough time/computation budget, we could compute $L_i(t)$ for all prototype opponents i and perform an online mirror descent update with an entropic Bregman divergence [261]. In a resource-constrained setting, we can only select a few of these losses, so we use EXP3 [16] to update $w(t)$. Unlike a standard adversarial bandit setting, where we pull just one arm (*e.g.*compute a loss $L_i(t)$) at every time step, we may have resources to compute up to N_w losses in parallel at any given time (the same indices J_k discussed in Section 3.3.1). Denote our unbiased subgradient estimate as $\gamma(t)$:

$$\gamma_i(t) = \frac{1}{N_w} \sum_{k=1}^{N_w} \frac{L_i(t)}{w_i(t)} \mathbf{1}\{J_k = i\}. \tag{3.7}$$

Algorithm 3.2 describes our slightly modified EXP3 algorithm, which has the following expected regret.

Proposition 3.2. *Let* $z := \frac{d-1}{N_w} + 1$. *Algorithm 3.2 run for* T *iterations with stepsize* $\eta = \sqrt{\frac{2\log(d)}{zT}}$ *has expected regret bounded by* $\sum_{t=1}^{T} \mathbb{E}\left[\gamma(t)^T (w(t) - w^\star)\right] \leq \sqrt{2zT\log(d)}$.

Algorithm 3.2 EXP3 with N_w arm-pulls per iteration

Input: Stepsize sequence η_t, $w(0) := \mathbf{1}/d$, steps T
for $t = 0$ **to** $T - 1$
  Sample N_w indices $J_k \overset{\text{i.i.d.}}{\sim} \text{Categorical}(w(t))$
  Compute $\gamma(t)$ (Equation (3.7))
  $w_i(t+1) := \frac{w_i(t)\exp(-\eta_t \gamma_i(t))}{\sum_{j=1}^{d} w_j(t)\exp(-\eta_t \gamma_j(t))}$

See Appendix B.2 for the proof. This regret bound looks similar to that if we simply ran N_w standard EXP3 steps per iteration t (in which case $z = d/N_w$). However, our approach enables parallel computation which is critical in our time-constrained setting. Note that the "multiple-play" setting we propose here has been studied before with better regret bounds but higher computational complexity per iteration [285, 309]. We prefer our approach for its simplicity and ability to be easily combined with the robust-cost computation.

3.4 Experiments

In this section we first describe the AR environment used to conduct our experiments. Next we explore the hyperparameters of the algorithms in Section 3.2 and 3.3, identifying a preferred configuration. Then we consider the overarching hypothesis: online adaptation can improve the performance of robust control strategies. We show the statistically significant results affirming the theory and validate the approach's performance on real vehicles.

The experiments use an existing low-cost $1/10^{th}$-scale, Ackermann-steered AV (Figure 3.2). Additionally, we create a simulator and an associated OpenAI Gym API [51] suitable for distributed computing. The simulator supports multiple agents as well as deterministic executions. We experimentally determine the physical parameters of the agent models for simulation and use SLAM to build the virtual track as a mirror of a real location (see Figure 3.4). The hardware specifications, software, and simulator are open-source[3] (see Appendices B.3 and B.4 for details).

The agent software uses a hierarchical planner [107] similar to Ferguson et al. [96]. The key difference is the use of a masked autoregressive flow (MAF) [237] which provides the generative model for goal states, $G(\theta)$. Belief inference and robust cost computation require sampling and evaluating the likelihood of goal states. MAFs can evaluate likelihoods quickly but generate samples slowly. Inspired by Oord et al. [220] we overcome this inefficency by

[3] https://github.com/travelbureau/f0_icml_code

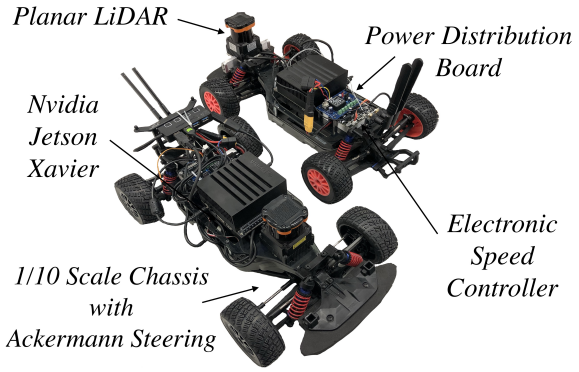


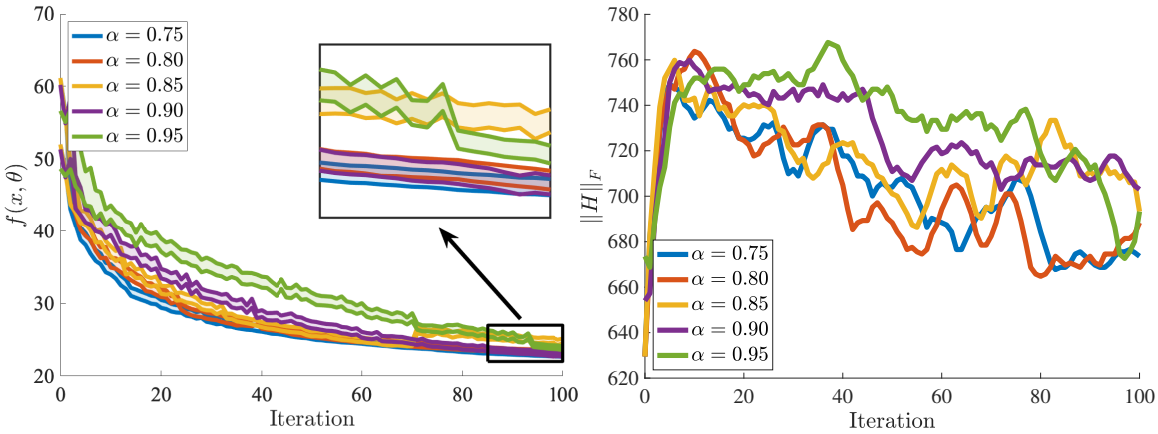
Figure 3.2: Components of the 1/10-scale vehicle

training a "student" inverse autogressive flow (IAF) [158] on MAF samples. Given a sample of goals from the IAF, the agent synthesizes dynamically feasible trajectories following McNaughton [196]. Each sample is evaluated according to Equation 3.4; the weights of the cost functions are learned by AAdaPT (and formal definitions of the cost components are in Appendix B.4). Belief updates use Algorithm 3.2 using the MAF to compute the losses $L_i(t)$.

3.4.1 Offline population synthesis

We run AAdaPT with $L = 5$ populations, $D = 160$ configurations per population, and $T = 100$ iterations. For vertical MCMC steps, we randomly sample 16 configuratons per population and perform $V = 2$ iterations of 5 hit-and-run proposals. Furthermore, we perform $E = DL^2/\alpha^{t/(L-1)}$ horizontal steps (motivated by the fact fact that "tunneling" from the highest-temperature level to the coldest takes $O(L^2)$ accepted steps). Finally, for training θ, we use Adam [156] with a learning rate of 10^{-4}.

Figure 3.3 shows results with 5 choices for the most influential hyperparameter, the annealing rate: $\alpha \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$. Figure 3.3(a) displays 95%-confidence intervals for the mean laptime in the coldest level. The annealing rates $\alpha \in \{0.75, 0.80, 0.90\}$ all result in comparable performance of 22.95 ± 0.14 (mean $\pm$ standard error) seconds at the end of the two-lap run. Figure 3.3(b) illustrates a metric for measuring diversity, the Frobenius norm of the Mahalanobis distance matrix (3.3). We see that $\alpha = 0.9$ results in the highest diversity while also attaining the best performance. Thus, in further experimentation, we use the results from the run conducted with $\alpha = 0.9$.



(a) Performance vs. iteration

(b) Diversity vs. iteration

Figure 3.3: Hyperparameter selection for AAdaPT. (a) 95%-confidence intervals for $f(x, \theta)$ in the coldest temperature level. (b) Frobenius norm of the Mahalanobis distance matrix H (3.3). The value $\alpha = 0.9$ achieves the best performance and diversity.

(a) Rollouts from 5 agents

(b) Snapshot trajectories

Figure 3.4: Qualitative illustrations of multimodal behavior in the learned population of cost functions

Figure 3.4 illustrates qualitative differences between cost functions. Figure 3.4(a) displays trajectories for agents driven using 5 cost functions sampled from the learned DPP. The cornering behavior is quite different between the trajectories. Figure 3.4(b) displays the trajectories chosen by all 160 agents in the population at $\beta_1(T)$ at various snapshots along the track. There is a wider spread of behavior near turns than areas where the car simply drives straight.

Table 3.1: The effect of distributional robustness on aggressiveness

Agent	% of iTTC values < 0.5s
$\rho/N_w = 0.001$	7.86± 0.90
$\rho/N_w = 0.025$	6.46± 0.78
$\rho/N_w = 0.2$	4.75± 0.65
$\rho/N_w = 0.4$	5.41± 0.74
$\rho/N_w = 0.75$	5.50± 0.82
$\rho/N_w = 1.0$	5.76± 0.84

3.4.2 Simulated experiments

We conduct a series of tests in simulation to determine the effects of distributional robustness and adaptivity on overall safety and performance. For a given robustness level $\rho/N_w \in \{0.001, 0.025, 0.2,$
$0.4, 0.75, 1.0\}$ (with $N_w = 8$ for all experiments), we simulate 40 two-lap races against each of the $d = 10$ diverse opponents sampled from the DPP. For fair comparisons, half of the races have the opponent starting on the outside and the other half with the opponent on the inside of the track. Importantly, these experiments involve only the most elite policies from the temperature level $\beta_1(T)$. Since the physical characteristics of the vehicles are identical, win rates between elite policies significantly greater than 0.5 are meaningful. In contrast, against a set of weaker opponents sampled via DPP from the 3^{rd} temperature level $\beta_3(T)$, the win-rate (fraction of races that our agent from the coldest temperature wins) is 0.848 ± 0.012.

Effects of distributional robustness We test the hypothesis that distributional robustness results in more conservative policies. For every race both agents have a fixed robustness level ρ and no adaptivity. To measure aggressiveness/conservativeness, we consider instantaneous time-to-collision (iTTC) of the vehicles during the race (see Appendix B.6). Smaller iTTC values imply more dangerous scenarios and more aggressive policies. In Table 3.1, we track the rate at which iTTC < 0.5 seconds. As expected, aggressiveness decreases with robustness (the rate of small iTTC values decreases as ρ increases). The trend is $a + b\log(\rho)$, where $a = 5.16 \pm 0.34$ and $b = -0.36 \pm 0.10$ ($R^2 = 0.75$).

Table 3.2: The effect of adaptivity on win-rate

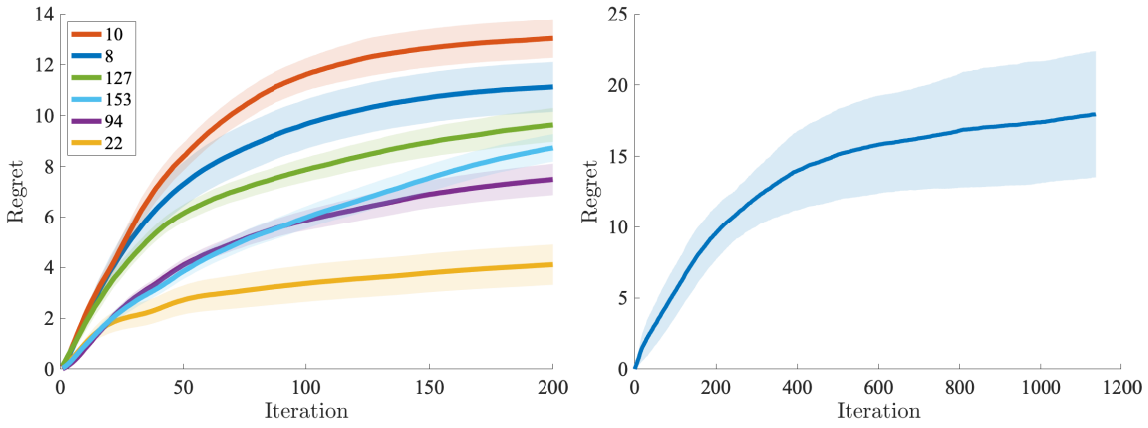
Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.593± 0.025	0.588± 0.025	0.84
$\rho/N_w = 0.025$	0.593± 0.025	0.600± 0.024	0.77
$\rho/N_w = 0.2$	0.538± 0.025	0.588± 0.025	0.045
$\rho/N_w = 0.4$	0.503± 0.025	0.573± 0.025	0.0098
$\rho/N_w = 0.75$	0.513± 0.025	0.593± 0.025	0.0013
$\rho/N_w = 1.0$	0.498± 0.025	0.590± 0.025	0.00024

Effects of adaptivity Now we investigate the effects of online learning on the outcomes of races. Figure 3.5(a) shows that Algorithm 3.2 identifies the opponent vehicle within approximately 150 timesteps (15 seconds), as illustrated by the settling of the regret curve.[4] Given evidence that the opponent model can be identified, we investigate whether adaptivity improves performance, as measured by win-rate. Table 3.2 displays results of paired t-tests for multiple robustness levels (with a null-hypothesis that adaptivity does not change the win-rate). Each test compares the effect of adaptivity for our agent on the 400 paired trials (and the opponents are always nonadaptive). Adaptivity significantly improves performance for the larger robustness levels $\rho/N_w \geq 0.2$. As hypothesized above, adaptivity automatically increases aggressiveness as the agent learns about its opponent and samples fewer of the other arms to compute the empirical robust cost (3.5). This effect is more prominent when robustness levels are greater, where adaptivity brings the win-rate back to its level without robustness ($\rho/N_w = 0.001$). Thus, the agent successfully balances safety and performance by combining distributional robustness with adaptivity.

3.4.3 Real-world validation

The real world experiments consist of races between agents 22 and 33; we examine the transfer of the opponent modeling approach from simulation to reality. In Figure 3.5(b) we plot 33's cumulative regret; it takes roughly 4 times as many observations relative to simulation-based experiments to identify the opponent (agent 22). We demonstrate the qualitative properties of the experiments in a video of real rollouts synchronized with

[4] We omit 3 of the regret lines for clarity in the plot.



(a) Simulation (b) Real

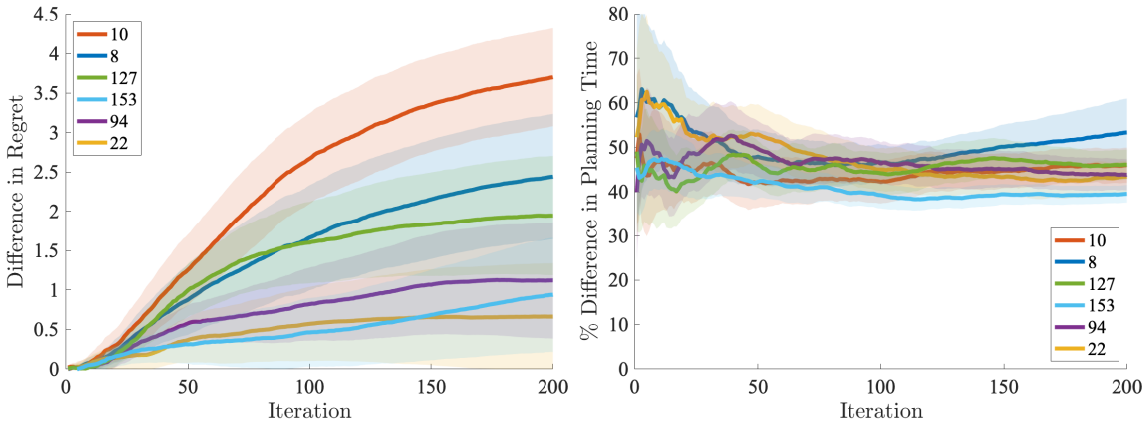
Figure 3.5: 95%-confidence intervals for regret using $N_w = 8$ arms in (a) simulation and (b) reality. The legend in (a) denotes opponent id and the opponent in (b) has id 22. Our agent has id 33.

corresponding simulations.[5] State estimation error and measurement noise drive the gap between simulated and real performance. First, both vehicle poses are estimated with a particle filter, whereas simulation uses ground-truth states. Since we infer beliefs about an opponent's policy based on a prediction of their actions at a given state, pose estimation error negatively impacts the accuracy of this inference. Second, the simulator only captures the geometry of the track; in reality glass and metal surfaces significantly affect the LIDAR range measurements, which in turn impact the MAF and IAF networks. The convergence of the cumulative regret in Figure 3.5(b) reflects that, despite the simulation/reality gap, our simulation-trained approach transfers to the real world. Diminishing the effect of the simulation/reality gap is the subject of future work (see Appendix B.5).

3.4.4 Approximation analysis

Sampling N_w indices $J_k \overset{\text{i.i.d.}}{\sim} P_0(t)$ allows us to quickly compute the approximate robust cost (Section 3.3.1) and perform a bandit-style update to the ambiguity set (Section 3.3.2). Now we analyze the time-accuracy tradeoff of performing this sampling approximation rather than using all d prototypical opponents at every time step. Figure 3.6(a) shows the difference in regret for the same experiments as in Figure 3.5(a) if we perform full online mirror-descent updates. Denoting the simulations in Figure 3.5(a) as S and those with

[5] https://youtu.be/7Yat9FZzE4g



(a) Difference in regret

(b) Difference in planning time

Figure 3.6: 95%-confidence intervals for the (a) difference in regret and (b) percent difference in cumulative planning time when using sampling approximations vs. online mirror descent. Online mirror descent yields lower regret at the expense of longer planning times.

the full mirror descent update as M, we compute difference as $\text{Regret}_S - \text{Regret}_M$. As expected, the difference is positive, since receiving the true gradient is better than the noisy estimate (3.7). Similarly, Figure 3.6(b) shows the percent increase in cumulative planning time for the same pairs (sampling vs. full online mirror descent), where percent increase is given by $100(\text{Time}_M - \text{Time}_S)/\text{Time}_S$. As the agent learns who the opponent is, it draws many repeats in the N_w arms, whereas the full mirror descent update always performs d computations. As a result, the percent increase in cumulative iteration time approaches a contant of approximately $1.5\times$. All of these comparisons are done in simulation, where the agent is not constrained to perform actions in under 100 milliseconds. Performing a full mirror descent update is impossible on the real car, as it requires too much time.

3.4.5 Out-of-distribution opponents

Now we measure performance against two agents—OOD1 and OOD2—that are not in the distribution developed by our offline population synthesis approach (see Appendix B.6.2 for details on each agent's policy). We perform only simulated experiments, as we are unable to perform further real-world experimentation at the time of writing due to the COVID-19 pandemic. For given robustness levels $\rho/N_w \in \{0.001, 1.0\}$ and $N_w = 8$ for all experiments, we perform 180 two-lap races against each of the two human-created racing agents. Again, for fair comparison, half of the experiments have the opponent start on the outside and

Table 3.3: The effect of adaptivity on win-rate vs. OOD1

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.633±0.036	0.683±0.035	0.280
$\rho/N_w = 1.0$	0.483±0.037	0.717±0.034	5.721e-6

Table 3.4: The effect of adaptivity on win-rate vs. OOD2

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	0.494±0.037	0.589±0.037	0.059
$\rho/N_w = 1.0$	0.572±0.037	0.739±0.033	0.001

half on the inside. Tables 3.3 and 3.4 show the results. Overall, the trends match those of the in-distribution opponents. Namely, adaptivity significantly increases the win-rate when robustness is high ($\rho/N_w = 1.0$), whereas for low robustness ($\rho/N_w = 0.001$) there is no significant change. Interestingly, adaptivity with robustness not only recovers but surpasses the win-rate of the non-adaptive non-robust policy. We hypothesize that, because out-of-distribution opponents do not match any of the learned prototypes, maintaining an uncertainty over belief automatically helps the agent plan against the "surprising" out-of-distribution actions. Validation of this hypothesis by comparing performance against more out-of-distribution opponents is an interesting direction for future work. Overall, we observe that even against out-of-distribution opponents, we achieve the overall goal of balancing performance and safety.

3.5 Discussion

The central hypothesis of this chapter is that distributionally robust evaluation of plans relative to the agent's belief state about opponents, which is updated as new observations are made, can lead to policies achieving the same performance as non-robust approaches without sacrificing safety. To evaluate this hypothesis we identify a natural division of the

underlying problem. First, we parameterize the set of possible opponents via population-based synthesis without requiring expert demonstrations. Second, we propose an online opponent-modeling framework which enables the application of distributionally robust optimization (DRO) techniques under computational constraints. We provide strong empirical evidence that distributional robustness combined with adaptivity enables a principled method automatically trading between safety and performance. Also, we demonstrate the transfer of our methods from simulation to real autonomous racecars. The addition of recursive feasibility arguments for stronger safety guarantees could improve the applicability of these techniques to real-world settings. Furthermore, although autonomous racing is the focus of our experiments in this chapter, future work should explore the generality of our approach in other settings such as human-robot interaction.

In this chapter as well as the previous one, we have considered relatively narrow, domain-specific definitions of safety: misclassification rates and episode lengths in Chapter 2 and percentage of time with small iTTC during a race in this chapter. In the next chapter, we generalize these notions of understanding and measuring safety. In particular, Chapter 4 presents a unified framework with which to measure safety that is well-suited for safety-critical algorithms—that is, measuring the *probability of failure*.

Part II

Testing Safety-Critical ML Systems

Chapter 4

The risk-based framework

The most important questions of life... are indeed, for the most part, only problems of probability.

— Pierre-Simon Laplace, *Théorie Analytique des Probabilités*

In this chapter, we present a framework with which to evaluate safety-critical machine learning systems. In such settings, failure is extremely costly, so the average-case test performance benchmark that is ubiquitous in current machine-learning practice is insufficient. Furthermore, because such systems interact with humans, model failure can result as a consequence of other stochastic agents, so defining notions of "correctness" is difficult. As such, formal verifications which rely on such binary specifications of correctness of an algorithm are inappropriate. Even if such a specification could be devised, model complexity often prevents viability of formal verification or other traditional software-testing techniques (similar to the NP-hardness results of Chapter 2). Because failures are costly yet always possible, we take the perspective of risk. Namely, we prioritize learning the *most likely* failure modes and characterizing a system's safety by its *probability of failure*. This approach, which we call the *risk-based framework*, is generally applicable to safety-critical machine-learning systems, but for concreteness, we frame this chapter within the context of autonomous driving. Chapter 5 broadens the scope to include other application domains.

While recent developments in autonomous vehicle (AV) technology highlight substantial progress, we lack tools for rigorous and scalable testing. Real-world testing, the *de facto* evaluation environment, places the public in danger, and, due to the rare nature of accidents, will require billions of miles in order to statistically validate performance claims.

We implement a simulation framework that can test an entire modern autonomous driving system, including, in particular, systems that employ deep-learning perception and control algorithms. Using adaptive importance-sampling methods to accelerate rare-event probability evaluation, we estimate the probability of an accident under a base distribution governing standard traffic behavior. We demonstrate our framework on a highway scenario, accelerating system evaluation by 2-20 times over naive Monte Carlo sampling methods and 10-300P times (where P is the number of processors) over real-world testing.

4.1 Introduction

Recent breakthroughs in deep learning have accelerated the development of autonomous vehicles (AVs); many research prototypes now operate on real roads alongside human drivers. While advances in computer-vision techniques have made human-level performance possible on narrow perception tasks such as object recognition, several fatal accidents involving AVs underscore the importance of testing whether the perception and control pipeline—when considered as a *whole system*—can safely interact with humans. Unfortunately, testing AVs in real environments, the most straightforward validation framework for system-level input-output behavior, requires prohibitive amounts of time due to the rare nature of serious accidents [260]. Concretely, a recent study [149] argues that AVs need to drive "hundreds of millions of miles and, under some scenarios, hundreds of billions of miles to create enough data to clearly demonstrate their safety." Alteratively, formally verifying an AV algorithm's "correctness" [171, 7, 256, 217] is difficult since all driving policies are subject to crashes caused by other drivers [260]. It is unreasonable to ask that the policy be safe under *all* scenarios. Unfortunately, ruling out scenarios where the AV should not be blamed is a task subject to logical inconsistency, combinatorial growth in specification complexity, and subjective assignment of fault.

Motivated by the challenges underlying real-world testing and formal verification, we consider a probabilistic paradigm—which we call a *risk-based framework*—where the goal is to evaluate the *probability of an accident* under a base distribution representing standard traffic behavior. By assigning learned probability values to environmental states and agent behaviors, our risk-based framework considers performance of the AV's policy under a data-driven model of the world. To efficiently evaluate the probability of an accident, we implement a photo-realistic and physics-based simulator that provides the AV with

perceptual inputs (*e.g.* video and range data) and traffic conditions (*e.g.* other cars and pedestrians). The simulator allows parallelized, faster-than-real-time evaluations in varying environments (*e.g.* weather, geographic locations, and aggressiveness of other cars).

Formally, we let P_0 denote the base distribution that models standard traffic behavior and $X \sim P_0$ be a realization of the simulation (*e.g.* weather conditions and driving policies of other agents). For an objective function $f : \mathcal{X} \to \mathbb{R}$ that measures "safety"—so that low values of $f(x)$ correspond to dangerous scenarios—our goal is to evaluate the probability of a dangerous event

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma) \tag{4.1}$$

for some threshold γ. Our risk-based framework is agnostic to the complexity of the ego-policy and views it as a black-box module. Such an approach allows, in particular, deep-learning based perception systems that make formal verification methods intractable.

An essential component of this approach is to estimate the base distribution P_0 from data; we use public traffic data collected by the US Department of Transportation [286]. While such datasets do not offer insights into how AVs interact with human agents—this is precisely why we design our simulator—they illustrate the range of standard human driving behavior that the base distribution P_0 must model. We use imitation learning [250, 242, 243, 131, 21] to learn a generative model for the behavior (policy) of environment vehicles; unlike traditional imitation learning, we train an ensemble of models to characterize a distribution of human-like driving policies.

As serious accidents are rare (p_γ is small), we view this as a *rare-event simulation* [13] problem; naive Monte Carlo sampling methods require prohibitively many simulation roll-outs to generate dangerous scenarios and estimate p_γ. To accelerate safety evaluation, we use adaptive importance-sampling methods to learn alternative distributions P_θ that generate accidents more frequently. Specifically, we use the cross-entropy algorithm [247] to iteratively approximate the optimal importance sampling distribution. In contrast to simple classical settings [247, 308] which allow analytic updates to P_θ, our high-dimensional search space requires solving convex optimization problems in each iteration (Section 4.2). To address numerical instabilities of importance sampling estimators in high dimensions, we carefully design search spaces and perform computations in logarithmic scale. Our implementation produces 2-20 times as many rare events as naive Monte Carlo methods, independent of the complexity of the ego-policy.

In addition to accelerating evaluation of p_γ, learning a distribution P_θ that *frequently*

Figure 4.1: Multi-lane highway driving on I-80: (left) real image, (right) rendered image from simulator

generates realistic dangerous scenarios $X_i \sim P_\theta$ is useful for engineering purposes. The importance-sampling distribution P_θ not only efficiently samples dangerous scenarios, but also ranks them according to their likelihoods under the base distribution P_0. This capability enables a deeper understanding of failure modes and prioritizes their importance to improving the ego-policy.

As a system, our simulator allows fully distributed rollouts, making our approach orders of magnitude cheaper, faster, and safer than real-world testing. Using the asynchronous messaging library ZeroMQ [130], our implementation is fully-distributed among available CPUs and GPUs; our rollouts are up to $30\mathsf{P}$ times faster than real time, where P is the number of processors. Combined with the cross-entropy method's speedup, we achieve 10-$300\mathsf{P}$ speedup over real-world testing.

In what follows, we describe components of our open-source toolchain, a photo-realistic simulator equipped with our data-driven risk-based framework and cross-entropy search techniques. The toolchain can test an AV as a *whole system*, simulating the driving policy of the ego-vehicle by viewing it as a black-box model. The use of adaptive-importance sampling methods motivates a unique simulator architecture (Section 4.3) which allows real-time updates of the policies of environment vehicles. In Section 4.4, we test our toolchain by considering an end-to-end deep-learning-based ego-policy [43] in a multi-agent highway scenario. Figure 4.1 shows one configuration of this scenario in the real world along with rendered images from the simulator, which uses Unreal Engine 4 [103]. Our experiments show that we accelerate the assessment of rare-event probabilities with respect to naive Monte Carlo methods as well as real-world testing. We believe our open-source framework

is a step towards a rigorous yet scalable platform for evaluating AV systems, with the broader goal of understanding how to reliably deploy deep-learning systems in safety-critical applications.

4.2 Rare-event simulation

To motivate our risk-based framework, we first argue that formally verifying correctness of a AV system is infeasible due to the challenge of defining "correctness." Consider a scenario where an AV commits a traffic violation to avoid collision with an out-of-control truck approaching from behind. If the ego-vehicle decides to avoid collision by running through a red light with no further ramifications, is it "correct" to do so? The "correctness" of the policy depends on the extent to which the traffic violation endangers nearby humans and whether any element of the "correctness" specification explicitly forbids such actions. That is, "correctness" as a binary output is a concept defined by its exceptions, many elements of which are subject to individual valuations [45].

Instead of trying to verify correctness, we begin with a continuous measure of safety $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is space of traffic conditions and behaviors of other vehicles. The prototypical example in this chapter is the minimum time-to-collision (TTC) (see Appendix C.1 for its definition) to other environmental agents over a simulation rollout. Rather than requiring safety for all $x \in \mathcal{X}$, we relax the deterministic verification problem into a probabilistic one where we are concerned with the probability under standard traffic conditions that $f(X)$ goes below a safety threshold. Given a distribution P_0 on $\mathcal{X}$, our goal is to estimate the rare event probability $p_\gamma := P_0(f(X) \leq \gamma)$ based on simulated rollouts $f(X_1), \ldots, f(X_n)$. Note that the safety function f need not make judgements regarding correctness, fault, or blame (although it is flexible enough such that constructive notions of blame like the RSS approach of Shalev-Shwartz et al. [260] can be components of f). Rather, the burden of proof for the importance of various modes of failure is placed on their probability of occurrence. Furthermore, although we consider only a scalar function f in this work, the framework can be generalized to include multi-objective criteria, in which case the the rare-event probability becomes one of searching for a Pareto frontier below some threshold γ.

As accidents are rare and p_γ is near 0, we treat this as a rare-event simulation problem;

see [56, 13, Chapter VI] for an overview of this topic. First, we briefly illustrate the well-known difficulty of naive Monte Carlo simulation when p_γ is small. From a sample $X_i \overset{\text{i.i.d.}}{\sim} P_0$, the naive Monte Carlo estimate is $\widehat{p}_{N,\gamma} := \frac{1}{N}\sum_{i=1}^N \mathbf{1}\{f(X_i) \le \gamma\}$. As p_γ is small, we use relative accuracy to measure our performance, and the central limit theorem implies the relative accuracy is approximately

$$\left|\frac{\widehat{p}_{N,\gamma}}{p_\gamma} - 1\right| \overset{\text{dist}}{\approx} \sqrt{\frac{1-p_\gamma}{Np_\gamma}}\,|Z| + o(1/\sqrt{N}) \quad \text{for } Z \sim \mathsf{N}(0,1).$$

For small p_γ, we require a sample of size $N \gtrsim 1/(p_\gamma \epsilon^2)$ to achieve ϵ-relative accuracy, and if $f(X)$ is light-tailed, the sample size must grow exponentially in γ.

Cross-entropy method As an alternative to a naive Monte Carlo estimator, we consider (adaptive) importance sampling [13], and we use a model-based optimization procedure to find a good importance-sampling distribution. The optimal importance-sampling distribution for estimating p_γ has the conditional density $p^\star(x) = \mathbf{1}\{f(x) \le \gamma\}\, p_0(x)/p_\gamma$, where p_0 is the density function of P_0: as $p_0(x)/p^\star(x) = p_\gamma$ for all x satisfying $\mathbf{1}\{f(x) \le \gamma\}$, the estimate $\widehat{p}^\star_{N,\gamma} := \frac{1}{N}\sum_{i=1}^N \frac{p_0(X_i)}{p^\star(X_i)}\mathbf{1}\{f(X_i) \le \gamma\}$ is exact. This sampling scheme is, unfortunately, *de facto* impossible, because we do not know p_γ. Instead, we use a parameterized importance sampler P_θ and employ an iterative model-based search method to modify θ so that P_θ approximates $P^\star$.

The cross-entropy method [247] iteratively tries to find $\theta^\star \in \operatorname{argmin}_{\theta\in\Theta} D_{\text{kl}}(P^\star \| P_\theta)$, the Kullback-Leibler projection of $P^\star$ onto the class of parameterized distributions $\mathcal{P} = \{P_\theta\}_{\theta\in\Theta}$. Over iterations k, we maintain a surrogate distribution $q_k(x) \propto \mathbf{1}\{f(x) \le \gamma_k\}\, p_0(x)$ where $\gamma_k \ge \gamma$ is a (potentially random) proxy for the rare-event threshold γ, and we use samples from P_θ to update θ as an approximate projection of Q onto $\mathcal{P}$. The motivation underlying this approach is to update θ so that P_θ upweights regions of $\mathcal{X}$ with low objective value (*i.e.* unsafe) $f(x)$. We fix a quantile level $\rho \in (0,1)$—usually we choose $\rho \in [0.01, 0.2]$—and use the ρ-quantile of $f(X)$ where $X \sim P_{\theta_k}$ as γ_k, our proxy for the rare event threshold γ (see [133] for alternatives). We have the additional challenge that the ρ-quantile of $f(X)$ is unknown, so we approximate it using i.i.d. samples $X_i \sim P_{\theta_k}$. Compared to applications of the cross-entropy method [247, 308] that focus on low-dimensional problems permitting analytic updates to θ, our high-dimensional search space requires solving convex optimization problems in each iteration. To address numerical challenges in computing likelihood

Algorithm 4.1 Cross-Entropy Method

1: Input: Quantile $\rho \in (0,1)$, Stepsizes $\{\alpha_k\}_{k\in\mathbb{N}}$, Sample sizes $\{N_k\}_{k\in\mathbb{N}}$, Number of iterations K
2: Initialize: $\theta_0 \in \Theta$
3: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
4: Sample $X_{k,1}, \ldots, X_{k,N_k} \overset{\text{i.i.d.}}{\sim} P_{\theta_k}$
5: Set γ_k as the minimum of γ and the ρ-quantile of $f(X_{k,1}), \ldots, f(X_{k,N_k})$
6: $\theta_{k+1} = \operatorname{argmax}_{\theta\in\Theta} \left\{\alpha_k \theta^\top D_{k+1} + (1-\alpha_k)\theta^\top \nabla A(\theta_k) - A(\theta)\right\}$

ratios in high-dimensions, our implementation carefully constrains the search space and we compute likelihoods in logarithmic scale.

We now rigorously describe the algorithmic details. First, we use natural exponential families as our class of importance samplers $\mathcal{P}$.

Definition 1. *The family of density functions $\{p_\theta\}_{\theta\in\Theta}$, defined with respect to base measure μ, is a* natural exponential family *if there exists a sufficient statistic Γ such that $p_\theta(x) = \exp(\theta^\top \Gamma(x) - A(\theta))$ where $A(\theta) = \log \int_{\mathcal{X}} \exp(\theta^\top \Gamma(x)) d\mu(x)$ is the log partition function and $\Theta := \{\theta \mid A(\theta) < \infty\}$.*

Given this family, we consider idealized updates to the parameter vector θ_k at iteration k, where we compute projections of a mixture of Q_k and P_{θ_k} onto $\mathcal{P}$

$$\begin{aligned}
\theta_{k+1} &= \operatorname*{argmin}_{\theta\in\Theta} D_{\mathrm{kl}}\left(\alpha_k Q_k + (1-\alpha_k) P_{\theta_k} \| P_\theta\right) \\
&= \operatorname*{argmax}_{\theta\in\Theta} \left\{\alpha_k \mathbb{E}_{Q_k}[\log p_\theta(X)] + (1-\alpha_k)\mathbb{E}_{\theta_k}[\log p_\theta(X)]\right\} \\
&= \operatorname*{argmax}_{\theta\in\Theta} \left\{\alpha_k \theta^\top \mathbb{E}_{Q_k}[\Gamma(X)] + (1-\alpha_k)\theta^\top \nabla A(\theta_k) - A(\theta)\right\}. \qquad (4.2)
\end{aligned}$$

The term $\mathbb{E}_{Q_k}[\Gamma(X)]$ is unknown in practice, so we use a sampled estimate. For $X_{k,1}, \ldots, X_{k,N_k} \overset{\text{i.i.d.}}{\sim} P_{\theta_k}$, let γ_k be the ρ-quantile of $f(X_{k,1}), \ldots, f(X_{k,N_k})$ and define

$$D_{k+1} := \frac{1}{N_k}\sum_{i=1}^{N_k} \frac{q_k(X_{k,i})}{p_{\theta_k}(X_{k,i})}\Gamma(X_{k,i}) = \frac{1}{N_k}\sum_{i=1}^{N_k} \frac{p_0(X_{k,i})}{p_{\theta_k}(X_{k,i})}\mathbf{1}\left\{f(X_{k,i}) \le \gamma_k\right\}\Gamma(X_{k,i}). \qquad (4.3)$$

Using the estimate D_{k+1} in place of $\mathbb{E}_{Q_k}[\Gamma(X)]$ in the idealized update (4.2), we obtain Algorithm 4.1. To select the final importance sampling distribution from Algorithm 4.1, we choose θ_k with the lowest ρ-quantile of $f(X_{k,i})$. We observe that this choice consistently

improves performance over taking the last iterate or Polyak averaging. Letting θ_{ce} denote the parameters for the importance sampling distribution learned by the cross-entropy method, we sample $X_i \overset{\text{i.i.d.}}{\sim} P_{\theta_{\text{ce}}}$ and use $\widehat{p}_{N,\gamma} := \frac{1}{N}\sum_{i=1}^{N} \frac{p_0(X_i)}{p_{\theta_{\text{ce}}}(X_i)} \mathbf{1}\{f(X_i) \leq \gamma\}$ as our final importance-sampling estimator for p_γ.

In the context of our rare-event simulator, we use a combination of Beta and Normal distributions for P_θ. The sufficient statistics Γ include (i) the parameters of the generative model of behaviors that our imitation-learning schemes produce and (ii) the initial poses and velocities of other vehicles, pedestrians, and obstacles in the simulation. Given a current parameter θ and realization from the model distribution P_θ, our simulator then (i) sets the parameters of the generative model for vehicle policies and draws policies from this model, and (ii) chooses random poses and velocities for the simulation. Our simulator is one of the largest-scale applications of cross-entropy methods.

4.3 Simulation framework

Two key considerations in our risk-based framework influence design choices for our simulation toolchain: (1) learning the base distribution P_0 of nominal traffic behavior via data-driven modeling, and (2) testing the AV as a *whole system*. We now describe how our toolchain achieves these goals.

4.3.1 Data-driven generative modeling

While our risk-based framework (cf. Section 4.2) is a concise, unambiguous measure of system safety, the rare-event probability p_γ is only meaningful insofar as the base distribution P_0 of road conditions and the behaviors of other (human) drivers is estimable. Thus, to implement our risk-based framework, we first learn a base distribution P_0 of nominal traffic behavior. Using the highway traffic dataset NGSim [286], we train policies of human drivers via imitation learning [250, 242, 243, 131, 21]. Our data consists of videos of highway traffic [286], and our goal is to create models that imitate human driving behavior even in scenarios distinct from those in the data. We employ an ensemble of generative adversarial imitation learning (GAIL) [131] models to learn P_0. Our approach is motivated by the observation that reducing an imitation-learning problem to supervised learning—where we simply use expert data to predict actions given vehicle states—suffers from poor performance in regions of the state space not encountered in data [242, 243]. Reinforcement-learning techniques

have been observed to improve generalization performance, as the imitation agent is able to explore regions of the state space in simulation during training that do not necessarily occur in the expert data traces.

Generically, GAIL is a minimax game between two functions: a discriminator D_ϕ and a generator G_ξ (with parameters ϕ and ξ respectively). The discriminator takes in a state-action pair (s, u) and outputs the probability that the pair came from real data, $\mathbb{P}(\text{real data})$. The generator takes in a state s and outputs a conditional distribution $G_\xi(s) := \mathbb{P}(u \mid s)$ of the action u to take given state s. In our context, $G_\xi(\cdot)$ is then the (learned) policy of a human driver given environmental inputs s. Training the generator weights ξ occurs in a reinforcement-learning paradigm with reward $-\log(1 - D_\phi(s, G_\xi(s)))$. We use the model-based variant of GAIL (MGAIL) [21] which renders this reward fully differentiable with respect to ξ over a simulation rollout, allowing efficient model training. GAIL has been validated by Kuefler et al. [166] to realistically mimic human-like driving behavior from the NGSim dataset across multiple metrics. These include the similarity of low-level actions (speeds, accelerations, turn-rates, jerks, and time-to-collision), as well as higher-level behaviors (lane change rate, collision rate, hard-brake rate, etc). See Appendix C.3 for a reference to an example video of the learned model driving in a scenario alongside data traces from human drivers.

Our importance sampling and cross-entropy methods use not just a single instance of model parameters ξ, but rather a distribution over them to form a generative model of human driving behavior. To model this distribution, we use a (multivariate normal) parametric bootstrap over a trained ensemble of generators $\xi^i,\ i = 1, \ldots, m$. Our models ξ^i are high-dimensional ($\xi \in \mathbb{R}^d,\ d > m$) as they characterize the weights of large neural networks, so we employ the graphical lasso [99] to fit the inverse covariance matrix for our ensemble. This approach to modeling uncertainty in neural-network weights is similar to the bootstrap approach of Osband et al. [221]. Other approaches include using dropout for inference [101] and variational methods [117, 42, 157].

While several open source driving simulators have been proposed [81, 258, 232], our problem formulation requires unique features to allow sampling from a continuous distribution of driving policies for environmental agents. Conditional on each sample of model parameters ξ, the simulator constructs a (random) rollout of vehicle behaviors according to G_ξ. Our simulator is designed to efficiently execute and update these policies as new samples ξ are drawn for each rollout. At the time this research was conducted, other simulators

did not have this functionality. Since then, the CARLA simulator [81] has significantly increased its functionality.

4.3.2 System architecture

The second key characteristic of our framework is that it enables black-box testing the AV as a *whole system*. Flaws in complex systems routinely occur at poorly specified interfaces between components, as interactions between processes can induce unexpected behavior. Consequently, solely testing subcomponents of an AV control pipeline separately is insufficient [2]. Moreover, it is increasingly common for manufacturers to utilize software and hardware artifacts for which they do not have any whitebox model [126, 62]. We provide a concise but extensible language-agnostic interface to our benchmark world model so that common AV sensors such as cameras and lidar can provide the necessary inputs to induce vehicle actuation commands.

Our simulator is a distributed, modular framework, which is necessary to support the inclusion of new AV systems and updates to the environment-vehicle policies. A benefit of this design is that simulation rollouts are simple to parallelize. In particular, we allow instantiation of multiple simulations simultaneously, without requiring that each include the entire set of components. For example, a desktop may support only one instance of Unreal Engine but could be capable of simulating 10 physics simulations in parallel; it would be impossible to fully utilize the compute resource with a monolithic executable wrapping all engines together. Our architecture enables instances of the components to be distributed on heterogeneous GPU compute clusters while maintaining the ability to perform meaningful analysis locally on commodity desktops. In Appendix C.1, we detail our scenario specification, which describes how Algorithm 4.1 maps onto our distributed architecture.

4.4 Experiments

In this section, we demonstrate our risk-based framework on a multi-agent highway scenario. As the rare-event probability of interest p_γ gets smaller, the cross-entropy method learns to sample more rare events compared to naive Monte Carlo sampling; we empirically observe that the cross-entropy method produces 2-20 times as many rare events as its naive counterpart. Our findings hold across different ego-vehicle policies, base distributions P_0,

and scenarios.

To highlight the modularity of our simulator, we evaluate the rare-event probability p_γ on two different ego-vehicle policies. The first is an instantiation of an imitation learning (non-vision) policy which uses lidar as its primary perceptual input. Secondly, we investigate a vision-based controller (vision policy), where the ego-vehicle drives with an end-to-end highway autopilot network [43], taking as input a rendered image from the simulator (and lidar observations) and outputting actuation commands. See Appendix C.2 for a summary of network architectures used.

We consider a scenario consisting of six agents, five of which are considered part of the environment. The environment vehicles' policies follow the distribution learned in Section 4.3.1. All vehicles are constrained to start within a set of possible initial configurations consisting of pose and velocity, and each vehicle has a goal of reaching the end of the approximately 2 km stretch of road. Fig. 4.1 shows one such configuration of the scenario, along with rendered images from the simulator. We create scene geometry based on surveyors' records and photogrammetric reconstructions of satellite imagery of the portion of I-80 in Emeryville, California where the traffic data was collected [286].

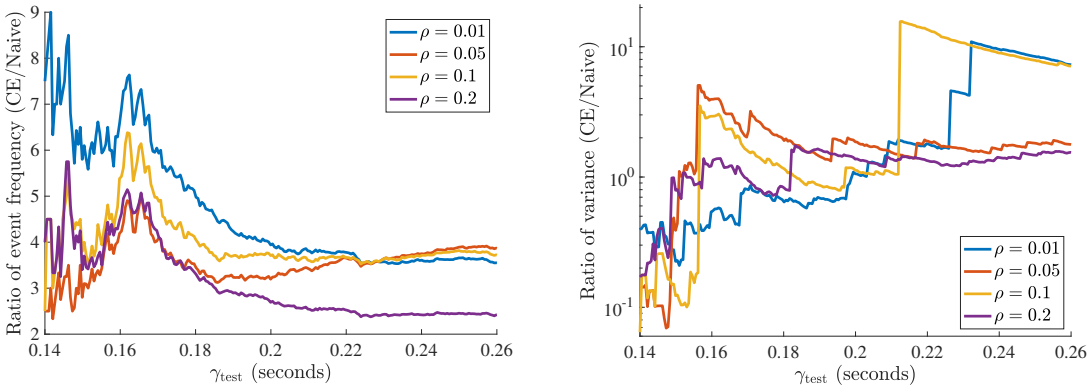
Simulation parameters We detail our postulated base distribution P_0. Letting m denote the number of vehicles, we consider the random tuple $X = (S, T, W, V, \xi)$ as our simulation parameter where the pair $(S, T) \in \mathbb{R}_+^{m\times 2}$ indicates the two-dimensional positioning of each vehicle in their respective lanes (in meters), W the orientation of each vehicle (in degrees), and V the initial velocity of each vehicle (in meters per second). We use $\xi \in \mathbb{R}^{404}$ to denote the weights of the last layer of the neural network trained to imitate human-like driving behavior. Specifically, we set $S \sim 40\text{Beta}(2, 2) + 80$ with respect to the starting point of the road, $T \sim 0.5\text{Beta}(2, 2) - 0.25$ with respect to the lane's center, $W \sim 7.2\text{Beta}(2, 2) - 3.6$ with respect to facing forward, and $V \sim 10\text{Beta}(2, 2) + 10$. We assume $\xi \sim \mathcal{N}(\mu_0, \Sigma_0)$, with the mean and covariance matrices learned via the ensemble approach outlined in Section 4.3.1. The neural network whose last layer is parameterized by ξ describes the policy of environment vehicles; it takes as input the state of the vehicle and lidar observations of the surrounding environment (see Appendix C.2 for more details). Throughout this section, we define our measure of safety $f : \mathcal{X} \to \mathbb{R}$ as the minimum time-to-collision (TTC) over the simulation rollout. We calculate TTC from the center of mass of the ego vehicle; if the ego-vehicle's body crashes into obstacles, we end the simulation

before the TTC can further decrease (see Appendix C.1 for details).

Cross-entropy method Throughout our experiments, we impose constraints on the space of importance samplers (adversarial distributions) for feasibility. Numerical stability considerations predominantly drive our hyperparameter choices. For model parameters ξ, we also constrain the search space to ensure that generative models G_ξ maintain reasonably realistic human-like policies (recall Sec. 4.3.1). For S, T, W, and V, we let $\{\text{Beta}(\alpha, \beta) : \alpha, \beta \in [1.5, 7]\}$ be the model space over which the cross-entropy method searches, scaled and centered appropriately to match the scale of the respective base distributions. We restrict the search space of distributions over $\xi \in \mathbb{R}^{404}$ by searching over $\{\mathsf{N}(\mu, \Sigma_0) : \|\mu - \mu_0\|_\infty \leq .01\}$, where (μ_0, Σ_0) are the parameters of the base (bootstrap) distribution. For our importance sampling distribution P_θ, we use products of the above marginal distributions. These restrictions on the search space mitigate numerical instabilities in computing likelihood ratios within our optimization routines, which is important for our high-dimensional problems.

We first illustrate the dependence of the cross-entropy method on its hyperparameters. We choose to use a non-vision ego-vehicle policy as a test bed for hyperparameter tuning, since this allows us to take advantage of the fastest simulation speeds for our experiments. We focus on the effects (in Algorithm 4.1) of varying the most influential hyperparameter, $\rho \in (0, 1]$, which is the quantile level determining the rarity of the observations used to compute the importance sampler θ_k. Intuitively, as ρ approaches 0, the cross-entropy method learns importance samplers P_θ that up-weight unsafe regions of $\mathcal{X}$ with lower $f(x)$, increasing the frequency of sampling rare events (events with $f(X) \leq \gamma$). In order to avoid overfitting θ_k as $\rho \to 0$, we need to increase N_k as ρ decreases. Our choice of N_k is borne out of computational constraints as it is the biggest factor that determines the run-time of the cross-entropy method. Consistent with prior works [247, 135], we observe empirically that $\rho \in [0.01, 0.2]$ is a good range for the values of N_k deemed feasible for our computational budget ($N_k = 1000 \sim 5000$). We fix the number of iterations at $K = 100$, number of samples taken per iteration at $N_k = 5000$, step size for updates at $\alpha_k = 0.8$, and $\gamma = 0.14$. As we see below, we consistently observe that the cross-entropy method learns to sample significantly more rare events, despite the high-dimensional nature ($d \approx 500$) of the problem.

To evaluate the learned parameters, we draw $n = 10^5$ samples from the importance sampling distribution to form an estimate of p_γ. In Figure 4.2, we vary ρ and report the



(a) Ratio of number of rare events vs. threshold

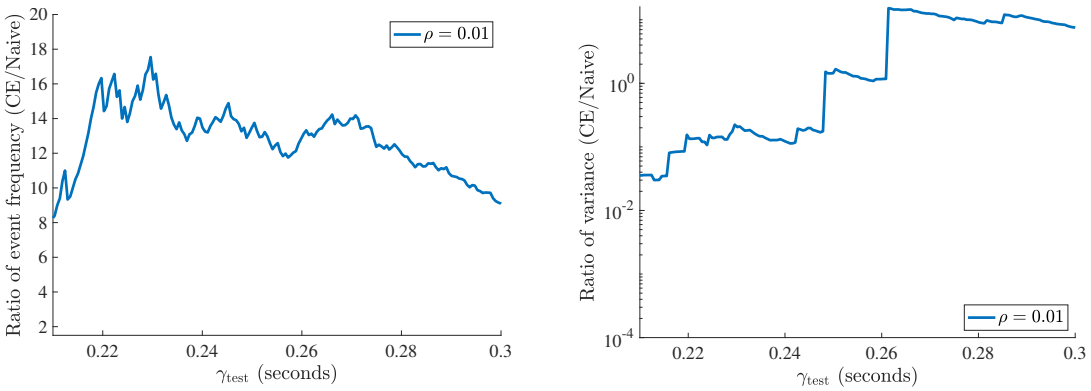
(b) Ratio of variance vs. threshold

Figure 4.2: The ratio of (a) number of rare events and (b) estimation variance for p_γ between cross-entropy method and naive MC sampling for the non-vision ego policy. Rarity is inversely proportional to γ, and, as expected, we see the best performance for our method over naive MC at small γ.

Search Algorithm	$\gamma_{\text{test}} = 0.14$	$\gamma_{\text{test}} = 0.15$	$\gamma_{\text{test}} = 0.19$	$\gamma_{\text{test}} = 0.20$
Naive 1300K	(12.4±3.1)e-6	(80.6±7.91)e-6	(133±3.2)e-5	(186±3.79)e-5
Cross-entropy 100K	(19.8±8.88)e-6	(66.1 ± 15)e-6	(108± 9.51)e-5	(164 ± 14)e-5
Naive 100K	(20±14.1)e-6	(100± 31.6)e-6	(132±11.5)e-5	(185±13.6)e-5

Table 4.1: Estimate of rare-event probability p_γ (non-vision ego policy) with standard errors. For the cross-entropy method, we show results for the learned importance sampling distribution with $\rho = 0.01$.

relative performance of the cross-entropy method compared to naive Monte Carlo sampling. Even though we set $\gamma = 0.14$ in Algorithm 4.1, we evaluate the performance of all models with respect to multiple threshold levels γ_{test}. We note that as ρ approaches 0, the cross-entropy method learns to frequently sample increasingly rare events; the cross-entropy method yields 3-10 times as many dangerous scenarios, and achieves 2-16 times variance reduction depending on the threshold level γ_{test}. In Table 4.1, we contrast the estimates provided by naive Monte Carlo and the importance sampling estimator provided by the cross-entropy method with $\rho = 0.01$; to form a baseline estimate, we run naive Monte Carlo with $1.3 \cdot 10^6$ samples. For a given number of samples, the cross-entropy method with $\rho = 0.01$ provides more precise estimates for the rare-event probability $p_\gamma \approx 10^{-5}$ over naive Monte Carlo.



(a) Ratio of number of rare events vs. threshold

(b) Ratio of variance vs. threshold

Figure 4.3: The ratio of (a) number of rare events and (b) estimation variance for p_γ between cross-entropy method and naive MC sampling for the vision-based ego policy.

Search Algorithm	$\gamma_{\text{test}} = 0.22$	$\gamma_{\text{test}} = 0.23$	$\gamma_{\text{test}} = 0.24$	$\gamma_{\text{test}} = 0.25$
Cross-entropy 50K	(5.87±1.82)e-5	(13.0± 2.94)e-5	(19.0 ± 3.14)e-5	(4.52 ± 1.35)e-4
Naive 50K	(11.3±4.60)e-5	(20.6±6.22)e-5	(43.2±9.00)e-5	(6.75±1.13)e-4

Table 4.2: Estimate of rare-event probability p_γ (vision-based ego policy) with standard errors. For the cross-entropy method, we show results for the learned importance sampling distribution with $\rho = 0.01$.

We now leverage the tuned hyperparameter ($\rho = 0.01$) for our main experiment: evaluating the probability of a dangerous event for the vision-based ego policy. We find that the hyperparameters for the cross-entropy method generalize, allowing us to produce good importance samplers for a very different policy without further tuning. Based on our computational budget (with our current implementation, vision-based simulations run about 15 times slower than simulations with only non-vision policies), we choose $K = 20$ and $N_k = 1000$ for the cross-entropy method to learn a good importance sampling distribution for the vision-based policy (although we also observe similar behavior for N_k as small as 100). In Figure 4.3, we illustrate again that the cross-entropy method learns to sample dangerous scenarios more frequently (Figure 4.3(a))—up to 18 times that of naive Monte Carlo—and produces importance sampling estimators with lower variance (Figure 4.3(b)). As a result, our estimator in Table 4.2 is better calibrated compared to that computed from naive Monte Carlo.

Qualitative analysis We provide a qualitative interpretation for the learned parameters of the importance sampler. For initial velocities, angles, and positioning of vehicles, the importance sampler shifts environmental vehicles to box in the ego-vehicle and increases the speeds of trailing vehicles by 20%, making accidents more frequent. We also observe that the learned distribution for initial conditions have variance 50% smaller than that of the base distribution, implying concentration around adversarial conditions. Perturbing the policy weights ξ for GAIL increases the frequency of risky high-level behaviors (lane-change rate, hard-brake rate, etc.). An interesting consequence of using our definition of TTC from the center of the ego vehicle (cf. Appendix C.1) as a measure of safety is that dangerous events $f(X) \leq \gamma_{\text{test}}$ (for small γ_{test}) include frequent sideswiping behavior, as such accidents result in smaller TTC values than front- or rear-end collisions. See Appendix C.3 for a reference to supplementary videos that exhibit the range of behavior across many levels γ_{test}. The modularity of our simulation framework easily allows us to modify the safety objective to an alternative definition of TTC or even include more sophisticated notions of safety, *e.g.* temporal-logic specifications or implementations of responsibility-sensitive safety (RSS) [260, 241].

4.5 Related work and discussion

Given the complexity of AV software and hardware components, it is unlikely that any single method will serve as an oracle for certification. Many existing tools are complementary to our risk-based framework. In this section, we compare and contrast representative results in testing, verification, and simulation.

AV testing generally consists of three paradigms. The first, largely attributable to regulatory efforts, uses a finite set of basic competencies (*e.g.* the Euro NCAP Test Protocol [255]); while this methodology is successful in designing safety features such as airbags and seat-belts, the non-adaptive nature of static testing is less effective in complex software systems found in AVs. Alternatively, real-world testing—deployment of vehicles with human oversight—exposes the vehicle to a wider variety of unpredictable test conditions. However, as we outlined above, these methods pose a danger to the public and require prohibitive number of driving hours due to the rare nature of accidents [149]. Simulation-based falsification (in our context, simply finding any crash) has also been successfully utilized [284]; this approach does not maintain a link to the likelihood of the occurrence of a particular

event, which we believe to be key in acting to prioritize and correct AV behavior.

Formal verification methods [171, 7, 256, 217] have emerged as a candidate to reduce the intractability of empirical validation. A verification procedure considers whether the system can *ever* violate a specification and returns either a proof that there is no such execution or a counterexample. Verification procedures require a white-box description of the system (although it may be abstract), as well as a mathematically precise specification. Due to the impossibility of certifying safety in *all* scenarios, these approaches [260] require further specifications that assign blame in the case of a crash. Such assignment of blame is impossible to completely characterize and relies on subjective notions of fault. Our risk-based framework allows one to circumvent this difficulty by only using a measure of safety that does not assign blame (*e.g.* TTC) and replacing the specifications that assign blame with a probabilistic notion of how likely the accident is. While this approach requires a learned model of the world P_0—a highly nontrivial statistical task in itself—the adaptive importance sampling techniques we employ can still efficiently identify dangerous scenarios even when P_0 is not completely accurate. Conceptually, we view verification and our framework as complementary; they form powerful tools that can evaluate safety *before* deploying a fleet for real-world testing.

Even given a consistent and complete notion of blame, verification remains highly intractable from a computational standpoint. Efficient algorithms only exist for restricted classes of systems in the domain of AVs, and they are fundamentally difficult to scale. Specifically, AVs—unlike previous successful applications of verification methods to application domains such as microprocessors [18]—include both continuous and discrete dynamics. This class of dynamics falls within the purview of hybrid systems [187], for which exhaustive verification is largely undecidable [127].

Verifying individual components of the perception pipeline, even as standalone systems, is a nascent, active area of research (see [11, 70, 23] and many others). Current subsystem verification techniques for deep neural networks [139, 152, 281] do not scale to state-of-the-art models and largely investigate the robustness of the network with respect to small perturbations of a single sample. There are two key assumptions in these works; the label of the input is unchanged within the radius of allowable perturbations, and the resulting expansion of the test set covers a meaningful portion of possible inputs to the network. Unfortunately, for realistic cases in AVs it is likely that perturbations to the state of the world which in turn generates an image *should* change the label. Furthermore, the combinatorial

nature of scenario configurations casts serious doubt on any claims of coverage.

In our risk-based framework, we replace the complex system specifications required for formal verification methods with a model P_0 that we learn via imitation-learning techniques. Generative adversarial imitation learning (GAIL) was first introduced by Ho and Ermon [131] as a way to directly learn policies from data and has since been applied to model human driving behavior by Kuefler et al. [166]. Model-based GAIL (MGAIL) is the specific variant of GAIL that we employ; introduced by Baram et al. [21], MGAIL's generative model is fully differentiable, allowing efficient model training with standard stochastic approximation methods.

The cross-entropy method was introduced by Rubinstein [246] and has attracted interest in many rare-event simulation scenarios [247, 165]. More broadly, it can be thought of as a model-based optimization method [135, 136, 137, 307, 138, 310]. With respect to assessing safety of AVs, the cross-entropy method has recently been applied in simple lane-changing and car-following scenarios in low dimensions [308]. Our work significantly extends this approach by implementing a photo-realistic simulator that can assess the deep-learning based perception pipeline along with the control framework. However, the cross-entropy method becomes brittle in high dimensions due to computations involving products of likelihood ratios, and it does not have guarantees for convergence to a good importance-sampling distribution. Chapter 5 considers the development of a novel rare-event simulation method that scales better with dimension and has rigorous convergence guarantees.

To summarize, a fundamental tradeoff emerges when comparing the requirements of our risk-based framework to other testing paradigms, such as real-world testing or formal verification. Real-world testing endangers the public but is still in some sense a gold standard. Verified subsystems provide evidence that the AV should drive safely even if the estimated distribution shifts, but verification techniques are limited by computational intractability as well as the need for both white-box models and the completeness of specifications that assign blame (*e.g.* [260]). In turn, our risk-based framework is most useful when the base distribution P_0 is accurate, but even when P_0 is misspecified, our adaptive importance sampling techniques can still efficiently identify dangerous scenarios, especially those that may be missed by verification methods assigning blame. Our framework offers significant speedups over real-world testing and allows efficient evaluation of black-box AV input/output behavior, providing a powerful tool to aid in the design of safe AVs.

Chapter 5

Neural bridge sampling for rare-event simulation

Chaos isn't a pit. Chaos is a ladder.

— Petyr Baelish, *Game of Thrones*

In the previous chapter, we introduced the risk-based framework, a formalism for testing safety-critical machine-learning systems. We also employed a basic technique—the cross-entropy method—to solve the rare-event simulation problem, the technical challenge underpinning the risk-based framework. In this chapter, we move beyond this basic technique and develop a novel rare-event simulation method that combines exploration, exploitation, and optimization techniques to find failure modes and estimate their rate of occurrence. We provide rigorous guarantees for the performance of our method in terms of both statistical and computational efficiency. Finally, we demonstrate the efficacy of our approach on a variety of scenarios, illustrating its usefulness as a tool for rapid sensitivity analysis and model comparison that are essential to developing and testing safety-critical autonomous systems.

5.1 Introduction

Data-driven and learning-based approaches have the potential to enable robots and autonomous systems that intelligently interact with unstructured environments. Unfortunately, evaluating the performance of the closed-loop system is challenging, limiting the

success of such methods in safety-critical settings. Even if we produce a deep reinforcement learning agent better than a human at driving, flying a plane, or performing surgery, we have no tractable way to certify the system's quality. Thus, currently deployed safety-critical autonomous systems are limited to structured environments that allow mechanisms such as PID control, simple verifiable protocols, or convex optimization to enable guarantees for properties like stability, consensus, or recursive feasibility (see *e.g.* [84, 206, 46]). The stylized settings of these problems and the limited expressivity of guaranteeable properties are barriers to solving unstructured, real-world tasks such as autonomous navigation, locomotion, and manipulation.

The goal of this chapter is to *efficiently* evaluate complex systems that lack safety guarantees and/or operate in unstructured environments. We assume access to a simulator to test the system's performance. Given a distribution $X \sim P_0$ of simulation parameters that describe typical environments for the system under test, our governing problem is to estimate the probability of an adverse event

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma). \tag{5.1}$$

The parameter γ is a threshold defining an adverse event, and $f : \mathcal{X} \to \mathbb{R}$ measures the safety of a realization x of the agent and environment (higher values are safer). In this chapter, we assume P_0 is known; the generative-modeling and system-identification literatures (*e.g.* [14, 115, 224]) provide several approaches to learn or specify P_0. A major challenge for solving problem (5.1) is that the better an agent is at performing a task (*i.e.* the smaller p_γ is), the harder it is to confidently estimate p_γ—one rarely observes events with $f(x) \leq \gamma$. For example, when P_0 is light-tailed, the sample complexity of estimating p_γ using naive Monte Carlo samples grows exponentially [56].

Problem (5.1) is often solved in practice by naive Monte Carlo estimation methods, the simplest of which *explore* the search space via random samples from P_0. These methods are unbiased and easy to parallelize, but they exhibit poor sample complexity. Naive Monte Carlo methods can be improved by adding an adaptive component *exploiting* the most informative portions of random samples drawn from a sequence of approximating distributions $P_0, P_1, \ldots, P_K$. However, standard adaptive Monte Carlo methods (*e.g.* [60]), though they may use first-order information on the distributions P_k themselves, fail to use first-order information about f to improve sampling; we explicitly leverage this to accelerate convergence

of the estimate through *optimization*.

Naive applications of first-order optimization methods in the estimation problem (5.1)—for example biasing a sample in the direction $-\nabla f(x)$ to decrease $f(x)$—also require second-order information to correct for the distortion of measure that such transformations induce. Consider the change of variables formula for distributions $\rho(y) = \rho(g^{-1}(y)) \cdot |\det J_{g^{-1}}(y)|$ where $y = g(x)$. When $g(x)$ is a function of the gradient $\nabla f(x)$, the volume distortion $|\det J_{g^{-1}}(y)|$ is a function of the Hessian $\nabla^2 f(x)$. Hessian computation, if even defined, is unacceptably expensive for high-dimensional spaces $\mathcal{X}$ and/or simulations that involve the time-evolution of a dynamical system; our approach avoids any Hessian computation. In contrast, gradients $\nabla f(x)$ can be efficiently computed for many closed-loop systems [1, 222, 306, 177] or through the use of surrogate methods [301, 75, 21, 88].

To that end, we propose *neural bridge sampling*, a technique that combines *exploration, exploitation*, and *optimization* to efficiently solve the estimation problem (5.1). Specifically, we consider a novel Markov chain Monte Carlo (MCMC) scheme that moves along an adaptive ladder of intermediate distributions P_k (with corresponding unnormalized densities $\rho_k(x)$ and normalizing constants $Z_k := \int_{\mathcal{X}} \rho_k(x)dx$). This MCMC scheme iteratively transforms the base distribution P_0 to the distribution of interest $P_0 I\{f(x) \leq \gamma\}$. Neural bridge sampling adaptively balances exploration in the search space (via $\nabla \log \rho_0$) against optimization (via ∇f), while avoiding Hessian computations. Our final estimate $\hat{p}_\gamma$ is a function of the ratios Z_k/Z_{k-1} of the intermediate distributions P_k, the so-called "bridges" [30, 198]. We accurately estimate these ratios by warping the space between the distributions P_k using neural density estimation.

Contributions and outline Section 5.2 presents our method, while Section 5.3 provides guarantees for its statistical performance and overall efficiency. A major focus of this work is empirical, and accordingly, Section 5.4 empirically demonstrates the superiority of neural bridge sampling over competing techniques in a variety of applications: (i) we evaluate the sensitivity of a formally-verified system to domain shift, (ii) we consider design optimization for high-precision rockets, and (iii) we perform model comparisons for two learning-based approaches to autonomous navigation.

5.1.1 Related Work

Safety evaluation Several communities [71] have attempted to evaluate the closed-loop performance of cyber-physical, robotic, and embodied agents both with and without learning-based components. Existing solutions are predicated on the definition of the evaluation problem: verification, falsification, or estimation. In this chapter we consider a method that utilizes interactions with a gradient oracle in order to solve the estimation problem (5.1). In contrast to our approach, the verification community has developed tools (*e.g.* [163, 65, 6]) to investigate whether any adverse or unsafe executions of the system exist. Such methods can certify that failures are impossible, but they require that the model is written in a formal language (a barrier for realistic systems) and they require whitebox access to this formal model. Falsification approaches (*e.g.* [94, 80, 9, 312, 85, 231]) attempt to find *any* failure cases for the system (but not the overall probability of failure). Similar to our approach, some falsification approaches (*e.g.* [1, 306]) utilize gradient information, but their goal is to simply minimize $f(x)$ rather than solve problem (5.1). Adversarial machine learning is closely related to falsification; the key difference is the domain over which the search for falsifying evidence is conducted. Adversarial examples (*e.g.* [152, 188, 265, 281]) are typically restricted to an p-norm ball around a point from a dataset, whereas falsification considers all possible in-distribution examples. Both verification and falsification methods provide less information about the system under test than estimation-based methods: they return only whether or not the system satisfies a specification. When the system operates in an unstructured environment (*e.g.* driving in an urban setting), the mere existence of failures is trivial to demonstrate [260]. Several authors (*e.g.* [218, 298]) have proposed that it is more important in such settings to understand the overall frequency of failures as well as the relative likelihoods of different failure modes, motivating our approach.

Sampling techniques and density estimation When sampling rare events and estimating their probability, there are two main branches of related work: parametric adaptive importance sampling (AIS) [193, 216] and nonparametric sequential Monte Carlo (SMC) techniques [82, 77]. Both of these literatures are advanced forms of variance reduction techniques, and they are complementary to standard methods such as control variates [248, 129]. Parametric AIS techniques, such as the cross-entropy method [247], postulate a family of distributions for the optimal importance-sampling distribution, and they iteratively perform heuristic optimization problems to update the sampling distribution. SMC techniques

perform sampling from a sequence of probability distributions defined nonparametrically by the samples themselves. The SMC formalism encompasses particle filters, birth-death processes, and smoothing filters [76]. Our technique blends aspects of both of these communities: we include parametric warping distributions in the form of normalizing flows [224] within the SMC setting.

Our method employs bridge sampling [30, 198], which is closely related to other SMC techniques such as umbrella sampling [63], multilevel splitting [50, 60], and path sampling [108]. The operational difference between these methods is in the form of the intermediate distribution used to calculate the ratio of normalizing constants. Namely, the optimal umbrella sampling distribution is more brittle than that of bridge sampling [63]. Multilevel splitting employs hard barriers through indicator functions, whereas our approach relaxes these hard barriers with smoother exponential barriers. Path sampling generalizes bridge sampling by taking discrete bridges to a continuous limit; this approach is difficult to implement in an adaptive fashion.

The accuracy of bridge sampling depends on the overlap between intermediate distributions P_k. Simply increasing the number of intermediate distributions is inefficient, because it requires running more simulations. Instead, we employ a technique known as *warping*, where we map intermediate distributions to a common reference distribution [293, 197]. Specifically, we use normalizing flows [237, 158, 223, 224], which efficiently transform arbitrary distributions to standard Gaussians through a series of deterministic, invertible functions. Normalizing flows are typically used for probabilistic modeling, variational inference, and representation learning. Recently, Hoffman et al. [132] explored the benefits of using normalizing flows for reparametrizing distributions within MCMC; our warping technique encompasses this benefit and extends it to the SMC setting.

Beyond simulation This chapter assumes that the generative model of the operating domain P_0 is given, so all failures are in the modeled domain by definition. When deploying systems in the real world, anomaly detection [61] to discover distribution shifts is complementary to our approach (see *e.g.* Choi et al. [68], Nachman and Shih [204]). Another way to frame that problem is via distributional robustness [93, 207, 234], where we analyze the worst-case probability of failure under an uncertainty set composed of perturbations to P_0.

5.2 Proposed approach

As we note in Section 5.1, naive Monte Carlo measures probabilities of rare events inefficiently. Instead, we consider a sequential Monte Carlo approach: we decompose the rare-event probability p_γ into a chain of intermediate quantities, each of which is tractable to compute with standard Monte Carlo methods. Specifically, consider K distributions P_k with corresponding (unnormalized) probability densities ρ_k and normalizing constants $Z_k := \int_{\mathcal{X}} \rho_k(x)dx$. Let ρ_0 correspond to the density for P_0 and $\rho_\infty(x) := \rho_0(x)I\{f(x) \leq \gamma\}$ be the (unnormalized) conditional density for the region of interest. Then, we consider the following decomposition:

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma) = \mathbb{E}_{P_K}\left[\frac{Z_K}{Z_0}\frac{\rho_\infty(X)}{\rho_K(X)}\right], \qquad \frac{Z_K}{Z_0} = \prod_{k=1}^{K}\frac{Z_k}{Z_{k-1}}. \tag{5.2}$$

Although we are free to choose the intermediate distributions arbitrarily, we will show below that our estimate for each ratio Z_k/Z_{k-1} and thus p_γ is accurate insofar as the distributions sufficiently overlap (a concept we make rigorous in Section 5.3). Thus, the intermediate distributions act as bridges that iteratively steer samples from P_0 towards P_K. One special case is the multilevel splitting approach [148, 50, 298, 215], where $\rho_k(x) := \rho_0(x)I\{f(x) \leq L_k\}$ for levels $\infty =: L_0 > L_1 \ldots > L_K := \gamma$. In this work, we introduce an exponential tilting barrier [262]

$$\rho_k(x) := \rho_0(x)\exp\left(\beta_k\left[\gamma - f(x)\right]_-\right), \tag{5.3}$$

which allows us to take advantage of gradients $\nabla f(x)$. Here we use the "negative ReLU" function defined as $[x]_- := -[-x]_+ = xI\{x < 0\}$. We set $\beta_0 := 0$ and adaptively choose $\beta_k > \beta_{k-1}$. The parameter β_k tilts the distribution towards the distribution of interest: $\rho_k \to \rho_\infty$ as $\beta_k \to \infty$. In what follows, we describe an MCMC method that combines exploration, exploitation, and optimization to draw samples $X_i^k \sim P_k$. We then show how to compute the ratios Z_k/Z_{k-1} given samples from both P_{k-1} and P_k. Finally, we describe an adaptive way to choose the intermediate distributions P_k. Algorithm 5.1 summarizes the overall approach.

MCMC with an exponential barrier Gradient-based MCMC techniques such as the Metropolis-adjusted Langevin algorithm (MALA) [244, 239] or Hamiltonian Monte Carlo (HMC) [86, 211] use gradients $\nabla \log \rho_0(x)$ to efficiently explore the space $\mathcal{X}$ and avoid

Algorithm 5.1 Neural bridge sampling

Input: N samples $x_i^0 \overset{\text{i.i.d.}}{\sim} P_0$, MCMC steps T, step size $\alpha \in (0,1)$, stop condition $s \in (0,1)$
Initialize $k \leftarrow 0$, $\beta_0 \leftarrow 0$, $\log(\hat{p}_\gamma) \leftarrow 0$
while $\frac{1}{N}\sum_i I\{f(x_i^k) \leq \gamma\} < s$ **do**
  $\beta_{k+1} \leftarrow$ solve problem (5.8)
  for $i = 1$ to N, in parallel
    $x_i^{k+1} \overset{\text{i.i.d.}}{\sim} \text{Mult}(\{\rho_{k+1}(x_i^k)/\rho_k(x_i^k)\})$ // multinomial resampling
  for $t = 1$ to T
    for $i = 1$ to N, in parallel
      $x_i^{k+1} \leftarrow \text{WarpedHMC}(x_i^k, \theta_k)$ // Appendix D.1
  $\theta_{k+1} \leftarrow \text{argmin}$ problem (5.6) // train normalizing flow on $\{x_i^{k+1}\}$ via SGD
  $\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(Z_{k+1}/Z_k)$ // warped bridge estimate (5.5)
  $k \leftarrow k+1$
$\log(\hat{p}_\gamma) \leftarrow \log(\hat{p}_\gamma) + \log(\frac{1}{N}\sum_i I\{f(x_i^k) \leq \gamma\})$

inefficient random-walk behavior [90, 67]. Classical mechanics inspires the HMC approach: HMC introduces an auxiliary random momentum variable $v \in \mathcal{V}$ and generates proposals by performing Hamiltonian dynamics in the augmented state-space $\mathcal{X} \times \mathcal{V}$. These dynamics conserve volume in the augmented state-space, even when performed with discrete time steps [176].

By including the barrier $\exp\left(\beta_k \left[\gamma - f(x)\right]_-\right)$, we combine exploration with optimization; the magnitude of β_k in the barrier modulates the importance of ∇f (optimization) over $\nabla \log \rho_0$ (exploration), two elements of the HMC proposal (see Appendix D.1 for details). We discuss the adaptive choice for β_k below. Most importantly, we avoid any need for Hessian computation because the dynamics conserve volume. As Algorithm 5.1 shows, we perform MCMC as follows: given N samples $x_i^{k-1} \sim P_{k-1}$ and a threshold β_k, we first resample using their importance weights (exploiting the performance of samples that have lower function value than others) and then perform T HMC steps. In this work, we implement split HMC [259] which is convenient for dealing with the decomposition of $\log \rho_k(x)$ into $\log \rho_0(x) + \beta_k[\gamma - f(x)]_-$ (see Appendix D.1 for details).

Estimating Z_k/Z_{k-1} via bridge sampling Bridge sampling [30, 198] allows estimating the ratio of normalizing constants of two distributions by rewriting

$$E_k := \frac{Z_k}{Z_{k-1}} = \frac{Z_k^B/Z_{k-1}}{Z_k^B/Z_k} = \frac{\mathbb{E}_{P_{k-1}}[\rho_k^B(X)/\rho_{k-1}(X)]}{\mathbb{E}_{P_k}[\rho_k^B(X)/\rho_k(X)]}, \qquad \widehat{E}_k = \frac{\sum_{i=1}^N \rho_k^B(x_i^{k-1})/\rho_{k-1}(x_i^{k-1})}{\sum_{i=1}^N \rho_k^B(x_i^k)/\rho_k(x_i^k)}, \tag{5.4}$$

where ρ_k^B is the density for a bridge distribution between P_{k-1} and P_k, and Z_k^B is its associated normalizing constant. We employ the geometric bridge $\rho_k^B(x) := \sqrt{\rho_{k-1}(x)\rho_k(x)}$.

In addition to being simple to compute, bridge sampling with a geometric bridge enjoys the asymptotic performance guarantee that the relative mean-square error scales inversely with the Bhattacharyya coefficient, $G(P_{k-1}, P_k) = \int_{\mathcal{X}} \sqrt{\frac{\rho_{k-1}(x)}{Z_{k-1}} \frac{\rho_k(x)}{Z_k}} dx \in [0, 1]$ (see Appendix D.2.1 for a proof). This value is closely related to the Hellinger distance, $H(P_{k-1}, P_k) = \sqrt{2 - 2G(P_{k-1}, P_k)}$. In Section 5.3, we analyze the ramifications of this fact on the overall convergence of our method.

Neural warping Both HMC and bridge sampling benefit from warping samples x_i into a different space. As Betancourt [35] notes, HMC mixes poorly in spaces with ill-conditioned geometries. Girolami and Calderhead [112] and Hoffman et al. [132] explore techniques to improve mixing efficiency by minimizing shear in the corresponding Hamiltonian dynamics. One way to do so is to transform to a space that resembles a standard isotropic Gaussian [191].

Conveniently, transforming P_k to a common distribution (*e.g.* a Gaussian) also benefits the bridge-sampling estimator (5.4). As noted above, the error of the bridge estimator grows with the Hellinger distance between the distributions $H(P_{k-1}, P_k)$. However, normalizing constants Z_k are invariant to (invertible) transformations. Thus, transformations that warp the space between distributions reduce the error of the bridge-sampling estimator (5.4). Concretely, we consider invertible transformations W_k such that $y_i^k = W_k(x_i^k)$. For clarity of notation, we write probability densities over the space $\mathcal{Y}$ as ϕ, the corresponding distributions for Y^k as Q_k, and the inverse transformations $W_k^{-1}(y)$ as $V_k(y)$. Then we can write the bridge-sampling estimate (5.4) in terms of the transformed variables y. The numerator and denominator are as follows:

$$\mathbb{E}_{Q_{k-1}}\left[\frac{\phi_k^B(Y)}{\phi_{k-1}(Y)}\right] = \mathbb{E}_{Q_{k-1}}\left[\sqrt{\frac{\phi_k(Y)}{\phi_{k-1}(Y)}}\right] = \mathbb{E}_{Q_{k-1}}\left[\sqrt{\frac{\rho_k(V_k(Y))|\det J_{V_k}(Y)|}{\rho_{k-1}(V_{k-1}(Y))|\det J_{V_{k-1}}(Y)|}}\right], \tag{5.5a}$$

$$\mathbb{E}_{Q_k}\left[\frac{\phi_k^B(Y)}{\phi_k(Y)}\right] = \mathbb{E}_{Q_k}\left[\sqrt{\frac{\phi_{k-1}(Y)}{\phi_k(Y)}}\right] = \mathbb{E}_{Q_k}\left[\sqrt{\frac{\rho_{k-1}(V_{k-1}(Y))|\det J_{V_{k-1}}(Y)|}{\rho_k(V_k(Y))|\det J_{V_k}(Y)|}}\right]. \tag{5.5b}$$

By transforming all P_k into Q_k to resemble standard Gaussians, we reduce the Hellinger distance $H(Q_{k-1}, Q_k) \leq H(P_{k-1}, P_k)$. Note that the volume distortions in the expression (5.5) are functions of the transformation V_k, so they do not require computation of the Hessian $\nabla^2 f$. However, computing $\rho_k(V_k(y))$ requires evaluations of f (*e.g.* calls of the simulator). We consider the cost-benefit analysis of warping in Section 5.3.

Classical warping techniques include simple mean shifts or affine scaling [293, 197].

Similar to Hoffman et al. [132], we consider normalizing flows, a much more expressive class of transformations that have efficient Jacobian computations [224]. Specifically, given samples x_i^k, we train masked autoregressive flows (MAFs) [223] to minimize the empirical KL divergence between the transformed samples y_i^k and a standard Gaussian $D_{\mathrm{KL}}(Q_k \| \mathcal{N}(0, I))$. Parametrizing W_k by θ_k, this minimization problem is equivalent to:

$$\text{minimize}_\theta \sum_{i=1}^{N} -\log \left| \det J_{W_k}\left(x_i^k; \theta\right) \right| + \frac{1}{2} \left\| W_k\left(x_i^k; \theta\right) \right\|_2^2. \tag{5.6}$$

The KL divergence is an upper bound to the Hellinger distance; we found minimizing the former to be more stable than minimizing the latter. Furthermore, to improve training efficiency, we exploit the iterated nature of the problem and warm-start the weights θ_k with the trained values θ_{k-1} when solving problem (5.6) via stochastic gradient descent (SGD). As a side benefit, the trained flows can be repurposed as importance-samplers for the ladder of distributions from nominal behavior to failure.

Adaptive intermediate distributions Because we assume no prior knowledge of the system under test, we exploit previous progress to choose the intermediate β_k online; this is a key difference to our approach compared to other forms of sequential Monte Carlo (*e.g.* [209, 210]) which require a predetermined schedule for β_k. We define the quantities

$$a_k := \textstyle\sum_i^N I\{f(x_i^k) \leq \gamma\}/N, \quad b_k(\beta) := \sum_{i=1}^N \exp\left((\beta - \beta_k)[\gamma - f(x_i^k)]_-\right)/N. \tag{5.7}$$

The first is the fraction of samples that have achieved the threshold. The second is an importance-sampling estimate of E_{k+1} given samples $x_i^k \sim P_k$, written as a function of β. For fixed fractions $\alpha, s \in (0, 1)$ with $\alpha < s$, β_{k+1} solves the following optimization problem:

$$\text{maximize } \beta \text{ s.t. } \{b_k(\beta) \geq \alpha, \quad a_k / b_k(\beta) \leq s\}. \tag{5.8}$$

Since $b_k(\beta)$ is monotonically decreasing and $b_k(\beta) \geq a_k$, this problem can be solved efficiently via binary search. The constant α tunes how quickly we enter the tails of P_0 (smaller α means fewer iterations), whereas s is a stop condition for the last iteration. Choosing β_{k+1} via (5.8) yields a crude estimate for the ratio Z_{k+1}/Z_k as α (or a_{K-1}/s for the last iteration). The bridge-sampling estimate $\widehat{E}_{k+1}$ corrects this crude estimate once we have samples from the next distribution P_{k+1}.

5.3 Performance analysis

We can write the empirical estimator of the function (5.2) as

$$\hat{p}_\gamma = \prod_{k=1}^{K} \widehat{E}_k \frac{1}{N} \sum_{i=1}^{N} \frac{\rho_\infty(x_i^K)}{\rho_K(x_i^K)}, \tag{5.9}$$

where $\widehat{E}_k$ is given by the expression (5.4) without warping, or similarly, as a Monte Carlo estimate of the expression (5.5) with warping. We provide guarantees for both the time complexity of running Algorithm 5.1 (*i.e.* the iterations K) as well as the overall mean-square error of $\hat{p}_\gamma$. For simplicity, we provide results for the asymptotic (large N) and well-mixed MCMC (large T) limits. Assuming these conditions, we have the following:

Proposition 5.1. *Let $K_0 := \lfloor \log(p_\gamma)/\log(\alpha) \rfloor$. Then, for large N and T, $s \geq 1/3$, and $p_\gamma < s$, the total number of iterations in Algorithm 5.1 approaches $K \overset{\text{a.s.}}{\to} K_0 + I\{p_\gamma/\alpha^{K_0} < s\}$. Furthermore, for the non-warped estimator, the asymptotic relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ is*

$$\frac{2}{N}\sum_{k=1}^{K}\left(\frac{1}{G(P_{k-1},P_k)^2}-1\right) - \frac{2}{N}\sum_{k=1}^{K-1}\left(\frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})}-1\right) + \frac{1-s}{sN} + o\left(\frac{1}{N}\right). \tag{5.10}$$

In particular, if the inverse Bhattacharyya coefficients are bounded such that $\frac{1}{G(P_{k-1},P_k)^2} \leq D$ (with $D \geq 1$), then the asymptotic relative mean-square error satisfies $\mathbb{E}\left[\left(\frac{\hat{p}_\gamma}{p_\gamma}-1\right)^2\right] \leq 2KD/N$. For the warped estimator, replace $G(P_i, P_j)$ with $G(Q_i, Q_j)$ in the expression (5.10).

See Appendix D.2.1 for the proof. We provide some remarks about the above result. Intuitively, the first term in the bound (5.10) accounts for the variance of $\widehat{E}_k$. The denominator of $\widehat{E}_{k-1}$ and numerator of $\widehat{E}_k$ both depend on x_i^k; the second sum in (5.10) accounts for the covariance between those terms. Furthermore, the quantities in the bound (5.10) are all empirically estimable. In particular,

$$G(P_{k-1},P_k)^2 = \frac{Z_k^B}{Z_{k-1}}\frac{Z_k^B}{Z_k}, \qquad \frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})} = \frac{Z_k^C}{Z_k}\frac{Z_k}{Z_k^B}\frac{Z_k}{Z_{k+1}^B}, \tag{5.11}$$

where $Z_k^C/Z_k = \mathbb{E}_{P_k}\left[\rho_k^B(X)\rho_{k+1}^B(X)/\rho_k(X)^2\right]$. The last term in the bound (5.10) is the relative variance of the final Monte Carlo estimate $\sum_i I\{f(x_i^K) \leq \gamma\}/N$.

Overall efficiency The statistical efficiency outlined in Proposition 5.1 is pointless if it is accompanied by an overwhelming computational cost. We take the atomic unit of computation to be a query of the simulator, which returns both evaluations of $f(x)$ and $\nabla f(x)$; we assume other computations to be negligible compared to simulation. As such, the cost of Algorithm 5.1 is $N(1+KT)$ evaluations of the simulator without warping and $N(1+KT)+2KN$ with warping. Thus, the relative burden of warping is minimal, because training the normalizing flows to minimize $D_{\mathrm{KL}}(Q_k \| \mathcal{N}(0, I))$ requires no extra simulations. In contrast, directly minimizing $D_{\mathrm{KL}}(Q_{k-1} \| Q_k)$ would require extra simulations at each training step to evaluate $\rho_k(V_k(y))$.

Our method can exploit two further sources of efficiency. First, we can employ surrogate models for gradient computation and/or function evaluation during the T MCMC steps. For example, using a surrogate model for a fraction $d \leq 1 - 1/T$ of the MCMC iterations reduces the factor T to $T_s := (1-d)T$ in the overall cost. Surrogate models have an added benefit of making our approach amenable for simulators that do not provide gradients. The second source of efficiency is parallel computation. Given C processors, the factor N in the cost drops to $N_c := \lceil N/C \rceil$.

The overall efficiency of the estimator (5.9)–relative error multiplied by cost [119]–depends on p_γ as $\log(p_\gamma)^2$. In contrast, the standard Monte Carlo estimator has cost N to produce an estimate with relative error $\frac{1-p_\gamma}{p_\gamma N}$. Thus, the relative efficiency gain for our estimator (5.9) over naive Monte Carlo is $O(1/(p_\gamma \log(p_\gamma)^2))$: the efficiency gains over naive Monte Carlo increase as p_γ decreases.

5.4 Experiments

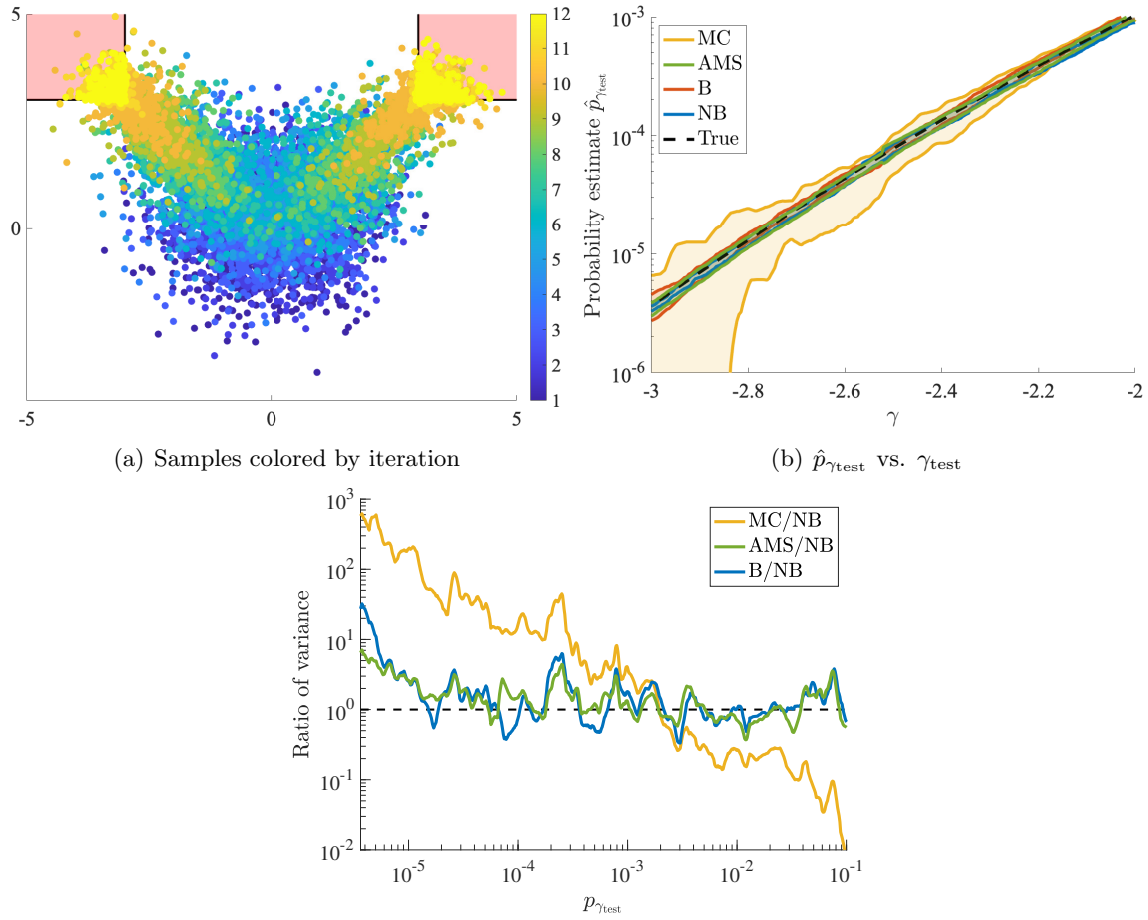
We evaluate our approach on a variety of scenarios showcasing its use in efficiently evaluating the safety of autonomous systems. We begin with a synthetic problem to illustrate the methodology concretely as well as highlight the pitfalls of using gradients naively. Then, we evaluate a formally-verified neural network controller [141] on the OpenAI Gym continuous MountainCar environment [201, 51] under a domain perturbation. Finally, we consider two examples of using neural bridge sampling as a tool for engineering design in high-dimensional settings: (a) comparing thruster sizes to safely land a rocket [38] in the presence of wind, and (b) comparing two algorithms on the OpenAI Gym CarRacing environment (which requires a surrogate model for gradients) [160].

We compare our method with naive Monte Carlo (MC) and perform ablation studies for the effects of neural warping (denoted as NB with warping and B without). We also provide comparisons with adaptive multilevel splitting (AMS) [50, 298, 215]. All methods are given the same computational budget as measured by evaluations of the simulator. This varies from 50,000-100,000 queries to run Algorithm 5.1 as determined by p_γ (see Appendix D.3 for details of each experiment's hyperparameters). However, despite running Algorithm 5.1 with a given γ, we evaluate estimates $\hat{p}_{\gamma_{\text{test}}}$ for all $\gamma_{\text{test}} \geq \gamma$. Larger γ_{test} require fewer queries to evaluate $\hat{p}_{\gamma_{\text{test}}}$ (as Algorithm 5.1 terminates early). Thus, we adjust the number of MC queries accordingly for each γ_{test}. Independently, we calculate the ground-truth values $p_{\gamma_{\text{test}}}$ for non-synthetic problems using a fixed, very large number of MC queries.

Synthetic problem We consider the two-dimensional function $f(x) = -\min(|x_{[1]}|, x_{[2]})$, where $x_{[i]}$ is the i^{th} dimension of $x \in \mathbb{R}^2$. We let $\gamma = -3$ and $P_0 = \mathcal{N}(0, I)$ (for which $p_\gamma = 3.6 \cdot 10^{-6}$). Note that $\nabla^2 f(x) = 0$ almost everywhere, yet $\nabla f(x)$ has negative divergence in the neighborhoods of $x_{[2]} = |x_{[1]}|$. Indeed, gradient descent collapses $x_i \sim P_0$ to the lines $x_{[2]} = |x_{[1]}|$, and the ill-defined nature of the Hessian makes it unsuitable to track volume distortions. Thus, simple gradient-based transformations used to find adversarial examples (*e.g.* minimize $f(x)$) should not be used for estimation in the presence of non-smooth functions, unless volume distortions can be quantified.

Figure 5.1(a) shows the region of interest in pink and illustrates the gradual warping of ρ_0 towards ρ_∞ over iterations of Algorithm 5.1. Figures 5.1(b) and 5.1(c) indicate that all adaptive methods outperform MC for $p_{\gamma_{\text{test}}} < 10^{-3}$. For larger $p_{\gamma_{\text{test}}}$, the overhead of the adaptive methods renders MC more efficient (Figure 5.1(c)). The linear trend of the yellow MC/NB line in Figure 5.1(c) aligns with the theoretical efficiency gain discussed in Section 5.3. Finally, due to the simplicity of the search space and the landscape of $f(x)$, the benefits of gradients and warping are not drastic. Specifically, as shown in Figure 5.1(c), all adaptive methods have similar confidence in their estimates except at very small $p_{\gamma_{\text{test}}} < 10^{-5}$, where NB outperforms AMS and B. The next example showcases the benefits of gradients as well as neural warping in a more complicated search space.

Sensitivity of a formally-verified controller under domain perturbation We consider a minimal reinforcement learning task, the MountainCar problem [201] (Figure 5.2(a)). Ivanov et al. [141] created a formally-verified neural network controller to achieve



(a) Samples colored by iteration

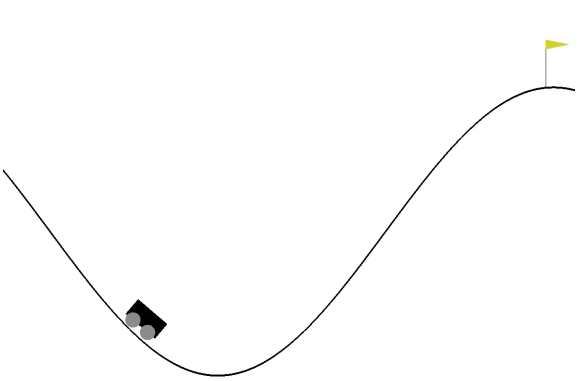
(b) $\hat{p}_{\gamma_{\text{test}}}$ vs. γ_{test}

(c) Ratio of variance vs. $p_{\gamma_{\text{test}}}$

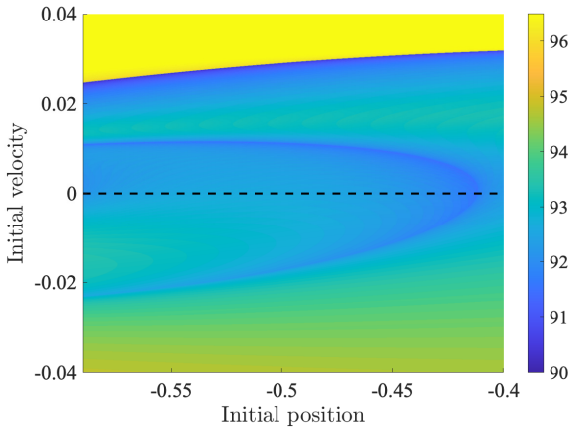
Figure 5.1: Experiments on a synthetic problem. 10 trials are used to calculate the 99% confidence intervals in (b) and variance ratios in (c). All adaptive methods perform similarly in this well-conditioned search space except at very small γ, where NB performs the best.

reward > 90 over all initial positions $\in [-0.59, -0.4]$ and 0 initial velocity (see Appendix D.3). The guarantees of formal verification hold only with respect to the specified domain; even small domain perturbations can affect system performance [142]. We illustrate this sensitivity by adding a small perturbation to the initial velocity $\sim \mathcal{N}(0, 10^{-4})$ and seek $p_\gamma := \mathbb{P}_0(\text{reward} \leq 90)$ for $P_0 = \text{Unif}(-0.59, -0.4) \times \mathcal{N}(0, 10^{-4})$. We measure the ground-truth failure rate as $p_\gamma = 1.6 \cdot 10^{-5}$ using 50 million naive Monte Carlo samples.

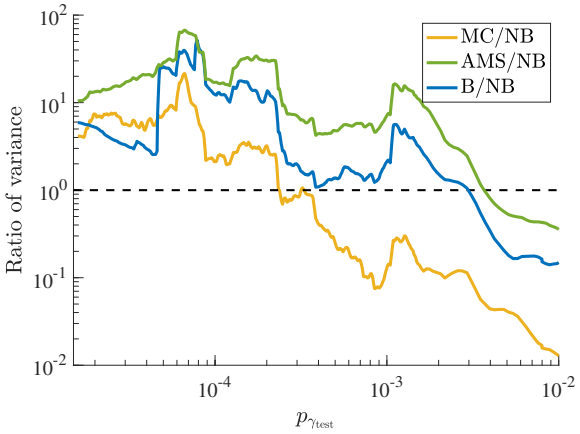
Figure 5.2(b) shows contours of $f(x)$. Notably, the failure region (dark blue) is an extremely irregular geometry with pathological curvature, which renders MCMC difficult for AMS and B [35]. Quantitatively, poor mixing adversely affects the performance of AMS



(a) The environment



(b) Contours of $f(x)$

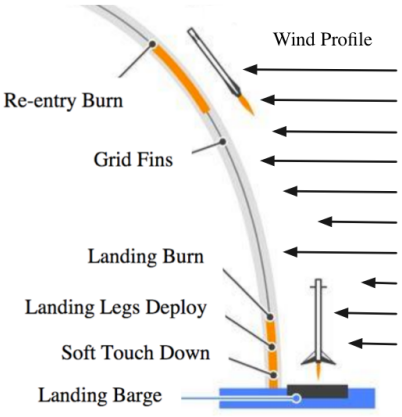


(c) Ratio of variance vs. $p_{\gamma_{\text{test}}}$

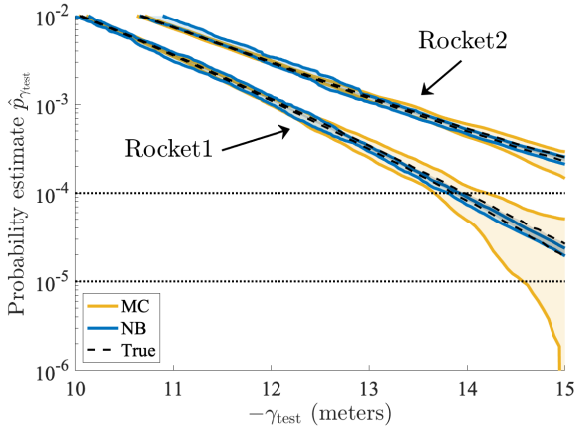
Figure 5.2: Experiments on the MountainCar environment. The dashed horizontal line in (b) is the line along which the controller is formally verified. 10 trials are used for the variance ratios in (c). The irregular geometry degrades performance of AMS and B, but B benefits slightly from gradients over AMS. NB uses gradients and neural warping to outperform all other techniques.

and B, and they perform even worse than MC (Figure 5.2(c)). Whereas gradients help B slightly over AMS, gradients and neural warping together help NB outperform all other methods. We next move to higher-dimensional systems.

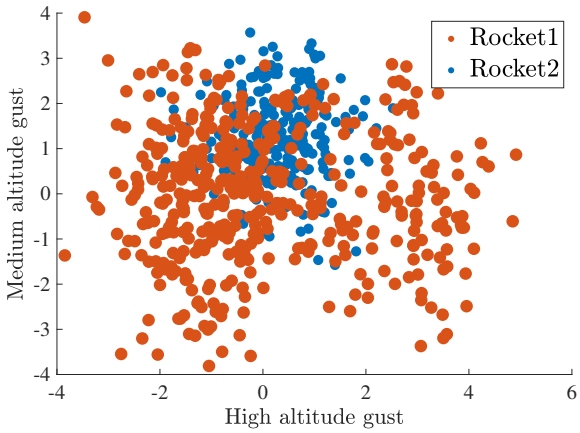
Rocket design We now consider the problem of autonomous, high-precision vertical landing of an orbital-class rocket (Figure 5.3(a)), a technology first demonstrated by SpaceX in 2015. Rigorous system-evaluation techniques such as our risk-based framework are powerful tools for quickly exploring design tradeoffs. In this experiment, the amount of thrust which



(a) Rocket landing scenario



(b) Failure rates for 2 rocket designs



(c) Failure modes

Figure 5.3: Rocket design experiments. NB's high-confidence estimates enable quick design iterations to either increase the landing pad radius or consider a third rocket that fails with probability $< 10^{-5}$. Low-dimensional visualization shows that Rocket2's failure types are more concentrated than those of Rocket1, even though Rocket2 has a higher overall probability of failure.

the rocket is capable of deploying to land safely must be balanced against the payload it is able to carry to space; stronger thrust increases safety but decreases payloads. We consider two rocket designs and we evaluate their respective probabilities of failure (not landing safely on the landing pad) for landing pad sizes up to 15 meters in radius. That is, $-f(x)$ is the distance from the landing pad's center at touchdown and $\gamma = -15$. We evaluate whether the rockets perform better than a threshold failure rate of 10^{-5}.

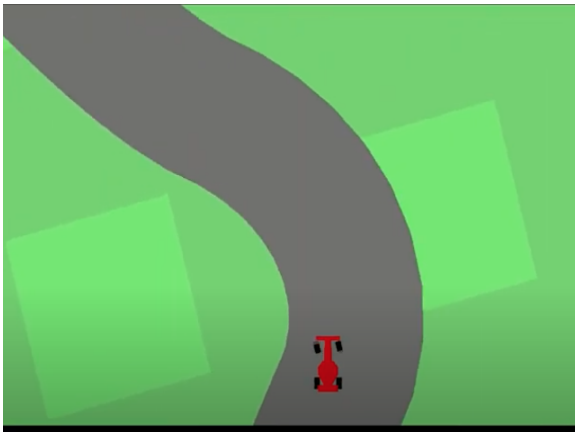
We let P_0 be the 100-dimensional search space parametrizing the sequence of wind-gusts during the rocket's flight. Appendix D.3 contains details for this parametrization

and the closed-loop simulation of the rocket's control law (based on industry-standard approaches [38, 238]). Figure 5.3(b) shows the estimated performance of the two rockets. We show only MC and NB for clarity; comparisons with other methods are in Table 5.1 (with ground-truth values calculated using 50 million naive Monte Carlo simulations). Whereas both NB and MC confidently estimate Rocket2's failure rate as higher than 10^{-4}, only NB confidently estimates Rocket1's failure rate as higher than 10^{-5}, letting engineers quickly judge whether to increase the size of the landing pad or build a better rocket.

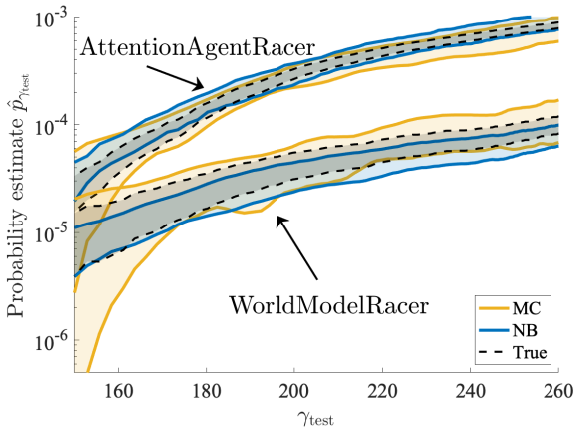
We can also distinguish between the modes of failure for the rockets. Namely, Figure 5.3(c) shows a PCA projection of failures (with $\gamma_{\text{test}} = -15$) onto 2 dimensions. Analysis of the PCA modes indicates that failures are dominated by high altitude and medium altitude gusts. Even though Rocket2 has a higher probability of failure, its failure mode is more concentrated than Rocket1's failures.

Car racing The CarRacing environment (Figure 5.4(a)) is a challenging reinforcement-learning task with a continuous action space and pixel observations. Similar observation spaces have been proposed for real autonomous vehicles (*e.g.* [19, 186, 295]). We compare two recent approaches, AttentionAgentRacer [279] and WorldModelRacer [118] that have similar average performance: they achieve average rewards of 903 ± 49 and 899 ± 46 respectively (mean $\pm$ standard deviation over 2 million trials). Both systems utilize one or more deep neural networks to plan in image-space, so neither has performance guarantees. We evaluate the probability of getting small rewards ($\gamma = 150$).

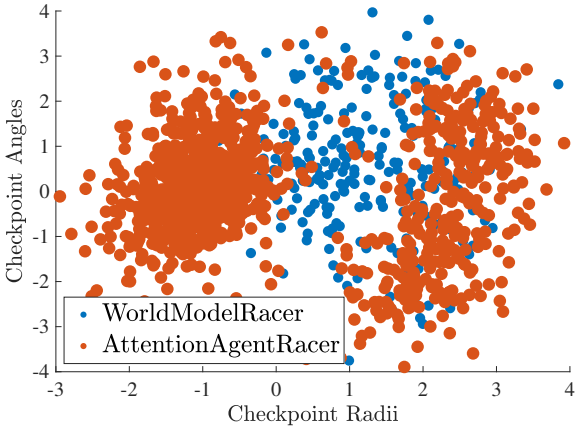
The 24-dimensional search space P_0 parametrizes the generation of the racing track (details are in Appendix D.3). This environment does not easily provide gradients due to presence of a rendering engine in the simulation loop. Instead, we fit a Gaussian process surrogate model to compute $\nabla f(x)$ (see Appendix D.3). As these experiments are extremely expensive (taking up to 1 minute per simulation), we only use 2 million naive Monte Carlo samples to compute the ground-truth failure rates. Figure 5.4(b) shows that, even though the two models have very similar average performance, their catastrophic failure curves are distinct. Furthermore, MC is unable to distinguish between the policies below rewards of 160 due to its high uncertainty, whereas NB clearly shows that WorldModelRacer is superior. Note that, because even the ground-truth has non-negligible uncertainty with 2 million samples, we only report the variance component of relative mean-square error in Table 5.1.



(a) The environment



(b) Failure rates for racecars



(c) Failure modes

Figure 5.4: CarRacing experiments. MC cannot distinguish between the policies below $\gamma_{\text{test}} = 160$. NB's high-confidence estimates enable model comparisons at extreme limits of failure. Low-dimensional visualization of the failure modes shows that the algorithms fail in distinct ways.

As with the rocket design experiments, we visualize the modes of failure (defined by $\gamma_{\text{test}} = 225$) via PCA in Figure 5.4(c). The dominant eigenvectors involve large differentials between radii and angles of consecutive checkpoints that are used to generate the racing tracks. AttentionAgentRacer has two distinct modes of failure, whereas WorldModelRacer has a single mode.

Table 5.1: Relative mean-square error $\mathbb{E}[(\hat{p}_\gamma/p_\gamma - 1)^2]$ over 10 trials

	Synthetic	MountainCar	Rocket1	Rocket2	AttentionAgentRacer	WorldModelRacer
MC	1.1821	0.2410	1.1039	0.0865	1.0866	0.9508
AMS	0.0162	0.5424	0.0325	0.0151	1.0211	0.8177
B	0.0514	0.3856	0.0129	0.0323	0.9030	0.7837
NB	**0.0051**	**0.0945**	**0.0102**	**0.0078**	**0.2285**	**0.1218**
p_γ	$3.6 \cdot 10^{-6}$	$1.6 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$\approx 2.5 \cdot 10^{-5}$	$\approx 9.5 \cdot 10^{-6}$

5.5 Discussion

There is a growing need for rigorous evaluation of safety-critical systems which contain components without formal guarantees (*e.g.* deep neural networks). Scalably evaluating the safety of such systems in the presence of rare, catastrophic events is a necessary component in enabling the development of trustworthy high-performance systems. Our proposed method, neural bridge sampling, employs three concepts—exploration, exploitation, and optimization—in order to evaluate system safety with provable statistical and computational efficiency. We demonstrate the performance of our method on a variety of reinforcement-learning and robotic systems, highlighting its use as a tool for continuous integration and rapid engineering design. In future work, we intend to investigate how efficiently sampling rare failures—like we propose here for *evaluation*—could also enable the *automated repair* of safety-critical reinforcement-learning agents.

Chapter 6

Conclusions and future directions

In order to make progress, one must leave the door to the unknown ajar—ajar only.
— Richard Feynman, *The Pleasure of Finding Things Out*

This thesis presents methods to improve the development and testing of safety-critical machine-learning systems. These two problems represent the bottlenecks to enabling ML algorithms as high-stakes decision-making systems. As such, we place particular emphasis on developing procedures that have both theoretical guarantees as well as practical implementations. Our overall approach has been to rigorously handle various forms of uncertainty within mathematical frameworks that lead to tractable optimization or search procedures. This chapter summarizes our findings and presents promising directions for further investigation.

In the first part of the thesis, we consider building safety-critical ML systems that are robust to modeled uncertainty sets. We consider small- and large-uncertainty regimes in Chapters 2 and 3 respectively. For small uncertainty sets, we consider a penalty formulation of a distributionally robust optimization procedure. When employed in the context of smooth loss functions, this procedure is provably fast and results in models with certificates of robustness. The beauty of this approach is its simplicity and wide applicability across many models, as the smoothness assumption is relatively easy to satisfy for ML models in many applications. Indeed, the upshot of this chapter is that smooth ML models may be preferable to standard non-smooth models (*e.g.* models with ReLU activations) from both

safety and computational standpoints. In addition to the empirical validation we have conducted here, further experimentation should be performed over larger models and datasets (*e.g.* the recent work of Xie et al. [303] seems promising in this respect). Furthermore, our theoretical results will benefit from tighter statistical bounds for the generalization performance of deep networks.

The methods of Chapter 2 are intractable for large uncertainty sets, so Chapter 3 considers the large-uncertainty regime using a different methodology. Because uncertainty is large, the focus of this chapter is to balance the tradeoff between safety and performance. Our approach is to learn a parametrization for the large uncertainty set that enables a tractable distributionally robust optimization procedure. In particular, we consider an extreme form of this setting where we must use synthetic data for the first stage of learning the uncertainty set, as we are unable to collect real data. We find that adaptivity is key in balancing the tradeoff between performance and safety; our experiments provide strong empirical evidence that being able to increase or decrease the size of the uncertainty set online enables performance that is both safe and performant. However, the methods of this chapter focused on the case-study of autonomous racing, and it is important to understand how to apply the procedure—the offline parametrization of uncertainty via synthetic data and the online robust adaptation algorithm—to other reinforcement-learning environments as well as supervised-learning settings. In particular, for these other scenarios there may be more rigorous ways to define the representation of the uncertainty set beyond the domain-specific knowledge that we have used for autonomous racing. For all of these settings, combining our procedure with more formal guarantees of safety (*e.g.* recursive feasibility for reinforcement-learning or formal verification for supervised learning) will enhance its ability to be used in real-world settings.

In the second part of the thesis, we formalize the testing of safety-critical ML systems through measuring the probability of failure, an approach we call the risk-based framework. This framework converts the safety-testing problem into one of searching for high-likelihood and *a priori* unknown failure modes. Specifically, the technical challenge becomes a rare-event simulation problem. Chapter 4 develops the three components of this framework—simulation, generative models, and a search algorithm—through the case-study of autonomous driving. Overall, the risk-based framework is a tractable alternative to traditional testing techniques; our approach is designed for efficient evaluation of modern safety-critical ML systems. Nevertheless, it is compatible with traditional techniques

such as formal verification or falsification for subcomponents. For example, these methods can be combined with our approach through the use of safety metrics that are built upon continuous relaxations of metric temporal logic; this combines notions of likelihood with those of fault or blame in the search process. Moreover, because our approach relies heavily on simulation, risk calculations are useful insofar as generative models are accurate. Thus, developing imitation-learning models that have guarantees regarding their ability to mimic real-world data-generating distributions is a clear avenue for continued research.

Finally, Chapter 5 develops upon the third component of the risk-based framework: the search algorithm used for rare-event simulation. Our technique, neural bridge sampling, is provably efficient and has guarantees for estimation precision. Crucially, many modern ML systems allow for gradients to be extracted from the simulator, so we develop the method to take advantage of this information when it is available. Our overall approach combines parametric and nonparametric approaches to sequential Monte Carlo schemes; learned parametric warping distributions allow us to improve estimation efficiency while nonparametric Markov chain Monte Carlo techniques correct for bias in the parametric distributions. Experiments showcase the benefits of this combination, as we outperform other techniques in performing rapid model comparison and sensitivity analysis over a variety of scenarios. An obvious avenue for further investigation is a more detailed convergence bound taking into account finite-step and potentially non-mixed Markov chain Monte Carlo schemes. Further experimentation with larger safety-critical systems is also important to evaluate the efficacy of the approach over other techniques.

6.1 Unifying development and testing

Perhaps the most important vector for future research is combining the two parts of this thesis into a unified framework for automating model governance. Imagine the following continuous integration process. First, we use the methods of Chapters 2 and 3 to develop a specific iteration of a safety-critical ML algorithm. Then, we test it using the methods of Chapters 4 and 5 to efficiently find failure scenarios in simulation as well as their likelihood of occurrence. To complete the feedback loop and continue the next iteration, we use these failures to inform the development of uncertainty sets for robustification. However, in traditional software development, this last piece of completing the loop is often an intensive manual process. This is one of the reasons that, despite enormous teams and capital pouring

into autonomous vehicle development over the past decade [154], fully self-driving cars are still considered to be years if not decades away from deployment [49].

Automating this feedback requires redesigning software from the ground up such that it is amenable to continuous updates (a paradigm shift that has been dubbed "Software 2.0" [151]); the traditional process of refactoring and recompiling code requires too much costly and inefficient human intervention. ML models are already well-designed to handle such types of continuous feedback—this is obviously the essence of gradient descent methods. However, automating overall model governance—the feedback between development and testing—requires re-architecting entire software stacks to be suitable for automated feedback rather than just individual ML subcomponents.

There are a variety of research questions that fall under this direction. How can we design architectures such that the feedback process has guarantees of convergence? What kinds of robustness certificates do we achieve using such a process? How do we optimally inject exogenous data into this feedback loop as objectives or performance criteria change? Furthermore, as we briefly discussed in Section 5.1.1, we can reverse this loop, using uncertainty sets (potentially developed via synthetic data) to inform a family of distributions over which to quantify risk. How can we efficiently compute this "robust risk"? How should we optimally tradeoff between performing the feedback loop in forward and reverse directions? The lenses of robust risk and its counterpart "risk-informed robustness" are natural extensions of this thesis. They represent exciting avenues for future research that can unify the notions of development and testing into a single framework for *governing* safety-critical ML.

6.2 Broader considerations

Having summarized the contributions of this thesis and outlined numerous frontiers for future research, we pause to reflect on the larger ramifications of building foundational theory and methods for developing and testing safety-critical ML systems. In particular, we take a moment to extend beyond the focused technical discussion of the previous chapters to broader ethical and societal considerations.

The underlying motivation for this thesis is that improving tools that ML practitioners have to robustify their systems and perform risk-estimation has the potential to provide a strong positive impact. Indeed, when safety-critical ML systems perform well, the result is

that, far from endangering society, they can actually *improve* overall safety. For example, in the case of autonomous vehicles, Sparrow and Howard [273] argue that it will be morally wrong not to deploy self-driving technology once performance exceeds human capabilities, and Kalra and Paddock [149] provide quantitative arguments corroborating this moral incentive. These arguments are readily transferrable to other applications of safety-critical ML such as medical devices, stock-market infrastructure, home-automation devices, and disability-assistance devices. Our work represents important tools for efficiently approaching this performance threshold and rigorously determining when it is actually achieved.

However, while the widespread availability of autonomy-enabled devices could benefit public health, there are many secondary risks associated with their development and deployment. First, many learning-based components of these systems will require massive and potentially invasive data collection [236]; preserving privacy of the public via federated learning [195] and differential privacy-based mechanisms [91] should remain important initiatives within the ML community. A second potential negative consequence of applications like autonomous vehicles is the use of the real-world as a "simulator" within a reinforcement-learning scheme by releasing "beta" autonomy features (*e.g.* Tesla Autopilot [151]). Unlike established industries such as aerospace [282], many potential safety-critical ML application domains currently lack regulation and standards; it is important to ensure that industry works with regulators to develop safety standards in a way that avoids regulatory capture. If widely adopted into regulatory frameworks, tools for improving robustness and estimating risk will enable rational decisions about the impact—positive or negative—of safety-critical ML systems before real lives are affected.

In a larger sense, the successful deployment of safety-critical ML across many domains could spark significant societal changes. For example, the autonomous applications described previously could become core components of weapons systems and military technology that are incompatible with (modern interpretations of) just war theory [272]. Similarly, automation of the transportation industry has the potential to rapidly destroy the economics of public infrastructure and cost millions of jobs [273]. Thus, Benkler [29] highlights that there is a growing need for the academic community to take action on defining the broader performance criteria to which we will hold artificial-intelligence applications. Specifically, beyond making ML applications pass an appropriate bar of safety, how will we ensure that they also abide by the ethical standards of our society? How can we build ML systems that we *trust*? This desideratum is less focused than the lenses of safety—robustness and

risk—that we have used to develop the technical advancements in this thesis. Rather, it requires a comprehensive integration of mathematical frameworks with legal, philosophical, and sociological schools of thought [54, 302]. This is, perhaps, a poignant message about the enduring importance of human judgement as it pertains to artificial intelligence. In order to entrust ML technology with our lives, technical advancements must inform and be informed by broader societal context.

Appendix A

Chapter 2 Appendices

A.1 Additional Experiments

A.1.1 MNIST attacks

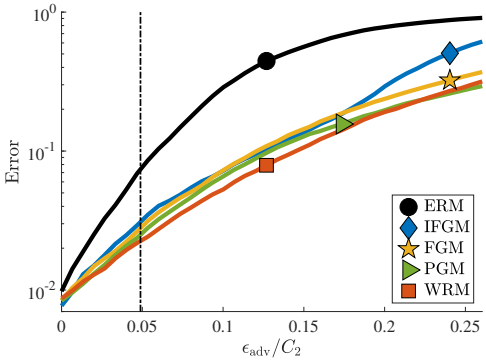
We repeat Figure 2.3 using FGM (tow row of Figure A.1) and IFGM (bottom row of Figure A.1) attacks. The same trends are evident as in Figure 2.3.

A.1.2 MNIST stability of loss surface

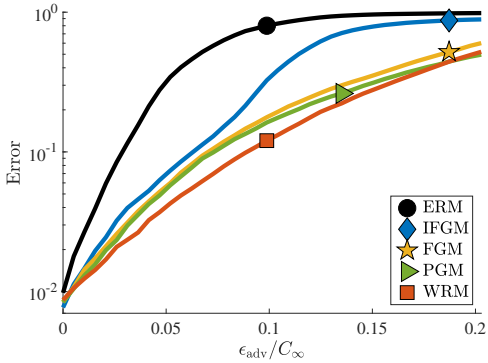
In Figure A.2, we repeat the illustration in Figure 2.4(b) for more digits. WRM's "misclassifications" are consistently reasonable to the human eye, as gradient-based perturbations actually transform the original image to other labels. Other models do not exhibit this behavior with the same consistency (if at all). Reasonable misclassifications correspond to having learned a data representation that makes gradients interpretable.

A.1.3 MNIST Experiments with varied γ

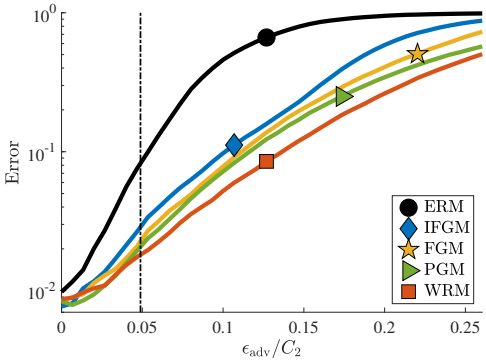
In Figure A.3, we choose a fixed WRM adversary (fixed γ_{adv}) and perturb WRM models trained with various penalty parameters γ. As the bound (2.11) with $\eta = \gamma$ suggests, even when the adversary has more budget than that used for training ($1/\gamma < 1/\gamma_{\text{adv}}$), degradation in performance is still *smooth*. Further, as we decrease the penalty γ, the amount of achieved robustness—measured here by test error on adversarial perturbations with γ_{adv}—has diminishing gains; this is again consistent with our theory which says that the inner problem (2.2b) is not efficiently computable for small γ.



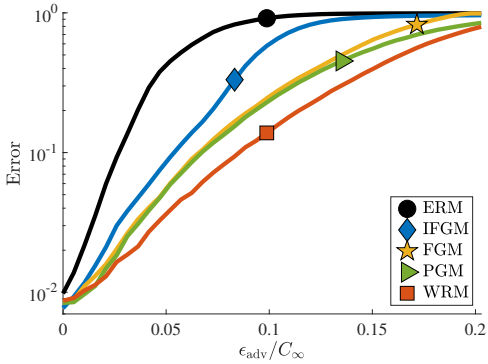
(a) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-FGM attack



(b) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-FGM attack



(c) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-IFGM attack



(d) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-IFGM attack

Figure A.1: Further attacks on the MNIST dataset. We illustrate test misclassification error vs. the adversarial perturbation level ϵ_{adv}. Top row: FGM attacks, bottom row: IFGM attacks. Left column: Euclidean-norm attacks, right column: ∞-norm attacks. The vertical bar in (a) and (c) indicates the perturbation level that was used for training the PGM, FGM, and IFGM models and the estimated radius $\sqrt{\widehat{\rho}_n(\theta_{\text{WRM}})}$.

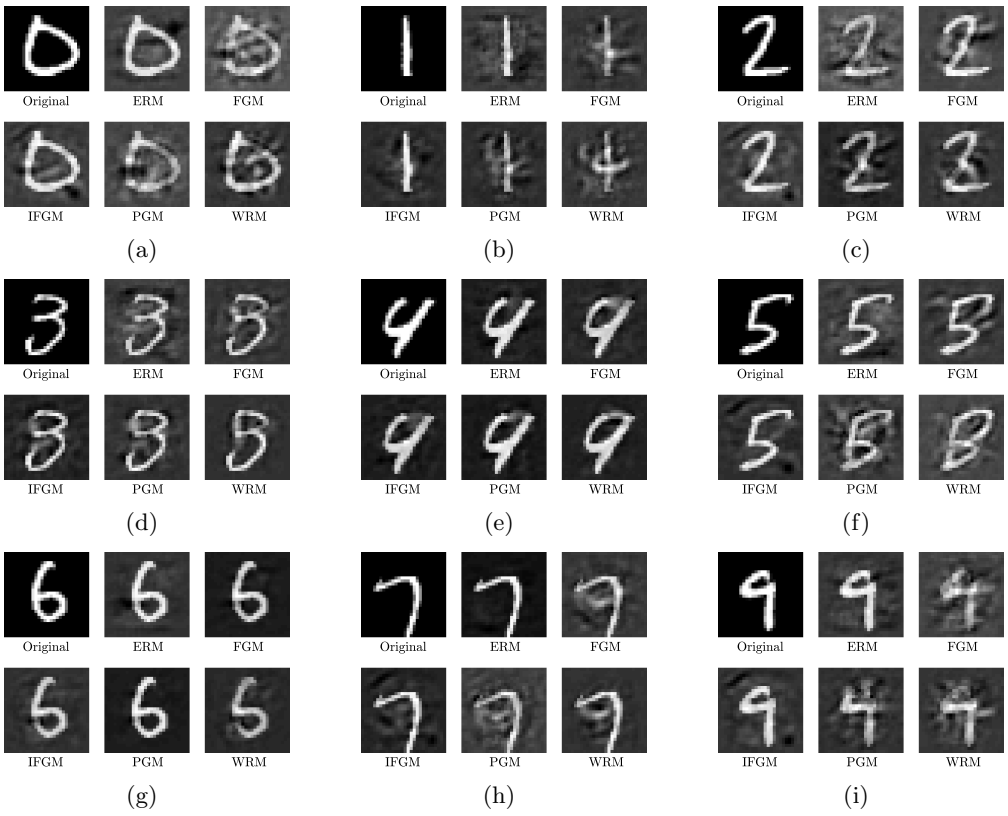
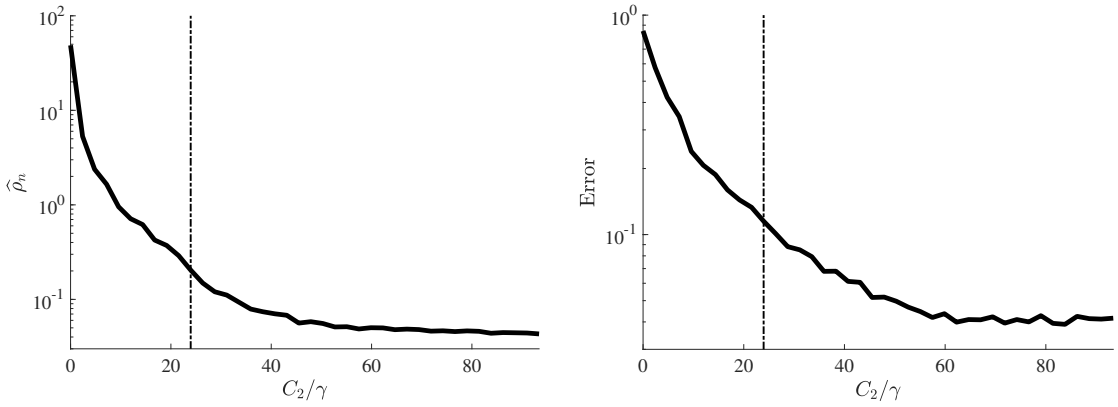


Figure A.2: Visualizing stability over inputs. We illustrate the smallest WRM perturbation (largest γ_{adv}) necessary to make a model misclassify a datapoint.



(a) $\widehat{\rho}_n$ vs. $1/\gamma$ (b) Test error vs. $1/\gamma$

Figure A.3: (a) Stability and (b) test error for a fixed adversary. We train WRM models with various levels of γ and perturb them with a fixed WRM adversary (γ_{adv} indicated by the vertical bar).

A.1.4 MNIST experiments with a larger adversarial budget

Figures A.4 and A.5 repeat Figures 2.2(b), 2.3, and A.1 for a larger training adversarial budget ($\gamma = 0.02C_2$) as well as larger test adversarial budgets. The distinctions in performance between various methods are less apparent now. For our method, the inner supremum is no longer strongly concave for over 10% of the data, indicating that we no longer have guarantees of performance. For large adversaries (i.e. large desired robustness values) our approach becomes a heuristic just like the other approaches.

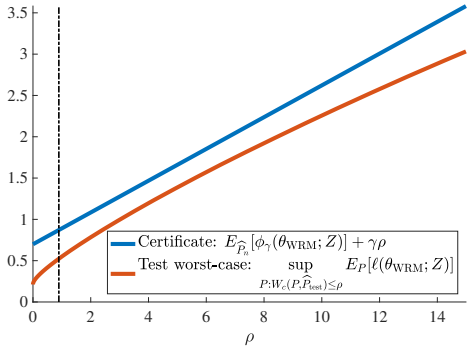
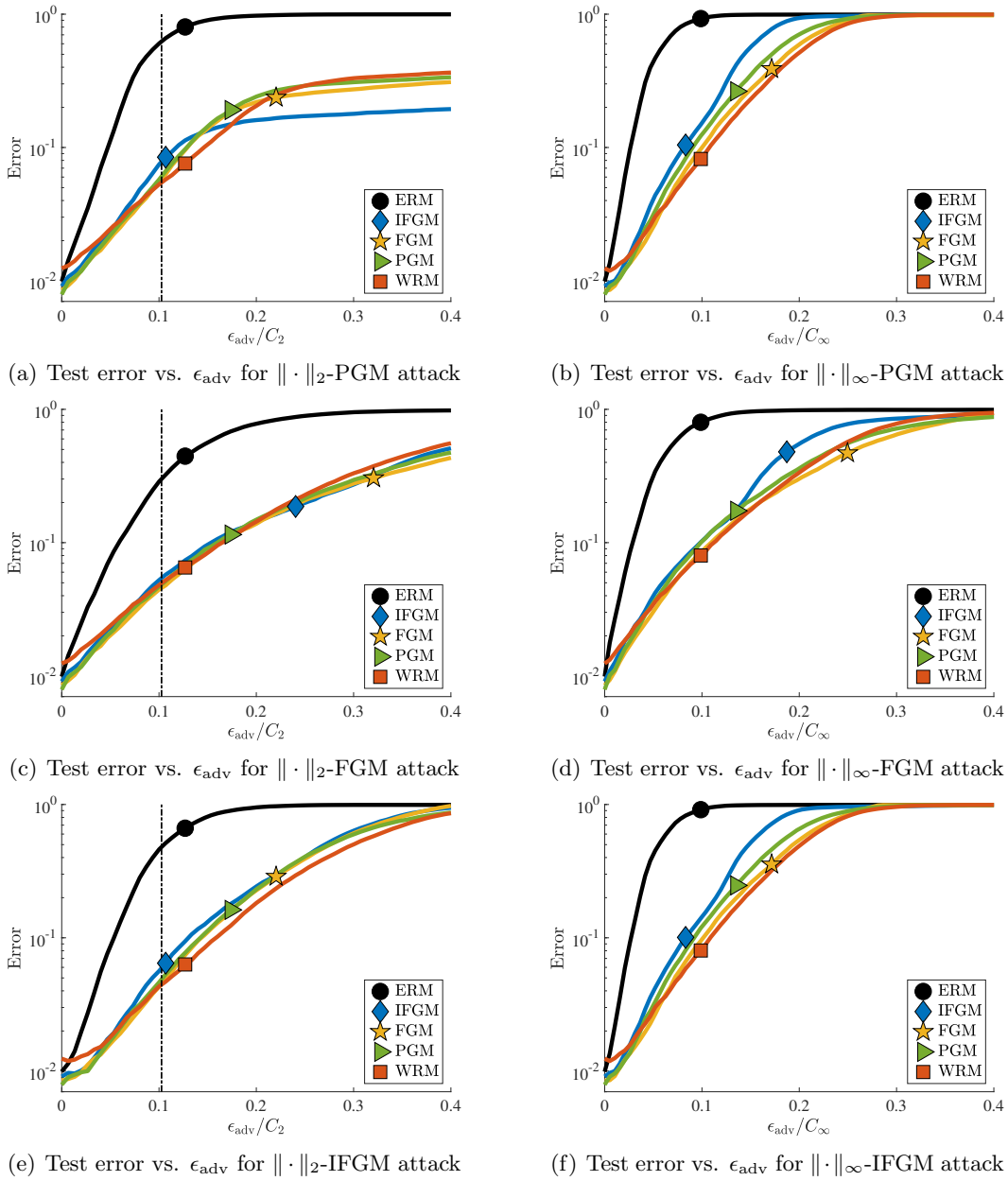


Figure A.4: Empirical comparison between certificate of robustness (2.11) (blue) and out-of-sample (test) worst-case performance (red) for the experiments on MNIST with a larger training adversary. The statistical error term $\epsilon_n(t)$ is omitted from the certificate. The vertical bar indicates the achieved level of robustness on the training set $\widehat{\rho}_n(\theta_{\mathrm{WRM}})$.



(a) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-PGM attack

(b) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-PGM attack

(c) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-FGM attack

(d) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-FGM attack

(e) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-IFGM attack

(f) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-IFGM attack

Figure A.5: Attacks on the MNIST dataset with larger (training and test) adversarial budgets. We illustrate test misclassification error vs. the adversarial perturbation level ϵ_{adv}. Top row: PGM attacks, middle row: FGM attacks, bottom row: IFGM attacks. Left column: Euclidean-norm attacks, right column: ∞-norm attacks. The vertical bar in (a), (c), and (e) indicates the perturbation level that was used for training the PGM, FGM, and IFGM models and the estimated radius $\sqrt{\widehat{\rho}_n(\theta_{\text{WRM}})}$.

A.1.5 MNIST ∞-norm experiments

We consider training FGM, IFGM, and PGM with $p = \infty$. We first compare with WRM trained in the same manner as before—with the squared Euclidean cost. Then, we consider a heuristic Lagrangian approach for training WRM with the squared ∞-norm cost.

Comparison with standard WRM

Our method (WRM) is trained to defend against $\|\cdot\|_2$-norm attacks by using the cost function

$$c((x, y), (x_0, y_0)) = \|x - x_0\|_2^2 + \infty \cdot \mathbf{1}\{y \neq y_0\}$$

with the convention that $0 \cdot \infty = 0$. Standard adversarial training methods often train to defend against $\|\cdot\|_\infty$-norm attacks, which we compare our method against in this subsection. Direct comparison between these approaches is not immediate, as we need to determine a suitable ϵ to train FGM, IFGM, and PGM in the ∞-norm that corresponds to the penalty parameter γ for the $\|\cdot\|_2$-norm that we use. Similar to the expression (2.23), we use

$$\epsilon := \mathbb{E}_{\widehat{P}_n}[\|T(\theta_{\text{WRM}}, Z) - Z\|_\infty] \tag{A.1}$$

as the adversarial training budget for FGM, IFGM and PGM with $\|\cdot\|_\infty$-norms. Because 2-norm adversaries tend to focus budgets on a subset of features, the resulting ∞-norm perturbations are relatively large. In Figure A.6 we show the results trained with a small training adversarial budget. In this regime, (large γ, small ϵ), WRM matches the performance of other techniques.

In Figure A.7 we show the results trained with a large training adversarial budget. In this regime (small γ, large ϵ), performance between WRM and other methods diverge. WRM, which provably defends against small perturbations, outperforms other heuristics against imperceptible attacks for both Euclidean and ∞ norms. Further, it outperforms other heuristics on natural images, showing that it consistently achieves a smaller price of robustness. On attacks with large adversarial budgets (large ϵ_{adv}), however, the performance of WRM is worse than that of the other methods (especially in the case of ∞-norm attacks). These findings verify that WRM is a practical alternative over existing heuristics for the moderate levels of robustness where our guarantees hold.

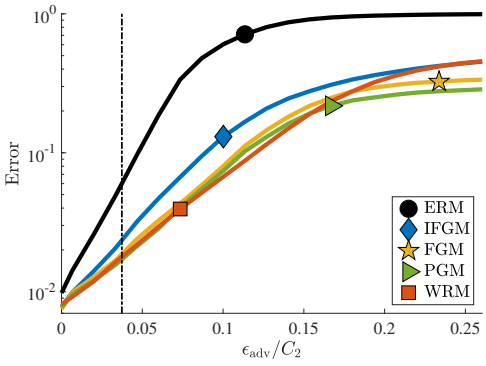
Comparison with $\|\cdot\|_\infty$-WRM

Our computational guarantees given in Theorem 2.1 does not hold anymore when we consider ∞-norm adversaries:

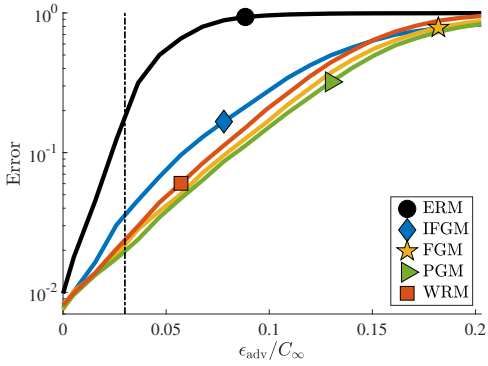
$$c((x,y),(x_0,y_0)) = \|x - x_0\|_\infty^2 + \infty \cdot \mathbf{1}\{y \neq y_0\}. \tag{A.2}$$

Optimizing the Lagrangian formulation (2.2b) with the ∞-norm is difficult since subtracting a multiple of the ∞-norm does not add (negative) curvature in all directions. In Appendix A.3, we propose a heuristic algorithm for solving the inner supremum problem (2.2b) with the above cost function (A.2). Our approach is based on a variant of proximal algorithms.

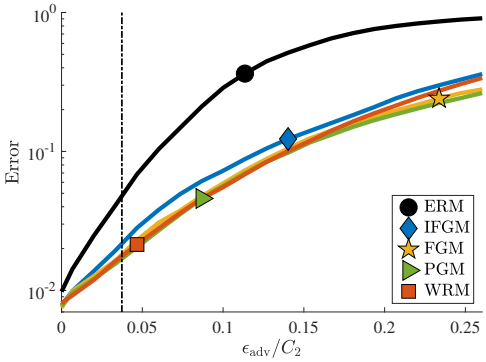
We compare our proximal heuristic introduced in Appendix A.3 with other adversarial training procedures that were trained against ∞-norm adversaries. Results are shown in Figure A.8 for a small training adversary and Figure A.9 for a large training adversary. We observe that similar trends as in Section A.1.5 hold again.



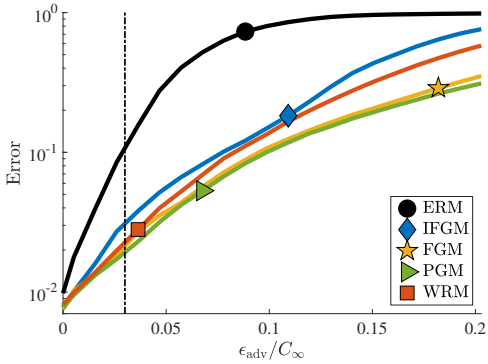
(a) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-PGM attack



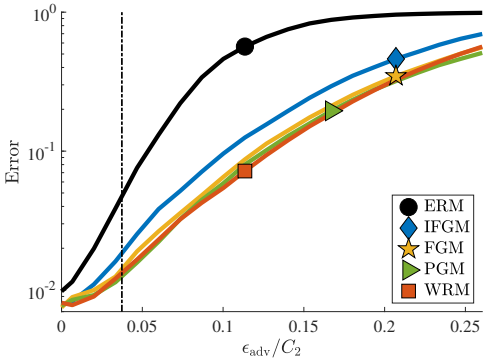
(b) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-PGM attack



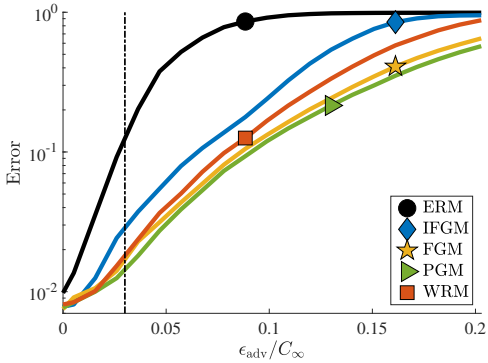
(c) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-FGM attack



(d) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-FGM attack

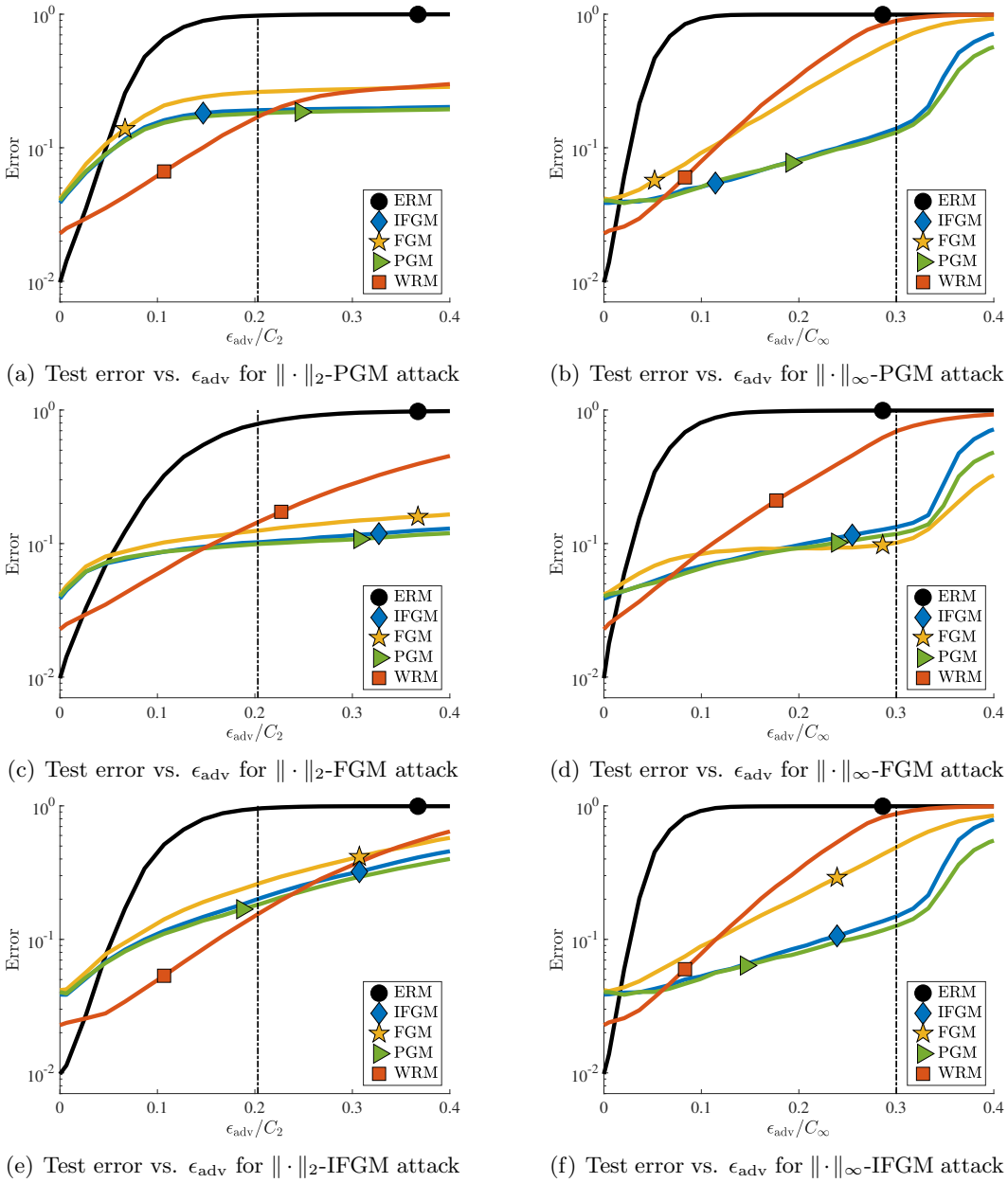


(e) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-IFGM attack



(f) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-IFGM attack

Figure A.6: Attacks on the MNIST dataset. We compare standard WRM with ∞-norm PGM, FGM, IFGM. We illustrate test misclassification error vs. the adversarial perturbation level ϵ_{adv}. Top row: PGM attacks, middle row: FGM attacks, bottom row: IFGM attacks. Left column: Euclidean-norm attacks, right column: ∞-norm attacks. The vertical bar in (a), (c), and (e) indicates the estimated radius $\sqrt{\widehat{\rho}_n(\theta_{\text{WRM}})}$. The vertical bar in (b), (d), and (f) indicates the perturbation level that was used for training the PGM, FGM, and IFGM models via (A.1).



(a) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-PGM attack

(b) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-PGM attack

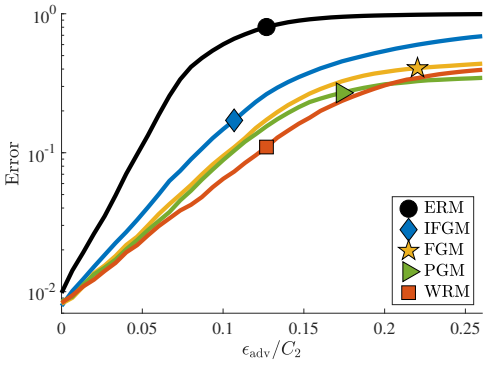
(c) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-FGM attack

(d) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-FGM attack

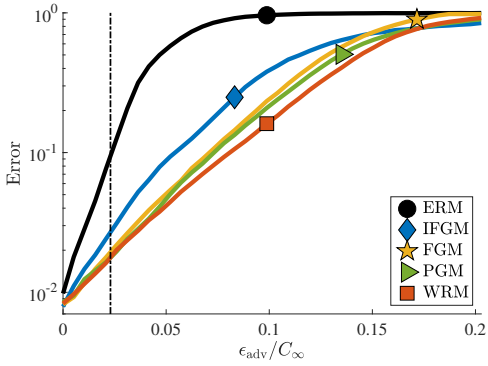
(e) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-IFGM attack

(f) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-IFGM attack

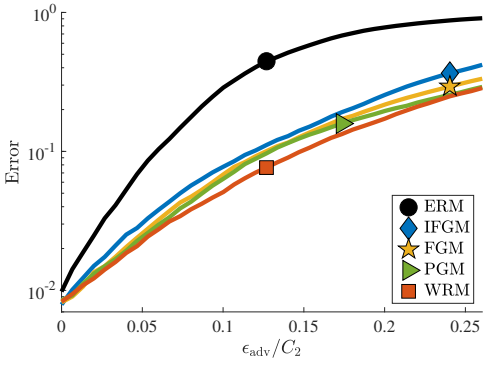
Figure A.7: Attacks on the MNIST dataset with larger (training and test) adversarial budgets. We compare standard WRM with ∞-norm PGM, FGM, IFGM models. We illustrate test misclassification error vs. the adversarial perturbation level ϵ_{adv}. Top row: PGM attacks, middle row: FGM attacks, bottom row: IFGM attacks. Left column: Euclidean-norm attacks, right column: ∞-norm attacks. The vertical bar in (a), (c), and (e) indicates the estimated radius $\sqrt{\widehat{\rho}_n(\theta_{\text{WRM}})}$. The vertical bar in (b), (d), and (f) indicates the perturbation level that was used for training the PGM, FGM, and IFGM models via (A.1).



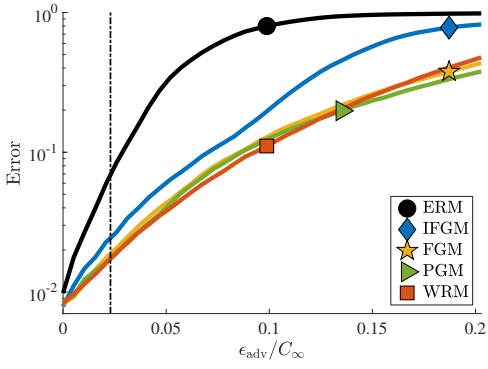
(a) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-PGM attack



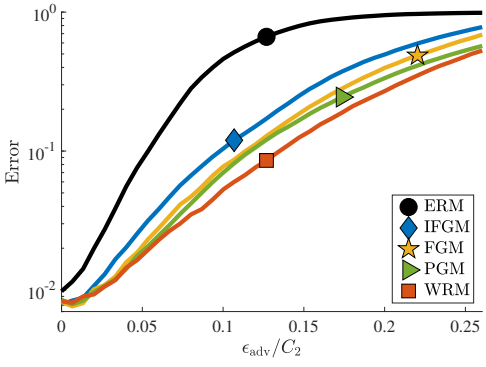
(b) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-PGM attack



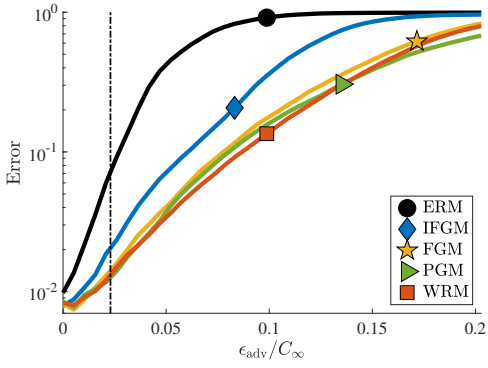
(c) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-FGM attack



(d) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-FGM attack

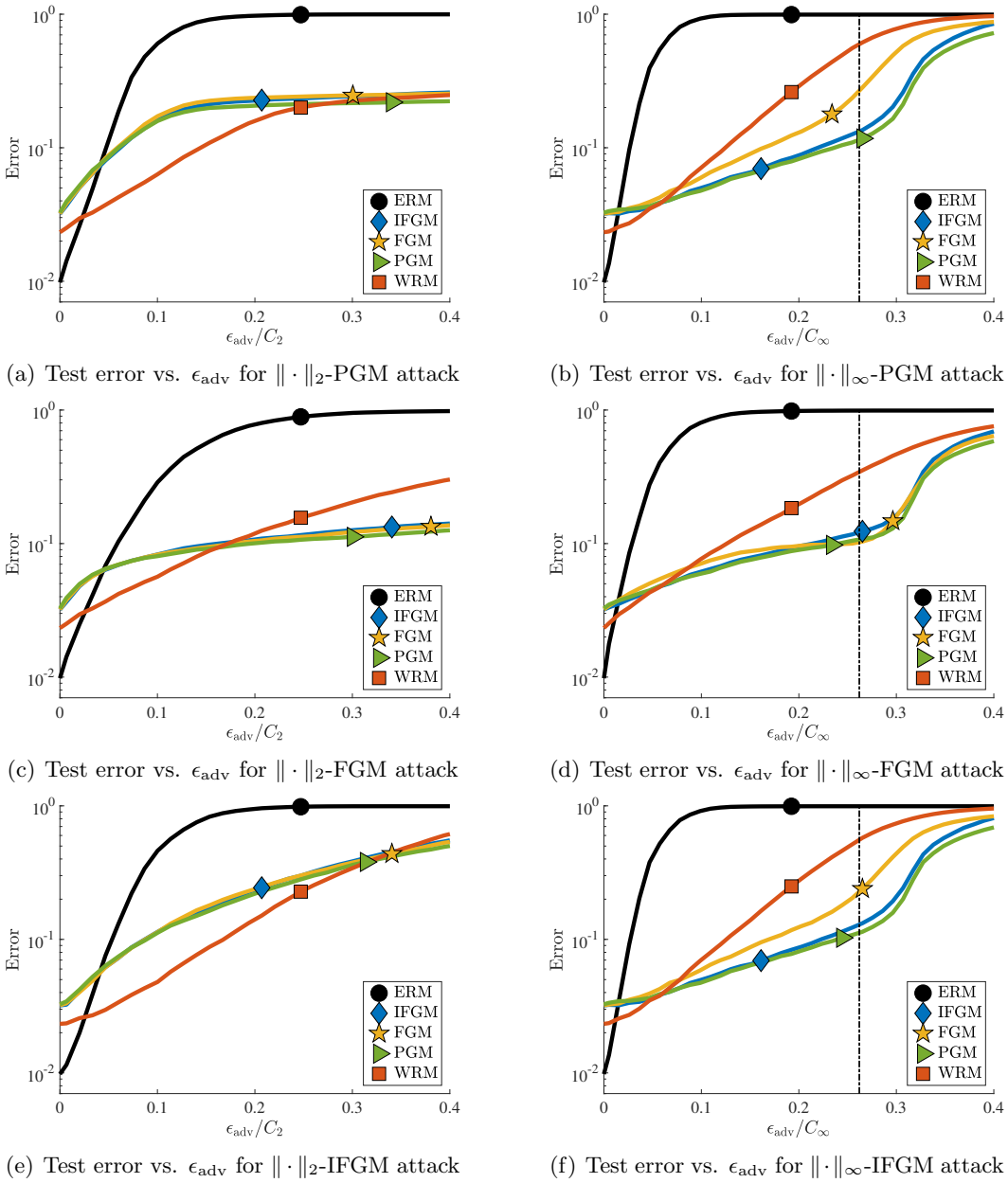


(e) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-IFGM attack



(f) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-IFGM attack

Figure A.8: Attacks on the MNIST dataset. All models are trained in the ∞-norm. We illustrate test misclassification error vs. the adversarial perturbation level ϵ_{adv}. Top row: PGM attacks, middle row: FGM attacks, bottom row: IFGM attacks. Left column: Euclidean-norm attacks, right column: ∞-norm attacks. The vertical bar in (b), (d), and (f) indicates the perturbation level that was used for training the PGM, FGM, and IFGM models and the estimated radius $\sqrt{\widehat{\rho}_n(\theta_{\text{WRM}})}$.



(a) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-PGM attack

(b) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-PGM attack

(c) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-FGM attack

(d) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-FGM attack

(e) Test error vs. ϵ_{adv} for $\|\cdot\|_2$-IFGM attack

(f) Test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-IFGM attack

Figure A.9: Attacks on the MNIST dataset with larger (training and test) adversarial budgets. All models are trained in the ∞-norm. We illustrate test misclassification error vs. the adversarial perturbation level ϵ_{adv}. Top row: PGM attacks, middle row: FGM attacks, bottom row: IFGM attacks. Left column: Euclidean-norm attacks, right column: ∞-norm attacks. The vertical bar in (b), (d), and (f) indicates the perturbation level that was used for training the PGM, FGM, and IFGM models and the estimated radius $\sqrt{\widehat{\rho}_n(\theta_{\text{WRM}})}$.

A.1.6 MNIST experiments when γ is chosen according to Section 2.4

We now present analysis of principled experiments on the MNIST dataset. For the chosen architecture, we find that it is difficult to satisfy the following three objectives: γ is larger than the bound given by Corollary 2.4, the model has high enough capacity such that the test accuracy on clean, unperturbed data is on par with standard benchmarks (less than 10% test error for the MNIST dataset), and the WRM model's performance differs appreciably from ERM's performance. Table A.1 tests ERM and WRM models trained with different γ with $\|\cdot\|_\infty$ PGM attacks of various ϵ_{adv}. Although all models enjoy good test accuracy on clean test examples, we see that when γ is large enough to satisfy the bound of Corollary 2.4, WRM and ERM do not differ in performance, as the perturbed examples during training are essentially the same as the originals.

To ameliorate this issue, we regularize the weights to decrease the bound of Corollary 2.4. Tables A.2, A.3, and A.4 present the same analysis for architectures with different l_2-regularization schemes. We choose to regularize the earlier layers (those closer to the inputs) more heavily than later layers (those closer to the output), as the bound in Corollary 2.4 scales with the norm of earlier layer weights exponentially in the depth of the network. We see that although

Overall, we see that high accuracy on clean test examples and appreciable adversarial robustness hold comes at the expense of $\gamma < \bar{\gamma}$ (the top rows of all tables). For $\gamma \gtrsim \bar{\gamma}$, high accuracy on clean test examples hold, but adversarial robustness seems similar to that of ERM (the bottom rows of all tables). For the most regularized models in Table A.4, the large values of γ achieves appreciable adversarial robustness, although the regularization is too heavy for good performance on clean examples. For this architecture, we are unable unable to easily discover a paramtrization that satisfied all three objectives.

γ	$\bar{\gamma}$	$\epsilon_{\text{adv}} = 0$	$\epsilon_{\text{adv}} = 0.05$	$\epsilon_{\text{adv}} = 0.10$	$\epsilon_{\text{adv}} = 0.15$	$\epsilon_{\text{adv}} = 0.20$
ERM	1.97×10^{6}	0.015	0.965	1	1	1
1	1.23×10^{6}	0.015	0.112	0.484	0.916	0.994
1×10^{2}	1.81×10^{6}	0.015	0.916	1	1	1
1×10^{4}	1.97×10^{6}	0.015	0.965	1	1	1
1×10^{6}	1.97×10^{6}	0.015	0.966	1	1	1
1×10^{7}	1.97×10^{6}	0.015	0.966	1	1	1

Table A.1: Performance of various WRM models. We show the smoothness upper bound ($\bar{\gamma}$) and test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-norm PGM attacks.

γ	$\bar{\gamma}$	$\epsilon_{\text{adv}} = 0$	$\epsilon_{\text{adv}} = 0.05$	$\epsilon_{\text{adv}} = 0.10$	$\epsilon_{\text{adv}} = 0.15$	$\epsilon_{\text{adv}} = 0.20$
ERM	7.77×10^{3}	0.045	0.318	0.855	0.992	0.999
1	8.69×10^{3}	0.041	0.183	0.606	0.925	0.996
1×10^{1}	6.56×10^{3}	0.046	0.303	0.835	0.991	1.000
1×10^{2}	7.70×10^{3}	0.048	0.326	0.858	0.992	1.000
1×10^{3}	7.84×10^{3}	0.046	0.320	0.857	0.992	0.999
1×10^{4}	7.75×10^{3}	0.048	0.329	0.860	0.993	1.000

Table A.2: Performance of various l_2-regularized WRM models. The regularization multiplier is 0.05 for the first half of the neural net layers and 0.01 for the latter half. We show the smoothness upper bound ($\bar{\gamma}$) and test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-norm PGM attacks.

γ	$\bar{\gamma}$	$\epsilon_{\text{adv}} = 0$	$\epsilon_{\text{adv}} = 0.05$	$\epsilon_{\text{adv}} = 0.10$	$\epsilon_{\text{adv}} = 0.15$	$\epsilon_{\text{adv}} = 0.20$
ERM	5.06×10^{3}	0.059	0.353	0.881	0.996	1.000
1	5.80×10^{3}	0.056	0.225	0.657	0.941	0.997
1×10^{1}	4.40×10^{3}	0.057	0.325	0.853	0.994	1
1×10^{2}	4.65×10^{3}	0.059	0.352	0.872	0.995	1
1×10^{3}	4.71×10^{3}	0.060	0.346	0.876	0.991	1.000
1×10^{4}	4.76×10^{3}	0.060	0.349	0.851	0.993	1.000

Table A.3: Performance of various l_2-regularized WRM models. The regularization multiplier is 0.1 for the first half of the neural net layers and 0.01 for the latter half. We show the smoothness upper bound ($\bar{\gamma}$) and test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-norm PGM attacks.

γ	$\bar{\gamma}$	$\epsilon_{\text{adv}} = 0$	$\epsilon_{\text{adv}} = 0.05$	$\epsilon_{\text{adv}} = 0.10$	$\epsilon_{\text{adv}} = 0.15$	$\epsilon_{\text{adv}} = 0.20$
ERM	5.76×10^2	0.117	0.433	0.852	0.988	1.000
1	5.87×10^2	0.117	0.364	0.717	0.952	0.996
1×10^1	7.61×10^2	0.116	0.429	0.832	0.986	1.000
1×10^2	8.03×10^2	0.121	0.442	0.849	0.986	1.000
5×10^2	8.06×10^2	0.120	0.443	0.857	0.990	1.000
1×10^3	8.04×10^2	0.115	0.436	0.846	0.987	1

Table A.4: Performance of various l_2-regularized WRM models. The regularization multiplier is 0.5 for the first half of the neural net layers and 0.01 for the latter half. We show the smoothness upper bound ($\bar{\gamma}$) and test error vs. ϵ_{adv} for $\|\cdot\|_\infty$-norm PGM attacks.

A.2 Proofs

A.2.1 Proof of Proposition 2.1

For completeness, we provide an alternative proof to that given in Blanchet and Murthy [39] using convex analysis. Our proof is less general, requiring the cost function c to be continuous and convex in its first argument. The below general duality result gives Proposition 2.1 as an immediate special case. Recalling Rockafellar and Wets [240, Def. 14.27 and Prop. 14.33], we say that a function $g : X \times Z \to \mathbb{R}$ is a *normal integrand* if for each α, the mapping

$$z \mapsto \{x \mid g(x, z) \leq \alpha\}$$

is closed-valued and measurable. We recall that if g is continuous, then g is a normal integrand [240, Cor. 14.34]; therefore, $g(x, z) = \gamma c(x, z) - \ell(\theta; x)$ is a normal integrand. We have the following theorem.

Theorem A.1. *Let f, c be such that for any $\gamma \geq 0$, the function $g(x, z) = \gamma c(x, z) - f(x)$ is a normal integrand. (For example, continuity of f and closed convexity of c is sufficient.) For any $\rho > 0$ we have*

$$\sup_{P:W_c(P,Q)} \int f(x)dP(x) = \inf_{\gamma \geq 0} \left\{ \int \sup_{x \in X} \{f(x) - \gamma c(x, z)\} \, dQ(z) + \gamma\rho \right\}.$$

Proof First, the mapping $P \mapsto W_c(P, Q)$ is convex in the space of probability measures. As taking $P = Q$ yields $W_c(Q, Q) = 0$, Slater's condition holds and we may apply standard (infinite dimensional) duality results [185, Thm. 8.7.1] to obtain

$$\begin{aligned} \sup_{P:W_c(P,Q)} \int f(x)dP(x) &= \sup_{P:W_c(P,Q)} \inf_{\gamma \geq 0} \left\{ \int f(x)dP(x) - \gamma W_c(P, Q) + \gamma\rho \right\} \\ &= \inf_{\gamma \geq 0} \sup_{P:W_c(P,Q)} \left\{ \int f(x)dP(x) - \gamma W_c(P, Q) + \gamma\rho \right\}. \end{aligned}$$

Now, noting that for any $M \in \Pi(P, Q)$ we have $\int f dP = \iint f(x)dM(x, z)$, we have that the rightmost quantity in the preceding display satisfies

$$\int f(x)dP(x) - \gamma \inf_{M \in \Pi(P,Q)} \int c(x, z)dM(x, z) = \sup_{M \in \Pi(P,Q)} \left\{ \int [f(x) - \gamma c(x, z)]dM(x, z) \right\}.$$

That is, we have

$$\sup_{P:W_c(P,Q)} \int f(x)dP(x) = \inf_{\gamma \geq 0} \sup_{P,M \in \Pi(P,Q)} \left\{ \int [f(x) - \gamma c(x,z)]dM(x,z) + \gamma \rho \right\}. \tag{A.3}$$

Now, we note a few basic facts. First, because we have a joint supremum over P and measures $M \in \Pi(P,Q)$ in expression (A.3), we have that

$$\sup_{P,M \in \Pi(P,Q)} \int [f(x) - \gamma c(x,z)]dM(x,z) \leq \int \sup_x [f(x) - \gamma c(x,z)]dQ(z).$$

We would like to show equality in the above. To that end, we note that if $\mathcal{P}$ denotes the space of regular conditional probabilities (Markov kernels) from Z to X, then

$$\sup_{P,M \in \Pi(P,Q)} \int [f(x) - \gamma c(x,z)]dM(x,z) \geq \sup_{P \in \mathcal{P}} \int [f(x) - \gamma c(x,z)]dP(x \mid z)dQ(z).$$

Recall that a conditional distribution $P(\cdot \mid z)$ is regular if $P(\cdot \mid z)$ is a distribution for each z and for each measurable A, the function $z \mapsto P(A \mid z)$ is measurable. Let $\mathcal{X}$ denote the space of all measurable mappings $z \mapsto x(z)$ from Z to X. Using the powerful measurability results of Rockafellar and Wets [240, Theorem 14.60], we have

$$\sup_{x \in \mathcal{X}} \int [f(x(z)) - \gamma c(x(z),z)]dQ(z) = \int \sup_{x \in X} [f(x) - \gamma c(x,z)]dQ(z)$$

because $f - c$ is upper semi-continuous, and the latter function is measurable. Now, let $x(z)$ be any measurable function that is ϵ-close to attaining the supremum above. Define the conditional distribution $P(\cdot \mid z)$ to be supported on $x(z)$, which is evidently measurable. Then using the preceding display, we have

$$\begin{aligned}
\int [f(x) - \gamma c(x,z)]dP(x \mid z)dQ(z) &= \int [f(x(z)) - \gamma c(x(z),z)]dQ(z) \\
&\geq \int \sup_{x \in X} [f(x) - \gamma c(x,z)]dQ(z) - \epsilon \\
&\geq \sup_{P,M \in \Pi(P,Q)} \int [f(x) - \gamma c(x,z)]dM(x,z) - \epsilon.
\end{aligned}$$

As $\epsilon > 0$ is arbitrary, this gives

$$\sup_{P,M\in\Pi(P,Q)} \int [f(x) - \gamma c(x,z)] dM(x,z) = \int \sup_{x\in X} [f(x) - \gamma c(x,z)] dQ(z)$$

as desired, which implies both equality (2.6) and completes the proof. □

A.2.2 Proof of Lemma 2.1

First, note that $z^\star(\theta)$ is unique and well-defined by the strong convexity of $f(\theta,\cdot)$. For Lipschitzness of $z^\star(\theta)$, we first argue that $z^\star(\theta)$ is continuous in θ. For any θ, optimality of $z^\star(\theta)$ implies that $\mathsf{g}_\mathsf{z}(\theta, z^\star(\theta))^T(z - z^\star(\theta)) \le 0$. By strong concavity, for any θ_1, θ_2 and $z_1^\star = z^\star(\theta_1)$ and $z_2^\star = z^\star(\theta_2)$, we have

$$\frac{\lambda}{2}\left\|z_1^\star - z_2^\star\right\|^2 \le f(\theta_2, z_2^\star) - f(\theta_2, z_1^\star) \text{ and } f(\theta_2, z_2^\star) \le f(\theta_2, z_1^\star) + \mathsf{g}_\mathsf{z}(\theta_2, z_1^\star)^T(z_2^\star - z_1^\star) - \frac{\lambda}{2}\left\|z_1^\star - z_2^\star\right\|^2.$$

Summing these inequalities gives

$$\lambda\left\|z_1^\star - z_2^\star\right\|^2 \le \mathsf{g}_\mathsf{z}(\theta_2, z_1^\star)^T(z_2^\star - z_1^\star) \le (\mathsf{g}_\mathsf{z}(\theta_2, z_1^\star) - \mathsf{g}_\mathsf{z}(\theta_1, z_1^\star))^T(z_2^\star - z_1^\star),$$

where the last inequality follows because $\mathsf{g}_\mathsf{z}(\theta_1, z_1^\star)^T(z_2^\star - z_1^\star) \le 0$. Using a cross-Lipschitz condition from above and Holder's inequality, we obtain

$$\lambda\left\|z_1^\star - z_2^\star\right\|^2 \le \left\|\mathsf{g}_\mathsf{z}(\theta_2, z_1^\star) - \mathsf{g}_\mathsf{z}(\theta_1, z_1^\star)\right\|_\star \left\|z_1^\star - z_2^\star\right\| \le L_{\mathsf{z}\theta}\left\|\theta_1 - \theta_2\right\|\left\|z_1^\star - z_2^\star\right\|,$$

that is,

$$\left\|z_1^\star - z_2^\star\right\| \le \frac{L_{\mathsf{z}\theta}}{\lambda}\left\|\theta_1 - \theta_2\right\|. \tag{A.4}$$

To see the second inequality, we show that $\bar{f}$ is differentiable with $\nabla\bar{f}(\theta) = \mathsf{g}_\theta(\theta, z^\star(\theta))$. By using a variant of the envelope (or Danskin's) theorem, we first show directional differentiability of $\bar{f}$. Recall that we say f is *inf-compact* if for all $\theta_0 \in \Theta$, there exists $\alpha > 0$ and a compact set $C \subset \Theta$ such that

$$\emptyset \ne \{z \in \mathcal{Z} : f(\theta, z) \le \alpha\} \subset C$$

for all θ in some neighborhood of θ_0 [44]. See Bonnans and Shapiro [44, Theorem 4.13] for a proof of the following result.

Lemma A.1. *Suppose that $f(\cdot, z)$ is differentiable in θ for all $z \in \mathcal{Z}$, and f, $\nabla_z f$ are continuous on $\Theta \times \mathcal{Z}$. If f is inf-compact, then $\bar{f}$ is directionally differentiable with*

$$\bar{f}'(\theta, d) = \sup_{z \in S(\theta)} \nabla_z f(\theta, z)^\top d$$

where $S(\theta) = \operatorname{argmin}_z f(\theta, z)$.

Now, note that from Assumption 2.2, we have

$$|f(\theta, z) - f(\theta_0, z) - \nabla_\theta f(\theta_0, z)^\top (\theta - \theta_0)| \leq L_{\theta\theta} \|\theta - \theta_0\|$$

from which it is easy to see that f is inf-compact. Applying Lemma A.1 to $\bar{f}$ and noting that $S(\theta)$ is unique by strong convexity of $f(\theta, \cdot)$, we have that $\bar{f}$ is directionally differentiable with $\nabla \bar{f}(\theta) = \mathsf{g}_\theta(\theta, z^\star(\theta))$. Since g_θ is continuous by Assumption 2.2 and $z^\star(\theta)$ is Lipschitz (A.4), we conclude that $\bar{f}$ is differentiable.

Finally, we have

$$\begin{aligned}
\|\mathsf{g}_\theta(\theta_1, z_1^\star) - \mathsf{g}_\theta(\theta_2, z_2^\star)\|_\star &\leq \|\mathsf{g}_\theta(\theta_1, z_1^\star) - \mathsf{g}_\theta(\theta_1, z_2^\star)\|_\star + \|\mathsf{g}_\theta(\theta_1, z_2^\star) - \mathsf{g}_\theta(\theta_2, z_2^\star)\|_\star \\
&\leq L_{\theta \mathsf{z}} \|z_1^\star - z_2^\star\| + L_{\theta\theta} \|\theta_1 - \theta_2\| \\
&\leq \left(L_{\theta\theta} + \frac{L_{\theta \mathsf{z}} L_{\mathsf{z}\theta}}{\lambda} \right) \|\theta_1 - \theta_2\|,
\end{aligned}$$

where we have used inequality (A.4) again. This is the desired result.

A.2.3 Proof of Theorem 2.1

Our proof is based on that of Ghadimi and Lan [110]. For shorthand, let $f(\theta, z; z_0) = \ell(\theta; z) - \gamma c(z, z_0)$, noting that we perform gradient steps with

$$g^t = \nabla_\theta f(\theta^t, \widehat{z}^t; z^t)$$

for $\widehat{z}^t$ an ϵ-approximate maximizer of $f(\theta, z; z^t)$ in z, and $\theta^{t+1} = \theta^t - \alpha_t g^t$. We assume $\alpha_t \leq \frac{1}{L_\phi}$ in the rest of the proof, which is satisfied for the constant stepsize $\alpha = \sqrt{\frac{\Delta_F}{L_\phi T \sigma^2}}$

and $T \geq \frac{L_\phi \Delta_F}{\sigma^2}$. By a Taylor expansion using the L_ϕ-smoothness of the objective F, we have

$$\begin{aligned} F(\theta^{t+1}) &\leq F(\theta^t) + \left\langle \nabla F(\theta^t), \theta^{t+1} - \theta^t \right\rangle + \frac{L_\phi}{2} \left\| \theta^{t+1} - \theta^t \right\|_2^2 \\ &= F(\theta^t) - \alpha_t \left\| \nabla F(\theta^t) \right\|_2^2 + \frac{L_\phi \alpha_t^2}{2} \left\| g^t \right\|_2^2 + \alpha_t \left\langle \nabla F(\theta^t), \nabla F(\theta^t) - g^t \right\rangle \\ &= F(\theta^t) - \alpha_t \left(1 - \frac{1}{2} L_\phi \alpha_t \right) \left\| \nabla F(\theta^t) \right\|_2^2 \\ &\qquad + \alpha_t \left(1 - L_\phi \alpha_t \right) \left\langle \nabla F(\theta^t), \nabla F(\theta^t) - g^t \right\rangle + \frac{L_\phi \alpha_t^2}{2} \left\| g^t - \nabla F(\theta^t) \right\|_2^2 . \end{aligned} \tag{A.5}$$

Recalling the definition (2.2b) of $\phi_\gamma(\theta; z_0) = \sup_{z \in \mathcal{Z}} f(\theta, z; z_0)$, we define the potentially biased errors $\delta^t = g^t - \nabla_\theta \phi_\gamma(\theta^t; z^t)$. Substituting the this into the progress guarantee (A.5), we have

$$\begin{aligned} F(\theta^{t+1}) &\leq F(\theta^t) - \alpha_t \left(1 - \frac{1}{2} L_\phi \alpha_t \right) \left\| \nabla F(\theta^t) \right\|_2^2 + \alpha_t \left(1 - L_\phi \alpha_t \right) \left\langle \nabla F(\theta^t), \nabla F(\theta^t) - \nabla_\theta \phi_\gamma(\theta; z^t) \right\rangle \\ &\quad - \alpha_t \left(1 - L_\phi \alpha_t \right) \left\langle \nabla F(\theta^t), \delta^t \right\rangle + \frac{L_\phi \alpha_t^2}{2} \left\| \nabla_\theta \phi_\gamma(\theta; z^t) + \delta^t - \nabla F(\theta^t) \right\|_2^2 \\ &= F(\theta^t) - \alpha_t \left(1 - \frac{1}{2} L_\phi \alpha_t \right) \left\| \nabla F(\theta^t) \right\|_2^2 + \alpha_t \left(1 - L_\phi \alpha_t \right) \left\langle \nabla F(\theta^t), \nabla F(\theta^t) - \nabla_\theta \phi_\gamma(\theta; z^t) \right\rangle \\ &\quad - \alpha_t \left(1 - L_\phi \alpha_t \right) \left\langle \nabla F(\theta^t), \delta^t \right\rangle \\ &\qquad + \frac{L_\phi \alpha_t^2}{2} \left(\left\| \delta^t \right\|_2^2 + \left\| \nabla_\theta \phi_\gamma(\theta; z^t) - \nabla F(\theta^t) \right\|_2^2 + 2 \left\langle \nabla_\theta \phi_\gamma(\theta; z^t) - \nabla F(\theta^t), \delta^t \right\rangle \right) . \end{aligned}$$

Using $\pm \langle a, b \rangle \leq \frac{1}{2} \|a\|_2^2 + \frac{1}{2} \|b\|_2^2$ in the preceding display, we get

$$\begin{aligned} F(\theta^{t+1}) &\leq F(\theta^t) - \frac{\alpha_t}{2} \left\| \nabla F(\theta^t) \right\|_2^2 + \alpha_t \left(1 - L_\phi \alpha_t \right) \left\langle \nabla F(\theta^t), \nabla F(\theta^t) - \nabla_\theta \phi_\gamma(\theta; z^t) \right\rangle \\ &\qquad + \frac{\alpha_t \left(1 + L_\phi \alpha_t \right)}{2} \left\| \delta^t \right\|_2^2 + L_\phi \alpha_t^2 \left\| \nabla_\theta \phi_\gamma(\theta; z^t) - \nabla F(\theta^t) \right\|_2^2 \end{aligned} \tag{A.6}$$

Letting $z_\star^t = \operatorname{argmax}_z f(\theta^t, z; z^t)$, note that the error δ^t satisfies

$$\begin{aligned} \left\| \delta^t \right\|_2^2 = \left\| \nabla_\theta \phi_\gamma(\theta^t; z^t) - \nabla_\theta f(\theta, \widehat{z}^t; z^t) \right\|_2^2 &= \left\| \nabla_\theta \ell(\theta, z_\star^t) - \nabla_\theta \ell(\theta, \widehat{z}^t) \right\|_2^2 \\ &\leq L_{\theta \mathbf{z}}^2 \| \widehat{z}^t - z_\star^t \|_2^2 \leq \frac{2 L_{\theta \mathbf{z}}^2}{\lambda} \epsilon, \end{aligned}$$

where the final inequality uses the $\lambda = \gamma - L_{\mathbf{zz}}$ strong-concavity of $z \mapsto f(\theta, z; z_0)$. For shorthand, let $\widehat{\epsilon} = \frac{2L_{\theta z}^2}{\gamma - L_{\mathbf{zz}}}\epsilon$. Taking conditional expectations in the bound (A.6) and using $\mathbb{E}[\nabla_\theta \phi_\gamma(\theta^t; z^t) \mid \theta^t] = \nabla F(\theta^t)$, we have

$$\begin{aligned}\mathbb{E}[F(\theta^{t+1}) - F(\theta^t) \mid \theta^t] &\leq -\frac{\alpha_t}{2}\left\|\nabla F(\theta^t)\right\|_2^2 + \frac{\alpha_t(1 + L_\phi \alpha_t)}{2}\widehat{\epsilon} + L_\phi \alpha_t^2 \left\|\nabla_\theta \phi_\gamma(\theta; z^t) - \nabla F(\theta^t)\right\|_2^2 \\ &\leq -\frac{\alpha_t}{2}\left\|\nabla F(\theta^t)\right\|_2^2 + \alpha_t \widehat{\epsilon} + L_\phi \alpha_t^2 \left\|\nabla_\theta \phi_\gamma(\theta; z^t) - \nabla F(\theta^t)\right\|_2^2,\end{aligned}$$

where we use the fact that $\alpha_t \leq \frac{1}{L_\phi}$. For a fixed stepsize α, taking the total expectation yields

$$\mathbb{E}\left[\left\|\nabla F(\theta^t)\right\|_2^2\right] - 2\widehat{\epsilon} \leq \frac{2}{\alpha}\mathbb{E}[F(\theta^t) - F(\theta^{t+1})] + 2L_\phi \alpha \sigma^2$$

since we have $\mathbb{E}[\|\nabla \phi_\gamma(\theta; Z) - \nabla F(\theta)\|_2^2] \leq \sigma^2$ by assumption. Summing over t, we have

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\left\|\nabla F(\theta^t)\right\|_2^2\right] - 2\widehat{\epsilon} &\leq \frac{2}{\alpha T}\left(F(\theta^0) - \mathbb{E}[F(\theta^T)]\right) + 2L_\phi \alpha \sigma^2 \\ &\leq \frac{2\Delta_F}{\alpha T} + 2L_\phi \alpha \sigma^2,\end{aligned}$$

where the latter inequality holds since $\inf_\theta F(\theta) \leq F(\theta^T)$. Plugging in $\alpha = \sqrt{\frac{\Delta_F}{L_\phi T \sigma^2}}$ gives the result.

A.2.4 Proof of Lemma 2.2

First, we introduce the decision reformulation of the problem: for some b, we ask whether there exists some u such that $\ell(\theta; z+u) \geq b$. The decision reformulation for an NPO problem is in NP, as a certificate for the decision problem can be verified in polynomial time. By appropriate scaling of θ, v, and w, Katz et al. [152] show that 3-SAT Turing-reduces to this decision problem: given an oracle D for the decision problem, we can solve an arbitrary instance of 3-SAT with a polynomial number of calls to D. The decision problem is thus NP-complete.

Now, consider an oracle O for the optimization problem. The decision problem Turing-reduces to the optimization problem, as the decision problem can be solved with one call to O. Thus, the optimization problem is NP-hard.

A.2.5 Proof of Theorem 2.2

We first show the bound (2.11). From the duality result (2.5), we have the deterministic result that

$$\sup_{P:W_c(P,Q)\le\rho} \mathbb{E}_Q[\ell(\theta;Z)] \le \gamma\rho + \mathbb{E}_Q[\phi_\gamma(\theta;Z)]$$

for all $\rho > 0$, distributions Q, and $\gamma \ge 0$. Next, we show that $\mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta;Z)]$ concentrates around its population counterpart at the usual rate [47].

First, we have that

$$\phi_\gamma(\theta;z) \in [-M_\ell, M_\ell],$$

because $-M_\ell \le \ell(\theta;z) \le \phi_\gamma(\theta;z) \le \sup_z \ell(\theta;z) \le M_\ell$. Thus, the functional $\theta \mapsto F_n(\theta)$ satisfies bounded differences [48, Thm. 6.2], and applying standard results on Rademacher complexity [22] and entropy integrals [288, Ch. 2.2] gives the result.

To see the second result (2.12), we substitute $\rho = \widehat{\rho}_n$ in the bound (2.11). Then, with probability at least $1 - e^{-t}$, we have

$$\sup_{P:W_c(P,P_0)\le\widehat{\rho}_n(\theta)} \mathbb{E}_P[\ell(\theta;Z)] \le \gamma\widehat{\rho}_n(\theta) + \mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta;Z)] + \epsilon_n(t).$$

Since we have

$$\sup_{P:W_c(P,\widehat{P}_n)\le\widehat{\rho}_n(\theta)} \mathbb{E}_P[\ell(\theta;Z)] = \mathbb{E}_{\widehat{P}_n}[\phi_\gamma(\theta;Z)] + \gamma\widehat{\rho}_n(\theta).$$

from the strong duality in Proposition 2.1, our second result follows.

A.2.6 Proof of Corollary 2.2

The result is essentially standard [288], which we now give for completeness. Note that for $\mathcal{F} = \{\ell(\theta;\cdot) : \theta \in \Theta\}$, any $(\epsilon, \|\cdot\|)$-covering $\{\theta_1, \ldots, \theta_N\}$ of Θ guarantees that $\min_i |\ell(\theta;X) - \ell(\theta_i;X)| \le L\epsilon$ for all θ, X, or

$$N(\mathcal{F}, \epsilon, \|\cdot\|_{L^\infty(\mathcal{X})}) \le N(\Theta, \epsilon/L, \|\cdot\|) \le \left(1 + \frac{\mathrm{diam}(\Theta)L}{\epsilon}\right)^d,$$

where $\operatorname{diam}(\Theta) = \sup_{\theta,\theta' \in \Theta} \|\theta - \theta'\|$. Noting that $|\ell(\theta; Z)| \leq L \operatorname{diam}(\Theta) + M_0 =: M_\ell$, we have the result.

A.2.7 Proof of Theorem 2.3

Define

$$P_n^*(\theta) := \operatorname*{argmax}_P \left\{ \mathbb{E}_P[\ell(\theta; Z)] - \gamma W_c(P, \widehat{P}_n) \right\},$$
$$P^*(\theta) := \operatorname*{argmax}_P \left\{ \mathbb{E}_P[\ell(\theta; Z)] - \gamma W_c(P, P_0) \right\}.$$

First, we show that $P^*(\theta)$ and $P_n^*(\theta)$ are attained for all $\theta \in \Theta$. We omit the dependency on θ for notational simplicity and only show the result for $P^*(\theta)$ as the case for $P_n^*(\theta)$ is symmetric. Let P^ϵ be an ϵ-maximizer, so that

$$\mathbb{E}_{P^\epsilon}[\ell(\theta; Z)] - \gamma W_c(P^\epsilon, P_0) \geq \sup_P \left\{ \mathbb{E}_P[\ell(\theta; Z)] - \gamma W_c(P_n, P_0) \right\} - \epsilon.$$

As $\mathcal{Z}$ is compact, the collection $\{P^{1/k}\}_{k \in \mathbb{N}}$ is a uniformly tight collection of measures. By Prohorov's theorem [36, Ch 1.1, p. 57], (restricting to a subsequence if necessary), there exists some distribution P^* on $\mathcal{Z}$ such that $P^{1/k} \overset{d}{\rightsquigarrow} P^*$ as $k \to \infty$. Continuity properties of Wasserstein distances [290, Corollary 6.11] then imply that

$$\lim_{k \to \infty} W_c(P^{1/k}, P_0) = W_c(P^*, P_0). \tag{A.7}$$

Combining (A.7) and the monotone convergence theorem, we obtain

$$\begin{aligned} \mathbb{E}_{P^*}[\ell(\theta; Z)] - \gamma W_c(P^*, P_0) &= \lim_{k \to \infty} \left\{ \mathbb{E}_{P^{1/k}}[\ell(\theta; Z)] - \gamma W_c(P^{1/k}, P_0) \right\} \\ &\geq \sup_P \left\{ \mathbb{E}_P[\ell(\theta; Z)] - \gamma W_c(P, P_0) \right\}. \end{aligned}$$

We conclude that P^* is attained for all P_0.

Next, we show the concentration result (2.18). Recall the definition (2.9) of the transportation mapping

$$T(\theta, z) := \operatorname*{argmax}_{z' \in \mathcal{Z}} \left\{ \ell(\theta; z') - \gamma c(z', z) \right\},$$

which is unique and well-defined under our strong concavity assumption that $\gamma > L_{\mathsf{zz}}$,

and smooth (recall Eq. (2.16)) in θ. Then by Proposition 2.1 (or by using a variant of Kantorovich duality [290, Chs. 9–10]), we have

$$\mathbb{E}_{P_n^*(\theta)}[\ell(\theta; Z) = \mathbb{E}_{\widehat{P}_n}[\ell(\theta; T(\theta; Z))] \quad \text{and} \quad \mathbb{E}_{P^*(\theta)}[\ell(\theta; Z) = \mathbb{E}_{P_0}[\ell(\theta; T(\theta; Z))]$$
$$W_c(P_n^*(\theta), \widehat{P}_n) = \mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), Z)] \quad \text{and} \quad W_c(P^*(\theta), P_0) = \mathbb{E}_{P_0}[c(T(\theta; Z), Z)].$$

We now proceed by showing the uniform convergence of

$$\mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), Z)] \quad \text{to} \quad \mathbb{E}_{P_0}[c(T(\theta; Z), Z)]$$

under both cases (i), that c is Lipschitz, and (ii), that ℓ is Lipschitz in z, using a covering argument on Θ. Recall inequality (2.16) (i.e. Lemma 2.1), which is that

$$\|T(\theta_1; z) - T(\theta_2; z)\| \le \frac{L_{\mathsf{z}\theta}}{[\gamma - L_{\mathsf{zz}}]_+} \|\theta_1 - \theta_2\| .$$

We have the following lemma.

Lemma A.2. *Assume the conditions of Theorem 2.3. Then for any $\theta_1, \theta_2 \in \Theta$,*

$$|c(T(\theta_1; z), z) - c(T(\theta_2; z), z)| \le \frac{L_{\mathsf{c}} L_{\mathsf{z}\theta}}{[\gamma - L_{\mathsf{zz}}]_+} \|\theta_1 - \theta_2\| .$$

Proof In the first case, that c is L_{c}-Lipschitz in its first argument, this is trivial: we have

$$|c(T(\theta_1; z), z) - c(T(\theta_2; z), z)| \le L_{\mathsf{c}} \|T(\theta_1; z) - T(\theta_2; z)\| \le \frac{L_{\mathsf{c}} L_{\mathsf{z}\theta}}{[\gamma - L_{\mathsf{zz}}]_+} \|\theta_1 - \theta_2\|$$

by the smoothness inequality (2.16) for T.

In the second case, that $z \mapsto \ell(\theta, z)$ is L_{c}-Lipschitz, let $z_i = T(\theta_i; z)$ for shorthand. Then we have

$$\begin{aligned}\gamma c(z_2, z) - \gamma c(z_1, z) &= \gamma c(z_2, z) - \ell(\theta_2, z_2) + \ell(\theta_2, z_2) - \gamma c(z_1, z) \\ &\le \gamma c(z_1, z) - \ell(\theta_2, z_1) + \ell(\theta_2, z_2) - \gamma c(z_1, z) = \ell(\theta_2, z_2) - \ell(\theta_2, z_1),\end{aligned}$$

and similarly,

$$\begin{aligned}\gamma c(z_2, z) - \gamma c(z_1, z) &= \gamma c(z_2, z) - \ell(\theta_1, z_1) + \ell(\theta_1, z_1) - \gamma c(z_1, z) \\ &\geq \gamma c(z_2, z) - \ell(\theta_1, z_1) + \ell(\theta_1, z_2) - \gamma c(z_2, z) = \ell(\theta_1, z_2) - \ell(\theta_1, z_1).\end{aligned}$$

Combining these two inequalities and using that

$$|\ell(\theta, z_2) - \ell(\theta, z_1)| \leq \gamma L_{\mathsf{c}} \|z_2 - z_1\|$$

for any θ gives the result. □

Using Lemma A.2 we obtain that $\theta \mapsto |\mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), \theta)] - \mathbb{E}_{P_0}[c(T(\theta; Z), Z)]|$ is $2L_{\mathsf{c}}L_{\mathsf{z}\theta}/[\gamma - L_{\mathsf{zz}}]_+$-Lipschitz. Let $\Theta_{\text{cover}} = \{\theta_1, \cdots, \theta_N\}$ be a $\frac{[\gamma - L_{\mathsf{zz}}]_+ t}{4L_{\mathsf{c}}L_{\mathsf{z}\theta}}$-cover of Θ with respect to $\|\cdot\|$. From Lipschitzness of $|\mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), Z)] - \mathbb{E}_{P_0}[c(T(\theta; Z), Z)]|$, we have that if for all $\theta \in \{\Theta_{\text{cover}}\}$,

$$|\mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), Z)] - \mathbb{E}_{P_0}[c(T(\theta; Z), \theta)]| \leq \frac{t}{2},$$

then it follows that

$$\sup_{\theta \in \Theta} |\mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), Z)] - \mathbb{E}_{P_0}[c(T(\theta; Z), Z)]| \leq t.$$

Under the first assumption (i), we have $|c(T(\theta; Z), Z)| \leq 2L_{\mathsf{c}}M_{\mathsf{z}}$. Applying Hoeffding's inequality, for any fixed $\theta \in \Theta$

$$\mathbb{P}\left(|\mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), Z)] - \mathbb{E}_{P_0}[c(T(\theta; Z), Z)]| \geq \frac{t}{2}\right) \leq 2\exp\left(-\frac{nt^2}{32L_{\mathsf{c}}^2 M_{\mathsf{z}}^2}\right).$$

Taking a union bound over $\theta_1, \cdots, \theta_N$, we conclude that

$$\mathbb{P}\left(\sup_{\theta \in \Theta} |\mathbb{E}_{\widehat{P}_n}[c(T(\theta; Z), Z)] - \mathbb{E}_{P_0}[c(T(\theta; Z), Z)]| \geq t\right) \leq 2N\left(\Theta, \frac{[\gamma - L_{\mathsf{zz}}]_+ t}{4L_{\mathsf{c}}L_{\mathsf{z}\theta}}, \|\cdot\|\right) \exp\left(-\frac{nt^2}{32L_{\mathsf{c}}^2 M_{\mathsf{z}}^2}\right)$$

which was our desired result (2.18).

Under the second assumption (ii), we have from the definition of the transport map T

$$\gamma c(T(\theta; z), z) \leq \ell(\theta; z) \leq M_\ell$$

and hence $|c(T(\theta;Z),Z)| \leq M_\ell/\gamma$. The result for the second case follows from an identical reasoning.

A.2.8 Proof of Proposition 2.2

From the chain rule, we can write down the Jacobian $J_x F_l(\theta;x)$ of $x \mapsto F_l(\theta;x)$ recursively.

Lemma A.3. *If σ_l is differentiable for all $l = 1,\dots,L$, then $x \mapsto F_l(\theta;x)$ is differentiable with*

$$J_x F_l(\theta;x) = \nabla\sigma_l(\theta_l \cdot F_{l-1}(\theta;x)) \cdot \theta_l \cdot J_x F_{l-1}(\theta;x)$$

for all $l = 1,\dots,L$. Using the convention $\prod_{l=1}^{L} A_l = A_L \cdots A_1$ for matrix products,

$$J_x F_l(\theta;x) = \prod_{l=1}^{L} \nabla\sigma_l(\theta_l \cdot F_{l-1}(\theta;x)) \cdot \theta_l.$$

To prove the first result of the proposition, we proceed by induction. For $l = 1$ and any $x, x' \in \mathcal{X}$,

$$\begin{aligned}\left\|F_1(\theta;x) - F_1(\theta;x')\right\|_2 &= \left\|\sigma_1(\theta_1 \cdot x) - \sigma_1(\theta_1 \cdot x')\right\|_2 \\ &\leq L_1^0 \left\|\theta_1\right\|_{\mathrm{op}} \left\|x - x'\right\|_2 .\end{aligned}$$

By induction, we conclude that for $l \geq 2$,

$$\begin{aligned}\left\|F_l(\theta;x) - F_l(\theta;x')\right\|_2 &= \left\|\sigma_l(\theta_l \cdot F_{l-1}(\theta;x)) - \sigma_l(\theta_l \cdot F_{l-1}(\theta;x'))\right\|_2 \\ &\leq L_l^0 \left\|\theta_l\right\|_{\mathrm{op}} \left\|F_{l-1}(\theta;x) - F_{l-1}(\theta;x')\right\|_2 \\ &\leq \left\|x - x'\right\|_2 \prod_{j=1}^{l} L_j^0 \left\|\theta_j\right\|_{\mathrm{op}} .\end{aligned}$$

To show that $JF_l(\theta;\cdot)$ is Lipschitz with respect to the operator norm, note from Lemma A.3

that

$$\begin{aligned} J_x F_l(\theta; x) - J_x F_l(\theta; x') &= \prod_{j=1}^{l} \left(\nabla\sigma_j(\theta_j \cdot F_{j-1}(\theta; x)) - \nabla\sigma_j(\theta_j \cdot F_{j-1}(\theta; x'))\right) \cdot \theta_j \\ &= \sum_{j=1}^{l} \underbrace{\left(\prod_{k=j+1}^{l} \nabla\sigma_k(\theta_k \cdot F_{k-1}(\theta; x')) \cdot \theta_k\right)}_{(a)} \\ &\qquad \cdot \underbrace{\left(\nabla\sigma_j(\theta_j \cdot F_{j-1}(\theta; x)) - \nabla\sigma_j(\theta_j \cdot F_{j-1}(\theta; x'))\right)}_{(b)} \\ &\qquad \cdot \underbrace{\theta_j \cdot J_x F_{j-1}(\theta; x)}_{(c)} \end{aligned} \tag{A.8}$$

where the last equality followed from telescoping summands. Here, we let $F_0(\theta; x) = x$ notational convenience and define the product $\prod_a^b \cdot = 1$ when $a > b$.

We now bound the operator norms of the three terms (a), (b), and (c). From Assumption 2.5 and the submultiplicativity of the operator norm, term (a) can be bounded by

$$\left\| \prod_{k=j+1}^{l} \nabla\sigma_k(\theta_k \cdot F_{k-1}(\theta; x')) \cdot \theta_k \right\|_{\text{op}} \leq \prod_{k=j+1}^{l} L_k^0 \left\|\theta_k\right\|_{\text{op}}.$$

From Assumption 2.5, we can bound term (b) by

$$\begin{aligned} \left\|\nabla\sigma_j(\theta_j \cdot F_{j-1}(\theta; x)) - \nabla\sigma_j(\theta_j \cdot F_{j-1}(\theta; x'))\right\|_{\text{op}} &\leq L_j^1 \left\|\theta_j \cdot (F_{j-1}(\theta; x) - F_{j-1}(\theta; x'))\right\|_2 \\ &\leq L_j^1 \left\|\theta_j\right\|_{\text{op}} \left(\prod_{k=1}^{j-1} L_k^0 \left\|\theta_k\right\|_{\text{op}}\right) \left\|x - x'\right\|_2 \end{aligned}$$

where the last inequality follows from the first part of the proof. Lastly, term (c) is bounded by from the first part of the proof

$$\left\|\theta_j \cdot J_x F_{j-1}(\theta; x)\right\|_{\text{op}} \leq \left\|\theta_j\right\|_{\text{op}} \prod_{k=1}^{j-1} L_k^0 \left\|\theta_k\right\|_{\text{op}}.$$

Collecting these bounds and applying them in the representation (A.8), triangle inequality and submultiplicativity of the operator norm yields

$$\begin{aligned}\left\|J_x F_l(\theta;x) - J_x F_l(\theta;x')\right\|_{\mathrm{op}} \leq \sum_{j=1}^{l} \left(\prod_{k=j+1}^{l} L_k^0 \left\|\theta_k\right\|_{\mathrm{op}}\right) \cdot L_j^1 \cdot \left\|\theta_j\right\|_{\mathrm{op}} \cdot \left(\prod_{k=1}^{j-1} L_k^0 \left\|\theta_k\right\|_{\mathrm{op}}\right) \\ \cdot \left\|x - x'\right\|_2 \cdot \left\|\theta_j\right\|_{\mathrm{op}} \cdot \left(\prod_{k=1}^{j-1} L_k^0 \left\|\theta_k\right\|_{\mathrm{op}}\right) \\ = \beta_l(\theta) \left\|x - x'\right\|_2 .\end{aligned}$$

A.2.9 Proof of Corollary 2.4

Denote the softmax function $\sigma_y : \mathbb{R}^K \to \mathbb{R}$

$$\sigma_y(x) := -\log p_y(x) = -\log \frac{\exp(x_y)}{\sum_{j=1}^{K} \exp(x_j)}.$$

We consider calculating L^0 and L^1 for this map. Note that

$$\nabla \sigma_y(x) = p(x) - e_y, \nabla^2 \sigma_y(x) = \operatorname{diag} p(x) - p(x)p(x)^T,$$

where $p(x) = \sum_{k=1}^{K} p_k(x) e_k$. Then,

$$L^0 = \sup_x \left\|\nabla \sigma_y(x)\right\|_2 = \sup_x \left\|p(x) - e_y\right\|_2 = \sqrt{2},$$

where the last equality results from the fact that $g(z) := \left\|z - e_y\right\|_2$ is convex in z and therefore $\sup_{z \in \mathcal{Z}} g(z)$ is attained at a corner of the probability simplex $\mathcal{Z} := \{z | z \geq 0, \sum_i z_i = 1\}$, namely any corner other than e_y.

The eigenvalues of $\operatorname{diag} p(x)$ are $p_i(x)$ and therefore satisfy $\lambda(\operatorname{diag} p(x)) \in [0, 1]$. The eigenvalues of $p(x)p(x)^T$ satisfy $\lambda(p(x)p(x)^T) \in \{0, \left\|p(x)\right\|_2^2\}$. Then, by Weyl's inequality, we have $\lambda(\nabla^2 \sigma_y(x)) \in [-\left\|p(x)\right\|_2^2, 1]$. Again using convexity of $\left\|\cdot\right\|_2$, we have $\sup_x \left\|p(x)\right\|_2 = 1$, as the supremum is attained at a corner of the probability simplex. Then $\lambda(\nabla^2 \sigma_y(x)) \in [-1, 1]$, whereby $L^1 = 1$.

An equivalent recursive way of expressing the Lipschitz constants $\alpha_l(\theta)$ and $\beta_l(\theta)$ (2.21)

is

$$\alpha_{l+1}(\theta) = L_{l+1}^0 \|\theta_{l+1}\|_{\text{op}} \alpha_l(\theta)$$
$$\beta_{l+1}(\theta) = \frac{\alpha_{l+1}(\theta)}{\alpha_l(\theta)}\beta_l(\theta) + \frac{L_{l+1}^1}{(L_{l+1}^0)^2}\alpha_{l+1}(\theta)^2,$$

with $\beta_0(\theta) = \alpha_0(\theta) = 1$. Then, considering an $(L+1)$-layer network where $\theta_{L+1} = I$ and σ_{L+1} is the softmax function, we have

$$\beta_{L+1}(\theta) = \sqrt{2}\beta_L(\theta) + \alpha_L(\theta)^2.$$

A.3 Proximal algorithm for $\|\cdot\|_\infty$-norm robustness

In this section, we give a efficient training algorithm that learns to defend against $\|\cdot\|_\infty$-norm perturbations. For simplicity, we assume $\mathcal{Z} = \mathbb{R}^m$ for the rest of this section. Let $\theta \in \Theta$ be some fixed model, $z^0 \in \mathcal{Z}$ a natural example[1] and define $f(z) := \ell(\theta; z)$ to ease notation. Concretely, we are interested in solving the optimization problem

$$\underset{z}{\text{maximize}}\, f(z) - \frac{\alpha}{2}\left\|z - z^0\right\|_\infty^2$$

Note that this is equivalent to computing the surrogate loss $\phi_\gamma(\theta; z^0) = \sup_{z\in\mathcal{Z}}\{\ell(\theta; z) - \gamma c(z, z^0)\}$ for $c(z, z^0) = \left\|z - z^0\right\|_\infty^2$ and $\alpha = 2\gamma$. Our following treatment can easily be modified for the supervised learning scenario $c((x, y), (x^0, y^0)) = \left\|x - x^0\right\|_\infty^2 + \infty \cdot \mathbf{1}\left\{y = y^0\right\}$ with the convention that $\infty \cdot 0 = 0$. To make our notation consistent with the optimization literature, we consider the minimization problem

$$\underset{z}{\text{minimize}} -f(z) + \frac{\alpha}{2}\left\|z - z^0\right\|_\infty^2. \tag{A.9}$$

A simple gradient descent algorithm applied to the problem (A.9) may be slow to converge in practice. Intuitively, this is because the subgradient of $z \mapsto \frac{1}{2}\left\|z - z^0\right\|_\infty^2$ is given by $\left\|z - z^0\right\|_\infty \cdot s$ where s is a m-dimensional vector taking values in $[-1, 1]$ whose coordinates are non-zero only when $|z_j - z_{0,j}| = \left\|z - z^0\right\|_\infty$. Hence, at any given iteration of gradient descent, the $\|\cdot\|_\infty$-norm penalty term only gets accounted for by at most a few coordinates.

To remedy this issue, we consider a proximal algorithm for solving the problem (A.9)

[1] We depart from our convention of denoting original datapoints as z_0 to ease forthcoming notation.

(see, for example, Parikh and Boyd [228] for an comprehensive review of proximal algorithms). For a function $g : \mathcal{Z} \to \mathbb{R}$ and a positive number $\lambda > 0$, the proximal operator for λg is defined by

$$\text{prox}_{\lambda g}(v) := \underset{z}{\text{argmin}} \left\{ g(z) + \frac{1}{2\lambda} \|z - v\|_2^2 \right\}.$$

Then, the proximal algorithm on the problem (A.9) consists of two steps at each iteration t: (i) for the smooth function $-f(z)$, take a gradient descent step at the current iterate z^t ($z^{t+\frac{1}{2}}$ below) and (ii) for the non-smooth function $\left\|z - z^0\right\|_\infty^2$, take a proximal step for the function $\frac{\lambda^t \alpha}{2} \left\|\cdot - z^0\right\|_\infty^2$ at $z^{t+\frac{1}{2}}$ (z^{t+1} below):

$$z^{t+\frac{1}{2}} = z^t + \lambda^t \nabla f(z^t), \qquad z^{t+1} = \text{prox}_{\frac{\lambda^t \alpha}{2} \|\cdot - z^0\|_\infty^2} \left(z^{t+\frac{1}{2}} \right). \tag{A.10}$$

The following proposition shows that we can compute the proximal step z^{t+1} efficiently, simply by sorting the vector $|z^{t+\frac{1}{2}} - z^0|$. We denote by v^t, the sorted vector of $|z^{t+\frac{1}{2}} - z^0|$ in **decreasing** order. In the proposition, we use the notation $[\cdot]_+ = \max(\cdot, 0)$.

Proposition A.1. *Define the scale parameter $\beta^t > 0$ by*

$$\beta^t := \frac{1}{1 + \alpha\lambda^t j^t} \sum_{i=1}^{j^t} v_i^t \quad \textit{where} \quad j^t := \max \left\{ j \in [m] : \sum_{i=1}^{j-1} v_i - \left(\frac{1}{\alpha\lambda^t} + (j-1) \right) v_j < 0 \right\}. \tag{A.11}$$

Then, z^{t+1} in the proximal update (A.10) is given by

$$z^{t+1} = z^{t+\frac{1}{2}} - \left[|z^{t+\frac{1}{2}} - z^0| - \beta^t \right]_+ \text{sign} \left(z^{t+\frac{1}{2}} - z^0 \right). \tag{A.12}$$

See Section A.3.1 for the proof of the proposition. From the proposition, we obtain the proximal procedure in Algorithm A.1 that can be used to (heuristically) solve for the approximate maximizer of $\ell(\theta; z) - \gamma c(z, z^0)$ in Algorithm 2.1. Roughly speaking, ignoring the truncation term in the proximal update (A.12), we have

$$z^{t+1} \approx z^0 + \beta^t \,\text{sign}(z^t + \lambda^t \nabla f(z^t) - z^0).$$

Here, we move towards the sign of $z^t + \lambda^t \nabla f(z^t) - z^0$ modulated by the term β^t, as opposed to just the sign of $\nabla f(z^t)$ for the iterated fast sign gradient method [116, 170].

Algorithm A.1 Proximal Algorithm for Maximizing $f(z) - \frac{\alpha}{2}\left\|z - z^0\right\|_\infty^2$

INPUT: Stepsizes λ^t
for $t = 0, \ldots, T-1$ **do**
  $z^{t+\frac{1}{2}} \leftarrow z^t + \lambda^t \nabla f(z^t)$
  $v^t \leftarrow \text{sort}(|z^{t+\frac{1}{2}} - z^0|, \text{dec})$
  Compute β^t as in (A.11)
  $z^{t+1} \leftarrow z^{t+\frac{1}{2}} - \left[|z^{t+\frac{1}{2}} - z^0| - \beta^t\right]_+ \text{sign}\left(z^{t+\frac{1}{2}} - z^0\right)$

A.3.1 Proof of Proposition A.1

In this proof, we drop the subscript on the iteration t to ease notation. We assume without loss of generality that $z^{t+\frac{1}{2}} - z^0 \neq 0$. For some convex, lower semi-continuous function $g : \mathbb{R}^m \to \mathbb{R}$, let $g^*(s) = \sup_s\{s^\top t - g(t)\}$ be the Fenchel conjuagte of g. From the Moreau decomposition [228, Section 2.5], we have

$$\text{prox}_g(w) + \text{prox}_{g^*}(w) = w$$

for any $w \in \mathbb{R}^m$. Noting that the conjugate of $z \mapsto \frac{\alpha\lambda}{2}\left\|z - z_0\right\|_\infty^2$ is given by $z \mapsto z^\top z_0 + \frac{1}{2\alpha\lambda}\left\|z\right\|_1^2$, we have

$$\text{prox}_{\frac{\alpha\lambda}{2}\|\cdot - z_0\|_\infty^2}(w) = w - \text{prox}_{\langle z^0, \cdot\rangle + \frac{1}{2\alpha\lambda}\|\cdot\|_1^2}(w) = w - \text{prox}_{\frac{1}{2\alpha\lambda}\|\cdot\|_1^2}(w - z^0)$$

Let us denote the sorted vector (in decreasing order) of $|w - z^0|$ by v. Then, in light of the preceeding display, it suffices to show that

$$\text{prox}_{\frac{1}{2\alpha\lambda}\|\cdot\|_1^2}(w - z^0) = [|w - z| - \beta^\star]_+ \text{sign}\left(w - z^0\right) \tag{A.13}$$

where $\beta^\star$ is defined as in (A.11). To show that equality (A.13) holds, note that the first order optimality conditions for

$$\text{prox}_{\frac{1}{2\alpha\lambda}\|\cdot\|_1^2}(w - z^0) = \underset{z}{\text{argmin}}\left\{\frac{1}{2}\left\|z\right\|_1^2 + \frac{\alpha\lambda}{2}\left\|z - (w - z^0)\right\|_2^2\right\}$$

is given by

$$\|z\|_1 \operatorname{sign}(z_i) + \alpha\lambda(z_i - w_i + z_i^0) = 0 \quad \text{if} \ \ |z_i| \neq 0 \tag{A.14a}$$

$$\|z\|_1 [-1, 1] - \alpha\lambda(w_i - z_i^0) \ni 0 \quad \text{if} \ \ |z_i| = 0. \tag{A.14b}$$

Now, we use the following elementary lemma.

Lemma A.4. *For $0 \neq v \geq 0$ with decreasing coordinates, the solution to the equation*

$$\sum_{i:v_i>\beta} (v_i - \beta) = \alpha\lambda\beta$$

exists and is given by

$$\beta^\star := \frac{1}{1 + \alpha\lambda j^\star} \sum_{i=1}^{j} v_i \quad \textit{where} \ \ j^\star := \max \left\{ j \in [m] : \sum_{i=1}^{j-1} v_i - \left(\frac{1}{\alpha\lambda} + (j-1) \right) v_j < 0 \right\}.$$

Proof of Lemma First, note that $\beta \mapsto \sum_{i:v_i>\beta}(v_i - \beta) - \alpha\lambda\beta =: h(\beta)$ is decreasing. Noting that $\|v\|_1 > 0$ and $-\alpha\lambda \|v\|_\infty < 0$, there exists β' such that $h(\beta') = 0$ and $\beta' \in (0, \|v\|_\infty)$. Since v_i's are decreasing and nonnegative, there exists j' such that $v_{j'} > \beta' \geq v_{j'+1}$ (we abuse notation and let $v_{m+1} := 0$). Then, we have

$$\sum_{i=1}^{j'-1}(v_i - v_{j'}) - \alpha\lambda v_{j'} < 0 \leq \sum_{i=1}^{j'}(v_i - v_{j'+1}) - \alpha\lambda v_{j'+1}.$$

That is, $j' = j^\star$. Solving for β' in

$$0 = h(\beta') = \sum_{i=1}^{j^\star} v_i - (\alpha\lambda + j^\star)\, \beta',$$

we obtain $\beta' = \beta^\star$ as claimed. □

Now, define

$$z^\star = \left[|w - z^0| - \beta^\star \right]_+ \operatorname{sign}\left(w - z^0\right).$$

Then, we have from Lemma A.4 that

$$\|z^\star\|_1 = \sum_{i:|w_i - z_i^0| > \beta^\star} (|w_i - z_i^0| - \beta^\star) = \sum_{i=1}^{j^\star} (v_i - \beta^\star) = \alpha\lambda\beta^\star.$$

If $z_i^\star > 0$, then $\operatorname{sign}(z_i^\star) = \operatorname{sign}(w_i - z_i^0)$ so that

$$\|z^\star\|_1 \operatorname{sign}(z_i) + \alpha\lambda(z_i^\star - w_i + z_i^0) = 0.$$

If $z_i^\star = 0$, then $|w_i - z_i^0| \leq \beta^\star$ and

$$\|z^\star\|_1 [-1, 1] - \alpha\lambda(w_i - z_i^0) = \alpha\lambda\beta^\star[-1, 1] - \alpha\lambda(w_i - z_i^0) \ni 0.$$

Hence, $z^\star$ satisfies the optimality condition (A.14) as desired.

Appendix B

Chapter 3 Appendices

B.1 Offline population synthesis

Here we provide extra details for Section 3.2.

Horizontal steps Horizontal steps occur as follows. Two random particles are sampled uniformly at random from adjacent temeprature levels. This forms a proposal for the swap, which is then accepted via standard MH acceptance conditions. Because the rest of the particles remain as-is, the acceptance condition reduces to a particualrly simple form (*cf.* Algorithm B.1).

Algorithm B.1 Horizontal swap

Sample $i \sim \text{Uniform}(1, 2, \ldots, L-1)$.
Sample $j, k \overset{\text{i.i.d.}}{\sim} \text{Uniform}(1, 2, \ldots, D)$.
Sample $p \sim \text{Uniform}([0, 1])$
Let $a = \min\left(1, e^{f(x^{i,j}, \theta^{i,j}) - f(x^{i+1,k}, \theta^{i+1,k})}\right)$
if $p < a^{\beta_i - \beta_{i+1}}$
  swap configurations $(x^{i,j}, \theta^{i,j})$ and $(x^{i+1,k}, \theta^{i+1,k})$

We ran our experiments on a server with 88 Intel Xeon cores @ 2.20 GHz. Each run of 100 iterations for a given hyperparameter setting α took 20 hours.

B.2 Online robust planning

Here we provide extra details for Section 3.3.

B.2.1 Solving problem (3.5)

We can rewrite the constraint $D_\phi\left(q \| \mathbf{1}/N_w\right) \leq \rho$ as $\|q - \mathbf{1}/N_w\|^2 \leq \rho/N_w$. Then, the partial Lagrangian can be written as

$$\mathcal{L}(q, \lambda) = \sum_i q_i c_i(t) - \frac{\lambda}{2}\left(\|q - \mathbf{1}/N_w\|^2 - \rho/N_w\right).$$

By inspection of the right-hand side, we see that, for a given λ, finding $v(\lambda) = \sup_{q \in \Delta} \mathcal{L}(q, \lambda)$ is equivalent to a Euclidean-norm projection of the vector $\mathbf{1}/N_w + c(t)/\lambda$ onto the probability simplex Δ. This latter problem is directly amenable to the methods of Duchi et al. [87].

B.2.2 Proof of Proposition 3.1

We redefine notation to suppress dependence of the cost C on other variables and just make explicit the dependence on the random index J. Namely, we let $C : \mathcal{J} \to [-1, 1]$ be a function of the random index J. We consider the convergence of

$$\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C(J)] \quad \text{to} \quad \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[C(J)].$$

To ease notation, we hide dependence on J and for a sample $J_1, \ldots, J_{N_w}$ of random vectors J_k, we denote $C_k := C(J_k)$ for shorthand, so that the C_k are bounded independent random variables. Our proof technique is similar in style to that of Sinha and Duchi [264]. We provide proofs for technical lemmas that follow in support of Proposition 3.1 that are shorter and more suitable for our setting (in particular Lemmas B.1 and B.3).

Treating $C = (C_1, \ldots, C_{N_w})$ as a vector, the mapping $C \mapsto \sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C]$ is a $\sqrt{\rho + 1}/\sqrt{N_w}$-Lipschitz convex function of independent bounded random variables. Indeed, letting $q \in \mathbb{R}_+^{N_w}$ be the empirical probability mass function associated with $Q \in \mathcal{P}_{N_w}$, we have $\frac{1}{N_w}\sum_{i=1}^{N_w}(N_w q_i)^2 \leq \rho + 1$ or $\|q\|_2 \leq \sqrt{(1+\rho)/N_w}$. Using Samson's sub-Gaussian concentration inequality [253] for Lipschitz convex functions of bounded random variables, we have with probability at least $1 - \delta$ that

$$\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C] \in \mathbb{E}\left[\sup_{Q \in \mathcal{P}_{N_w}} \mathbb{E}_Q[C]\right] \pm 2\sqrt{2}\sqrt{\frac{(1+\rho)\log\frac{2}{\delta}}{N_w}}. \tag{B.1}$$

By the containment (B.1), we only need to consider convergence of

$$\mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[C]\right] \quad \text{to} \quad \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[C],$$

which we do with the following lemma.

Lemma B.1 (Sinha and Duchi [264]). *Let $Z = (Z_1, \ldots, Z_{N_w})$ be a random vector of independent random variables $Z_i \overset{\text{i.i.d.}}{\sim} P_0$, where $|Z_i| \le M$ with probability 1. Let $C_\rho = \frac{2(\rho+1)}{\sqrt{1+\rho}-1}$. Then*

$$\mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] \ge \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z] - 4C_\rho M\sqrt{\frac{\log(2N_w)}{N_w}}$$

and

$$\mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] \le \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z].$$

See Appendix B.2.3 for the proof.

Combining Lemma B.1 with containment (B.1) gives the result.

B.2.3 Proof of Lemma B.1

Before beginning the proof, we first state a technical lemma.

Lemma B.2 (Ben-Tal et al. [28]). *Let ϕ be any closed convex function with domain $\operatorname{dom}\phi \subset [0,\infty)$, and let $\phi^*(s) = \sup_{t\ge 0}\{ts - \phi(t)\}$ be its conjugate. Then for any distribution P and any function $g : \mathcal{W} \to \mathbb{R}$ we have*

$$\sup_{Q:D_\phi(Q\|P)\le\rho} \int g(w)dQ(w) = \inf_{\lambda\ge 0,\eta}\left\{\lambda\int\phi^*\left(\frac{g(w)-\eta}{\lambda}\right)dP(w) + \rho\lambda + \eta\right\}.$$

See Appendix B.2.4 for the proof.

We prove the result for general ϕ-divergences $\phi(t) = t^k - 1$, $k \ge 2$. To simplify algebra, we work with a scaled version of the ϕ-divergence: $\phi(t) = \frac{1}{k}(t^k - 1)$, so the scaled population and empirical constraint sets we consider are defined by

$$\mathcal{P} = \left\{Q : D_\phi\left(Q\|P_0\right) \le \frac{\rho}{k}\right\} \quad \text{and} \quad \mathcal{P}_{N_w} := \left\{q : D_\phi\left(q\|\mathbf{1}/N_w\right) \le \frac{\rho}{k}\right\}.$$

$$\begin{aligned}
\mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] &= \mathbb{E}_{P_0}\left[\inf_{\lambda\geq 0,\eta} \frac{1}{N_w}\sum_{i=1}^{N_w} \lambda\phi^*\left(\frac{Z_i-\eta}{\lambda}\right) + \eta + \frac{\rho}{k}\lambda\right] \\
&\leq \inf_{\lambda\geq 0,\eta} \mathbb{E}_{P_0}\left[\frac{1}{N_w}\sum_{i=1}^{N_w} \lambda\phi^*\left(\frac{Z_i-\eta}{\lambda}\right) + \eta + \frac{\rho}{k}\lambda\right] \\
&= \inf_{\lambda\geq 0,\eta}\left\{\mathbb{E}_{P_0}\left[\lambda\phi^*\left(\frac{Z-\eta}{\lambda}\right)\right] + \frac{\rho}{k}\lambda + \eta\right\} \\
&= \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z].
\end{aligned}$$

This proves the upper bound in Lemma B.1.

Now we focus on the lower bound. For the function $\phi(t) = \frac{1}{k}(t^k - 1)$, we have $\phi^*(s) = \frac{1}{k^*}[s]_+^{k^*} + \frac{1}{k}$, where $1/k^* + 1/k = 1$, so that the duality result in Lemma B.2 gives

$$\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z] = \inf_{\eta}\left\{(1+\rho)^{1/k}\left(\frac{1}{N_w}\sum_{i=1}^{N_w}[Z_i-\eta]_+^{k^*}\right)^{\frac{1}{k^*}} + \eta\right\}.$$

Because $|Z_i| \leq M$ for all i, we claim that any η minimizing the preceding expression must satisfy

$$\eta \in \left[-\frac{1+(1+\rho)^{\frac{1}{k^*}}}{(1+\rho)^{\frac{1}{k^*}}-1}, 1\right]\cdot M. \tag{B.2}$$

For convenience, we first define the shorthand

$$S_{N_w}(\eta) := (1+\rho)^{1/k}\left(\frac{1}{N_w}\sum_{i=1}^{N_w}[Z_i-\eta]_+^{k^*}\right)^{\frac{1}{k^*}} + \eta.$$

Then it is clear that $\eta \leq M$, because otherwise we would have $S_{N_w}(\eta) > M \geq \inf_\eta S_{N_w}(\eta)$. Let the lower bound be of the form $\eta = -cM$ for some $c > 1$. Taking derivatives of the

objective $S_{N_w}(\eta)$ with respect to η, we have

$$\begin{aligned} S'_{N_w}(\eta) &= 1 - (1+\rho)^{1/k} \frac{\frac{1}{N_w}\sum_{i=1}^{N_w}[Z_i - \eta]_+^{k^*-1}}{\left(\frac{1}{N_w}\sum_{i=1}^{N_w}[Z_i-\eta]_+^{k^*}\right)^{1-\frac{1}{k^*}}} \\ &\le 1 - (1+\rho)^{1/k}\left(\frac{(c-1)M}{(c+1)M}\right)^{k^*-1} \\ &= 1 - (1+\rho)^{1/k}\left(\frac{c-1}{c+1}\right)^{k^*-1}. \end{aligned}$$

For any $c > c_{\rho,k} := \frac{(1+\rho)^{\frac{1}{k^*}}+1}{(1+\rho)^{\frac{1}{k^*}}-1}$, the preceding display is negative, so we must have $\eta \ge -c_{\rho,k}M$. For the remainder of the proof, we thus define the interval

$$U := [-Mc_{\rho,k}, M], \quad c_{\rho,k} = \frac{(1+\rho)^{\frac{1}{k^*}}+1}{(1+\rho)^{\frac{1}{k^*}}-1},$$

and we assume w.l.o.g. that $\eta \in U$.

Again applying the duality result of Lemma B.2, we have that

$$\begin{aligned} \mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] &= \mathbb{E}\left[\inf_{\eta\in U} S_{N_w}(\eta\right] \\ &= \mathbb{E}\left[\inf_{\eta\in U}\{S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)] + \mathbb{E}[S_{N_w}(\eta)]\}\right] \\ &\ge \inf_{\eta\in U}\mathbb{E}[S_{N_w}(\eta)] - \mathbb{E}\left[\sup_{\eta\in U}|S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)]|\right]. \end{aligned} \tag{B.3}$$

To bound the first term in expression (B.3), we use the following lemma.

Lemma B.3 (Sinha and Duchi [264]). *Let $Z \ge 0, Z \not\equiv 0$ be a random variable with finite $2p$-th moment for $1 \le p \le 2$. Then we have the following inequality:*

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n Z_i^p\right)^{\frac{1}{p}}\right] \ge \|Z\|_p - \frac{p-1}{p}\sqrt{\frac{2}{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\|Z\|_2, \tag{B.4a}$$

and if $\|Z\|_\infty \le C$, *then*

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] \ge \|Z\|_p - C\frac{p-1}{p}\sqrt{\frac{2}{n}}. \tag{B.4b}$$

See Appendix B.2.5 for the proof. Now, note that $[Z-\eta]_+ \in [0, 1+c_{\rho,k}]M$ and $(1+\rho)^{1/k}(1+c_{\rho,k}) =: C_{\rho,k}$. Thus, by Lemma B.3 we obtain that

$$\mathbb{E}[S_{N_w}(\eta)] \ge (1+\rho)^{1/k}\mathbb{E}\left[[Z-\eta]_+^{k^*}\right]^{1/k^*} + \eta - C_{\rho,k}M\frac{k^*-1}{k^*}\sqrt{\frac{2}{N_w}}.$$

Using that $\frac{k^*-1}{k^*} = \frac{1}{k}$, taking the infimum over η on the right hand side and using duality yields

$$\inf_\eta \mathbb{E}[S_{N_w}(\eta)] \ge \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z] - C_{\rho,k}\frac{M}{k}\sqrt{\frac{2}{N_w}}.$$

To bound the second term in expression (B.3), we use concentration results for Lipschitz functions. First, the function $\eta \mapsto S_{N_w}(\eta)$ is $\sqrt{1+\rho}$-Lipschitz in η. To see this, note that for $1 \le k^\star \le 2$ and $X \ge 0$, by Jensen's inequality,

$$\frac{\mathbb{E}[X^{k^\star-1}]}{(\mathbb{E}[X^{k^\star}])^{1-1/k^\star}} \le \frac{\mathbb{E}[X]^{k^\star-1}}{(\mathbb{E}[X^{k^\star}])^{1-1/k^\star}} \le \frac{\mathbb{E}[X]^{k^\star-1}}{\mathbb{E}[X]^{k^\star-1}} = 1,$$

so $S'_{N_w}(\eta) \in [1-(1+\rho)^{\frac{1}{k}}, 1]$ and therefore S_{N_w} is $(1+\rho)^{1/k}$-Lipschitz in η. Furthermore, the mapping $T : z \mapsto (1+\rho)^{\frac{1}{k}}(\frac{1}{N_w}\sum_{i=1}^{N_w}[z_i-\eta]_+^{k^*})^{\frac{1}{k^*}}$ for $z \in \mathbb{R}^{N_w}$ is convex and $(1+\rho)^{\frac{1}{k}}/\sqrt{N_w}$-Lipschitz. This is verified by the following:

$$\begin{aligned}
\left|T(z) - T(z')\right| &\le (1+\rho)^{1/k}\left|\left(\frac{1}{N_w}\sum_{i=1}^{N_w}\left|[z_i-\eta]_+ - \left[z'_i-\eta\right]_+\right|^{k^*}\right)^{\frac{1}{k^*}}\right| \\
&\le \frac{(1+\rho)^{1/k}}{{N_w}^{1/k^*}}\left|\left(\sum_{i=1}^{N_w}\left|z_i - z'_i\right|^{k^*}\right)^{\frac{1}{k^*}}\right| \\
&\le \frac{(1+\rho)^{1/k}}{\sqrt{N_w}}\|z-z'\|_2,
\end{aligned}$$

where the first inequality is Minkowski's inequality and the third inequality follows from the fact that for any vector $x \in \mathbb{R}^n$, we have $\|x\|_p \le n^{\frac{2-p}{2p}}\|x\|_2$ for $p \in [1,2]$, where these

denote the usual vector norms. Thus, the mapping $Z \mapsto S_{N_w}(\eta)$ is $(1+\rho)^{1/k}/\sqrt{N_w}$-Lipschitz continuous with respect to the ℓ_2-norm on Z. Using Samson's sub-Gaussian concentration result for convex Lipschitz functions, we have

$$\mathbb{P}\left(|S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)]| \geq \delta\right) \leq 2\exp\left(-\frac{N_w\delta^2}{2C_{\rho,k}^2 M^2}\right)$$

for any fixed $\eta \in \mathbb{R}$ and any $\delta \geq 0$. Now, let $\mathcal{N}(U,\epsilon) = \{\eta_1, \ldots, \eta_{N(U,\epsilon)}\}$ be an ϵ cover of the set U, which we may take to have size at most $N(U,\epsilon) \leq M(1+c_{\rho,k})\frac{1}{\epsilon}$. Then we have

$$\sup_{\eta\in U} |S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)] \leq \max_{i\in\mathcal{N}(U,\epsilon)} |S_{N_w}(\eta_i) - \mathbb{E}[S_{N_w}(\eta_i)]| + \epsilon(1+\rho)^{1/k}.$$

Using the fact that $\mathbb{E}[\max_{i\leq n}|X_i|] \leq \sqrt{2\sigma^2\log(2n)}$ for X_i all σ^2-sub-Gaussian, we have

$$\mathbb{E}\left[\max_{i\in\mathcal{N}(U,\epsilon)} |S_{N_w}(\eta_i) - \mathbb{E}[S_{N_w}(\eta_i)]|\right] \leq C_{\rho,k}\sqrt{2\frac{M^2}{N_w}\log 2N(U,\epsilon)}.$$

Taking $\epsilon = M(1+c_{\rho,k})/N_w$ gives that

$$\mathbb{E}\left[\sup_{\eta\in U} |S_{N_w}(\eta) - \mathbb{E}[S_{N_w}(\eta)]\right] \leq \sqrt{2}MC_{\rho,k}\sqrt{\frac{1}{N_w}\log(2N_w)} + \frac{C_{\rho,k}M}{N_w}.$$

Then, in total we have (using $C_\rho \geq C_{\rho,k}$, $k \geq 2$, and $N_w \geq 1$),

$$\begin{aligned}\mathbb{E}\left[\sup_{Q\in\mathcal{P}_{N_w}} \mathbb{E}_Q[Z]\right] &\geq \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z] - \frac{C_\rho M\sqrt{2}}{\sqrt{N_w}}\left(\frac{1}{k} + \sqrt{\log(2N_w)} + \frac{1}{\sqrt{2N_w}}\right) \\ &\geq \sup_{Q\in\mathcal{P}} \mathbb{E}_Q[Z] - 4C_\rho M\sqrt{\frac{\log(2N_w)}{N_w}}.\end{aligned}$$

This gives the desired result of the lemma.

B.2.4 Proof of Lemma B.2

Let $L \geq 0$ satisfy $L(w) = dQ(w)/dP(w)$, so that L is the likelihood ratio between Q and P. Then we have

$$\begin{aligned}
&\sup_{Q: D_\phi(Q\|P) \leq \rho} \int g(w) dQ(w) = \sup_{\int \phi(L) dP \leq \rho, \mathbb{E}_P[L]=1} \int g(w) L(w) dP(w) \\
&= \sup_{L \geq 0} \inf_{\lambda \geq 0, \eta} \left\{ \int g(w) L(w) dP(w) - \lambda \left(\int f(L(w)) dP(w) - \rho \right) - \eta \left(\int L(w) dP(w) - 1 \right) \right\} \\
&= \inf_{\lambda \geq 0, \eta} \sup_{L \geq 0} \left\{ \int g(w) L(w) dP(w) - \lambda \left(\int f(L(w)) dP(w) - \rho \right) - \eta \left(\int L(w) dP(w) - 1 \right) \right\},
\end{aligned}$$

where we have used that strong duality obtains because the problem is strictly feasible in its non-linear constraints (take $L \equiv 1$), so that the extended Slater condition holds [185, Theorem 8.6.1 and Problem 8.7]. Noting that L is simply a positive (but otherwise arbitrary) function, we obtain

$$\begin{aligned}
&\sup_{Q: D_\phi(Q\|P) \leq \rho} \int g(w) dQ(w) \\
&= \inf_{\lambda \geq 0, \eta} \int \sup_{\ell \geq 0} \left\{ (g(w) - \eta)\ell - \lambda \phi(\ell) \right\} dP(w) + \lambda \rho + \eta \\
&= \inf_{\lambda \geq 0, \eta} \int \lambda \phi^* \left(\frac{g(w) - \eta}{\lambda} \right) dP(w) + \eta + \rho \lambda.
\end{aligned}$$

Here we have used that $\phi^*(s) = \sup_{t \geq 0} \{st - \phi(t)\}$ is the conjugate of ϕ and that $\lambda \geq 0$, so that we may take divide and multiply by λ in the supremum calculation.

B.2.5 Proof of Lemma B.3

For $a > 0$, we have

$$\inf_{\lambda \geq 0} \left\{ \frac{a^p}{p \lambda^{p-1}} + \lambda \frac{p-1}{p} \right\} = a,$$

(with $\lambda = a$ attaining the infimum), and taking derivatives yields

$$\frac{a^p}{p \lambda^{p-1}} + \lambda \frac{p-1}{p} \geq \frac{a^p}{p \lambda_1^{p-1}} + \lambda_1 \frac{p-1}{p} + \frac{p-1}{p} \left(1 - \frac{a^p}{\lambda_1^p} \right) (\lambda - \lambda_1).$$

Using this in the moment expectation, by setting $\lambda_n = \sqrt[p]{\frac{1}{n} \sum_{i=1}^n Z_i^p}$, we have for any

$\lambda \geq 0$ that

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] = \mathbb{E}\left[\frac{\sum_{i=1}^n Z_i^p}{pn\lambda_n^{p-1}} + \lambda_n \frac{p-1}{p}\right]$$
$$\geq \mathbb{E}\left[\frac{\sum_{i=1}^n Z_i^p}{pn\lambda^{p-1}} + \lambda \frac{p-1}{p}\right] + \frac{p-1}{p}\mathbb{E}\left[\left(1 - \frac{\sum_{i=1}^n Z_i^p}{n\lambda^p}\right)(\lambda_n - \lambda)\right].$$

Now we take $\lambda = \|Z\|_p$, and we apply the Cauchy-Schwarz inequality to obtain

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] \geq \|Z\|_p - \frac{p-1}{p}\mathbb{E}\left[\left(1 - \frac{\frac{1}{n}\sum_{i=1}^n Z_i^p}{\|Z\|_p^p}\right)^2\right]^{\frac{1}{2}} \mathbb{E}\left[\left(\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}} - \|Z\|_p\right)^2\right]^{\frac{1}{2}}$$
$$= \|Z\|_p - \frac{p-1}{p\sqrt{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\mathbb{E}\left[\left(\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}} - \mathbb{E}[Z^p]^{\frac{1}{p}}\right)^2\right]^{\frac{1}{2}}$$
$$\geq \|Z\|_p - \frac{p-1}{p\sqrt{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{2}{p}} + \mathbb{E}[Z^p]^{\frac{2}{p}}\right]^{\frac{1}{2}}$$
$$\geq \|Z\|_p - \frac{p-1}{p}\sqrt{\frac{2}{n}}\sqrt{\mathrm{Var}(Z^p/\mathbb{E}[Z^p])}\|Z\|_2,$$

where the last inequality follows by the fact that the norm is non-decreasing in p.

In the case that we have the unifom bound $\|Z\|_\infty \leq C$, we can get tighter guarantees. To that end, we state a simple lemma.

Lemma B.4. *For any random variable $X \geq 0$ and $a \in [1, 2]$, we have*

$$\mathbb{E}[X^{ak}] \leq \mathbb{E}[X^k]^{2-a}\mathbb{E}[X^{2k}]^{a-1}$$

Proof For $c \in [0, 1]$, $1/p + 1/q = 1$ and $A \geq 0$, we have by Holder's inequality,

$$\mathbb{E}[A] = \mathbb{E}[A^c A^{1-c}] \leq \mathbb{E}[A^{pc}]^{1/p}\mathbb{E}[A^{q(1-c)}]^{1/q}$$

Now take $A := X^{ak}$, $1/p = 2 - a$, $1/q = a - 1$, and $c = \frac{2}{a} - 1$. □

First, note that $\mathbb{E}[Z^{2p}] \leq C^p\mathbb{E}[Z^p]$. For $1 \leq p \leq 2$, we can take $a = 2/p$ in Lemma B.4, so that we have

$$E[Z^2] \leq \mathbb{E}[Z^p]^{2-\frac{2}{p}}\mathbb{E}[Z^{2p}]^{\frac{2}{p}-1} \leq \|Z\|_p^p C^{2-p}.$$

Now, we can plug these into the expression above (using $\mathrm{Var} Z^p \leq \mathbb{E}[Z^{2p}] \leq C^p \|Z\|_p^p$), yielding

$$\mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} Z_i^p\right)^{\frac{1}{p}}\right] \geq \|Z\|_p - C\frac{p-1}{p}\sqrt{\frac{2}{n}}$$

as desired.

B.2.6 Proof of Proposition 3.2

We utilize the following lemma for regret of online mirror descent.

Lemma B.5. *The expected regret for online mirror descent with unbiased stochastic subgradient $\gamma(t)$ and stepsize η is*

$$\sum_{t=1}^{T} \mathbb{E}\left[\gamma(t)^T (w(t) - w^\star)\right] \leq \frac{\log(d)}{\eta} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{d} w_j(t)\gamma_j(t)^2\right] \tag{B.5}$$

See Appendix B.2.7 for the proof. Now we bound the right-hand term of the regret bound (B.5) in our setting. For this we utilize the following:

$$\begin{aligned}
\mathbb{E}\left[\gamma_i(t)^2 | w(t)\right] &= \frac{1}{N_w^2}\frac{L_i^2(t)}{w_i^2(t)}\mathbb{E}\left[\left(\sum_{k=1}^{N_w} \mathbf{1}\left\{J_k = i\right\}\right)^2 \middle| w(t)\right] \\
&= \frac{1}{N_w^2}\frac{L_i^2(t)}{w_i^2(t)}\left(N_w(N_w-1)w_i(t)^2 + N_w w_i(t)\right),
\end{aligned}$$

where the latter fact is simply the second moment for the sum of N_w random variables $\overset{\text{i.i.d.}}{\sim}$ Bernoulli$(w_i(t))$. Then,

$$\begin{aligned}
\sum_{i=1}^{d} w_i(t)\mathbb{E}\left[\gamma_i(t)^2 | w(t)\right] &= \sum_{i=1}^{d} L_i(t)^2\left(\frac{N_w - 1}{N_w} w_i(t) + \frac{1}{N_w}\right) \\
&\leq \sum_{i=1}^{d}\left(\frac{N_w - 1}{N_w} w_i(t) + \frac{1}{N_w}\right) \\
&= \frac{N_w - 1}{N_w} + \frac{d}{N_w} \\
&=: z.
\end{aligned}$$

Plugging in the prescribed $\eta = \sqrt{\frac{2\log(d)}{zT}}$ into the bound (B.5) yields the result.

B.2.7 Proof of Lemma B.5

We first show the more general regeret of online mirror descent with a Bregman divergence and then specialize to the entropic regularization case. Let $\psi(w)$ be a convex fuction and $\psi^*(\theta)$ its Fenchel conjugate. Define the Bregman divergence $B_\psi(w, w') = \psi(w) - \psi(w') - \nabla\psi(w')^T(w - w')$. In the following we use the subscript $\cdot_t$ instead of $(\cdot)(t)$ for clarity. The standard online mirror descent learner sets

$$w_t = \operatorname*{argmin}_w \left(\gamma_t^T w + \frac{1}{\eta} B_\psi(w, w_t) \right).$$

Using optimality of w_{t+1} in the preceding equation, we have

$$\begin{aligned}
\gamma_t^T(w_t - w^*) &= \gamma_t^T(w_{t+1} - w^*) + \gamma_t^T(w_t - w_{t+1}) \\
&\leq \frac{1}{\eta}(\nabla\psi(w_{t+1}) - \nabla\psi(w_t))^T(w^* - w_{t+1}) \\
&+ \gamma_t^T(w_t - w_{t+1}) \\
&= \frac{1}{\eta}\left(B_\psi(w^*, w_t) - B_\psi(w^*, w_{t+1}) - B_\psi(w_{t+1}, w_t)\right) \\
&+ \gamma_t^T(w_t - w_{t+1}).
\end{aligned}$$

Summing this preceding display over iterations t yields

$$\begin{aligned}
\sum_{t=1}^T \gamma_t^T(w_t - w^*) \leq \frac{1}{\eta} B_\psi(w^*, w_1) \\
+ \sum_{t=1}^T \left(-\frac{1}{\eta} B_\psi(w_{t+1}, w_t) + \gamma_t^T(w_t - w_{t+1}) \right)
\end{aligned}$$

Now let $\psi(w) = \sum_i w_i \log w_i$. Then, with $w_1 = \mathbf{1}/d$, $B\psi(w^*, w_1) \leq \log(d)$. Now we bound the second term with the following lemma.

Lemma B.6. *Let $\psi(x) = \sum_j x_j \log x_j$ and $x, y \in \Delta$ be defined by: $y_i = \frac{x_i \exp(-\eta g_i)}{\sum_j x_j \exp(-\eta g_j)}$ where $g \in \mathbb{R}_+^d$ is non-negative. Then*

$$-\frac{1}{\eta} B_\psi(y, x) + g^T(x - y) \leq \frac{\eta}{2} \sum_{i=1}^d g_i^2 x_i.$$

See Appendix B.2.8 for the proof. Setting $y = w_{t+1}$, $x = w_t$, and $g = \gamma_t$ in Lemma B.6

yields

$$\sum_{t=1}^{T} \gamma_t^T (w_t - w^*) \leq \frac{\log(d)}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \sum_{j=1}^{d} w_j(t) \gamma_j(t)^2.$$

Taking expectations on both sides yields the result.

B.2.8 Proof of Lemma B.6

Note that $B_\psi(y, x) = \sum_i y_i \log \frac{y_i}{x_i}$. Substituting the values for x and y into this expression, we have

$$\sum_i y_i \log \frac{y_i}{x_i} = -\eta g^T y - \sum_i y_i \log \left(\sum_j x_j e^{-\eta g_j} \right)$$

Now we use a Taylor expansion of the function $g \mapsto \log \left(\sum_j x_j e^{-\eta g_j} \right)$ around the point 0. If we define the vector $p_i(g) = x_i e^{-\eta g_i} / \left(\sum_j x_j e^{-\eta g_j} \right)$, then

$$\log \left(\sum_j x_j e^{-\eta g_j} \right) = \log(\mathbf{1}^T x) - \eta p(0)^T g + \frac{\eta^2}{2} g^\top \left(\operatorname{diag}(p(\widetilde{g})) - p(\widetilde{g}) p(\widetilde{g})^\top \right) g$$

where $\widetilde{g} = \lambda g$ for some $\lambda \in [0, 1]$. Noting that $p(0) = x$ and $\mathbf{1}^T x = \mathbf{1}^T y = 1$, we obtain

$$B_\psi(y, x) = \eta g^T (x - y) - \frac{\eta^2}{2} g^\top \left(\operatorname{diag}(p(\widetilde{g})) - p(\widetilde{g}) p(\widetilde{g})^\top \right) g,$$

whereby

$$-\frac{1}{\eta} B_\psi(y, x) + g^T (x - y) \leq \frac{\eta}{2} \sum_{i=1}^{d} g_i^2 p_i(\widetilde{g}). \tag{B.6}$$

Lastly, we claim that the function

$$s(\lambda) = \sum_{i=1}^{d} g_i^2 \frac{x_i e^{-\lambda g_i}}{\sum_j x_j e^{-\lambda g_j}}$$

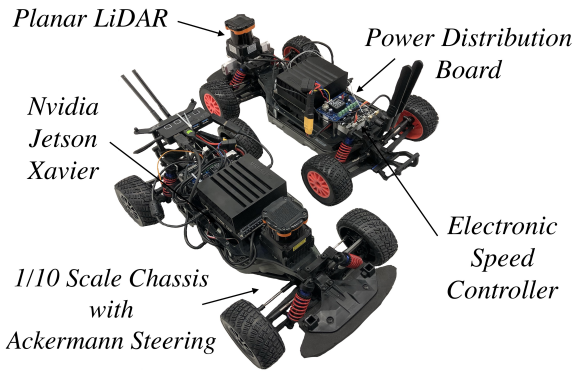


Figure B.1: Components of the 1/10 scale vehicle

is non-increasing on $\lambda \in [0, 1]$. Indeed, we have

$$\begin{aligned} s'(\lambda) &= \frac{\left(\sum_i g_i x_i e^{-\lambda g_i}\right)\left(\sum_i g_i^2 x_i e^{-\lambda g_i}\right)}{\left(\sum_i x_i e^{-\lambda g_i}\right)^2} - \frac{\sum_i g_i^3 x_i e^{-\lambda g_i}}{\sum_i x_i e^{-\lambda g_i}} \\ &= \frac{\sum_{ij} g_i g_j^2 x_i x_j e^{-\lambda g_i - \lambda g_j} - \sum_{ij} g_i^3 x_i x_j e^{-\lambda g_i - \lambda g_j}}{\left(\sum_i x_i e^{-\lambda g_i}\right)^2} \end{aligned}$$

Using the Fenchel-Young inequality, we have $ab \leq \frac{1}{3}|a|^3 + \frac{2}{3}|b|^{3/2}$ for any a, b so $g_i g_j^2 \leq \frac{1}{3}g_i^3 + \frac{2}{3}g_j^3$. This implies that the numerator in our expression for $s^{'}(\lambda)$ is non-positive. Thus, $s(\lambda) \leq s(0) = \sum_{i=1}^{d} g_i^2 x_i$ which gives the result when combined with inequality (B.6).

B.3 Hardware

The major components of the vehicle used in experiments are shown in Figure B.1. The chassis of the 1/10-scale vehicles used in experiments are based on a Traxxas Rally 1/10-scale radio-controlled car with an Ackermann steering mechanism. An electronic speed controller based on an open source design [289] controls the RPM of a brushless DC motor and actuates a steering servo. A power distribution board manages the power delivery from a lithium polymer (LiPo) battery to the onboard compute unit and sensors. The onboard compute unit is a Nvidia Jetson Xavier, a system-on-a-chip that contains 8 ARM 64 bit CPU cores and a 512 core GPU. The onboard sensor for localization is a planar LIDAR that operates at 40Hz with a maximum range of 30 meters. The electronic speed controller also provides odometry via the back EMF of the motor.

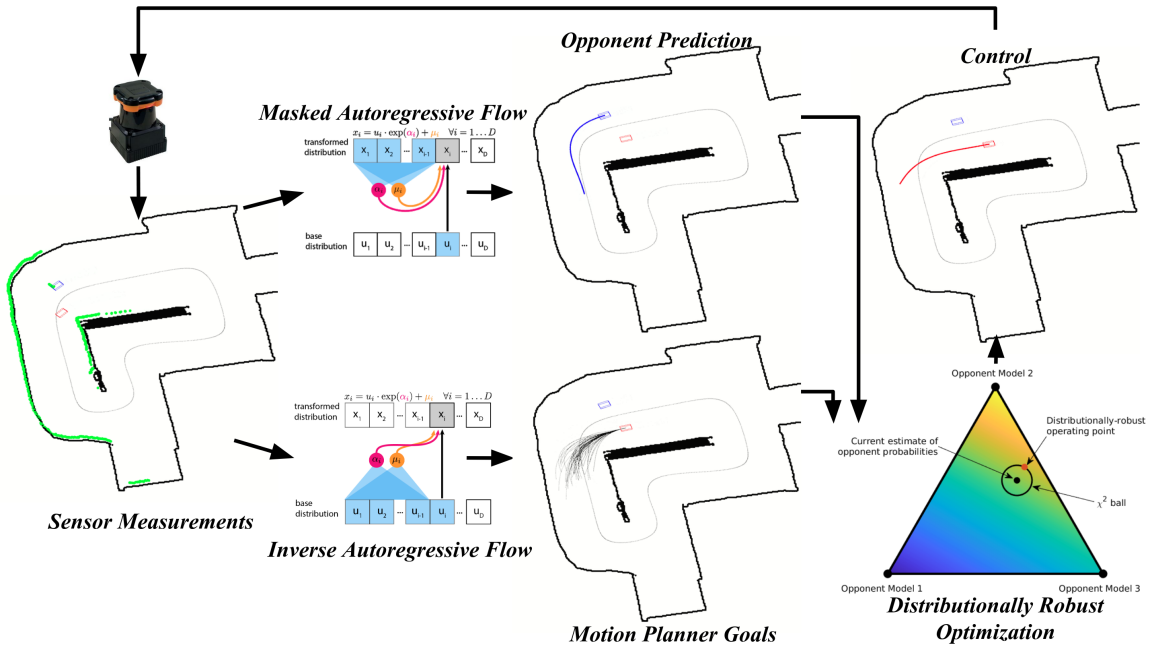


Figure B.2: FormulaZero implementation on vehicle. Online each agent measures the world using onboard sensors such as a planar LIDAR. Given the sensor measurement the vehicle performs opponent prediction via the use of a masked autoregressive flow and simultaneously selects motion planner goals using an inverse autoregressive flow. Given the set of goals each is evaluated within our DRO framework, the best goal is chosen, and a new control command is applied to the vehicle. Then, the process occurs again.

B.4 Vehicle Software Stack

This section gives a detailed overview of the software used onboard the vehicles. Figure B.2 gives a graphical overview.

B.4.1 Mapping

We create occupancy grid maps of tracks using Google Cartographer [128]. The map's primary use is as an efficient prior for vehicle localization algorithms. In addition, maps serve as a representation of the static portion of the simulation environment describing where the vehicle may drive and differentiating which (if any) portions of the LIDAR scan have line-of-sight to other agents. A feature of our system useful to other researchers is that any environment which can be mapped may be trivially added to the simulator described in Appendix B.5.

B.4.2 Localization

Due to the speeds at which the vehicles travel, localization must provide pose estimates at a rate of at least 20 Hz. Thus, to localize the vehicle we use a particle filter [294] that implements a ray-marching scheme on the GPU in order to efficiently simulate sensor observations in parallel. We add a small modification which captures the covariance of the pose estimate. We do not use external localization systems (*e.g.* motion capture cameras) in any experiment.

B.4.3 Planning

The vehicle software uses a hierarchical planner [107] similar to that of Ferguson et al. [96]. At the top level the planner receives a map and waypoints representing the centerline of the track; the goal is to traverse the track from start to finish. Unlike route planning in road networks, there are no routing decisions to be made. In more complex instances of our proposed environment, this module could be necessary for making strategic decisions such as pit stops. The second key difference is the mid-level planner. Whereas Ferguson et al. [96] uses a deterministic lattice of points, our vehicle draws samples from a neural autoregressive flow. Each sample contains a goal pose and speed profile. Given this specification, the local planner calculates a trajectory parameterized as a cubic spline, evaluates static and dynamic costs of the proposed plan in belief space, and selects the lowest cost option.

Sampling behavior proposals

There are two advantages to using a neural autoregressive flow in our planning framework. First, each agent in the population weights the individual components of its cost function differently; the flow enables the goal generation mechanism to learn a distribution which places more probability mass on the agent's preferences. Second, as planning takes place in the context of the other agent's actions, the ego-agent's beliefs can be updated by inverting the flow and estimating the likelihood of the other agent's actions under a given configuration of the cost function.

The goal-generation process utilizes an inverse autoregressive flow (IAF) [158]. The IAF samples are drawn from a density conditioned on a 101-dimensional observation vector composed of a subsampled LIDAR scan and current speed. Each sample is a 6 dimensional

vector: Δt, the perpendicular offset of the goal pose from the track's centerline; Δs, the arc-length along the track's centerline relative to the vehicle's current pose; $\Delta\theta$, the difference between the goal pose's heading angle and the current heading angle; three velocity offsets from the vehicle's current velocity at three equidistant knot points along the trajectory.

The second benefit of using a generative model for sampling behavior proposals is the ability to update an agent's beliefs about the opponent's policy type. As noted in Section 3.4, masked [223] and inverse autoregressive flows (MAF and IAF respectively) have complementary strengths. While sampling from a MAF is slow, density estimation using this architecture is fast. Thus, we use a MAF network trained to mimic the samples produced by the IAF for this task. The architectures of each network are the same, and we describe this architecture below.

The IAF and MAF networks used in this work have 5 MADE layers [223] each containing: a masked linear mapping ($\mathbb{R}^6 \to \mathbb{R}^{100}$), RELU layer, masked linear mapping ($\mathbb{R}^{100} \to \mathbb{R}^{100}$), RELU layer, and a final masked linear layer ($\mathbb{R}^{100} \to \mathbb{R}^{12}$). Note that output of a MADE layer includes both the transformed sample and the logarithm of the absolute value of the determinant of the Jacobian of the transformation. For sampling, the latter is discarded, and the transformed sample is passed to the next layer. In addition, the masking pattern is sequential and held constant during both training and inference. This choice was made to aid in debugging of experiments and to simplify communication during distributed training.

Each population member has a dedicated IAF model, which is trained iteratively according to the AADAPT algorithm described in Section 3.2 using the hyperparameters given in Section 3.4. We initialize each IAF with a set of weights which approximate an identity transformation for random pairs of samples from a normal distribution and simulated observations. In addition each population member also has a MAF model, which is trained using the same hyperparameters as the IAF but only after AADAPT has finished. Our code extends an existing library[1] created by other authors; we add support for the IAF architecture as well as generalize the network architecture to 3-dimensional tensors. The latter extension enables sampling from multiple agents' IAF models simultaneously and efficiently.

[1] https://github.com/kamenbliznashki/normalizing_flows

Table B.1: The resolution and ranges of the trajectory generator look-up table

Index	Resolution	Min	Max
Δx	0.1 m	-1.0 m	10.0 m
Δy	0.1 m	-8.0 m	8.0 m
$\Delta\theta$	$\pi/32$ rad	$-\pi/2$ rad	$\pi/2$ rad
κ_0	0.2 rad/m	-1.0 rad/m	1.0 rad/m

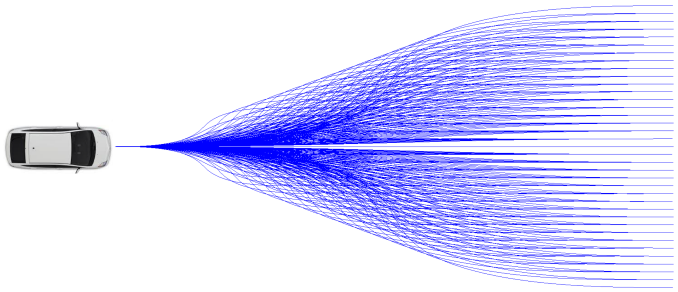


Figure B.3: Sample trajectories from the look-up table

Model Predictive Control

The goal of the trajectory generator is to compute kinematically and dynamically feasible trajectories that take the vehicle from its current pose to a set of sampled poses from the IAF. The trajectory generator combines approaches from [134, 205, 153, 196]. Each trajectory is represented by a cubic spiral with five parameters $p = [s, a, b, c, d]$ where s is the arc length of the spiral, and (a, b, c, d) encode the curvature at equispaced knot points along the trajectory. Powell's method or gradient descent can be used to find the spline parameters that (locally) minimize the sum of the Euclidean distance between the desired endpoint pose and the forward simulated pose. Offline, a lookup table of solutions for a dense grid of goal poses is precomputed, enabling fast trajectory generation online. Each trajectory is associated with an index which selects the Δx, Δy, and the $\Delta\theta$ of the goal pose relative to the current pose (where positive x is ahead of the vehicle and postiive y is to the left), and κ_0, the initial curvature of the trajectory. The resolution and the range of the table is listed in Table B.1. Figure B.3 shows a selection of trajectories. The point on the left of the figure is the starting pose of the vehicle, and the collection of goal poses is shown as the points on the right of the figure.

Trajectory Cost Functions

Each of the generated trajectories is evaluated with the weighted sum of the following cost functions. Note, in order to ensure safety, goals which would result in collision result in infinite cost and are automatically rejected prior to computing the robust cost, which operates only on finite-cost proposals.

1. **Trajectory length:** $c_{al} = s$, where $1/s$ is the arc length of each trajectory. Short and myopic trajectories are penalized.

2. **Maximum absolute curvature:** $c_{mc} = \max_i\{|\kappa_i|\}$, where κ_i are the curvatures at each point on a trajectory. Large curvatures are penalized to preserve smoothness of trajectories.

3. **Mean absolute curvature:** $c_{ac} = \frac{1}{N}\sum_{i=0}^{N}|\kappa_i|$, the notation is the same as c_{mc} and the effect of this feature is similar, but less myopic.

4. **Hysteresis loss:** Measured between the previous chosen trajectory and each of the sampled trajectories, $c_{hys} = ||\theta_{prev}^{[n_1,n_2]} - \theta^{[0,n_2-n_1]}||_2^2$, where θ_{prev} is the array of heading angles of each pose on the previous selected trajectory by the vehicle, θ is the array of heading angles of each pose on the trajectory being evaluated, and the ranges $[n_1, n_2]$ and $[0, n_2 - n_1]$ define contiguous portions of trajectories that are compared. Trajectories dissimilar to the previously selected trajectory are penalized.

5. **Lap progress:** Measured along the track from the start to the end point of each trajectory in the normal and tangential coordinate system, $c_p = \frac{1}{s_{end}-s_{start}}$, where s_{end} is the corresponding position in the tangential coordinate along the track of the end point of a trajectory, and s_{start} is that of the start point of a trajectory. Shorter progress in distance is penalized.

6. **Maximum acceleration:** $c_{ma} = \max_i |\frac{\Delta v_i}{\Delta t_i}|$ where Δv is the array of difference in velocity between adjacent points on a trajectory, and Δt is the array of corresponding time intervals between adjacent points. High maximum acceleration is penalized.

7. **Maximum absolute curvature change:** Measured between adjacent points along each trajectory, $c_{dk} = \max_i |\frac{\Delta \kappa_i}{\Delta t_i}|$. High curvature changes are penalized.

8. **Maximum lateral acceleration:** $c_{la} = \max_i\{|\kappa|_i v_i^2\}$, where κ and v are the arrays of curvature and velocity of all points on a trajectory. High maximum lateral accelerations are penalized.

9. **Minimum speed:** $c_{ms} = \frac{1}{(\min_i\{v_i\})_+}$. Low minimum speeds are penalized.

10. **Minimum range:** $c_{mr} = \min_i\{r_i\}$, where r is the array of range measurements (distance to static obstacles) generated by the simulator. Smaller minimum range is penalized, and trajectories with minimum ranges lower than a threshold are given infinite cost and therefore discarded.

11. **Cumulative inter-vehicle distance short:**

$$c_{dyshort} = \begin{cases} \infty, \text{ if } d(\texttt{ego}_i, \texttt{opp}_i) \leq \texttt{thresh} \\ \sum_{i=0}^{N_{short}} d(\texttt{ego}_i, \texttt{opp}_i), \text{ otherwise} \end{cases}$$

    Where the function $d()$ returns the instantaneous minimum distance between the two agents at point i, N_{short} is a point that defines the shorter time horizon for a trajectory of N points. Trajectories with infinite cost on the shorter time horizon are considered infeasible and discarded.

12. **Discounted cumulative inter-vehicle distance long:** $c_{dylong} = \sum_{i=N_{short}}^{N_{long}} 0.9^{i-N_{short}} \frac{1}{d(\texttt{ego}_i, \texttt{opp}_i)}$, where N_{long} is a point that defines the longer time horizon for a trajectory of N points. Note that $N_{short} < N_{long} < N$. Lower minimum distances between agents on the longer time horizon are penalized.

13. **Relative progress:** Measured along the track between the sampled trajectories' endpoints and the opponent's selected trajectory's endpoint, $c_{dp} = (s_{opp\_end} - s_{end})_+$, where s_{opp_end} is the position along the track in tangential coordinates of the endpoint of the opponent's chosen trajectory. Lagging behind the opponent is penalized.

Path tracker

Once a trajectory has been selected it is given to the path-tracking module. The goal of the path tracker is to compute a steering input which drives the vehicle to follow the desired trajectory. Our implementation uses a simple and industry-standard geometrical tracking method called pure pursuit [72, 270]. Due to the decoupling of the trajectory generation

and tracking modules it is possible for the tracker to run at a much higher frequency than the trajectory generator; this is essential for good performance.

B.4.4 Communication and system architecture

The ZeroMQ [130] messaging library is used to create interfaces between the FormulaZero software stack and the underlying ROS nodes that control and actuate the vehicle test bed. Unlike in the simulator, some aspects of the FormulaZero planning function operate non-deterministicaly and asynchronously. In particular we use a sink node to collect observations from ROS topics related to the various sensors on the vehicle in order to approximate the step-function present in the Gym API. When a planning cycle is complete, the trajectory is published back to ROS and tracked asynchronously using pure-pursuit as new pose estimates become available. Because perception is not the primary focus of this project we simplify the problem of detecting and tracking the other vehicle. In particular, each vehicle estimates its current pose in the map obtained by its onboard particle filter, and this information is communicated to the other vehicle via ZeroMQ over a local wireless network. Since tracking and detection has been well studied in robotics, solutions which rely less on communication could be explored by other future work.

B.5 Simulation Stack

The simulation stack includes a lightweight 2D physics engine with a dynamical vehicle model. Then on top of the physics engine, a multi-agent simulator with an OpenAI Gym [51] API is used to perform rollouts of the experiments.

B.5.1 Vehicle Dynamics

The single-track model in Althoff et al. [8] is chosen because it considers tire slip influences on the slip angle, which enables accurate simulation at physical limits of the vehicle test bed. It is also easily enables changes to the driving surface friction coefficient in simulation which allows the simulator to model a variety of road surfaces.

B.5.2 System Identification

Parameter identification was performed to derive the following vehicle parameters: mass, center of mass, moment of inertia, surface friction coefficient, tire cornering stiffness, and maximum acceleration/deceleration rates following the methods described in O'Kelly et al. [219].

B.5.3 Distributed Architecture

Due to the nature of the AAdaPT algorithm, the rollouts in a single vertical step do not need to be in sequence. The ZeroMQ messaging library is used to create a MapReduce [73] pattern between the task distributor, result collector, and the workers. Each worker receives the description of the configuration to be simualted, *e.g.* (x, θ). Then the workers asynchronously perform simulations and send results to the collector.

B.5.4 Addressing the simulation/reality gap

As noted in Section 3.4 there are several differences between the observations in simulated rollouts and reality. First, pose estimation errors are not present in the simulator. A simple fix would be to add Gaussian white noise to the pose observations returned by the simulator. We avoided this and other domain randomization techniques in order to preserve the determinism of the simulator, but we will investigate its effect in further experiments. Second, the LIDAR simulation does not account for material properties of the environment. In particular, surfaces such as glass do not produce returns, causing subsets of the LIDAR beams to be dropped. We hypothesize that simple data augmentation schemes which select a random set of indices to drop from simulated LIDAR observations would improve the robustness to such artifacts when the system is deployed on the real car; we are currently investigating this hypothesis.

B.6 Experiments

Additional videos of simulation runs are available.[2]

[2] https://youtu.be/8q0lZssbEI4

B.6.1 Instantaneous time-to-collision (iTTC)

Let $T_i(t)$ be the instantaneous time-to-collision between the ego vehicle and the i-th environment vehicle at time step t. The value $T_i(t)$ can be defined in multiple ways (see *e.g.* Sontges et al. [271]). Norden et al. [215] define it as the amount of time that would elapse before the two vehicles' bounding boxes intersect assuming that they travel at constant fixed velocities from the snapshot at time t. Note that this definition of TTC is distinct from that of Section C.1, and is a better indicator of danger for racing scenarios. Time-to-collision captures directly whether or not the ego-vehicle was involved in a crash. If it is positive no crash occurred, and if it is 0 or negative there was a collision.

B.6.2 Out-of-distribution agent strategies

In the following sections, we describe the human-created algorithms used in our out-of-distribution analysis.

OOD1: RRT* with MPC-based Opponent Prediction

This approach exploits the fact that the two-car racing scenario is similar to driving alone on the track with the only exception being during overtaking the opponent. This approach uses a costmap-based RRT* [150] planning algorithm. The agent first uses the opponent's current pose and velocity in the world, and uses Model-Predictive Control to calculate an open loop trajectory of N optimal inputs resulting in N+1 states based on a given cost function and constraints. Specifically, the optimization problem is constrained by a linearized version of the single track model described in Althoff et al. [8], and by the boundary values of the inputs and states of the vehicle. The cost function that the optimization tries to minimize consists of the trajectory length and input power requirement. The costmap used by RRT* also incorporates this predicted trajectory of the opponent vehicle by inflating the two-dimensional spline representing the prediction, and weighting the portion of the spline closer to the ego vehicle higher. RRT* samples the two dimensional space that the vehicle lies in. The path generated by RRT* is then tracked with the Pure Pursuit controller [72].

OOD2: RL-based Lane Switching

The second algorithm is based on a lane-switching planning strategy that uses an RL algorithm to make lane switching decisions, and filters out unsafe decisions using a collision

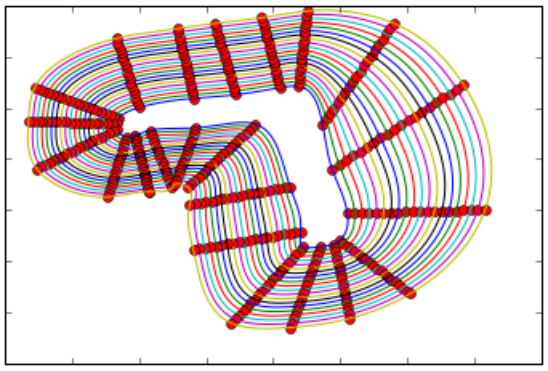


Figure B.4: Lanes that cover the track

indicator. First, as shown in B.4, different lanes going through numerous checkpoints on the track are created to cover the entirety of the race track. Then a network is trained to make lane switching decisions. The state of the RL problem consists of the sub-sampled LIDAR scans of the ego vehicle; the pose (x, y, θ) of the opponent car with respect to the ego vehicle; velocity (v_x, v_y) of the opponent vehicle with respect of the ego vehicle; projected distance from the ego vehicle's current position to all pre-defined paths. The reward of a rollout is zero in the beginning. At each timestep, the timestep itself is subtracted from the total reward. A rollout receives -100 as the reward when the ego agent collide with the environment or the other agent. And finally, if both agents finish 2 laps, the difference between lap times (positive if the ego agent wins) of the two agents are added to the reward. Clipped Double Q-Learning [100] is used to estimate the Q function and make the lane switching decisions. iTTC defined in Appendix B.6.1 is used as an indicator for future collisions. If any decisions made by the RL network would result in a collision indicated by the iTTC value, the safety function kicks in and makes the lane switching decision based on the collision indicator. Finally, ego vehicle actuation is provided by the same Pure Pursuit controller [72] tracking the selected lane. We used an existing implementation[3] of this algorithm.

[3] https://github.com/pnorouzi/rl-path-racing

Appendix C

Chapter 4 Appendices

C.1 Scenario specification

A scenario specification consists of a scenario description and outputs both p_γ (4.1), the accident rate, and a dataset consisting of initial conditions and the minimum time to collision, our continuous objective safety measure. Concretely, a scenario description includes

- a set of possible initial conditions, *e.g.* a range of velocities and poses for each agent
- a safety measure specification for the ego agent,
- a generative model of environment policies, an ego vehicle model,
- a world geometry model, *e.g.*a textured mesh of the static scene in which the scenario is to take place.

Given the scenario description, the search module creates physics and rendering engine worker instances, and Algorithm 4.1 then adaptively searches through many perturbations of conditions in the scenario, which we call scenario realizations. A set of scenario realizations may be mapped to multiple physics, rendering, and agent instantiations, evaluated in parallel, and reduced by a sink node which reports a measure of each scenarios performance relative to the specification.

In our implementation the safety measure is minimum time-to-collision (TTC). TTC is defined as the time it would take for two vehicles to intercept one another given that they each maintain their current heading and velocity [292]. The TTC between the ego-vehicle

and vehicle i is given by

$$TTC_i(t) = \frac{r_i(t)}{[-\dot{r}_i(t)]_+}, \tag{C.1}$$

where r_i is the distance between the ego vehicle and vehicle i, and $\dot{r}_i$ the time derivative of this distance (which is simply computed by projecting the relative velocity of vehicle i onto the vector between the vehicles' poses). The operator $[\cdot]_+$ is defined as $[x]_+ := \max(x, 0)$. We define $TTC_i(t) = \infty$ for $\dot{r}_i(t) \geq 0$.

Vehicles are described as oriented rectangles in the 2D plane. Since we are interested in the time it would take for the ego-vehicle to intersect the polygonal boundary of another vehicle on the road, we utilize a finite set of range and range measurements in order to approximate the TTC metric. For a given configuration of vehicles, we compute N uniformly spaced angles $\theta_1, \ldots, \theta_N$ in the range $[0, 2\pi]$ with respect to the ego vehicle's orientation and cast rays outward from the center of the ego vehicle. For each direction we compute the distance which a ray could travel before intersecting one of the M other vehicles in the environment. These form N range measurements $s_1, \ldots, s_N$. Further, for each ray s_i, we determine which vehicle (if any) that ray hit; projecting the relative velocity of this vehicle with respect to ego vehicle gives the range-rate measurement $\dot{s}_i$. Finally, we approximate the minimum TTC for a given simulation rollout X of length T discrete time steps by:

$$f(X) := \min_{t=0,\ldots,T} \left(\min_{i=1,\ldots,N} \frac{s_i(t)}{[-\dot{s}_i(t)]_+} \right),$$

where we again define the approximate instantaneous TTC as ∞ for $\dot{s}_i(t) \geq 0$. Note that this measure can approximate the true TTC arbitrarily well via choice of N and the discretization of time used by the simulator. Furthermore, note that our definition of TTC is with respect to the *center* of the ego vehicle touching the *boundary* of another vehicle. Crashing, on the other hand, is defined in our simulation as the intersection of boundaries of two vehicles. Thus, TTC values we evaluate in our simulation are nonzero even during crashes, since the center of the ego vehicle has not yet collided with the boundary of another vehicle.

C.2 Network architectures

The MGAIL generator model we use takes the same inputs as that of Kuefler et al. [166]—the dynamical states of the vehicle as well as virtual lidar beam reflections. Specifically, we take

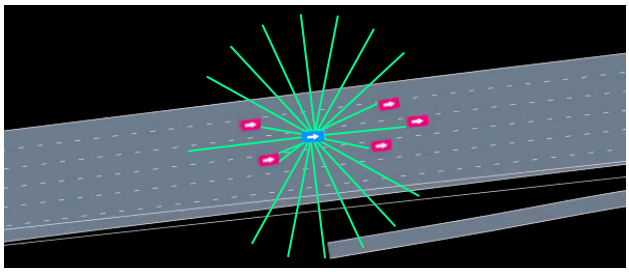


Figure C.1: Depiction of lidar sensor input used for GAIL models

as inputs: geometric parameters (vehicle length/width), dynamical states (vehicle speed, lateral and angular offsets with respect to the center and heading of the lane, distance to left and right lane boundaries, and local lane curvature), three indicators for collision, road departure, and traveling in reverse, and lidar sensor observations (ranges and range-rates of 20 lidar beams) as depicted in Figure C.1. The generator has two hidden layers of 200 and 100 neurons. The output consists of the mean and variance of normal distributions for throttle and steering commands; we then sample from these distributions to draw a given vehicle's action. The discriminator shares the same size for hidden layers. The forward model used to allow fully-differentiable training first encodes both the state and action through a 150 neuron layer and also adds a GRU layer to the state encoding. A Hadamard product of the results creates a joint embedding which is put through three hidden layers each of 150 neurons. The output is a prediction of the next state.

The end-to-end highway autopilot model is a direct implementation of Bojarski et al. [43] via the code found at the link `https://github.com/sullychen/autopilot-tensorflow`. In our implementation of the vision-based policy, this highway autopilot model uses rendered images to produce steering commands. Lidar inputs are used to generate throttle commands using the same network as the non-vision policy.

C.3 Supplementary videos

We have provided some videos to augment our analysis (available at `http://amansinha.org/docs/OKellySiNaDuTe18_videos.zip`):

- gail.mp4 provides an example of a trained GAIL model driving alongside data traces from real human drivers [286].

- Example videos from rollouts. The filenames start with "mttc =" to indicate the minimum TTC that resulted between the ego and any other vehicle during the rollout. Note that even crashes have nonzero values of TTC due to the definition we used for TTC from the center of the ego vehicle (cf. Appendix C.1). The videos are all played back at 2.5× real-time speed. The videos included in the supplement are:
  - Crashes:
    - mttc = 0.23 − crash.mp4
    - mttc = 0.30.mp4
    - mttc = 0.42.mp4
    - mttc = 0.56.mp4
  - Non-crashes:
    - mttc = 0.23 − nocrash.mp4
    - mttc = 0.79.mp4
    - mttc = 1.43.mp4
    - mttc = 2.01.mp4
    - mttc = 3.05.mp4
    - mttc = 6.00.mp4
    - mttc = 6.01.mp4
    - mttc = 10.11.mp4

  These videos contain overhead, RGB, segmented, and depth views. We also include higher-resolution RGB videos with the same base names as above but the extension "_hires.mp4".

Appendix D

Chapter 5 Appendices

D.1 Warped Hamiltonian Monte Carlo

In this section, we provide a brief overview of Hamiltonian Monte Carlo (HMC) as well as a specific rendition, split HMC [259], as it is used within our setting. Given "position" variables x and "momentum" variables v, we define the Hamiltonian for a dynamical system as $H(x, v)$ which can usually be written as $U(x) + K(v)$, where $U(x)$ is the potential energy and $K(v)$ is the kinetic energy. For MCMC applications, $U(x) = -\log(\rho_0(x))$ and we take $v \sim \mathcal{N}(0, I)$ so that $K(v) = \|v\|^2/2$. In HMC, we start at state x_i and sample $v_i \sim \mathcal{N}(0, I)$. We then simulate the Hamiltonian, which is given by the partial differential equations:

$$\dot{x} = \frac{\partial H}{\partial v}, \quad \dot{v} = -\frac{\partial H}{\partial x}.$$

Of course, this must be done in discrete time for most Hamiltonians that are not perfectly integrable. One notable exception is when x is Gaussian, in which case the dynamical system corresponds to the evolution of a simple harmonic oscillator (*i.e.* a spring-mass system). When done in discrete time, a symplectic integrator must be used to ensure high accuracy. After performing some discrete steps of the system (resulting in the state (x_f, v_f)), we negate the resulting momentum (to make the resulting proposal reversible), and then accept the state $(x_f, -v_f)$ using the standard Metropolis-Hastings criterion: $\min(1, \exp(-H(x_f, -v_f) + H(x_i, v_i)))$ [121].

The standard symplectic integrator—the leap-frog integrator—can be derived using the

Algorithm D.1 WarpedHMC

Input: Sample x, momentum $v \sim \mathcal{N}(0, I)$, transform V_θ and its inverse W_θ, scale factor β, step size ϵ
$y \leftarrow W_\theta(x)$
$\hat{v} \leftarrow v - 0.5\epsilon\beta I\{f(x) > \gamma\} J_{V_\theta}(y) \nabla f(x)$
$\hat{y} \leftarrow y\cos(\epsilon) + \hat{v}\sin(\epsilon)$
$\hat{v} \leftarrow \hat{v}\cos(\epsilon) - y\sin(\epsilon)$
$\hat{x} \leftarrow V_\theta(\hat{y})$
$\hat{v} \leftarrow \hat{v} - 0.5\epsilon\beta I\{f(\hat{x}) > \gamma\} J_{V_\theta}(\hat{y}) \nabla f(\hat{x})$
$\hat{v} \leftarrow -\hat{v}$
$x \leftarrow \hat{x}$ with probability $\min(1, \exp(-H(\hat{y}, \hat{v}) + H(y, v)))$
Return x

following symmetric decomposition of the Hamiltonian (performing a symmetric decomposition retains the reversibility of the dynamics): $H(x, v) = U(x)/2 + K(v) + U(x)/2$. Using simple Euler integration for each term individually results in the following leap-frog step of step-size ϵ:

$$\begin{aligned} v_{1/2} &= v_i - \frac{\epsilon}{2}\frac{\partial U(x_i)}{\partial x} \\ x_f &= x_i + \epsilon\frac{\partial K(v_{1/2})}{\partial v} \\ v_f &= v_{1/2} - \frac{\epsilon}{2}\frac{\partial U(x_f)}{\partial x}, \end{aligned}$$

where each step simply simulates the individual Hamiltonian $H_1(x, v) = U(x)/2$, $H_2(x, v) = K(v)$, or $H_3(x, v) = U(x)/2$ in sequence. As presented by Shahbaba et al. [259], this same decomposition can be done in the presence of more complicated Hamiltonians. In particular, consider the Hamiltonian $H(x, v) = U_1(x) + U_0(x) + K(v)$. We can decompose this in the following manner: $H_1(x, v) = U_1(x)/2$, $H_2(x, v) = U_0(x) + K(v)$, and $H_3(x, v) = U_1(x)/2$. We can apply Euler integration to the momentum v for the first and third Hamiltonians and the standard leap-frog step to the second Hamiltonian (or even analytic integration if possible). For this work, we have $U_0(x) = -\log \rho_0(x)$ and $U_1(x) = -\beta[\gamma - f(x)]_-$.

To account for warping, the modifications needed to the HMC steps above are simple. When performing warping, we simply perform HMC for a Hamiltonian $\hat{H}(y, v)$ that is defined with respect to the warped position variable y, where $x = V_\theta(y)$ for given parameters θ. By construction of the normalizing flows, we assume $y \sim \mathcal{N}(0, I)$, so that we can perform the dynamics for $\hat{H}_2(y, v)$ analytically. Furthermore, the Jacobian $J_{V_\theta}(y)$ is necessary for performing the Euler integration of $H_1(y, v)$ and $H_3(y, v)$. This is summarized in Algorithm

D.1. Note that we always perform the Metropolis-Hastings acceptance with respect to the true Hamiltonian H, rather than the Hamiltonian $\hat{H}$ that assumes perfect training of the normalizing flows.

D.2 Performance analysis

D.2.1 Proof of Proposition 5.1

We begin with showing the convergence of the number of iterations. To do this, we first show almost sure convergence of β_k in the limit $N \to \infty$. We note that in the optimization problem (5.8), β_k is a feasible point, yielding $b_k(\beta) = 1$. Thus, $\beta_{k+1} \geq \beta_k \geq \beta_0 := 0$. Due to this growth of β_k with k, we have

$$\frac{Z_{k+1}}{Z_k} = \mathbb{E}_{P_k}\left[\frac{\rho_{k+1}(X)}{\rho_k(X)}\right] \leq 1,$$

$$\begin{aligned}
\mathbb{P}_k(f(X) \leq \gamma) &= \mathbb{E}_{P_{k+1}}\left[\frac{Z_{k+1}}{Z_k}\frac{\rho_k(X)}{\rho_{k+1}(X)} I\{f(X) \leq \gamma)\}\right] \\
&= \frac{Z_{k+1}}{Z_k}\mathbb{E}_{P_{k+1}}\left[I\{f(X) \leq \gamma)\}\right] \\
&\leq \mathbb{P}_{k+1}(f(X) \leq \gamma).
\end{aligned}$$

By the unfiorm convergence of empirical measures offered by the Glivenko-Cantelli Theorem, the value $a_k \to \mathbb{P}_k(f(X) \leq \gamma)$ almost surely. Then, the stop condition can be rewritten as $b_k(\beta) \geq a_k/s \to \mathbb{P}_k(f(X) \leq \gamma)/s \geq p_\gamma/s$. Since $b_k(\beta)$ is monotonically decreasing in the quantity $\beta - \beta_k$, this constraint gives an upper bound for β_{k+1}, and, as a result, all β_k are almost surely bounded from above and below. We denote this interval as $\mathcal{B}$.

Now, we consider the convergence of the solutions to the finite N versions of problem (5.8), denoted β_k^N, to the "true" optimizers β_k in the limit as $N \to \infty$. Leaving the dependence on β_k implicit for the moment, we consider the random variable $Y := g(X;\beta) := \exp\left((\beta - \beta_k)[\gamma - f(X)]_-\right)$. Then, since $\beta \in \mathcal{B}$ is bounded and g is continuous in β, we can state the Glivenko-Cantelli convergence of the empirical measure uniformly over $\mathcal{B}$: $\sup_{\beta\in\mathcal{B}} \|F^N(Y) - F(Y)\|_\infty \to 0$ almost surely, where F is the cumulative distribution function for Y. Note that the constraints in the problem (5.8) can be rewritten as expectations of this random variable Y. Furthermore, the function g is strictly monotonic in β

(and therefore invertible) for non-degenerate $f(X)$ (*i.e.* $f(x) > \gamma$ for some non-negligible measure under P_0). Thus, we have almost sure convergence of the argmin β_{k+1}^N to β_{k+1}.

Until now, we have taken dependence on β_k implicitly. Now we make the dependence explicit to show the final step of convergence. In particular, we can write β_{k+1} as a function of β_k (along with their empirical counterparts), For concreteness, we consider the following decomposition for two iterations:

$$|\beta_2^N(\beta_1^N) - \beta_2(\beta_1)| \leq |\beta_2^N(\beta_1^N) - \beta_2(\beta_1^N)| + |\beta_2(\beta_1^N) - \beta_2(\beta_1)|.$$

We have already shown above that the first term on the right hand side vanishes almost surely. By the same reasoning, we know that $\beta_1^N \to \beta_1$ almost surely. The second term also vanishes almost surely since $\beta_{k+1}(\beta)$ is a continuous mapping. This is due to the fact that the constraint functions in problem (5.8) are continuous functions of both β and β_k along with the invertibility properties discussed previously. Then, we simply extend the telescoping series above for any k and similarly show that all terms vanish almost surely. This shows the almost sure convergence for all β_k up to some K.

Now we must show that K is bounded and almost surely converges to a constant. To do this we explore the effects of the optimization procedure. Assuming the stop condition (the second constraint) does not activate, the first constraint in problem (5.8) has the effect of making $\mathbb{Z}_{k+1}/Z_k = \alpha$ (almost surely), which implies $\mathbb{P}_{k+1}(f(X) \leq \gamma) = \mathbb{P}_k(f(X) \leq \gamma)/\alpha$. In other words, we magnify the event of interest by a factor of $1/\alpha$. The second constraint can be rewritten as $\mathbb{P}_{k+1}(f(X) \leq \gamma) \leq s$. Thus, we magnify the probability of the region of interest by factors of α unless doing so would increase the probability to greater than s. In that case, we conclude with setting the probability to s (since $\mathbb{P}_\beta(f(X) \leq \gamma)$ is monotonically increasing in β). In this way, we have 0 iterations for $p_\gamma \in [s, 1]$, 1 iteration for $p_\gamma \in [\alpha s, s)$, 2 iterations for $p_\gamma \in [\alpha^2 s, \alpha s)$, and so on. Then, the total number of iterations is (almost surely) $\lfloor \log(p_\gamma)/\log(\alpha) \rfloor + I\{p_\gamma/\alpha^{\lfloor \log(p_\gamma)/\log(\alpha) \rfloor} < s\}$.

Now we move to the relative mean-square error of $\hat{p}_\gamma$. We employ the delta method, whereby, for large N, this is equivalent to $\mathrm{Var}(\log(\hat{p}_\gamma))$ (up to terms $o(1/N)$). For notational

convenience, we decompose $\widehat{E}_k$ into its numerator and denominator:

$$A_k(X) := \rho_k^B(X)/\rho_{k-1}(X), \qquad \widehat{A}_k := \frac{1}{N}\sum_{i=1}^{N} A_k(x_i^{k-1})$$
$$B_k(X) := \rho_k^B(X)/\rho_k(X), \qquad \widehat{B}_k := \frac{1}{N}\sum_{i=1}^{N} B_k(x_i^{k}).$$

By construction (and assumption of large T), Algorithm 5.1 has a Markov property that each iteration's samples x_i^k are independent of the previous iterations' samples x_i^{k-1} given β_k. For shorthand, let $\beta_{0:k}$ denote all $\beta_0, \ldots, \beta_k$. Conditioning on $\beta_{0:k}$, we have

$$\mathrm{Var}(A_k) = \mathrm{Var}\left(\mathbb{E}[A_k|\beta_{0:k}]\right) + \mathbb{E}\left[\mathrm{Var}\left(A_k|\beta_{0:k}\right)\right].$$

Since $\beta_{0:k}$ approaches constants almost surely as $N \to \infty$, the first term vanishes and the second term is the expectation of a constant. In particular, the second term is as follows:

$$\begin{aligned}
\mathrm{Var}\left(A_k|\beta_{0:k}\right) &= \mathbb{E}\left[A_k^2|\beta_{0:k}\right] - \left(\mathbb{E}\left[A_k|\beta_{0:k}\right]\right)^2 \\
&= \mathbb{E}_{P_{k-1}}\left[\frac{\rho_k(X)}{\rho_{k-1}(X)}\right] - \left(\mathbb{E}_{P_{k-1}}\left[\sqrt{\frac{\rho_k(X)}{\rho_{k-1}(X)}}\right]\right)^2 \\
&= \frac{Z_k}{Z_{k-1}} - \left(\frac{Z_k^B}{Z_{k-1}}\right)^2.
\end{aligned}$$

Similarly, $\mathrm{Var}(B_k|\beta_{0:k}) = Z_{k-1}/Z_k - (Z_k^B/Z_k)^2$. Next we look at the covariance terms:

$$\mathrm{Cov}(A_{k-1}, A_k) = \mathrm{Cov}\left(\mathbb{E}[A_{k-1}|\beta_{0:k}], \mathbb{E}[A_k|\beta_{0:k}]\right) + \mathbb{E}\left[\mathrm{Cov}\left(A_{k-1}, A_k|\beta_{0:k}\right)\right].$$

Again, the first term vanishes since $\beta_{0:k}$ approach constants as $N \to \infty$. By construction, the second term is also 0 since the quantities are conditionally independent. Similarly, $\mathrm{Cov}(B_{k-1}, B_k) = 0$ and $\mathrm{Cov}(A_i, B_j) = 0$ for $j \neq i-1$. However, there is a nonzero

covariance for the quantities that depend on the same distribution:

$$\begin{aligned}\mathrm{Cov}\left(B_k, A_{k+1}|\beta_{0:k+1}\right) &= \mathbb{E}\left[B_k A_{k+1}|\beta_{0:k+1}\right] - \mathbb{E}\left[B_k|\beta_{0:k+1}\right]\mathbb{E}\left[A_{k+1}|\beta_{0:k+1}\right] \\ &= \mathbb{E}_{P_k}\left[\frac{\sqrt{\rho_{k-1}(X)\rho_{k+1}(X)}}{\rho_k(X)}\right] - \frac{Z_{k+1}^B}{Z_k}\frac{Z_k^B}{Z_k} \\ &= \frac{Z_k^C}{Z_k} - \frac{Z_{k+1}^B}{Z_k}\frac{Z_k^B}{Z_k}.\end{aligned}$$

By the large T assumption, the samples x_i^k and x_j^k are independent for all $i \neq j$ given β_k. Then we have

$$\mathrm{Var}(\widehat{A}_k|\beta_{0:k}) = \mathrm{Var}(A_k|\beta_{0:k})/N, \;\; \mathrm{Var}(\widehat{B}_k|\beta_{0:k}) = \mathrm{Var}(B_k|\beta_{0:k})/N,$$

$$\mathrm{Cov}(\widehat{B}_k, \widehat{A}_{k+1}|\beta_{0:k+1}) = \mathrm{Cov}(B_k, A_{k+1}|\beta_{0:k+1})/N.$$

The last term in $\hat{p}_\gamma$, $\frac{1}{N}\sum_{i=1}^N \frac{\rho_\infty(x_i^K)}{\rho_K(x_i^K)}$, reduces to a simple Monte Carlo estimate since $\frac{\rho_\infty(X)}{\rho_K(X)} = I\{f(X) \leq \gamma\}$. Furthermore, this quantity is independent of all other quantities given $\beta_{0:K}$ and, as noted above, approaches s almost surely as $N \to \infty$.

Putting this all together, the delta method gives (as $N \to \infty$ so that $\beta_{0:K}$ approach constants almost surely),

$$\begin{aligned}\mathrm{Var}(\log(\hat{p}_\gamma)) &\to \sum_{k=1}^K \left(\frac{\mathrm{Var}(\widehat{A}_k)}{(Z_k^B/Z_{k-1})^2} + \frac{\mathrm{Var}(\widehat{B}_k)}{(Z_k^B/Z_k)^2}\right) - 2\sum_{k=1}^{K-1}\frac{\mathrm{Cov}(\widehat{B}_k, \widehat{A}_{k+1})}{Z_{k+1}^B Z_k^B/Z_k^2} + \frac{1-s}{sN} \\ &= \frac{2}{N}\sum_{k=1}^K\left(\frac{Z_{k-1}Z_k}{(Z_k^B)^2} - 1\right) - \frac{2}{N}\sum_{k=1}^{K-1}\left(\frac{Z_k^C Z_k}{Z_k^B Z_{k+1}^B} - 1\right) + \frac{1-s}{sN} + o\left(\frac{1}{N}\right).\end{aligned}$$

The Bhattacharrya coefficient can be written as

$$G(P_{k-1}, P_k) = \int_{\mathcal{X}} \sqrt{\frac{\rho_{k-1}(x)}{Z_{k-1}}\frac{\rho_k(x)}{Z_k}}dx = \frac{Z_k^B}{\sqrt{Z_{k-1}Z_k}}.$$

Furthermore, we have

$$\frac{G(P_{k-1}, P_{k+1})}{G(P_{k-1}, P_k)G(P_k, P_{k+1})} = \frac{Z_k^C}{\sqrt{Z_{k-1}Z_{k+1}}}\frac{\sqrt{Z_{k-1}Z_k}}{Z_k^B}\frac{\sqrt{Z_k Z_{k+1}}}{Z_{k+1}^B} = \frac{Z_k^C Z_k}{Z_k^B Z_{k+1}^B},$$

yielding this final result

$$\mathrm{Var}(\log(\hat{p}_\gamma)) \to \frac{2}{N}\sum_{k=1}^{K}\left(\frac{1}{G(P_{k-1},P_k)^2}-1\right) - \frac{2}{N}\sum_{k=1}^{K-1}\left(\frac{G(P_{k-1},P_{k+1})}{G(P_{k-1},P_k)G(P_k,P_{k+1})}-1\right) + \frac{1-s}{sN} + o\left(\frac{1}{N}\right). \quad \text{(D.1)}$$

We remark that a special case of this formula is for $K = 1$ and $s = 1$ (so only the first term survives), which is the relative mean-square error for a single bridge-sampling estimate $\widehat{E}_k$.

Now, since $G(P, Q) \geq 0$, the terms in the second sum are ≥ -1 so that the second sum is $\leq 2(K-1)/N$. Furthermore, since $s \geq 1/3$, the last term is also $\leq 2/N$. Thus, if we have $\frac{1}{G(P_{k-1},P_k)^2} \leq D$ (with $D \geq 1$), then the asymptotic relative mean-square error (D.1) is $\leq 2KD/N$ (up to terms $o\left(\frac{1}{N}\right)$).

When performing warping, we follow the exact same pattern as the above results, conditioning on both $\beta_{0:k}$ and $W_{0:k}$, where W_0 is defined as the identity mapping. We follow the same almost-sure convergence proof for W_k as above for β_k, which requires compactness of $\theta \in \Theta$, continuity of W with respect to θ and x, and that we actually achieve the minimum in problem (5.6). Although the first two conditions are immediate in most applications, the last condition can be difficult to satisfy for deep neural networks due to the nonconvexity of the optimization problem.

D.3 Experimental setups

D.3.1 Hyperparameters

The number of samples N affects the absolute performance of all of the methods tested, but not their relative performance with respect to each other. For all experiments, we use $N = 1000$ for B and NB to have adequate absolute performance given our computational budget (see below for the computing architecture used). Other hyperparameters were tuned on the synthetic problem and fixed for the rest of the experiments (with the exception of the MAF architecture for the rocket experiments). The hyperparameters were chosen as follows.

When performing Hamiltonian dynamics for a Gaussian variable, a time step of 2π results in no motion and time step of π results in a mode reversal, where both the velocity and position are negated. The π time step is in this sense the farthest exploration that can occur in phase space (which can be intuitively understood by recognizing that the phase diagram of a simple spring-mass system is a unit circle). Thus, we considered $T = 4, 8, 12,$

and 16 with time steps π/T. We found that $T = 8$ provided reasonable exploration (as measured by autocorrelations and by the bias of the final estimator $\hat{p}_\gamma$) and higher values of T did not provide much more benefit. For B, we allowed 2 more steps $T = 10$ to keep the computational cost the same across B and NB. Similarly, for AMS, we set $T = 10$. We also performed tuning online for the time step to keep the accepatance ratio between 0.4 and 0.8. This was done by setting the time step to $\sin^{-1}(\min(1, \sin(t)\exp((p - C)/2))$, where t is the current time step, p is the running acceptance probability for a single chain and $C = 0.4$ if $p < 0.4$ or 0.8 if $p > 0.8$. This was done after every T HMC steps.

For the step size of the bridge, we considered $\alpha \in \{0.01, 0.1, 0.3, 0.5\}$. Smaller α results in fewer iterations and better computational efficiency. However, we found that very small α made MAF training difficult (see below for the MAF architectures used). We settled on $\alpha = 0.3$, which provided reasonable computational efficiency (no more than 11 iterations for the synthetic problem) as well as stable MAF training. For AMS, we followed the hyperparameter settings of Webb et al. [298]. Namely, we chose a culling fraction of $\alpha_{\text{AMS}} = 10\%$, where α_{AMS} sets the fraction of particles that are removed and rejuvenated at each iteration [298].

The MAF architectures for the synthetic, MountainCar, and CarRacing experiments were set at 5 MADE units, each with 1 hidden layer of 100 neurons. Because the rocket search space is very high dimensional, we decreased the MAF size for computational efficiency: we set it at 2 MADE units, each with hidden size 400 units. We used 100 epochs for training, a batch size of 100, a learning rate of 0.01 and an exponential learning-rate decay with parameter 0.95.

Given the above parameters, the number of simulations for each experiment varies based on the final probability in question p_γ (smaller values result in more simulations due to having a higher number of iterations K). We had runs of 111000, 101000, 91000, 71000, 91000, and 101000 simulations respectively for the synthetic, MountainCar, Rocket1, Rocket2, AttentionAgentRacer, and WorldModelRacer environments. We used these values as well as the ground truth p_γ values to determine the number of particles allowed for AMS, $N_{\text{AMS}} = 920, 910, 820, 780, 820, 910$ respectively, as AMS has a total cost of $N_{\text{AMS}}(1 + \alpha_{\text{AMS}} T K_{\text{AMS}})$, where $K_{\text{AMS}} \approx \log(p_\gamma)/\log(1 - \alpha_{\text{AMS}})$.

For the surrogate Gaussian process regression model for CarRacing, we retrained the model on the most recent N simulations after every NT simulations (*e.g.* after every T HMC iterations). This made the amortized cost of training the surrogate model negligible

compared to performing the simulations themselves. We used a Matern kernel with parameter $\nu = 2.5$. We optimized the kernel hyperparameters using an L-BFGS quasi-Newton solver.

Computing infrastructure and parallel computation Experiments were carried out on commodity CPU cloud instances, each with 96 Intel Xeon cores @ 2.00 GHz and 85 GB of RAM. AMS, B, and NB are all designed to work in a Map-Reduce paradigm, where a central server orchestrates many worker jobs followed by synchronization step. AMS requires more iterations and fewer parallel worker threads per iteration than B and NB. In particular, whereas B and NB perform N parallel jobs per iteration, AMS only performs $\alpha_{\text{AMS}} N_{\text{AMS}}$ parallel jobs per iteration. Thus, B and NB take advantage of massive scale and parallelism much more than AMS.

D.3.2 Environment details

MountainCar

The MountainCar environment considers a simple car driving on a mountain road. The car can sense horizontal distance s as well as its velocity v, and may send control inputs u (the amount of power applied in either the forward or backward direction). The height of the road is given by: $h(s) = 0.45\sin(3s) + 0.55$. The speed of the car, v, is a function of s and u only. Thus, the discrete time dynamics are: $s_{k+1} = s_k + v_{k+1}$ and $v_{k+1} = v_k + 0.0015u_k - 0.0025\cos(3s_k)$. For a given episode the agent operating the car receives a reward of $-0.1u_k^2$ for each control input and 100 for reaching the goal state.

In this experiment we explore the effect of domain shift on a formally verified neural network. We utilize the neural network designed by Ivanov et al. [141]; it contains two hidden layers, each of 16 neurons, for a total of 337 parameters. For our experiments we use the trained network parameters available at: `https://github.com/Verisig/verisig`. Ivanov et al. [141] describe a layer-by-layer approach to verification which over-approximates the reachable set of the combined dynamics of the environment and the neural network. An encoding of this system (network and environment) is developed for the tool Flow* [65] which constructs the (overapproximate) reachable set via a Taylor approximation of the combined dynamics.

The MountainCar environment is considered solved if a policy achieves an average reward of 90 over 100 trials. The authors instead seek to prove that the policy will achieve a

reward of at least 90 for any initial condition. By overapproximating the reachable states of the system, they show that the car always receives a total reward greater than 90 and achieves the goal in less than 115 steps for a subset of the intial conditions $\hat{p}_0 \in [-0.59, -0.4]$.

Rocket design

The system under test is a rocket spacecraft with dynamics $m\ddot{p} = f - mge_3$, where $m > 0$ is the mass, $p(t) \in \mathbf{R}^3$ is the position, and e_3 is the unit vector in the z-direction. While it is possible to synthesize optimal trajectories for an idealized model of the system, significant factors such as wind and engine performance (best modeled as random variables) are unaccounted for [38]. Without feedback control, even small uncorrected tracking errors result in loss of the vehicle. In the case of disturbances the authors suggest two approaches: (1) a feedback control law which tracks the optimal trajectory (2) receding horizon model predictive control. The system we consider tracks an optimal trajectory using a feedback control law. Namely, the optimal trajectory is given by the minimum fuel solution to a linearized mode of the dynamics. Specifically, we consider the thrust force discretized in time with a zero-order hold, such that f_k applied for time $t \in [(k-1)h, kh]$ for a time step $h = 0.2$. Then, the reference thrust policy solves the following convex optimization problem

$$\begin{aligned}
&\text{minimize} \sum_{i=1}^{K} \|f_k\|_2 \\
&\text{such that } p_K = v_K = 0, \|f_k\| \leq F_{\max}, \\
&\qquad v_{k+1} - v_k = \frac{h}{m} f_k - hge_3, \\
&\qquad p_{k+1} - p_k = \frac{h}{2}\left(v_k + v_{k+1}\right), \\
&\qquad (p_3)_k \geq 0.5\|((p_1)_k, (p_2)_k)\|_2,
\end{aligned}$$

where the last constraint is a minimum glide slope and $F_{\max}$ is a maximum thrust value for the nominal thrusters. This results in the thrust profile $f^\star$. The booster thrusters correct for disturbances along the flight. The disturbances at every point in time follow a mixture of Gaussians. Namely, we consider 3 wind gust directions, $w_1 = (1,1,1)/\sqrt(3)$, $w_2 = (0,1,0)$, and $w_3 = (1,0,0)$. For every second in time, the wind follows a mixture:

$$W \sim \mathcal{N}(0, I) + w_1 B + w_2 \hat{B} + (1 - \hat{B}) w_3,$$

where $B \sim$ Bernoulli(1/3) and $\hat{B} \sim$ Bernoulli(1/2). This results in 5 random variables for each second, or a total of 100 random variables since we have a 20 second simulation. The wind intensity experienced by the rocket is a linear function of height (implying a simplistic laminar boundary layer): $f_w = CWp_3$ for a constant C. Finally, the rocket has a proportional feedback control law for the booster thrusters to the errors in both the position p_k and velocity v_k:

$$f_{\text{feedback,k}} = \text{clip-by-norm}(f_k^\star - K_p(p_k - p_k^\star) - K_v(v_k - v_k^\star)).$$

The maximum norm for clip-by-norm is aF_{max}, where $a = 1.15$ for Rocket1 and $a = 1.1$ for Rocket2, indicating that the boosters are capable of providing 15% or 10% of the thrust of the main engine.

Car Racing

We compare the failure rate of agents solving the car-racing task utilizing the two distinct approaches ([118] and [279]). The car racing task differs from the other experiments due to the inclusion of a (simple) renderer in the system dynamics. At each the step the agent recieves a reward of $-0.1 + \mathcal{I}_{newtile}(1000/N) - \mathcal{I}_{offtrack}(100)$ where N is the total number of tiles visited in the track. The environment is considered solved if the agent returns an average reward of 900 over 100 trials. The search space P_0 is the inherent randomness involved with generating a track. The track is generated by selecting 12 checkpoints in polar coordinates, each with radian value uniformly in the interval $[2\pi i/12, 2\pi(i+1)/12)$ for $i = 0, \ldots 11$, and with radius uniformly in the interval $[R/3, R]$, for a given constant value R. This results in 24 parameters in the search space. The policies used for testing are described below:

AttentionAgent Tang et al. [279] utilize a simple self-attention module to select patches from a 96x96 pixel observation. First the input image is normalized then a sliding window approach is used to extract N patches of size $M \times M \times 3$ which are flattened and arranged into a matrix of size $3M^2 \times N$. The self-attention module is used to compute the attention matrix A and importance vector (summation of each column of A). A feature extraction operation is applied to the top K elements of the sorted importance vector and the selected features are input to a neural network controller. Both the attention module and the controller

are trained together via CMA-ES. Together, the two modules contain approximately 4000 learnable parameters.

WorldModel The agent of Ha and Schmidhuber [118] first maps a top-down image of the car on track via a variational autoencoder to a latent vector z. Given z, the world model M utilizes a recurrent-mixture density network [37] to model the distribution of future possible states $P(z_{t+1} \mid a_t, z_t, h_t)$. Note that h_t, the hidden state of the RNN. Finally, a simple linear controller C maps the concatenation of z_t and h_t to the action, a_t.

Bibliography

[1] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius. Functional gradient descent method for metric temporal logic specifications. In *2014 American Control Conference*, pages 2312–2317. IEEE, 2014.

[2] H. Abbas, M. O'Kelly, A. Rodionova, and R. Mangharam. Safe at any speed: A simulation-based test harness for autonomous vehicles. LNCS. Springer, 2018.

[3] J. Abernethy and A. Rakhlin. Beating the adaptive bandit with high probability. In *2009 Information Theory and Applications Workshop*, pages 280–289. IEEE, 2009.

[4] N. Agarwal, B. Bullins, E. Hazan, S. M. Kakade, and K. Singh. Online control with adversarial disturbances. *arXiv preprint arXiv:1902.08721*, 2019.

[5] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[6] M. Althoff. An introduction to cora 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[7] M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

[8] M. Althoff, M. Koschi, and S. Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.

[9] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on*

Tools and Algorithms for the Construction and Analysis of Systems, pages 254–257. Springer, 2011.

[10] O. Arenz, M. Zhong, G. Neumann, et al. Efficient gradient-free variational inference using policy search. 2018.

[11] S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592. , 2014.

[12] K. Arulkumaran, A. Cully, and J. Togelius. Alphastar: An evolutionary computation perspective. *arXiv preprint arXiv:1902.01724*, 2019.

[13] S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Springer, 2007.

[14] K. J. Åström and P. Eykhoff. System identification—a survey. *Automatica*, 7(2): 123–162, 1971.

[15] K. J. Åström and B. Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[16] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multi-armed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[17] J. A. Bagnell, A. Y. Ng, and J. G. Schneider. Solving uncertain markov decision processes. 2001.

[18] C. Baier, J.-P. Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[19] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[20] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[21] 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, 2017.

[22] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[23] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6241–6250. , 2017.

[24] C. J. Bélisle, H. E. Romeijn, and R. L. Smith. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266, 1993.

[25] A. Bemporad and M. Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 1999.

[26] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2010.

[27] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[28] A. Ben-Tal, D. den Hertog, A. D. Waegenaere, B. Melenberg, and G. Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[29] Y. Benkler. Don't let industry write the rules for ai. *Nature*, 569(7754):161–162, 2019.

[30] C. H. Bennett. Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, 22(2):245–268, 1976.

[31] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[32] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dkebiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[33] D. Bertsimas and M. Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

[34] D. Bertsimas, V. Gupta, and N. Kallus. Data-driven robust optimization. *arXiv:1401.0212 [math.OC]*, 2013. URL http://arxiv.org/abs/1401.0212.

[35] M. Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[36] P. Billingsley. *Convergence of Probability Measures*. Wiley, Second edition, 1999.

[37] C. M. Bishop. Mixture density networks. Technical report, Citeseer, 1994.

[38] L. Blackmore. Autonomous precision landing of space rockets. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*. National Academies Press, 2017.

[39] J. Blanchet and K. Murthy. Quantifying distributional model risk via optimal transport. *arXiv:1604.01446 [math.PR]*, 2016. URL https://arxiv.org/abs/1604.01446.

[40] J. Blanchet, Y. Kang, and K. Murthy. Robust Wasserstein profile inference and applications to machine learning. *arXiv:1610.05627 [math.ST]*, 2016.

[41] J. Blanchet, Y. Kang, F. Zhang, and K. Murthy. Data-driven optimal transport cost selection for distributionally robust optimizatio. *arXiv:1705.07152 [stat.ML]*, 2017.

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

[43] 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 arXiv:1604.07316*, 2016.

[44] J. F. Bonnans and A. Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

[45] J.-F. Bonnefon, A. Shariff, and I. Rahwan. The social dilemma of autonomous vehicles. *Science*, 352(6293):1573–1576, 2016. ISSN 0036-8075. doi: 10.1126/science.aaf2654. URL http://science.sciencemag.org/content/352/6293/1573.

[46] F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[47] S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: a survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

[48] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: a Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[49] N. E. Boudette. Despite high hopes, self-driving cars are 'way in the future'. *New York Times*, 17, 2019.

[50] C.-E. Bréhier, T. Lelièvre, and M. Rousset. Analysis of adaptive multilevel splitting algorithms in an idealized case. *ESAIM: Probability and Statistics*, 19:361–394, 2015.

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

[52] T. Brown, D. Mane, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. In *Machine Learning and Computer Security Workshop, Neural Information Processing Systems*, 2017.

[53] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[54] M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong, et al. Toward trustworthy ai development: Mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*, 2020.

[55] S. Bubeck, N. Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1): 1–122, 2012.

[56] J. Bucklew. *Introduction to rare event simulation*. Springer Science & Business Media, 2013.

[57] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[58] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[59] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[60] F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.

[61] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[62] M. Cheah, S. A. Shaikh, J. Bryans, and H. N. Nguyen. Combining third party components securely in automotive systems. In *IFIP International Conference on Information Security Theory and Practice*, pages 262–269. Springer, 2016.

[63] M.-H. Chen, Q.-M. Shao, and J. G. Ibrahim. *Monte Carlo methods in Bayesian computation*. Springer Science & Business Media, 2012.

[64] X. Chen, M. Sim, and P. Sun. A robust optimization perspective on stochastic programming. *Operations Research*, 55(6):1058–1071, 2007.

[65] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[66] Y. Chen, G. Lan, and Y. Ouyang. Accelerated schemes for a class of variational inequalities. *arXiv:1403.4164 [math.OC]*, 2014.

[67] Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu. Fast mixing of metropolized hamiltonian monte carlo: Benefits of multi-step gradients. *arXiv preprint arXiv:1905.12247*, 2019.

[68] H. Choi, E. Jang, and A. A. Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

[69] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[70] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728. , 2016.

[71] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer. A survey of algorithms for black-box safety validation. *arXiv preprint arXiv:2005.02979*, 2020.

[72] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[73] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[74] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pages 4188–4197, 2018.

[75] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[76] P. Del Moral. Feynman-kac formulae. In *Feynman-Kac Formulae*, pages 47–93. Springer, 2004.

[77] P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[78] E. Delage and Y. Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, 2010.

[79] W. Ding and S. Shen. Online vehicle trajectory prediction using policy anticipation network and optimization-based context reasoning. *arXiv preprint arXiv:1903.00847*, 2019.

[80] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

[81] 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.

[82] A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[83] J. Doyle, K. Glover, P. Khargonekar, and B. Francis. State-space solutions to standard h_2 and h_∞ control problems. In *1988 American Control Conference*, pages 1691–1696. IEEE, 1988.

[84] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

[85] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia. Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *International Conference on Computer Aided Verification*, pages 432–442. Springer, 2019.

[86] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[87] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto thel1-ball for learning in high dimensions. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008. doi: 10.1145/1390156.1390191. URL http://dx.doi.org/10.1145/1390156.1390191.

[88] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

[89] J. C. Duchi, P. W. Glynn, and H. Namkoong. Statistics of robust optimization: A generalized empirical likelihood approach. *arXiv:1610.03425 [stat.ML]*, 2016. URL https://arxiv.org/abs/1610.03425.

[90] A. Durmus, E. Moulines, et al. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017.

[91] C. Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[92] G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv:1703.11008 [cs.LG]*, 2017.

[93] P. M. Esfahani and D. Kuhn. Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *arXiv:1505.05116 [math.OC]*, 2015.

[94] J. M. Esposito, J. Kim, and V. Kumar. Adaptive rrts for validating hybrid robotic control systems. In *Algorithmic Foundations of Robotics VI*, pages 107–121. Springer, 2004.

[95] Federation Internationale de l'Automobile. Formula one 2019 results. https://www.formula1.com/en/results.html/2019/, 2019.

[96] D. Ferguson, T. M. Howard, and M. Likhachev. Motion planning in urban environments. *Journal of Field Robotics*, 25(11-12):939–960, 2008.

[97] C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[98] U. Food, D. Administration, et al. Artificial intelligence and machine learning in software as a medical device. *Silverspring: US Food and Drug Administration*, 2019.

[99] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[100] S. Fujimoto, H. Van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[101] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learningearning*, pages 1050–1059, 2016.

[102] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction. In *Robotics: Science and Systems*, volume 1, 2015.

[103] E. Games. Unreal engine 4 documentation. *URL https://docs. unrealengine. com/latest/INT/index. html*, 2015.

[104] R. Gao and A. J. Kleywegt. Distributionally robust stochastic optimization with wasserstein distance. *arXiv:1604.02199 [math.OC]*, 2016.

[105] R. Gao, X. Chen, and A. J. Kleywegt. Wasserstein distributional robustness and regularization in statistical learning. *arXiv:1712.06050 [cs.LG]*, 2017.

[106] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014.

[107] E. Gat, R. P. Bonnasso, R. Murphy, et al. On three-layer architectures. *Artificial intelligence and mobile robots*, 195:210, 1998.

[108] A. Gelman and X.-L. Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185, 1998.

[109] C. J. Geyer. Markov chain monte carlo maximum likelihood. 1991.

[110] S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[111] I. Gilboa and M. Marinacci. Ambiguity and the bayesian paradigm. In *Readings in formal epistemology*, pages 385–439. Springer, 2016.

[112] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[113] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[114] J. Goh and M. Sim. Distributionally robust optimization and its tractable approximations. *Operations Research*, 58(4):902–917, 2010.

[115] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[116] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[117] A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[118] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[119] J. M. Hammersley and D. C. Handscomb. Monte carlo methods. 1964.

[120] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[121] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[122] A. J. Hawkins. Inside the lab where waymo is building the brains for its driverless cars. *The Verge*, 2018.

[123] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813, 2016.

[124] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[125] W. He, J. Wei, X. Chen, N. Carlini, and D. Song. Adversarial example defenses: Ensembles of weak defenses are not strong. *arXiv:1706.04701 [cs.LG]*, 2017.

[126] H. Heinecke, K.-P. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leflour, J.-L. Maté, K. Nishikawa, and T. Scharnhorst. Automotive open system architecture-an industry-wide initiative to manage the complexity of emerging automotive e/e-architectures. Technical report, SAE Technical Paper, 2004.

[127] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.

[128] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

[129] T. C. Hesterberg and B. L. Nelson. Control variates for probability and quantile estimation. *Management Science*, 44(9):1295–1312, 1998.

[130] P. Hintjens. *ZeroMQ: messaging for many applications.* " O'Reilly Media, Inc.", 2013.

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

[132] M. Hoffman, P. Sountsov, J. V. Dillon, I. Langmore, D. Tran, and S. Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

[133] T. Homem-de Mello. A study on the cross-entropy method for rare-event probability estimation. *INFORMS Journal on Computing*, 19(3):381–394, 2007.

[134] T. M. Howard. *Adaptive model-predictive motion planning for navigation in complex environments*. Carnegie Mellon University, 2009.

[135] J. Hu and P. Hu. On the performance of the cross-entropy method. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 459–468. IEEE, 2009.

[136] J. Hu and P. Hu. Annealing adaptive search, cross-entropy, and stochastic approximation in global optimization. *Naval Research Logistics (NRL)*, 58(5):457–477, 2011.

[137] J. Hu, P. Hu, and H. S. Chang. A stochastic approximation framework for a class of randomized optimization algorithms. *IEEE Transactions on Automatic Control*, 57 (1):165–178, 2012.

[138] J. Hu, E. Zhou, and Q. Fan. Model-based annealing random search with stochastic averaging. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 24(4):21, 2014.

[139] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.

[140] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[141] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the*

22nd ACM International Conference on Hybrid Systems: Computation and Control, pages 169–178. ACM, 2019.

[142] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Case study: verifying the safety of an autonomous racing car with a neural network controller. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–7, 2020.

[143] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[144] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[145] A. Juditsky, A. Nemirovski, and C. Tauvel. Solving variational inequalities with the stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

[146] L. P. Kaelbling. The foundation of efficient robot learning. *Science*, 369(6506):915–916, 2020. ISSN 0036-8075. doi: 10.1126/science.aaz7597. URL https://science.sciencemag.org/content/369/6506/915.

[147] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[148] H. Kahn and T. Harris. Estimation of particle transmission by random sampling. 1951.

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

[150] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[151] A. Karpathy. Software 2.0. *Medium. com*, 2017.

[152] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[153] A. Kelly and B. Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[154] C. F. Kerry and J. Karsten. Gauging investment in self-driving cars. *Brookings Institution, October*, 16, 2017.

[155] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2, 2011.

[156] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[157] 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.

[158] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[159] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[160] O. Klimov. Carracing-v0. 2016. *URL https://gym. openai. com/envs/CarRacing-v0*.

[161] M. J. Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[162] J. Z. Kolter and E. Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv:1711.00851 [cs.LG]*, 2017. URL https://arxiv.org/abs/1711.00851.

[163] S. Kong, S. Gao, W. Chen, and E. Clarke. dreach: δ-reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015.

[164] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018.

[165] D. P. Kroese, R. Y. Rubinstein, and P. W. Glynn. The cross-entropy method for estimation. *Handbook of Statistics: Machine Learning: Theory and Applications*, 31: 19–34, 2013.

[166] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. Imitating driver behavior with generative adversarial networks. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 204–211. IEEE, 2017.

[167] D. Kuhn, P. M. Esfahani, V. A. Nguyen, and S. Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In *Operations Research & Management Science in the Age of Analytics*, pages 130–166. INFORMS, 2019.

[168] A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[169] P. R. Kumar. A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23(3):329–380, 1985.

[170] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *arXiv:1611.01236 [cs.CV]*, 2016.

[171] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*, pages 585–591. Springer, 2011.

[172] H. Lam. Recovering best statistical guarantees via the empirical divergence-based distributionally robust optimization. *Operations Research*, 2018. URL http://arXiv.org/abs/1605.09349.

[173] H. Lam and E. Zhou. Quantifying input uncertainty in stochastic optimization. In *Proceedings of the 2015 Winter Simulation Conference*. IEEE, 2015.

[174] J. Lee and M. Raginsky. Minimax statistical learning and domain adaptation with Wasserstein distances. *arXiv:1705.07815 [cs.LG]*, 2017.

[175] R. Lee, O. J. Mengshoel, A. Saksena, R. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer. Adaptive stress testing: Finding failure events with reinforcement learning. *arXiv preprint arXiv:1811.02188*, 2018.

[176] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*, volume 14. Cambridge University Press, 2004.

[177] K. Leung, N. Arechiga, and M. Pavone. Back-propagation through stl specifications: Infusing logical structure into gradient-based methods.

[178] G. Lewis-Kraus. The great ai awakening. *The New York Times Magazine*, 14:12, 2016.

[179] A. Liniger and J. Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 2019.

[180] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. Kochenderfer. Algorithms for verifying deep neural networks (2019). *arXiv preprint arxiv:1903.06758*.

[181] S. Lohr. The age of big data. *New York Times*, 11(2012), 2012.

[182] L. Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86(3):443–461, 1999.

[183] L. Lovász and S. Vempala. Hit-and-run is fast and fun. *preprint, Microsoft Research*, 2003.

[184] L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

[185] D. Luenberger. *Optimization by Vector Space Methods*. Wiley, 1969.

[186] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[187] J. Lygeros. Lecture notes on hybrid systems. In *Notes for an ENSIETA workshop*, 2004.

[188] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083 [stat.ML]*, 2017.

[189] A. Majumdar and R. Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic foundations of robotics X*, pages 543–558. Springer, 2013.

[190] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3932–3939. IEEE, 2017.

[191] O. Mangoubi and A. Smith. Rapid mixing of hamiltonian monte carlo on strongly log-concave distributions. *arXiv preprint arXiv:1708.07114*, 2017.

[192] E. Marinari and G. Parisi. Simulated tempering: a new monte carlo scheme. *EPL (Europhysics Letters)*, 19(6):451, 1992.

[193] A. W. Marshall. The use of multi-stage sampling schemes in monte carlo computations. Technical report, RAND CORP SANTA MONICA CALIF, 1954.

[194] J. Matyas. Random optimization. *Automation and Remote control*, 26(2):246–253, 1965.

[195] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[196] M. McNaughton. Parallel algorithms for real-time motion planning. 2011.

[197] X.-L. Meng and S. Schilling. Warp bridge sampling. *Journal of Computational and Graphical Statistics*, 11(3):552–586, 2002.

[198] X.-L. Meng and W. H. Wong. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, pages 831–860, 1996.

[199] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. *arXiv:1507.00677 [stat.ML]*, 2015.

[200] 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):529–533, 2015.

[201] A. W. Moore. Efficient memory-based learning for robot control. Technical report, University of Cambridge, Computer Laboratory, 1990.

[202] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[203] J.-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[204] B. Nachman and D. Shih. Anomaly detection with density estimation. *Physical Review D*, 101(7):075042, 2020.

[205] B. Nagy and A. Kelly. Trajectory generation for car-like robots using cubic curvature polynomials. *Field and Service Robots*, 11, 2001.

[206] S. Nakamoto and A. Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 2008.

[207] H. Namkoong and J. C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in neural information processing systems*, pages 2208–2216, 2016.

[208] H. Namkoong and J. C. Duchi. Variance regularization with convex objectives. In *Advances in Neural Information Processing Systems 30*, 2017.

[209] R. M. Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

[210] R. M. Neal. Estimating ratios of normalizing constants using linked importance sampling. *arXiv preprint math/0511216*, 2005.

[211] R. M. Neal. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

[212] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5949–5958, 2017.

[213] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[214] A. Nilim and L. El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[215] J. Norden, M. O'Kelly, and A. Sinha. Efficient black-box assessment of autonomous vehicle safety. *arXiv preprint arXiv:1912.03618*, 2019.

[216] M.-S. Oh and J. O. Berger. Adaptive importance sampling in monte carlo integration. *Journal of Statistical Computation and Simulation*, 41(3-4):143–168, 1992.

[217] M. O'Kelly, H. Abbas, S. Gao, S. Shiraishi, S. Kato, and R. Mangharam. Apex: Autonomous vehicle plan verification and execution. volume 1, Apr 2016.

[218] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in Neural Information Processing Systems*, pages 9827–9838, 2018.

[219] M. O'Kelly, H. Zheng, J. Auckley, A. Jain, K. Luong, and R. Mangharam. Technical Report: TunerCar: A Superoptimization Toolchain for Autonomous Racing. Technical Report UPenn-ESE-09-15, University of Pennsylvania, September 2019. https://repository.upenn.edu/mlab_papers/122/.

[220] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, pages 3918–3926, 2018.

[221] 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.

[222] Y. V. Pant, H. Abbas, and R. Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1235–1240. IEEE, 2017.

[223] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[224] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

[225] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv:1602.02697 [cs.CR]*, 2016.

[226] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[227] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[228] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

[229] B. Peters. The age of big data. *URL: https://www. forbes. com/sites/bradpeters/2012/07/12/the-age-of-big-data*, 2012.

[230] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017.

[231] X. Qin, N. Aréchiga, A. Best, and J. Deshmukh. Automatic testing and falsification with dynamically constrained reinforcement learning. *arXiv preprint arXiv:1910.13645*, 2019.

[232] C. Quiter and M. Ernst. Deepdrive. https://github.com/deepdrive/deepdrive, 2018.

[233] A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. *arXiv:1801.09344 [cs.LG]*, 2018. URL https://arxiv.org/abs/1801.09344.

[234] H. Rahimian and S. Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

[235] N. Ratliff, J. A. Bagnell, and M. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[236] C. Ré. Software 2.0 and snorkel: beyond hand-labeled data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2876–2876, 2018.

[237] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1530–1538. JMLR. org, 2015.

[238] J. Ridderhof and P. Tsiotras. Minimum-fuel powered descent in the presence of random disturbances. In *AIAA Scitech 2019 Forum*, page 0646, 2019.

[239] G. O. Roberts and O. Stramer. Langevin diffusions and metropolis-hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357, 2002.

[240] R. T. Rockafellar and R. J. B. Wets. *Variational Analysis*. Springer, New York, 1998.

[241] N. Roohi, R. Kaur, J. Weimer, O. Sokolsky, and I. Lee. Self-driving vehicle verification towards a benchmark. *arXiv preprint arXiv:1806.08810*, 2018.

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

[243] 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, 2011.

[244] P. J. Rossky, J. Doll, and H. Friedman. Brownian dynamics as smart monte carlo simulation. *The Journal of Chemical Physics*, 69(10):4628–4633, 1978.

[245] A. Rozsa, M. Gunther, and T. E. Boult. Towards robust deep neural networks with bang. *arXiv:1612.00138 [cs.CV]*, 2016.

[246] R. Y. Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In *Stochastic optimization: algorithms and applications*, pages 303–363. Springer, 2001.

[247] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: A unified approach to Monte Carlo simulation, randomized optimization and machine learning*. Information Science & Statistics, Springer Verlag, NY, 2004.

[248] R. Y. Rubinstein and R. Marcus. Efficiency of multivariate control variates in monte carlo simulation. *Operations Research*, 33(3):661–677, 1985.

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

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

[251] A. Sadat, M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. *arXiv preprint arXiv:1910.04586*, 2019.

[252] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, volume 2. Ann Arbor, MI, USA, 2016.

[253] P. Samson. Concentration of measure inequalities for Markov chains and ϕ-mixing processes. *Annals of Probability*, 28(1):416–461, 2000.

[254] A. Satariano and N. Kumar. The massive hedge fund betting on ai. *Bloomberg Markets, September*, 27:2017, 2017.

[255] R. Schram, A. Williams, and M. van Ratingen. Implementation of autonomous emergency braking (aeb), the next step in euro ncap's safety assessment. *ESV, Seoul*, 2013.

[256] S. A. Seshia, D. Sadigh, and S. S. Sastry. Formal methods for semi-autonomous driving. In *Proceedings of the 52nd Annual Design Automation Conference*, page 148. ACM, 2015.

[257] S. Shafieezadeh-Abadeh, P. M. Esfahani, and D. Kuhn. Distributionally robust logistic regression. In *Advances in Neural Information Processing Systems*, pages 1576–1584, 2015.

[258] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.

[259] B. Shahbaba, S. Lan, W. O. Johnson, and R. M. Neal. Split hamiltonian monte carlo. *Statistics and Computing*, 24(3):339–349, 2014.

[260] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[261] S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[262] D. Siegmund. Importance sampling in the monte carlo study of sequential tests. *The Annals of Statistics*, pages 673–684, 1976.

[263] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[264] A. Sinha and J. C. Duchi. Learning kernels with random features. In *Advances in Neural Information Processing Systems 29*, 2016.

[265] A. Sinha, H. Namkoong, and J. Duchi. Certifying some distributional robustness with principled adversarial training. In *Proceedings of the Sixth International Conference on Learning Representations*, 2018. arXiv:1710.10571 [cs.LG].

[266] A. Sinha, M. O'Kelly, J. Duchi, and R. Tedrake. Neural bridge sampling for evaluating safety-critical autonomous systems. *arXiv preprint arXiv:2008.10581*, 2020.

[267] A. Sinha, M. O'Kelly, H. Zheng, R. Mangharam, J. Duchi, and R. Tedrake. Formulazero: Distributionally robust online adaptation via offline population synthesis. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. arXiv:2003.03900 [cs.LG].

[268] E. Smirnova, E. Dohmatob, and J. Mary. Distributionally robust reinforcement learning. *arXiv preprint arXiv:1902.08708*, 2019.

[269] R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

[270] J. M. Snider et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[271] S. Sontges, M. Koschi, and M. Althoff. Worst-case analysis of the time-to-react using reachable sets. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1891–1897. IEEE, 2018.

[272] R. Sparrow. Killer robots. *Journal of applied philosophy*, 24(1):62–77, 2007.

[273] R. Sparrow and M. Howard. When human beings are like drunk robots: Driverless vehicles, ethics, and the future of transport. *Transportation Research Part C: Emerging Technologies*, 80:206–215, 2017.

[274] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[275] R. H. Swendsen and J.-S. Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

[276] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv:1312.6199 [cs.CV]*, 2013.

[277] C. Szepesvári and M. L. Littman. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural computation*, 11(8):2017–2060, 1999.

[278] A. Tamar, S. Mannor, and H. Xu. Scaling up robust mdps using function approximation. In *International Conference on Machine Learning*, pages 181–189, 2014.

[279] Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. *arXiv preprint arXiv:2003.08165*, 2020.

[280] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.

[281] V. Tjeng and R. Tedrake. Verifying neural networks with mixed integer programming. *arXiv:1711.07356 [cs.LG]*, 2017.

[282] W. F. Tosney and P. G. Cheng. Space safety is no accident how the aerospace corporation promotes space safety. In *Space Safety is No Accident*, pages 101–108. Springer, 2015.

[283] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv:1705.07204 [stat.ML]*, 2017.

[284] C. E. Tuncali, T. P. Pavlic, and G. Fainekos. Utilizing s-taliro as an automatic test generation framework for autonomous vehicles. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 1470–1475. IEEE, 2016.

[285] T. Uchiya, A. Nakamura, and M. Kudo. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory*, pages 375–389. Springer, 2010.

[286] US Department of Transportation – FHWA. Ngsim – next generation simulation, 2008.

[287] J. Van Den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[288] A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer, New York, 1996.

[289] B. Vedder. Vedder electronic speed controller. URL https://vesc-project.com/documentation.

[290] C. Villani. *Optimal Transport: Old and New*. Springer, 2009.

[291] G. Vinnicombe. Frequency domain uncertainty and the graph topology. *IEEE Transactions on Automatic Control*, 38(9):1371–1383, 1993.

[292] K. Vogel. A comparison of headway and time to collision as safety indicators. *Accident analysis & prevention*, 35(3):427–433, 2003.

[293] A. F. Voter. A monte carlo method for determining free-energy differences and transition state theory rate constants. *The Journal of chemical physics*, 82(4):1890–1899, 1985.

[294] C. Walsh and S. Karaman. Cddt: Fast approximate 2d ray casting for accelerated localization. abs/1705.01167, 2017. URL http://arxiv.org/abs/1705.01167.

[295] D. Wang, C. Devin, Q.-Z. Cai, P. Krähenbühl, and T. Darrell. Monocular plan view networks for autonomous driving. *arXiv preprint arXiv:1905.06937*, 2019.

[296] Z. Wang, R. Spica, and M. Schwager. Game theoretic motion planning for multi-robot racing. In *Distributed Autonomous Robotic Systems*, pages 225–238. Springer, 2019.

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

[298] S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar. A statistical approach to assessing neural network robustness. 2018.

[299] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. J. Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

[300] G. Williams, B. Goldfain, P. Drews, J. M. Rehg, and E. A. Theodorou. Autonomous racing with autorally vehicles and differential games. *arXiv preprint arXiv:1707.04540*, 2017.

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

[302] J. M. Wing. Trustworthy ai. *arXiv preprint arXiv:2002.06276*, 2020.

[303] C. Xie, M. Tan, B. Gong, A. Yuille, and Q. V. Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020.

[304] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009.

[305] H. Xu, C. Caramanis, and S. Mannor. A distributional interpretation of robust optimization. *Mathematics of Operations Research*, 37(1):95–110, 2012.

[306] S. Yaghoubi and G. Fainekos. Falsification of temporal logic requirements using gradient based local search in space and time. *IFAC-PapersOnLine*, 51(16):103–108, 2018.

[307] Z. B. Zabinsky. *Stochastic adaptive search for global optimization*, volume 72. Springer Science & Business Media, 2013.

[308] D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. LeBlanc. Accelerated evaluation of automated vehicles in car-following maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):733–744, 2018.

[309] D. P. Zhou and C. J. Tomlin. Budget-constrained multi-armed bandits with multiple plays. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[310] E. Zhou and J. Hu. Gradient-based adaptive stochastic search for non-differentiable optimization. *IEEE Transactions on Automatic Control*, 59(7):1818–1832, 2014.

[311] K. Zhou, J. C. Doyle, and K. Glover. Robust and optimal control. 1996.

[312] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*, pages 1–10, 2014.