Title

Formal Error Bounds in Neural Networks, LiDAR Localization, and Generative Models in Closed-Loop Learning

Author

Marchi, Matteo

Publication Date

2025

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Formal Error Bounds in Neural Networks, LiDAR Localization, and Generative Models in Closed-Loop Learning

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical and Computer Engineering

by

Matteo Marchi

2025

ABSTRACT OF THE DISSERTATION

Formal Error Bounds in Neural Networks, LiDAR Localization,
and Generative Models in Closed-Loop Learning

by

Matteo Marchi
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2025
Professor Paulo Tabuada, Co-Chair
Professor Ankur M Mehta, Co-Chair

In the modern world, we are witnessing a rapid acceleration in the adoption of complex and poorly understood systems, such as neural networks, and their use in processing high-dimensional sensor data like cameras and LiDAR. While it is hard to deny the effectiveness and impact of these systems, their theoretical understanding remains elusive. Unfortunately, this makes their use in closed-loop control systems hazardous, as many probabilistic characterizations of their errors fail to hold in this regime, and the tools used to prove properties such as safety and stability (e.g., Input-to-State-Stability theory) are not applicable. Furthermore, while substantial work exists on applying machine learning to control problems—mostly empirical in nature—far less has explored the application of control-theoretic tools to the analysis of machine learning systems.

The primary aim of this work is to bridge these gaps by providing rigorous worst-case error, safety, and stability guarantees for control systems with neural network or black-box components in the loop, as well as improving the theoretical analysis of these systems through control-theoretical tools. Specifically, this work focuses on: 1. Developing a neural network architecture

that enjoys deterministic error bounds. 2. Deriving worst-case pose estimate error guarantees for LiDAR localization. 3. Performing a formal analysis of the long-term dynamics of generative models that are trained on their own synthetic data.

The first part of this dissertation builds on prior work showing that arbitrary depth residual networks (as opposed to the classical result involving arbitrary width) enjoy universal approximation capabilities in the uniform norm sense. However, these results lack a training procedure. We address this gap by developing an architecture and training algorithm exploiting monotonicity that result in residual neural networks with deterministic error bounds by construction. Additionally, such deterministic error bounds enable formal safety and stability guarantees to be proven when using these networks in control loops. Thus, we develop a framework based on Input-to-State-Stability (ISS) that exploits these deterministic errors when using neural networks as state observers, or feedback controllers.

The second part considers the problem of LiDAR-based localization. In particular, it looks at the so-called point cloud registration problem, which is a core routine of most localization and Simultaneous Localization And Mapping (SLAM) algorithms. The literature is rich with a wide variety of algorithms that attempt to solve the problem, from the classical Iterative Closest Point (ICP) to feature and learning-based approaches. However, the existing methods lack in near unanimity the ability to provide bounds on the pose estimation error they provide. In Part II we present a simple and fast point cloud registration algorithm called PASTA (Provably Accurate Simple Transformation Alignment), provide an extensive formal analysis of its worst-case estimation error, and experimentally verify its effectiveness. Such an algorithm, due its formal error bounds, and fast execution time, can be used as a supervisor for other localization algorithms that would otherwise not enjoy worst-case error bounds.

Finally, in the third part, we consider a topic very relevant to recent developments: generative models. As these models rapidly spread in popularity and use, the synthetic data they generate enters the internet, and, in turn, becomes part of the datasets used to train the next generation of generative models. There are rising concerns (supported by empirical observations) about the long-

term consequences of this process, with fears that it may lead to the internet and these models to "degenerate" over time. In this work we analyze the learning dynamics of generative models that are fed back their own produced content in addition to their original training dataset, with particular focus on the effect of "temperature", a parameter typically used to modulate the sampling of these models. Using tools from control theory, we show that, unless a sufficient amount of external data is introduced at each iteration, any non-trivial temperature leads the model to asymptotically degenerate. In fact, either the generative distribution collapses to a small set of outputs, or becomes uniform over a large set of outputs.

The dissertation of Matteo Marchi is approved.

Suhas N Diggavi

Brett Thomas Lopez

Bahman Gharesifard

Ankur M Mehta, Committee Co-Chair

Paulo Tabuada, Committee Co-Chair

University of California, Los Angeles

2025

To Eden, my friends, and all who have shaped the person I am today.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

It is that section where you have a heartfelt acknowledgment of all the wonderful people in your life who have helped you along the way to where you are now. As someone who feels chronically awkward displaying emotions, it is hard for me to write, but it would be harder still to keep it all to myself and not share my appreciation for the great people who have intersected my life. I hope I am able to convey the gratitude I have for meeting all of you.

Of all people, my advisor, Prof. Paulo Tabuada, could not possibly be left unrecognized in a dissertation. Thank you for giving me this opportunity and for jump-starting this incredible journey; my life would be completely different had you not been here. Thank you for all the support and guidance you have provided over the years and for being understanding when things did not go as expected. Your passion and inquisitiveness for research are incredible, and I truly believe you deeply care for all your students.

Of course, I have to acknowledge everyone on my committee. Prof. Gharesifard, I am sure you hear this a lot, but you really are a wonderfully enjoyable and fun presence to be around. I loved all the discussions we had (together with everyone else in the lab), both research- and non-research-related. To Prof. Diggavi, it has been a pleasure collaborating with you, and to Prof. Lopez and Prof. Mehta, thank you for being on my committee and patiently listening to my ramblings during my quals exam and defense.

To everyone in the lab, I want to give you the most heartfelt thanks I can. I was shy and awkward when I joined, but you all have been impossibly warm and welcomed me without hesitation! To Alimzhan Sultangazin and Tzanis Anevlavis, it was always fun to be in the lab with you guys around! To Luigi Pannocchi, I am so glad I had another Italian to share my perplexities with! To Marcus Lucas, your infectious calm is always welcome. To Lucas Fraile, you have pulled triple duty—being a great lab mate, friend, and roommate! To Yskandar Gas, you have the best sense of humor and always made it a pleasure to be there. To Jonathan Bunton, you have been the most incredible friend, and I really miss all our rambling sessions in the lab! To all the new students—Rahal Nanayakkara, João Pedro Silvestre, Alvaro Rodriguez Abella, Mitalee Oza, and

Valen Yamamoto—it has been a pleasure to meet you (even if briefly for some of you) and I really hope you enjoy the journey ahead.

To my family in Italy, I need to thank you for supporting me and helping me become the person I am today. I wouldn't have had the courage to undertake such a great step if you hadn't been there.

Finally, to Eden Haney, my deepest thank you for existing and for choosing to be on this crazy ride with me. I could never have imagined that on this journey, I would meet someone like you, and even less that I would be married to you! You have been there during my deepest lows and my highest highs, and you unlocked a part of me that I was not sure even existed. I can safely say that you taught me to actually live life, and I am so grateful to have you in mine!

I can only conclude with something Yska always used to say (and he's so right): "What a life!"

VITA

2014 B.S. Control and Automation Engineering,
Politecnico di Milano, Milan, Italy.

2017 M.S. Control and Automation Engineering,
Politecnico di Milano, Milan, Italy.

2019-2025 PhD Candidate,
Department of Electrical and Computer Engineering,
University of California, Los Angeles, CA, USA.

PUBLICATIONS

M. Marchi, S. Soatto, P. Chaudhari and P. Tabuada, "Heat Death of Generative Models in Closed-Loop Learning", *63rd IEEE Conference on Decision and Control*, 2024.

M. Marchi, J. Bunton, J. Pedro Silvestre and P. Tabuada, "A Framework for Time-Varying Optimization via Derivative Estimation", *22nd European Control Conference*, 2024.

M. Marchi, J. Bunton, Y. Gas, B. Gharesifard and P. Tabuada, "Sharp Performance Bounds for PASTA", *IEEE Control Systems Letters*, 2023.

M. Marchi, J. Bunton, B. Gharesifard and P. Tabuada, "LiDAR Point Cloud Registration with Formal Guarantees", *61st IEEE Conference on Decision and Control*, 2022.

M. Marchi, J. Bunton, B. Gharesifard and P. Tabuada, "Safety and Stability Guarantees for Control Loops with Deep Learning Perception", *IEEE Control Systems Letters*, 2021.

M. Marchi, B. Gharesifard and P. Tabuada, "Training Deep Residual Networks for Uniform Approximation Guarantees", *3rd Conference on Learning for Dynamics and Control,* 2021.

Part I

Deterministic Error Bounds for Residual Neural Networks

It has recently been shown that deep residual networks with sufficiently high depth, but bounded width, are capable of universal approximation in the supremum norm sense. Based on these results, we show how to modify existing training algorithms for deep residual networks to provide approximation bounds for the test error, in the supremum norm, based on the training error. Our methods are based on control-theoretic interpretations of these networks both in discrete and continuous time, and establish that it is enough to suitably constrain the set of parameters being learned in a way that is compatible with most currently used training algorithms.

These results have immediate control applications, as deep learning is currently used in the perception pipeline of autonomous systems, such as when estimating the system state from camera and LiDAR measurements. While this practice is typical, hard guarantees on the worst-case behavior of the closed-loop system are rare. In this part, we further leverage our results on neural network approximation, combined with classical input-to-state stability (ISS) properties, and show how to design deep neural networks for state estimation that guarantee the safety and stability of the resulting closed-loop system.

CHAPTER 1

Introduction

Deep learning [LBH15] has profoundly changed the way in which many engineering problems are solved, with computer vision being a particularly striking example. Deep neural networks can now perform complex tasks like gesture recognition [OK17], obstacle detection for self driving cars [RGP17], and many more [VDD18]. Although similar benefits may be expected by integrating machine learning with control, we must contend with the safety critical nature of many control applications. Unfortunately, the statistical guarantees typically provided in machine learning, such as probably approximately correct learning bounds, cannot be directly used to establish formal guarantees of safety and performance of control loops. Despite these difficulties, several research efforts are underway to tackle this problem. For example, there are several recent papers studying control related tasks that employ data-driven controllers or perception maps [DMR20, COM19, ALS19], and others that further study classical problems such as the Linear Quadratic Regulator [DMM19] or the Kalman Filter [TMP20] in settings where the underlying parameters are to be learned as new information arrives, while still guaranteeing some level of performance. On the other hand, control-theoretic techniques, particularly ideas from robust and optimal control theory, have been useful in developing training algorithms for neural networks [SFP20, LCT17].

In this work, we offer one additional contribution towards this effort. It was recently established by [TG20], using control-theoretic techniques, that deep residual networks have the power of universal approximation with respect to the infinity norm. In other words, given a continuous function $f : E \to \mathbb{R}^n$ to be learned, defined on a compact set $E \subset \mathbb{R}^n$, and given a desired accuracy $\varepsilon \in \mathbb{R}^+$, there exists a deep residual network implementing the function ϕ satisfying $\sup_{x \in E} \|f(x) - \phi(x)\|_\infty \leq \varepsilon$. Such a result has obvious relevance in the context of a control loop

since one can use well-established nonlinear control analysis techniques to study the effect of using ϕ instead of f by using the upper bound ε on the mismatch between f and ϕ. Unfortunately, the results of [TG20] are not constructive and, in particular, they do not provide training procedures for deep residual networks that guarantee such bounds. Moreover, the results are established for the continuous limit of deep residual networks given by continuous-time control systems. These shortcomings are addressed in this work. We show that most training algorithms, and gradient descent in particular, can be modified to offer similar approximation guarantees without the need to take the continuous limit.

At the technical level, we make the following two contributions: 1) we show how to modify training algorithms so that an upper bound on the infinity norm of the test error can be computed from an upper bound on the training error; 2) we show the training error can be made as small as desired by increasing depth (although we do not guarantee that any particular training algorithm can achieve it, as the training of deep networks is known to be a non-convex problem).

Interestingly, the guarantees provided in this work are *deterministic* which contrasts with the mainstream approach in machine learning that typically only provides *probabilistic* guarantees [SB14, AB09]. Underlying this difference are the assumptions made on the training data. Whereas classical learning theory assumes the training data to be generated according to some distribution, we make no assumptions on how it is generated. However, our bounds are based on how well the data covers the domain of the function to be learned. This is similar to robustness bounds in the Input to State Stability framework: nothing[1] is assumed about disturbances and the bound on the state depends on the concrete disturbance being applied. In our case, nothing is assumed about the training data and the bound depends on the actual data that was used for training.

[1]Except for being essentially bounded.

CHAPTER 2

Uniform Norm Universal Approximation

In this section we review the results of [TG20] upon which the results of this work are based.

2.1 Residual networks as control systems

One of the key insights exploited by [TG20] is that residual neural networks can be thought of as the forward Euler discretization of continuous-time control systems. This observation, first made by [Wei17, HR17, LZL18], allows us to use control theoretic techniques to analyze deep residual networks.

Let us consider a deep residual network modeled by the discrete-time control system:

$$z(k+1) = z(k) + s(k)\Sigma(W(k)z(k) + b(k))\,, \tag{2.1}$$

where $k \in \mathbb{N}_0$ indexes the layers of the network, $z(k) \in \mathbb{R}^n$ models the contents of the n neurons in layer k, $(s(k), W(k), b(k)) \in \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ are the weights (interpreted as inputs), and $\Sigma(z) = (\sigma(z_1), \ldots, \sigma(z_n))$ is defined by the scalar activation function $\sigma : \mathbb{R} \to \mathbb{R}$. In particular, we consider deep residual networks that have fixed width equal to n. To help the reader distinguish between discrete-time models and continuous-time models, we reserve z for the state of discrete-time models and x for the state of continuous-time models. Before introducing the continuous analogue of (2.1), we recall the notion of flow. The flow $Z^k : \mathbb{R}^n \to \mathbb{R}^n$ of (2.1) under the input $(s, W, b) : \{0, 1, \ldots, \ell\} \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ is the map Z^k taking the state $z \in \mathbb{R}^n$ to the state $Z^k(z)$ reached from z at time $k \in \{0, 1, \ldots, \ell + 1\}$ by the solution of (2.1) under the input (s, W, b).

Discrete-time control systems of the form (2.1) can be viewed as the time discretization of the

continuous-time control system:

$$\dot{x}(t) = s(t)\Sigma(W(t)x(t) + b(t))\,, \tag{2.2}$$

where $x(t)$ and $(s(t), W(t), b(t))$ take values in the same sets as in (2.1), except they are indexed by $t \in \mathbb{R}_0^+$ modeling continuous time. The weights are now functions defined on $[0, \tau]$ that we interpret as open-loop inputs parameterized by time $t \in [0, \tau]$. Similarly to the discrete-time case, we define the flow $X^t : \mathbb{R}^n \to \mathbb{R}^n$ induced by (2.2) and by a choice of inputs $(s, W, b) : [0, \tau] \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ to be the function mapping the state $x \in \mathbb{R}^n$ to the state $X^t(x)$ reached at time $t \in [0, \tau]$ from x by the solution of (2.2) under the input (s, W, b).

2.2 Controllability

The interpretation of deep residual networks as continuous-time control systems allows the problem of training a network to be recast as the problem of designing an open-loop control input.

Let us assume that we seek to train a network so as to learn a continuous function $f : E \to \mathbb{R}^n$ where $E \subset \mathbb{R}^n$ is a compact set. We are given a collection of samples of this function, i.e., a collection of d pairs $(x^i, f(x^i))$, with $i \in \{1, \dots, d\}$, and our objective is to choose a time $\tau \in \mathbb{R}^+$ and a control input $(s, W, b) : [0, \tau] \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ so that the resulting flow X^t satisfies $X^\tau(x^i) = f(x^i)$ for all $i \in \{1, \dots, d\}$. We emphasize that, independently of the number of samples d, we seek a *single* control input. In other words, we seek a single input to concurrently control d copies of the control system (2.2), each initialized at one of the points x^i in the sample set. To make this idea formal, we introduce the ensemble control system:

$$\dot{X}(t) = \Big[s(t)\Sigma(W(t)X_{\bullet 1}(t) + b(t))|s(t)\Sigma(W(t)X_{\bullet 2}(t) + b(t))|\dots|s(t)\Sigma(W(t)X_{\bullet d}(t) + b(t))\Big]\,, \tag{2.3}$$

where $X_{\bullet i}$ represents the i-th column of $X(t) \in \mathbb{R}^{n\times d}$. If we define $X^{\text{init}} = \Big[x^1|x^2|\dots|x^d\Big]$ and $X^{\text{fin}} = \Big[f(x^1)|f(x^2)|\dots|f(x^d)\Big]$, we can express the network training problem as the design of a time $\tau \in \mathbb{R}^+$ and an input $(s, W, b) : [0, \tau] \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ so that the flow X^t of (2.3) satisfies $X^\tau(X^{\text{init}}) = X^{\text{fin}}$. Note that the ensemble control system (2.3), is formed by d exact

copies of (2.2) whereas the literature on ensemble control, e.g., [LK06, HS14, Bro07], mostly deals with ensembles of different control systems, with the exception of [AS20] where a setting similar to the one here is considered.

To summarize, the ability to train a network relies on the controllability of the ensemble control system (2.3). Let us recall the notion of controllability.

Definition 2.1. *Control system* (2.3) *is said to be controllable on a submanifold* M *of* $\mathbb{R}^{n\times d}$ *if, given any points* $X^{\text{init}}, X^{\text{fin}} \in M$ *there exist* $\tau \in \mathbb{R}^+$ *and a control input* $(s, W, b) : [0, \tau] \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ *so that the flow* X^t *of* (2.3) *under said input satisfies* $X^\tau(X^{\text{init}}) = X^{\text{fin}}$.

In order to state one of the key results of [TG20], showing that controllability holds on an open, dense, and connected subset of $\mathbb{R}^{n\times d}$, we introduce a mild assumption on the activation function σ stated in terms of its derivative denoted by $D\sigma$:

Assumption 2.1. *We assume that* $D\sigma \geq 0$, $\overline{D\sigma} := \sup_{x\in\mathbb{R}} D\sigma < \infty$, *and that there exists* $k \in \mathbb{N}_0$ *such that* $\xi = D^k\sigma$ *is injective and satisfies a quadratic differential equation* $D\xi = a_0 + a_1\xi + a_2\xi^2$ *with* $a_2 \neq 0$.

Assumption 2.1 is quite mild. It is satisfied, e.g., by the logistic function, hyperbolic tangent, tangent, and soft plus. Moreover, it also holds for the ReLU by regarding this function as the limit of the soft plus function, see [TG20] for more details.

Theorem 2.1 ([TG20])**.** *Let* $N \subset \mathbb{R}^{n\times d}$ *be the set defined by:*

$$N = \left\{ F \in \mathbb{R}^{n\times d} \mid \prod_{1\leq i<j\leq d} (F_{\ell i} - F_{\ell j}) = 0,\ \ell \in \{1, \dots, n\} \right\}. \tag{2.4}$$

Suppose that Assumption 2.1 holds. If $n > 1$*, then the ensemble control system* (2.3) *is controllable on the submanifold* $M = \mathbb{R}^{n\times d}\backslash N$.

Theorem 2.1 shows that given any finite set of samples defining X^{init} and X^{fin}, as described above, only two situations can occur: 1) either $X^{\text{init}}, X^{\text{fin}} \in M$ and $X^\tau(X^{\text{init}}) = X^{\text{fin}}$ or; 2) $X^{\text{init}} \notin M$ or $X^{\text{fin}} \notin M$ and $\|X^\tau(X^{\text{init}}) - X^{\text{fin}}\| \leq \varepsilon$ for any chosen norm $\|\cdot\|$ and accuracy

$\varepsilon \in \mathbb{R}^+$. The latter case holds in virtue of M being an open and dense subset of $\mathbb{R}^{n\times d}$. In other words, deep residual networks can memorize exactly almost any finite set of samples. Moreover, those finite sets that cannot be exactly memorized can be approximated to an arbitrary accuracy.

2.3 Uniform approximation

The second main result of [TG20] extends Theorem 2.1 from finite sets to the whole domain of the function to be learned. This is done by using a deep residual network with $n + 1$ neurons to learn a function $f : E \to \mathbb{R}^n$, with $E \subset \mathbb{R}^n$ a compact set. The additional neuron allows non-monotone functions to be approximated while using a monotone flow. The monotonicity property is crucial to the L_∞ norm approximation result, and is the main ingredient behind our results in Chapter 3. To recall this result more precisely, since f and the flow X^t have different domains and co-domains, we introduce a linear injection $\alpha : \mathbb{R}^n \to \mathbb{R}^{n+1}$, a linear projection $\beta : \mathbb{R}^{n+1} \to \mathbb{R}^n$, and measure the error between f and the learned function $\beta \circ X^\tau \circ \alpha$ by:

$$\|f - \beta \circ X^\tau \circ \alpha\|_{L^\infty(E)} := \sup_{x\in E} \|f(x) - \beta \circ X^\tau \circ \alpha(x)\|_\infty.$$

Theorem 2.2. *Let $n > 1$ and suppose that Assumption 2.1 holds. Then, for every continuous function $f : \mathbb{R}^n \to \mathbb{R}^n$, for every compact set $E \subset \mathbb{R}^n$, and for every $\varepsilon \in \mathbb{R}^+$ there exist a time $\tau \in \mathbb{R}^+$, a linear injection $\alpha : \mathbb{R}^n \to \mathbb{R}^{n+1}$, a linear projection $\beta : \mathbb{R}^{n+1} \to \mathbb{R}^n$, and an input $(s, W, b) : [0, \tau] \to \mathbb{R} \times \mathbb{R}^{(n+1)\times(n+1)} \times \mathbb{R}^{n+1}$ so that the flow $X^t : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ of (2.2) with state space $\mathbb{R}^{n+1}$ under the said input satisfies:*

$$\|f - \beta \circ X^\tau \circ \alpha\|_{L^\infty(E)} \leq \varepsilon.$$

By interpreting α and β as linear layers, the first and last, respectively, we conclude that deep residual networks can approximate any continuous function arbitrarily well with respect to the supremum norm. [TG20] select α as the fixed function $\alpha(x) = (x, \mathbf{1}^T x)$ and β as the linear function $\beta(x, y) = x + \kappa \mathbf{1} y$ where $\kappa \in \mathbb{R}$ has to be appropriately chosen for each f being approximated, and $\mathbf{1} \in \mathbb{R}^n$ is the vector whose entries are all one.

CHAPTER 3

Error Bounds and Training

In this chapter, we show how the approximation error between a function to be learned and the function implemented by a deep residual network can be bounded by the training error. Monotonicity plays a key role in our approach and thus we start by reviewing this concept.

Definition 3.1 (First-orthant partial order on $\mathbb{R}^n$)**.** *The first-orthant partial order $\preceq$ on $\mathbb{R}^n$ is defined by $x \preceq y$ if and only if $x_i \leq y_i$ for all $i \in \{1, \ldots, n\}$, where $\leq$ denotes the usual total order on $\mathbb{R}$.*

Monotonicity of a map can now be introduced by making use of the preceding partial order. Although monotonicity can be defined with respect to other orders, the first-orthant order will simplify the analysis.

Definition 3.2 (Monotone map)**.** *We say that a function $\phi : \mathbb{R}^n \to \mathbb{R}^m$ is monotone if for any $x, y \in \mathbb{R}^n$:*

$$x \preceq y \implies \phi(x) \preceq \phi(y)\,.$$

We say that a flow Z^k is monotone if the map $Z^k : \mathbb{R}^n \to \mathbb{R}^n$ is monotone for each $k \in \mathbb{N}$.

Given a set $E \subset \mathbb{R}^n$, we denote, respectively, by $\sup E$ and $\inf E$ the least upper bound of all the elements in E and the greatest lower bound of all the elements in E with respect to the order $\preceq$. Moreover, given two points $x, z \in \mathbb{R}^n$ with $x \preceq z$, we denote by $[x, z]$ the set defined by:

$$[x, z] = \{y \in \mathbb{R}^n \mid x \preceq y \preceq z\}.$$

Flows of deep residual networks can be made monotone by enforcing certain constraints on the inputs.

Proposition 3.1. *Consider the discrete-time control system* (2.1) *modeling a deep residual network and assume Assumption 2.1 holds. Any flow Z^k induced by* (2.1) *using an input $(s, W, b) : \{0, 1, \dots, \ell\} \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ satisfying:*

$$s(k)w_{ij}(k) \geq 0, \quad 1 + s(k)w_{ii}(k)\overline{D\sigma} \geq 0, \qquad \forall i, j \in \{1, 2, \dots, n\}, i \neq j, \quad k \in \{0, 1, \dots, \ell\}, \tag{3.1}$$

is monotone.

Proof. Let $k \in \mathbb{N}_0$ and consider the map $\phi^k : \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$\phi^k(z) = z + s(k)\Sigma(W(k)z + b(k)) .$$

Note that for all $i, j \in \{1, 2, \dots, n\}$ with $i \neq j$, we have that

$$\frac{\partial \phi_i^k}{\partial x_j}(x) = s(k)D\sigma(W(k)x + b(k))w_{ij}(k) .$$

Since $D\sigma \geq 0$ and $s(k)w_{ij}(k) \geq 0$, by assumption, we have $\frac{\partial \phi_i^k}{\partial x_j} \geq 0$ for all $i \neq j$. Moreover, for $i = j$ we have:

$$\frac{\partial \phi_i^k}{\partial x_i}(x) = 1 + s(k)D\sigma(W(k)x + b(k))w_{ii}(k) .$$

It now follows from the assumption $1 + s(k)w_{ii}(k)\overline{D\sigma} \geq 0$, alongside with the fact that $\overline{D\sigma} \geq 0$, that $\frac{\partial \phi_i^k}{\partial x_j} \geq 0$ holds for $i = j$. Hence, by [HS06, Lemma 5.1], we conclude that ϕ^k is monotone.

Finally, since:

$$Z^k = \phi^{k-1} \circ \phi^{k-2} \circ \cdots \circ \phi^0 ,$$

and the composition of monotone functions is monotone, Z^k is monotone. □

We now prove one of the main results of this work, a deterministic bound for the approximation error based on the training error. Although reminiscent of a generalization bound that holds for **any** set of sample points, the bound in the following result depends on the **specific** set of samples.

The result makes use of a discrete-time control system of the following form, representing a residual neural network, where the state $z(k) \in \mathbb{R}^{n+1}$ is a tuple $z(k) = (z_1(k), z_2(k))$ with

$z_1(k) \in \mathbb{R}^n$ and $z_2(k) \in \mathbb{R}$, whose dynamics are described by:

$$z(k+1) = (z_1(k+1), z_2(k+1)) = (z_1(k) + s(k)\Sigma(W(k)z_1(k) + b(k)),\ z_2(k))\,. \tag{3.2}$$

The dynamics are split into two independent parts. The first is a control system of form (2.1), while the second has no dynamics and just propagates the initial value of z_2 through the layers. We denote by Z_1^k and Z_2^k the flows of the first and second part of (3.2), respectively.

Theorem 3.1. *Consider the discrete-time control system* (3.2)*, modeling a deep residual network, and suppose that Assumption 2.1 holds. Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a continuous function, and let $E \subset \mathbb{R}^n$ be a compact set. For any finite set $E_{\text{samples}} \subset \mathbb{R}^n$ satisfying $E \subseteq [\inf E_{\text{samples}}, \sup E_{\text{samples}}]$, let $\delta \in \mathbb{R}^+$ be the smallest number satisfying:*

$$\forall x \in E,\ \exists \underline{x}, \overline{x} \in E_{\text{samples}}, \quad \|\overline{x} - \underline{x}\|_\infty \leq \delta \text{ and } \underline{x} \preceq x \preceq \overline{x}.$$

For any flow Z^k induced by (3.2) *and by an input $(s, W, b) : \{0, 1, \ldots, \ell\} \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ satisfying the constraints* (3.1)*, we have:*

$$\|f - \beta \circ Z^{\ell+1} \circ \alpha\|_{L^\infty(E)} \leq 3\|f - \beta \circ Z^{\ell+1} \circ \alpha\|_{L^\infty(E_{\text{samples}})} + 2\omega_f(\delta) + 2n|\kappa|\delta, \tag{3.3}$$

where ω_f is the modulus of continuity of f, $\alpha : \mathbb{R}^n \to \mathbb{R}^{n+1}$ is given by $\alpha(x) = (x, \mathbf{1}^T x)$, $\beta : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is given by $\beta(x, y) = x + \kappa \mathbf{1} y$, $\mathbf{1} \in \mathbb{R}^n$ is the vector whose entries are all 1, and $\kappa \in \mathbb{R}$.

Proof. Observing the structure of (3.2) we can write for any $x \in E$:

$$\beta \circ Z^{\ell+1} \circ \alpha(x) = \beta \circ (Z_1^{\ell+1}(x), Z_2^{\ell+1}(\mathbf{1}^T x)) = Z_1^{\ell+1}(x) + \kappa \mathbf{1} Z_2^{\ell+1}(\mathbf{1}^T x) = Z_1^{\ell+1}(x) + \kappa \mathbf{1}\mathbf{1}^T x\,.$$

Then:

$$\left\|f - \beta \circ Z^{\ell+1} \circ \alpha\right\|_{L^\infty(E_{\text{samples}})} = \left\|f - Z_1^{\ell+1} - \kappa \mathbf{1}\mathbf{1}^T\right\|_{L^\infty(E_{\text{samples}})} = \left\|\tilde{f} - Z_1^{\ell+1}\right\|_{L^\infty(E_{\text{samples}})}, \tag{3.4}$$

where $\tilde{f} = f - \kappa \mathbf{1}\mathbf{1}^T$. Then, since $\tilde{f}$ is a continuous function and $Z_1^{\ell+1}$ is monotone, we can apply Lemma A.3 of [TG20] to obtain:

$$\begin{aligned}\left\|f - \beta \circ Z^{\ell+1} \circ \alpha\right\|_{L^\infty(E)} &= \left\|\tilde{f} - Z_1^{\ell+1}\right\|_{L^\infty(E)} \\ &\leq 3\left\|\tilde{f} - Z_1^{\ell+1}\right\|_{L^\infty(E_{\text{samples}})} + 2\omega_{\tilde{f}}(\delta) \\ &\leq 3\left\|f - \beta \circ Z^{\ell+1} \circ \alpha\right\|_{L^\infty(E_{\text{samples}})} + 2\omega_{\tilde{f}}(\delta).\end{aligned}$$

To conclude the proof, it remains to be shown that $\omega_{\tilde{f}}(\delta) \leq \omega_f(\delta) + n|\kappa|\delta$. This can be seen by observing that for any $x, y \in \mathbb{R}^n$, we have:

$$\begin{aligned}\left\|\tilde{f}(x) - \tilde{f}(y)\right\|_\infty &= \left\|f(x) + f(y) - \kappa \mathbf{1}\mathbf{1}^T(x-y)\right\|_\infty \\ &\leq \|f(x) + f(y)\|_\infty + |\kappa|\left\|\mathbf{1}\mathbf{1}^T\right\|_\infty \|x - y\|_\infty \\ &\leq \omega_f(\|x-y\|_\infty) + n|\kappa|\,\|x-y\|_\infty\,,\end{aligned}$$

yielding the claim. □

Note that the bound (3.3) can be used as a stopping criterion for the training. Since $\|f - \beta \circ Z^{\ell+1} \circ \alpha\|_{L^\infty(E_{\text{samples}})}$ is known, some knowledge of an upper bound for ω_f will directly give us an upper bound for the approximation error. Hence, the question arises as to how to train deep residual networks so that the assumptions of Theorem 3.1 hold. This can be done by training a deep residual network with any of the usual iterative optimization techniques (such as gradient descent), as long as the parameters s, W are suitably constrained to satisfy (3.1). We now make this idea precise.

Consider a deep residual network described by (3.2) with $n+1$ neurons and $\ell+1$ layers enhanced with a layer implementing the linear map $\alpha(x) = (x, \mathbf{1}^T x)$ functioning as layer 0, and another layer implementing the linear map $\beta(x, y) = x + \kappa \mathbf{1} y$ functioning as layer $\ell+2$. Note that while layer 0 is fixed, the parameter κ in layer $\ell+2$ will also be learned. If we denote by $\theta(k)$, $k \in \{1, 2, \ldots, \ell+2\}$ the parameters of each layer, we have:

$$\theta(k) = (s(k), W(k), b(k)) \in \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n, \quad k \in \{1, 2, \ldots, \ell+1\}$$

$$\theta(\ell+2) = \kappa \in \mathbb{R}.$$

To ensure that the parameters $\theta(k)$, $k \in \{1, 2, \ldots, \ell+1\}$, satisfy the constraints (3.1), we project them to the closest point in the set defined by (3.1) using the projection $\mathrm{proj} : \mathbb{R}^{n^2+n+1} \to \mathbb{R}^{n^2+n+1}$ defined by the solution of a quadratic optimization problem:

$$\mathrm{proj}(\theta(k)) = \begin{cases} \arg\min_{\theta' \in \mathbb{R}^{n^2+n+1}} & \|\theta' - \theta(k)\|^2 \\ \text{s.t.} & s' w'_{ij} \geq 0, \quad i,j \in \{1,\ldots,n\}, i \neq j \ , \\ & 1 + s' w'_{ii} \overline{D\sigma} \geq 0, \quad i \in \{1,\ldots,n\} \end{cases} \tag{3.5}$$

where $\theta' = (s', W', b')$ is the variable we optimize over (note that b' does not appear in the constraints, so can always be taken as $b' = b(k)$). Since the constraints (3.1) are independent for each layer, the parameters of each layer can be independently projected. Although problem (3.5) is non-convex, many efficient heuristics exist for solving quadratic programs, see [PB17] and references therein. Furthermore, we note that the optimal W' can be computed explicitly for a given fixed s', so that the projection can be recast as a single-variable non-linear optimization problem.

Let now Θ be the full set of parameters of the network $\Theta = (\theta(1), \ldots, \theta(\ell+1), \theta(\ell+2))$. Any iterative training algorithm can be written in the form:

$$\Theta_{i+1} = \psi(\Theta_i),$$

for a suitably defined ψ (that we assume here to encode all the information about the problem, such as the available training data), and modified to:

$$\begin{aligned} \widetilde{\Theta}_{i+1} &= \psi(\Theta_i) \\ \Theta_{i+1} &= (\mathrm{proj}(\tilde{\theta}_{i+1}(1)), \mathrm{proj}(\tilde{\theta}_{i+1}(2)), \ldots, \mathrm{proj}(\tilde{\theta}_{i+1}(\ell+1)), \tilde{\theta}_{i+1}(\ell+2)), \end{aligned}$$

so that the parameters θ at each iteration satisfy the constraints (3.1). Even though we do not plan to dwell on this topic here, it is worth pointing out that the projected gradient descent has convergence properties similar to those of normal gradient descent [ABS13]. In fact, constraining the weights in neural networks is a recurring idea in the literature, see, e.g., [CZ14, DV10].

As a concluding remark, it is interesting to note that penalizing the magnitude of κ during training, (that would typically be done for purposes of regularization), can be justified on the basis of Theorem 3.1, where $|\kappa|$ appears in the final approximation guarantee, indicating a potential tradeoff between training and test errors.

CHAPTER 4

Revisiting Universal Approximation

The bound (3.3) provided in Theorem 3.1 provides information about the approximation error based on the training error. However, we do not know if low training error is achievable when the constraints (3.1) on the inputs are enforced. We will show this to be the case by building upon the results of [TG20].

In analogy with (3.2), we present the following control system, where $x(t) \in \mathbb{R}^{n+1}$, $x_1(t) \in \mathbb{R}^n$, and $x_2(t) \in \mathbb{R}$:

$$\dot{x}(t) = (\dot{x}_1(t), \dot{x}_2(t)) = (s(t)\Sigma(W(t)x_1(t) + b(t)),\, 0)\,. \tag{4.1}$$

Like in the discrete case, the dynamics are split into two independent parts, with the first a control system of form (2.2), while the second has no dynamics. We denote respectively by X_1^t and X_2^t the flows of the first and second part of (4.1).

We first introduce a variant of Corollary 4.5s of [TG20].

Theorem 4.1. *Let $n > 1$ and suppose that Assumption 2.1 holds. Then, for every continuous function $f : \mathbb{R}^n \to \mathbb{R}^n$, for every compact set $E \subset \mathbb{R}^n$, and for every $\varepsilon \in \mathbb{R}^+$ there exist a time $\tau \in \mathbb{R}^+$ and an input $(s, W, b) : [0, \tau] \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ satisfying:*

$$s(t)w_{ij}(t) \geq 0 \qquad \forall i, j \in \{1, 2, \ldots, n\}, i \neq j, t \in [0, \tau], \tag{4.2}$$

so that the flow $X^t : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ of (4.1) *under the said input satisfies:*

$$\|f - \beta \circ X^\tau \circ \alpha\|_{L^\infty(E)} \leq \varepsilon,$$

where $\alpha : \mathbb{R}^n \to \mathbb{R}^{n+1}$ is given by $\alpha(x) = (x, \mathbf{1}^T x)$, and $\beta : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is given by $\beta(x, y) = x + \kappa \mathbf{1} y$ as in Theorem 3.1.

Note that this result differs from Theorem 2.2, as it places additional constraints (4.2) on the inputs, and assumes that the flow X^τ obeys (4.1). However, the proof of Theorem 2.2 only requires minor modifications to account for (4.1) and (4.2).

Proof. We first consider the flow of the first component X_1^τ only. Proposition A.4 and Theorem 4.4 of [TG20] will still hold for X_1^τ, provided that there exist choices of inputs (s, W, b) satisfying the constraints (4.2) that induce the same set of vector fields used in their proofs. The vector fields used in Proposition A.4 are:

$$\begin{cases} X_i^+ = \sigma(c)\frac{\partial}{\partial x_i}, & X_i^- = -X_i^+ \\ Y_i^+ = \sigma(x_i)\frac{\partial}{\partial x_i}, & Y_i^- = -\sigma(x_i)\frac{\partial}{\partial x_i} \end{cases}, \quad i \in \{1, 2, \dots, 2n\},$$

for some c such that $\sigma(c) \neq 0$. Respectively, these can be realized in our case by the choices $(s, W, b) = (\pm 1, 0, ce_i)$ and $(s, W, b) = (\pm 1, E_{ii}, ce_i)$, where e_i is the vector with 1 in its i-th entry and 0 in every other entry, and E_{ij} is the matrix with 1 in its ij-th entry and 0 in all other entries (note that the constraints (4.2) only apply to off-diagonal entries). An additional class of vector fields that is used in Theorem 4.4 of [TG20] is given by

$$Z_{ij} = \sigma(x_j)\frac{\partial}{\partial x_i}, \quad i, j \in \{1, \dots, 2n\}.$$

Note that these vector fields are realizable in our case by the choice $(s, W, b) = (1, E_{ij}, 0)$.

This ensures that Proposition A.4 and Theorem 4.4 hold for X_1^τ. Consequently, Corollary 4.5 of [TG20] will hold for the complete flow X^τ establishing this result. □

We can now establish that the training error can be made as small as desired, provided that the network depth is large enough, by Euler discretization of the continuous-time input from Theorem 4.1.

Theorem 4.2. *Consider the discrete-time control system* (3.2)*, modeling a deep residual network, and suppose that Assumption 2.1 holds. Then, for every continuous function $f : \mathbb{R}^n \to \mathbb{R}^n$ with $n > 1$, for every compact set $E \subset \mathbb{R}^n$, and for every $\varepsilon \in \mathbb{R}^+$ there exist a time $\ell \in \mathbb{N}$ and an input*

$(s, W, b) : \{0, 1, \ldots, \ell\} \to \mathbb{R} \times \mathbb{R}^{n\times n} \times \mathbb{R}^n$ *satisfying* (3.1)*, so that the flow* $Z^k : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ *of* (3.2) *under the said input satisfies:*

$$\|f - \beta \circ Z^{\ell+1} \circ \alpha\|_{L^\infty(E)} \leq \varepsilon, \tag{4.3}$$

where $\alpha : \mathbb{R}^n \to \mathbb{R}^{n+1}$ *is given by* $\alpha(x) = (x, \mathbf{1}^T x)$*, and* $\beta : \mathbb{R}^{n+1} \to \mathbb{R}^n$ *is given by* $\beta(x, y) = x + \kappa \mathbf{1} y$ *as in Theorem 3.1.*

Proof. . According to Theorem 4.1, there exist a $\beta : \mathbb{R}^{n+1} \to \mathbb{R}^n$ and a piecewise constant control input[1] (s, W, b) for (4.1) satisfying (4.2) inducing a flow X^t so that $\|f - h\|_{L^\infty(E)} \leq \frac{\varepsilon}{2}$ with h defined by $h(x) = \beta \circ X^\tau \circ \alpha(x)$. Consider now the function $g(x) = \beta \circ Z^{\ell+1} \circ \alpha(x)$, where Z^k is the flow induced by (3.2) under some input. Then, for any $x \in E$:

$$\begin{aligned} \|f(x) - g(x)\|_\infty &\leq \|f(x) - h(x)\|_\infty + \|h(x) - g(x)\|_\infty \\ &\leq \frac{\varepsilon}{2} + \left\|X_1^\tau(x) + \kappa \mathbf{1}\mathbf{1}^T x - Z_1^{\ell+1}(x) - \kappa \mathbf{1}\mathbf{1}^T x\right\|_\infty \\ &\leq \frac{\varepsilon}{2} + \left\|X_1^\tau(x) - Z_1^{\ell+1}(x)\right\|_\infty . \end{aligned} \tag{4.4}$$

Let (s, W, b) be the continuous-time input for (4.1) that results in the continuous-time flow X^t. We now construct a discrete-time input (s', W', b') for (3.2), so that the resulting flow Z^k satisfies:

$$\left\|X_1^\tau(x) - Z_1^{\ell+1}(x)\right\|_\infty \leq \frac{\varepsilon}{2}. \tag{4.5}$$

By Euler forward integration (see [Atk08]), there exists a sufficiently small $T \in \mathbb{R}^+$, so that the flow of (2.1) under the input $(s'(k), W'(k), b'(k)) = (Ts(kT), W(kT), b(kT))$, $k \in \{1, 2, \ldots, \lfloor \tau/T \rfloor\}$ satisfies (4.5) with $\ell = \lfloor \tau/T \rfloor$, provided the solution of (2.2) has bounded second derivative and the right-hand side of (2.2) is Lipschitz continuous in the state variable. Since the input (s, W, b) is piecewise constant, the solution of (2.2) can be seen as the composition of analytic flows (the right-hand side of (2.2) is analytic for constant inputs in virtue of Assumption 2.1), one per each constant component of the input. Since the solution is defined on the compact $[0, \tau]$, its second derivative is bounded.

[1]The controllability results of [TG20] only rely on being able to switch between a finite set of vector fields which is achieved by piecewise constant inputs.

By combining (4.4) with (4.5) we obtain inequality (4.3), and hence to conclude the proof it suffices to show that the inputs (s', W', b') satisfy (3.1). The first requirement in (3.1):

$$s'(k)w'_{ij}(k) \geq 0,$$

is immediately satisfied, since $\forall t \in [0, \tau]$ we have that $s(t)w_{ij}(t) \geq 0$. In order to show that the second requirement in (3.1) is satisfied, we first let:

$$\bar{s} = \max_{[0,t]} |s(t)|, \quad \bar{w}_{ij} = \max_{[0,t]} |w_{ij}(t)| \,,$$

and select T such that:

$$T \leq \min_{i,j \in \{1,\dots,n\}} (\bar{s}\bar{w}_{ij}\overline{D\sigma})^{-1} \,.$$

Therefore, we have that:

$$1 + Ts(kT)w_{ii}(kT)\overline{D\sigma} = 1 + s'(k)w'_{ii}(k)\overline{D\sigma} \geq 0 \,.$$

□

CHAPTER 5

Neural Networks in Control Loops

Neural network architectures with deterministic error bounds, such as that presented in the previous chapters, have an immediate application in a control setting. This property can be exploited to develop control applications that involve machine learning components in the loop, while retaining provable safety and stability guarantees.

5.1 Motivation

The resurgence of neural networks in the last decade, especially with the advent of deep learning [LBH15], has facilitated progress on notoriously difficult problems such as image classification [LKB17] and natural speech processing [YHP18]. This progress naturally led to widespread use of deep neural networks in the perception pipeline of autonomous systems [Ack17, LHA20]. Given the safety-critical nature of such systems, it is imperative to provide formal safety, stability, and robustness certificates when deep neural networks are used in a closed-loop control system.

Incorporating neural networks in closed-loop control has been extensively studied [Son93], typically with the underlying assumption that the neural network can produce arbitrarily small approximation errors [SMO04, LJY99]. However, this is often unrealistic and unless the approximation errors are appropriately quantified and their impact on safety and stability is studied [ZPZ00], the analysis of the resulting closed-loop systems remains incomplete. We address the current gap in analysis by providing safety and stability guarantees when deep neural networks are used in the perception pipeline with a suitably robust control law.

Existing techniques to establish the stability of closed-loop systems containing neural net-

works [AGS13, YSA20, JL20] are based on estimates of the neural network's Lipschitz constant by, for instance, imposing incremental quadratic type constraints during training [FRH19]. Moreover, most results focus on L_2 type estimates, whereas we target guarantees in the uniform norm, which are necessary to exploit ISS properties and provide safety and stability certificates. Other recent approaches consist of performing output range and reachability analysis of the networks [KL20, DJS18, XTR18] by solving an optimization problem or performing formal verification, or rely on special dynamic network structure [SP99].

In this work, we consider a neural network acting as either a state observer or an output feedback control law for a dynamical system. Our main contribution consists in bridging the gap between the classical notion of input-to-state stability and newly established worst-case guarantees for neural networks, providing concise, hard guarantees on the resulting closed-loop system behavior.

5.2 Problem statement

Consider the nonlinear control system:

$$\dot{x}(t) = f(x(t), u(t), d(t)) \tag{5.1}$$

$$y(t) = h(x(t)), \tag{5.2}$$

where $t \in \mathbb{R}_{\geq 0}$, $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ the control input, $d(t) \in \mathbb{R}^o$ the disturbance, $y(t) \in \mathbb{R}^p$ the output, and $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^o \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^p$ are smooth maps.

Although we focus on using output measurements to provide a reliable state estimate $\widehat{x}$ for the controller, suppose for the moment that a feedback controller $u = k(x)$ has been designed under the assumption that the state x is known. We will then consider two scenarios. In the first, the controller is used with a state estimate provided by a deep neural network. In the second scenario, the deep neural network directly provides the control input and the controller is used in the training of such network. The state estimate of the state $x(t)$ at time t provided by the neural network, in

the first scenario, is computed from a finite sequence of $\ell + 1$ outputs and ℓ inputs. To this end, consider the measurements:

$$S_{\text{meas}}(t, \ell) = (y(t), y(t - \tau), \ldots, y(t - \ell\tau), y(t), u(t - \tau), \ldots, u(t - \ell\tau)),$$

with sampling time $\tau \in \mathbb{R}_{>0}$. The next definition describes our observability assumption.

Definition 5.1. *The control system* (5.1)-(5.2) *is said to be ℓ-observable if there exists a map $r : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^n$ satisfying:*

$$r(S_{\text{meas}}(t, \ell)) = x(t), \tag{5.3}$$

for any solution $x(t)$ of (5.1)-(5.2)*.*

This notion of observability is restatement of the notion of uniform observability on compact sets provided in [Han09].

Given the unknown continuous function $r : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^n$ returning the current state from current and past inputs and outputs, we want to build a neural network implementing the function $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^n$ approximating r with respect to the L^∞ norm. The function ψ is to be constructed only from the data $S_{\text{meas}}(t, \ell)$ and the evaluation of r on $S_{\text{meas}}(t, \ell)$.

For simplicity of presentation, and in the interest of space, we assume the sampling time τ is small enough so that we can neglect the effect of only having state estimates produced at the time instants $q\tau$, $q \in \mathbb{Z}_{\geq 0}$. In other words, we assume that state estimates are produced for every $t \in \mathbb{R}_{\geq 0}$ and, consequently, control inputs are also produced at every $t \in \mathbb{R}_{\geq 0}$. Eliminating this simplifying assumption can be done, e.g., by resorting to the sampled-data techniques described in [NT04] and would not substantially alter the results.

We can now formalize the first problem addressed in this work.

Problem 1. (Deep neural network observer design): *Design a deep neural network to implement the map $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^n$ producing the state estimate $\widehat{x}$ from the sequence of inputs and outputs $S_{\text{meas}}(t, \ell)$ so that the closed-loop system:*

$$\dot{x} = f(x, k(\widehat{x}), d),$$

is safe and/or stable.

When a deep neural network is used to directly provide control inputs, it instead implements the function $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^n$ approximating $k \circ r$ with respect to the L^∞ norm. Once again, ψ is to be constructed only from the data $S_{\text{meas}}(t, \ell)$ and the evaluation of $k \circ r$ on $S_{\text{meas}}(t, \ell)$. With this formalism, we now introduce the second problem addressed in this work.

Problem 2. (Deep neural network controller design): *Design a deep neural network to implement the map $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^m$ producing the control input $\widehat{u}$ from the sequence of inputs and outputs $S_{\text{meas}}(t, \ell)$ so that the closed-loop system:*

$$\dot{x} = f(x, \widehat{u}, d),$$

is safe and/or stable.

In the next sections we provide some key details on the approximation power of deep residual neural networks in preparation to address these problems in Sections 7.2 and 7.3.

To train a neural network under this setup we require samples from the map r in (5.3). Although this may seem restricting, several practical scenarios fit this setup. For example, a drone with a mounted camera or LiDAR could rely on a neural network to implement the map from perception data to the attitude of the drone. The samples of the map r can be built through experiments in a controlled scenario where the true state of the drone is separately obtained from a camera localization system. Once the neural network is trained on this data, however, the drone would be able to operate purely from perception data.

CHAPTER 6

Deterministic Generalization Guarantees

It was established in the previous chapters that, under suitable monotonicity assumptions, we can provide a generalization error with a deterministic upper bound, and the generalization guarantees are based on the set of samples E_S providing "good coverage" of the set E. To formalize this idea, we use the abbreviation *w.r.t.* of *with respect to* and introduce the notion of δ-cover w.r.t. $\preceq$.

Definition 6.1. **(δ-cover w.r.t. $\preceq$):** *Let E_S and E be subsets of $\mathbb{R}^n$. The set E_S is a δ-cover of E w.r.t. the partial order $\preceq$ for some $\delta \in \mathbb{R}_{\geq 0}$, if for any $x \in E$, there exist $\underline{x}, \overline{x} \in E_S$ such that:*

$$x \in [\underline{x}, \overline{x}] \quad \textit{and} \quad \|\underline{x} - \overline{x}\|_\infty \leq \delta.$$

Recalling that E_S is a finite set of samples, we call E_S a finite δ-cover *of E w.r.t. $\preceq$.*

The following key result from Theorem 3.1 provides the desired upper bound, provided the approximating function ϕ can be decomposed as the sum of a monotone and a linear function. For convenience, we restate a version of the result in the lemma below.

Lemma 6.1. (Generalization Lemma): *Let $g : \mathbb{R}^n \to \mathbb{R}^n$ be a continuous function, let $E \subset \mathbb{R}^n$ be a compact set, and let E_S be a finite δ-cover of E w.r.t. to $\preceq$. For any function $\psi : \mathbb{R}^n \to \mathbb{R}^n$ of the form $\psi = \phi + A$, where $\phi : \mathbb{R}^n \to \mathbb{R}^n$ is a monotone function and $A : \mathbb{R}^n \to \mathbb{R}^n$ is a linear map, the generalization error is upper bounded by:*

$$\|g - \psi\|_{L^\infty(E)} \quad \leq \quad 3\|g - \psi\|_{L^\infty(E_S)} + 2\omega_g(\delta) + 2\|A\|_\infty \delta, \tag{6.1}$$

where ω_g is a modulus of continuity of g on E and $\|A\|_\infty$ is the operator ∞-norm of the map A.

The assumption provided by the equality $\psi = \phi + A$ immediately raises three questions: 1) are there deep neural networks that implement such functions? 2) would implementing such functions prevent the neural network from having a small training error? 3) are there training algorithms that enforce the desired monotone plus linear assumption?

The first two questions were addressed in [TG20] for the class of deep residual neural networks. It was shown that for any continuous function $g : \mathbb{R}^n \to \mathbb{R}^n$ there exists a deep residual neural network with $n + 1$ neurons per layer implementing an approximating function of the desired form $\phi + A$ and achieving arbitrarily small training error (see Theorem 4.4 and Corollary 4.5 in [TG20]). The third question was addressed in the previous chapters by showing that most training algorithms can be modified with a projection step to enforce certain constraints on the network weights, thereby satisfying the desired monotonicity properties.

CHAPTER 7

Safety and Stability Guarantees

In Section 7.2, we use Lemma 6.1 to establish local Input-to-State Stability for the closed-loop system. Finally, in Section 7.3, we leverage this result to prove safety of the closed-loop system, expressed as local Input-to-State Safety.

7.1 From density to the δ-cover property

The major requirement in Lemma 6.1, in addition to monotonicity, is that the set of sample points $E_S \subseteq \mathbb{R}^n$ is a finite δ-cover of E w.r.t. $\preceq$. This condition can be hard to verify in practice, as it involves the relative position of the samples and the points in the set E, as governed by the partial order.

Alternatively, we provide a condition that relies only on how densely the points of E_S fill E, which we can then use to verify the finite-δ cover w.r.t. $\preceq$ condition for a subset $E' \subset E$. Ideally, we would like the δ-property to be verified for the whole E, so that we can claim the error guarantee in as much of the space as possible. As the density of samples increases, the subset where our guarantee holds, E', approaches the whole set E, as desired.

Lemma 7.1. *Let $E \subseteq \mathbb{R}^n$ be a compact set and E_S be a finite η-cover of E, i.e., there exists an $\eta \in \mathbb{R}_{\geq 0}$ such that for any $x \in E$ there exists an $x_s \in E_S$ satisfying $\|x - x_s\|_\infty \leq \eta$. Further assume that E contains a ball of radius 2η in the infinity-norm and let $E' \subset E$ be the set:*

$$E' = \left\{ x \in E \ : \ \inf_{x' \notin E} \|x - x'\|_\infty \geq 2\eta \right\}. \tag{7.1}$$

Then $E' \neq \emptyset$, and E_S is a finite δ-cover, with $\delta = 4\eta$, of E' w.r.t. to $\preceq$.

Proof. Observe that, since E contains a ball of radius 2η, the center of this ball belongs to E' thus implying non-emptiness of E'. If x is a point in E', the set:

$$\{x' \in E \ : \ x' \preceq x, \ \|x - x'\|_\infty \leq 2\eta\},$$

is an ℓ_∞ ball of radius η. Because E is a finite η-cover of E, this ℓ_∞ ball contains a point $\underline{x} \in E_S$ such that $\underline{x} \preceq x$. A dual argument follows to construct $\overline{x}$, and by the triangle inequality we find:

$$\|\underline{x} - \overline{x}\|_\infty \leq \|\underline{x} - x\|_\infty + \|x - \overline{x}\|_\infty \leq 4\eta.$$

Because x was arbitrary, this holds for all of E', thus E_S is a finite δ-cover w.r.t. to $\preceq$ of E' for $\delta = 4\eta$. □

7.2 Closed-loop stability guarantees

We start by discussing the case where the deep neural network produces state estimates and the controller $u = k(x)$ enforces Input-to-State Stability (ISS) w.r.t. state estimation errors and disturbances (Problem 1). This means the solution $x(t)$ of the closed-loop control system:

$$\dot{x} = f(x, k(x + e), d), \tag{7.2}$$

where the disturbance input d and the state estimate error $e = \widehat{x} - x$ are assumed to be essentially bounded, satisfies:

$$\|x(t)\| \ \leq \ \beta(\|x(0)\|, t) + \gamma_d\left(\|d\|_{L^\infty(\mathbb{R}_{\geq 0})}\right) + \ \gamma_e\left(\|e\|_{L^\infty(\mathbb{R}_{\geq 0})}\right), \tag{7.3}$$

for all $t \in \mathbb{R}_{\geq 0}$, some class $\mathcal{KL}$ function β and class $\mathcal{K}$ functions[1] γ_d and γ_e. In (7.3) we used the notation $\|x\|$ to denote any norm in $\mathbb{R}^n$ and $\|d\|_{L^\infty(\mathbb{R}_{\geq 0})}$ to denote the uniform norm defined as:

$$\|d\|_{L^\infty(\mathbb{R}_{\geq 0})} \triangleq \sup_{t \in \mathbb{R}_{\geq 0}} \|d(t)\|.$$

[1]A class $\mathcal{K}$ function $\gamma : \mathbb{R} \to \mathbb{R}$ is strictly increasing with $\gamma(0) = 0$. A class $\mathcal{KL}$ function $\beta : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a continuous function such that $\beta(\cdot, s) \in \mathcal{K}$ and $\beta(r, \cdot)$ is strictly decreasing with $\lim_{s \to \infty} \beta(r, s) = 0$.

An alternative setting that we will later use consists of designing a controller $u = k(x)$ that enforces ISS with respect to disturbance d and the actuation error e for the closed-loop system:

$$\dot{x} = f(x, k(x) + e, d), \tag{7.4}$$

implying the same inequality (7.3) with this new definition of e and different β, γ_d, and γ_e. This second perspective will allow us to provide results when using a deep neural network to produce the control input u instead of the state (Problem 2).

The key observation is that by combining the ISS assumption (7.3) with the generalization bound in Lemma 6.1, we obtain local ISS w.r.t. the disturbance d. Local ISS (LISS) only requires the ISS inequality (7.3) to hold for sufficiently small disturbances d and initial conditions $x(0)$. The LISS property, following [SW96], is defined in reference to a zero-invariant compact set $G \subset \mathbb{R}^n$, where zero-invariance means that trajectories of (5.1) with the disturbance d identically zero and $x(0) \in G$ remain inside G for all future times. A formal definition of LISS is provided below, where we use $\|\cdot\|_G$ to denote the distance to the set G w.r.t a chosen norm $\|\cdot\|$.

Definition 7.1. (Local input-to-state stability (LISS) to a set): *The control system* (5.1)*, with* $u(t) = k(x(t))$*, for some choice of controller* $k : \mathbb{R}^n \to \mathbb{R}^m$*, is locally input-to-state stable (LISS) w.r.t.* d *to a compact zero-invariant set* $G \subset \mathbb{R}^n$*, if there exist* $\rho \in \mathbb{R}_{\geq 0}$*,* $\gamma \in \mathcal{K}$*, and* $\beta \in \mathcal{KL}$ *s.t. for any* $x(0)$*,* d*, and* t *satisfying* $\|x(0)\| \leq \rho$*,* $\|d\|_{L^\infty(\mathbb{R}_{\geq 0})} \leq \rho$*, and* $t \geq 0$ *we have:*

$$\|x(t)\|_G \leq \beta\left(\|x(0)\|_G, t\right) + \gamma\left(\|d\|_{L^\infty(\mathbb{R}_{\geq 0})}\right). \tag{7.5}$$

In our scenario, we consider a compact region $F \subset \mathbb{R}^n$ of the state space containing a ball centered at the equilibrium to be stabilized, which we take to be the origin without loss of generality. We define $h(F)$ as the set:

$$h(F) = \{y \in \mathbb{R}^p \mid y = h(x),\ x \in F\},$$

and let E be $(h(F))^{\ell+1} \times (k(F))^{\ell}$, where $(h(F))^{\ell+1}$ is the $\ell + 1$ fold Cartesian product of $h(F)$ with itself, and therefore contains all the sequences of $\ell + 1$ outputs corresponding to sequences of states remaining in F. Similarly, the set $(k(F))^{\ell}$ contains all the sequences of ℓ inputs.

We assume E_S is a finite η-cover of E, representing the available samples for training. Using Lemma 7.1, this set of samples is a finite 4η-cover w.r.t. $\preceq$ of some $E' \subseteq E$, defined as in (7.1). We then require that $r(E')$, the set of states for which Lemma 6.1 applies with r as introduced in Definition 5.1, contains a neighborhood of the origin, with the intention of proving LISS to a subset of $r(E')$. Lemma 6.1 technically requires r to have the same domain and co-domain, whereas we are interested in mapping sequences of outputs and inputs in $(h(F))^{\ell+1} \times (k(F))^{\ell} \subset \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ to states in $F \subset \mathbb{R}^n$. However, we can always embed $\mathbb{R}^n$ in $\mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ so that the domain and co-domain are of the same dimension and the lemma's conditions are satisfied.

For convenience, we define the radius $\|S\|$ of a compact set $S \subset \mathbb{R}^n$ containing the origin, w.r.t. a norm $\|\cdot\|$ as:

$$\|S\| \triangleq \sup_{\substack{\zeta \in \mathbb{R}_{\geq 0} \\ \{x \in \mathbb{R}^n \,:\, \|x\| \leq \zeta\} \subseteq S}} \zeta. \tag{7.6}$$

We are now in a position to state one of the main results.

Theorem 7.1. *Let $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ be the map producing state estimates from observations, assumed to be the sum of a monotone function ϕ and linear function A, and let* (7.3) *be satisfied. Then, there exist $\mu, \bar{\eta} \in \mathbb{R}_{\geq 0}$ such that if $\|r - \psi\|_{L^\infty(E_S)} \leq \mu$ and E_S is a finite η-cover of $E \subset \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ with $\eta \leq \bar{\eta}$, the control system* (7.2) *in closed-loop with the controller $u(t) = k(\psi(S_{\text{meas}}(t, \ell)))$ is LISS w.r.t. the disturbance input d to a non-empty compact set $G \subset F$ containing the origin.*

Proof. Because E_S is a finite η-cover of E, by Lemma 7.1, E_S is a finite 4η-cover w.r.t. $\preceq$ of the set $E' \subset E$. Further, by Lemma 6.1, the function ψ satisfies $\|r - \psi\|_{L^\infty(E')} \leq \xi$, where:

$$\xi = 3\|r - \psi\|_{L^\infty(E_S)} + 2\omega_r(4\eta) + 8\|A\|_\infty \eta.$$

Consider any solution $x(t)$ of (7.2) with initial condition $x(0)$ satisfying:

$$\beta(\|x(0)\|, 0) + \gamma_d\left(\|d\|_{L^\infty(\mathbb{R}_{\geq 0})}\right) + \gamma_e(\xi) < \|r(E')\|. \tag{7.7}$$

This solution will remain in $r(E')$ for at least some time $t_1 \in \mathbb{R}_{>0}$, since the set of initial conditions satisfying (7.7) defines a strict open subset of $r(E')$. Moreover,

$$\|x - \widehat{x}\|_{L^\infty([0,t_1])} = \|e\|_{L^\infty([0,t_1])} \leq \xi, \tag{7.8}$$

and by the ISS properties of the controller in (7.3), we have:

$$\|x(t)\| \leq \beta(\|x(0)\|, t) + \gamma_d\left(\|d\|_{L^\infty([0,t_1])}\right) + \gamma_e(\xi). \tag{7.9}$$

Therefore, the solution will remain in $r(E')$ for all $t \in \mathbb{R}_{>0}$, by iteratively considering the state $x(t_1)$ as a new initial condition satisfying (7.7).

We now verify that there exist μ, $\bar{\eta}$ such that the set of initial conditions satisfying (7.7) is not empty. In particular, for any μ and $\bar{\eta}$ satisfying:

$$3\mu + 2\omega_r(4\bar{\eta}) + 8\|A\|_\infty\bar{\eta} \leq \gamma_e^{-1}(\|r(E')\|),$$

we have:

$$\xi \leq \gamma_e^{-1}(\|r(E')\|). \tag{7.10}$$

Hence, (7.7) will be satisfied as long as:

$$\|x(0)\| \leq \rho \quad \text{and} \quad \|d\|_{L^\infty(\mathbb{R}_{\geq 0})} \leq \rho, \tag{7.11}$$

for some ρ that satisfies:

$$\beta(\rho, 0) + \gamma_d(\rho) < \|r(E')\| - \gamma_e(\xi). \tag{7.12}$$

The constant ρ is guaranteed to exist since the left hand side is a class $\mathcal{K}$ function of ρ, and the right hand side is non-negative by virtue of (7.10). Let us now denote the solution of (7.2) at time t, with initial condition x_0 and zero disturbance, by $x(t, x_0, 0)$, and let:

$$O(B(\gamma_e(\xi))) = \{z \in \mathbb{R}^n \ : \ z = x(t, x_0, 0), t \in \mathbb{R}_{\geq 0},\ x_0 \in B(\gamma_e(\xi))\},$$

where $B(\gamma_e(\xi))$ denotes the closed ball, for the $\|\cdot\|$ norm, of radius $\gamma_e(\xi)$ centered on the origin. Note that (7.9) conforms to the definition of ISpS in Proposition VI.3 of [SW96], noting that the results only hold when (7.11) is satisfied, we conclude that the system is locally ISS to the set $G = \overline{O(B(\gamma_e(\xi))}$, the topological closure of $O(B(\gamma_e(\xi))$. □

Theorem 7.1 establishes LISS for the closed-loop system (7.2) when a neural network is used for state estimation. Moreover, the size of the set G, towards which trajectories converge, is directly related to the error in the neural network's training, which can in principle be made arbitrarily small [TG20]. We note that the LISS result only holds for sufficiently small training error μ and sufficiently high sample density (i.e., sufficiently small η in the finite η-cover E_S) and, although the result is stated existentially, concrete values and conditions are given in the proof for μ, η, ρ, and G.

If we directly estimate the stabilizing input from observations, we have a similar result.

Theorem 7.2. *Let $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ be the map producing approximate inputs from observations, assumed to be the sum of a monotone function ϕ and linear function A, and let* (7.4) *be satisfied. Then, there exist $\mu, \bar{\eta} \in \mathbb{R}_{\geq 0}$ such that if $\|k \circ r - \psi\|_{L^\infty(E_S)} \leq \mu$ and E_S is a finite η-cover of $E \subset \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ with $\eta \leq \bar{\eta}$, the control system* (7.4) *in closed-loop with the controller $u(t) = \psi(S_{\text{meas}}(t, \ell))$ is LISS w.r.t. the disturbance input d to a non-empty compact set $G \subset F$ containing the origin.*

The proof of this statement is completely analogous to that of Theorem 7.1, with the exception that r is replaced by $k \circ r$ in the argument and we reference the dual ISS condition of (7.3) involving $e = \widehat{u} - k(x)$.

Choosing whether to use a neural network to estimate the system state and apply Theorem 7.1, or to use a neural network directly as a controller and instead use Theorem 7.2, may depend on the achievable training errors in each regime, and the modulus of continuity ω_r versus $\omega_{k \circ r}$. This comparison, however, assumes one can construct a stabilizing nominal controller k that is ISS with respect to measurement errors and actuation errors for the same system. In practice, it may be easier to design a controller that is ISS with respect to actuation errors than measurement errors [Fre95]. In particular, for control-affine systems there are general methods to construct a globally asymptotically stabilizing control law ISS with respect to actuation errors [Son89], but no (general) such techniques exist for measurement errors.

7.3 Closed-loop safety guarantees

In the previous section, we formally established stability of closed-loop systems using deep neural networks. Of equal importance, however, is the notion of safety, interpreted here as the forward invariance of some set of safe states $C \subset \mathbb{R}^n$. In this section we study Problems 1 and 2 in regards to safety.

We assume again that an a priori controller $u = k(x)$ is designed, but this time to enforce Input-to-State Safety (ISSf) of a set of safe states $C \subset \mathbb{R}^n$. The ISSf notion was first introduced in [RJ16] and later studied in the context of control barrier functions in [KA19]. Even though the definition of ISSf in [KA19] only calls for forward invariance of an inflation of C, it is known from the results in [XTG15] that existence of barrier functions, and a fortiori of ISSf barrier functions, implies stability of such sets. Having this in mind, we directly define ISSf by requiring stability of the inflated sets.

The closed-loop control system (7.2), with $u = k(x)$ for some choice of controller $k : \mathbb{R}^n \to \mathbb{R}^m$, is Input-to-State Safe (ISSf) w.r.t to: 1) the set of safe states $C \subset \mathbb{R}^n$, assumed to be compact; 2) the state estimation error e and; 3) the disturbance d, if every solution of (7.2) satisfies:

$$\|x(t)\|_C \;\leq\; \beta(\|x(0)\|_C, t) + \gamma_d\left(\|d\|_{L^\infty(\mathbb{R}_{\geq 0})}\right) + \gamma_e\left(\|e\|_{L^\infty(\mathbb{R}_{\geq 0})}\right), \tag{7.13}$$

for all $t \in \mathbb{R}_{\geq 0}$ and for some class $\mathcal{KL}$ function β and class $\mathcal{K}$ functions γ_d and γ_e.

Because of ISSf's close relationship to ISS, we can also define it locally, where we say the system is locally ISSf (LISSf) w.r.t. the disturbance d to a compact, zero-invariant safe set G, if there exists a $\rho \in \mathbb{R}_{\geq 0}$ such that the system is ISSf w.r.t. d to G for any $x(0)$, d, satisfying $\|x(0)\|_G \leq \rho$, $\|d\|_{L^\infty(R_{\geq 0})} \leq \rho$, for all $t \in \mathbb{R}_{\geq 0}$. Note that this definition is identical to LISS in Definition 7.1, with G a safe set.

Given a safe set $C \subset \mathbb{R}^n$ and an $\varepsilon \in \mathbb{R}_{\geq 0}$, we define the inflated sets:

$$B_C(\varepsilon) = \{x \in \mathbb{R}^n \;:\; \|x\|_C \leq \varepsilon\}. \tag{7.14}$$

From (7.13), if the estimation error e is zero, then the state will converge to the inflated set

$B_C\left(\gamma_d(\|d\|_{L^\infty(\mathbb{R}_{\geq 0})})\right)$. As established in [KA19], the size of this inflated set increases monotonically with the magnitude of the disturbance d.

We again consider a compact region $F \subset \mathbb{R}^n$ containing the set C and let E be $(h(F))^{\ell+1} \times (k(F))^{\ell}$, the set of all sequences of $\ell+1$ outputs and inputs of the system from states in F. We assume E_s is a finite η-cover of E, which represents a finite set of samples for training that, by Lemma 7.1, is a 4η-cover w.r.t. $\preceq$ of some $E' \subset E$, as defined in (7.1). We further require that $C \subseteq r(E')$, since $r(E')$ is the region where formal guarantees can be made about the quality of the state estimates produced by a deep neural network.

Analogously to the definition of the radius of a compact set in (7.6), we define the radius of some compact subset $S \subset \mathbb{R}^n$ containing C w.r.t. the safe set $C \subset \mathbb{R}^n$, $\|S\|_C$ as:

$$\|S\|_C \triangleq \sup_{\substack{\zeta \in \mathbb{R}_{\geq 0} \\ \{x \in \mathbb{R}^n \,:\, \|x\|_C \leq \zeta\}}} \zeta. \tag{7.15}$$

We now make a statement parallel to Theorem 7.1 for safety.

Theorem 7.3. *Let $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ be the map producing state estimates from observations, assumed to be the sum of a monotone function ϕ and a linear function A, and let* (7.13) *be satisfied. Then, there exist μ, $\overline{\eta} \in \mathbb{R}_{\geq 0}$ such that if $\|r - \psi\|_{L^\infty(E_S)} \leq \mu$, and E_S is a finite η-cover of $E \subset \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ with $\eta \leq \overline{\eta}$, the control system* (7.2) *in closed-loop with the controller $u(t) = k(\psi(S_{\text{meas}}(t, \ell)))$ is LISS (or equivalently, locally ISSf) w.r.t. d to a non-empty compact set C' containing C.*

Proof. The proof is identical to the proof of Theorem 7.1 once we replace $\|x(t)\|$ with $\|x(t)\|_C$, replace $\|r(E')\|$ with $\|r(E')\|_C$ as defined in (7.15), and replace $B(\gamma_e(\xi))$ with $B_C(\gamma_e(\xi))$ as defined in (7.14). Because of its similarity to the previous proof, we omit the details here. □

Although the statement of Theorem 7.3 gives an existential result for C', the size of this set when compared to C can be bounded using an inequality of the same form as (7.9) (with $\|\cdot\|$ replaced by $\|\cdot\|_C$), thus scaling in size with $\|d\|_{L^\infty(\mathbb{R}_{\geq 0})}$ and ξ. In addition, when the disturbance d and

estimation error e are absent, we recover the ISSf property corresponding to the ideal controller, implying $C' = C$.

Theorem 7.3 establishes that when using a neural network as a state estimator, we can still ensure the local ISS of a slightly larger set of safe states. The result can be viewed as a generalization of Theorem 7.1, since the latter ensues by taking the safe set C to be the singleton containing the origin.

As before, we make a statement similar to Theorem 7.3 for neural networks directly producing control inputs.

Theorem 7.4. *Let $\psi : \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell} \to \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ be the map producing approximate inputs from observations, assumed to be the sum of a monotone function ϕ and a linear function A, and let* (7.13) *be satisfied. Then, there exist $\mu, \overline{\eta} \in \mathbb{R}_{\geq 0}$ such that if $\|k \circ r - \psi\|_{L^\infty(E_S)} \leq \mu$ and E_S is a finite η-cover of $E \subset \mathbb{R}^{p(\ell+1)} \times \mathbb{R}^{m\ell}$ with $\eta \leq \overline{\eta}$, the control system* (7.4)*, in closed-loop with the controller $u(t) = \psi(S_{\text{meas}}(t, \ell))$ is LISS (or equivalently, locally ISSf) w.r.t. d to a non-empty compact set C' containing C.*

CHAPTER 8

Conclusions

In this part, we showed how to modify existing training algorithms for deep residual networks so that approximation bounds can be given in the supremum norm. These results are different from the typical approximation guarantees in the literature in that they are deterministic and are based on the sample set used for training. They are applicable to scenarios where the domain of the function to be learned is known and can be appropriately sampled. Although not all applications satisfy these requirements, we regard these results as useful first steps to obtain hard guarantees with a view towards integrating deep networks within a control loop.

Further, we proposed a suite of results ensuring safety and stability of closed-loop systems with deep neural networks in the perception pipeline. Our findings most crucially hinge on the prior results on uniform approximation and generalization capabilities of deep residual neural networks. We believe it is possible to build upon these approximation and generalization results to provide even stronger guarantees.

Part II

Formal Error Bounds for LiDAR Localization

LiDAR is a widely used sensor for self-localization and SLAM algorithms. One key step in using LiDAR data for localization is the alignment of two LiDAR scans taken from different poses, a process called scan-matching or point cloud registration. Most existing algorithms for this problem are heuristic in nature and local, meaning they may not produce accurate results under poor initialization. Moreover, existing methods give no guarantee on the quality of their output, which can be detrimental for safety-critical tasks. In this work, we present PASTA (Provably Accurate Simple Transformation Alignment), a low-complexity *global* point registration method that is suitable for point clouds generated from LiDAR scans. This algorithm is global and does not rely on point-to-point correspondences, which are typically absent in LiDAR data. Moreover, and to the best of our knowledge, we offer the first point cloud registration algorithm with *provable error bounds* and verify experimentally the effectiveness of our method.

CHAPTER 9

Introduction

LiDAR[1] sensing is quickly becoming commonplace in autonomous vehicles and robots. LiDAR sensors are accurate, inexpensive, and provide rich environment data, enabling a wealth of successful work in localization and Simultaneous Localization and Mapping (SLAM) [HKR16, CCC16, DV20, CCL18]. To use LiDAR data effectively, these works typically rely on a crucial sub-routine for solving the point cloud registration problem.

Point cloud registration, also referred to as scan-matching, asks for the appropriate transformation relating two measurements of the same environment to one another. Typically, this transformation is simply a rotation and translation, but some more general settings also consider scaling and warps as well [YSC20]. This problem has a standard closed-form solution when the point clouds from each measurement are also equipped with point-to-point correspondences, or when the point clouds are uniformly spaced with no noise [Hor87]. In LiDAR sensor data, however, the associated point clouds have very non-uniform spacing and rarely contain perfect correspondences between points.

Other point cloud registration algorithms we might use for localization fall into one of three categories: local (iterative), global, or learning-based. While we refrain from listing all existing methods in detail (see [PCS15] for an excellent review), we provide some remarks to contextualize our work.

Arguably the most well-known local method for point cloud registration is Iterative Closest Point (ICP) [BM92]. The algorithm follows a simple loop of choosing correspondences (using

[1] Light Detection and Ranging.

nearest neighbors under the estimated transformation) then using them to estimate a new associated transformation. Since its inception, countless variants of ICP have been developed to handle more structured environments [SHT09]. All of these ICP variants, however, rely on the same alternating optimization approach, meaning they converge to an incorrect transformation when initialized poorly. Moreover, when the correspondences (point-wise, planar, or otherwise) fail to exist, such as in point clouds from LiDAR data, ICP-based methods may fail entirely.

In contrast, global algorithms require no initialization to estimate a transformation. These methods are typically based on branch-and-bound techniques to solve the ICP problem [YLJ13], or on random-sampling consensus [RBB09]. While global methods can have performance that is superior even to well-initialized local algorithms, they often suffer from prohibitively long runtimes or rely on carefully crafted features persisting between measurements.

Various machine learning algorithms have also been developed to solve the point cloud registration problem. In particular, machine learning methods are well-suited for extracting features from raw point cloud data [QSM17], or even for finding correspondences between features in multiple point clouds [GZW19]. The identified features or correspondences can then be directly handed to any of the aforementioned local or global algorithms [YSC21]. The major downfall of these methods is their need for large amounts of labeled data coming from the deployment environment and the inherent brittleness of learned features.

Despite their practical success, these algorithms *cannot be used in safety-critical applications*, as guaranteed safe control requires the state to be either known exactly, or formal bounds on its uncertainty. [DTC21]. To the authors' knowledge, only one other point cloud registration algorithm, TEASER++ [YSC21], provides formal guarantees, but relies on the presence of at least a few point-to-point correspondences that may be absent in LiDAR point clouds. While this problem can be circumvented by extracting point-features from the data, most scenarios require significant feature engineering.

In this work, we present a new algorithm named PASTA (Provably Accurate Simple Transformation Alignment), which has precisely these missing guarantees. Moreover, it operates without

requiring the existence of point correspondences, making it ideal for low-density LiDAR data. In our analysis of PASTA, we prove that its simple, globally valid algorithm has an explicit bound on the error between its output and the true transformation relating two point clouds. As already pointed out, these bounds may be instrumental, for example, in guaranteeing the safety of autonomous systems, but also in defining a supervisor for other heuristic and learning-based methods.

9.1 Notation

We begin by introducing some notation and concepts. The results we prove hold in $\mathbb{R}^n$ for any $n \in \mathbb{N}$, but when dealing with point clouds coming from LiDAR data, n will be 2 or 3.

1. A point cloud is a finite set with $d \in \mathbb{N}$ elements:

$$X = \{x^{(1)}, x^{(2)}, \ldots, x^{(d)}\}, \tag{9.1}$$

   where $x^{(i)} \in \mathbb{R}^n$ for each $i \in \{1, 2, \ldots, d\}$.

2. Given a point cloud X, its convex hull is given by:

$$H = \left\{ x = \sum_{i=1}^{m} \lambda_i x^{(i)} \,\middle|\, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0 \right\}. \tag{9.2}$$

3. Given a compact set $H \subset \mathbb{R}^n$, and the Lebesgue measure μ on $\mathbb{R}^n$, we denote the volume of H as:

$$|H| = \mu(H) = \int_H \mathrm{d}\mu, \tag{9.3}$$

   and define its radius as:

$$\rho(H) = \min_{q \in \mathbb{R}^n} \max_{x \in H} \|x - q\|. \tag{9.4}$$

4. We define the first moment of a compact set $H \subset \mathbb{R}^n$ as:

$$c = |H|^{-1} \int_{x \in E} x \, \mathrm{d}\mu. \tag{9.5}$$

5. We define the second moment of a compact set $H \subset \mathbb{R}^n$ relative to its first moment c as:

$$\Sigma = |H|^{-1} \int_{x \in H} (x - c)(x - c)^T \, \mathrm{d}\mu. \tag{9.6}$$

6. Given an orthornormal matrix $R \in \mathcal{SO}(n)$ and a set $H \subset \mathbb{R}^n$, we denote the rotated set as $RH \subset \mathbb{R}^n$:

$$RH = \{z \in \mathbb{R}^n \mid z = Rx, \ x \in H\}. \tag{9.7}$$

7. Given two compact sets $H_1, H_2 \subset \mathbb{R}^n$ of non-zero measure, we introduce a notion of "overlap" between them. This overlap is represented by a number $\delta \in [0, 1]$, and is defined as follows:

$$\delta = \frac{|H_1 \cap H_2|}{\max\{|H_1|, |H_2|\}}. \tag{9.8}$$

   Intuitively, when $\delta = 0$, H_1 and H_2 are disjoint, and when $\delta = 1$, $H_1 = H_2$.

8. Integral expressions without a specified domain are interpreted as over all of $\mathbb{R}^n$.

Note that for a convex hull H constructed from a (non-empty) point cloud, the moments are well-defined. Moreover, by partitioning H into a set of disjoint simplices (triangles in $\mathbb{R}^2$, tetrahedrons in $\mathbb{R}^3$), we can compute the moments of H as a weighted sum of the moments of these simplices, using just the coordinates of their vertices (see Appendix).

9.2 Problem statement

Given a point cloud $\{r_1^{(i)}\}_{i=1}^{m_1}$, $r_1^{(i)} \in \mathbb{R}^n$ coming from a LiDAR sensor located[2] at the pose (p_1, R_1), and a second point cloud $\{r_2^{(i)}\}_{i=1}^{m_2}$ from a pose (p_2, R_2) (both in some global coordinate frame) estimate the relative translation vector $\hat{p}$ and rotation matrix $\hat{R}$ that transforms the first pose into the second, i.e., such that $\hat{R}R_1 = R_2$ and $\hat{p} = R_1^T(p_2 - p_1)$.

[2]A pose is a pair $(p, R) \in \mathbb{R}^n \times \mathcal{SO}(n)$, where p denotes a translation vector and R denotes a rotation matrix between a fixed reference frame and a frame fixed to the LiDAR. The location of the LiDAR frame is the origin of the rays used by the sensor to perform the distance measurements.

CHAPTER 10

PASTA: Provably Accurate Simple Transformation Alignment

Our approach draws inspiration from methods for 2D image alignment based on principal component analysis [SB11, RL18]. The intuition behind these methods is as follows: the eigenvectors of the covariance matrix of a cloud of points span the principal axes of the data set. If the point clouds are related by a simple rotation and translation, these axes undergo the same rotation. As a consequence, it is possible to reconstruct the rotation between two point clouds by simply matching the eigenvectors of their covariance matrices. Once the rotation is known, we can easily find the translation by matching the first moments of the data sets.

This idea does not directly apply to point clouds generated by a LiDAR sensor, as even without occlusions the points are not related by a pure rotation and translation. This issue arises because the point clouds from LiDAR measurements are not uniformly distributed over the environment when the sensor changes pose, shown more clearly in Figure 10.1. To avoid this problem, PASTA first generates the *convex hull* of the point cloud, and then computes *its* first and second moment, rather than using the points alone.

10.1 Ideal case

PASTA is most easily understood for two compact sets $H_1, H_2 \subset \mathbb{R}^n$ that are perfectly related by a rigid transformation $H_2 = RH_1 + p$. Under this relationship, the first and second moments of H_1 and H_2 are related by the expressions:

$$\begin{aligned} c_2 &= Rc_1 + p \\ \Sigma_2 &= R\Sigma_1 R^T. \end{aligned} \tag{10.1}$$

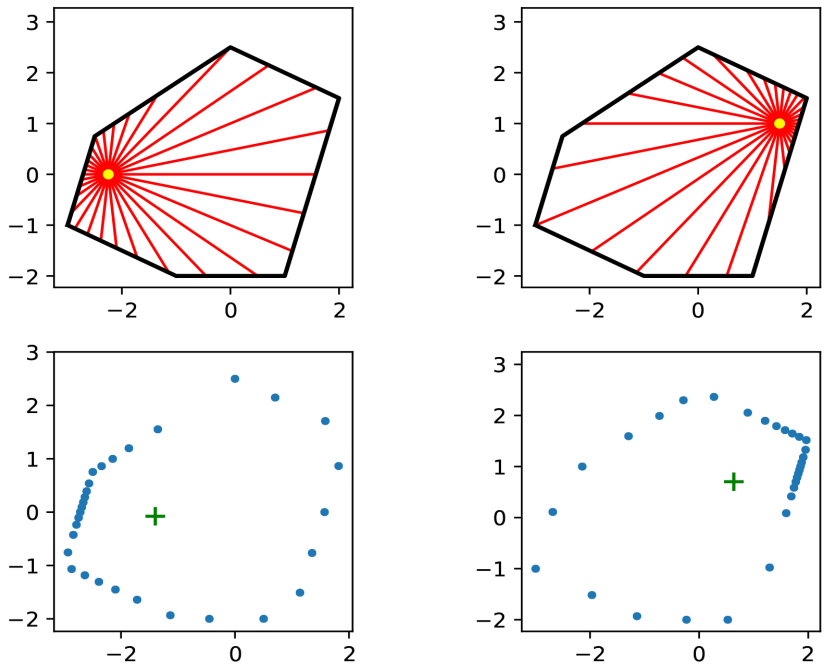

Figure 10.1: Top row: LiDAR rays from different positions in a 2D environment. Bottom row: corresponding distance measurements converted into a point cloud. The average of the points (green cross) differs between the two measurements.

These equations can easily be obtained by directly computing c_2 and Σ_2, using the change of variables $x = Rz + p$:

$$\begin{aligned} c_2 &= \frac{\int_{H_2} x \, \mathrm{d}\mu}{\int_{H_2} \mathrm{d}\mu} = \frac{\int_{H_1} (p + Rz)\det(R) \, \mathrm{d}\mu}{\int_{H_1} \det(R) \, \mathrm{d}\mu} \\ &= p + R|H_1|^{-1} \int_{H_1} z \, \mathrm{d}\mu \\ &= p + Rc_1, \end{aligned} \tag{10.2}$$

noting that $\det(R) = 1$ by the orthonormality of R. Similarly:

$$\begin{aligned} \Sigma_2 &= \frac{\int_{H_2} (x - c_2)(x - c_2)^T \, \mathrm{d}\mu}{\int_{H_2} \mathrm{d}\mu} \\ &= \frac{\int_{H_2} (x - (p + Rc_1))(x - (p + Rc_1))^T \, \mathrm{d}\mu}{\int_{H_2} \mathrm{d}\mu}. \end{aligned} \tag{10.3}$$

Using the change of variables $x = p + Rz$, we obtain:

$$\begin{aligned}\Sigma_2 &= \frac{\int_{H_1} R(z - c_1)(z - c_1)^T R^T \, \mathrm{d}\mu}{\int_{H_1} \mathrm{d}\mu} \\ &= R\Sigma_1 R^T.\end{aligned} \tag{10.4}$$

PASTA's estimate is then defined as the solution of (10.1) in terms of (p, R). In particular, the solution is given by

$$\begin{aligned}R &= V_2 V_1^T \\ p &= c_2 - Rc_1,\end{aligned} \tag{10.5}$$

where $V_1, V_2 \in \mathbb{R}^{n \times n}$ are the matrices whose columns are the unit-eigenvectors of Σ_1, Σ_2 respectively. We can verify this solution by direct substitution, noting that because R is orthogonal, Eq. (10.1) implies that Σ_1 and Σ_2 have the same eigenvalues, and thus their eigenvalue decompositions are:

$$\begin{aligned}\Sigma_1 &= V_1 \Lambda_1 V_1^T = V_1 \Lambda V_1^T \\ \Sigma_2 &= V_1 \Lambda_2 V_1^T = V_2 \Lambda V_2^T,\end{aligned} \tag{10.6}$$

where $\Lambda \in \mathbb{R}^{n \times n}$ is the diagonal matrix containing the eigenvalues of Σ_1 and Σ_2. It is then straightforward to verify that (10.1) is satisfied by the solution (10.5).

10.2 Algorithm

Before proceeding further, we clarify some technical details regarding the solution (10.5). First, there is a sign ambiguity on the eigenvectors of R, (note that both $V_2 V_1^T$ and $-V_2 V_1^T$ are valid solutions of (10.1)). This ambiguity can be resolved by using additional information about the sets H_1 and H_2. Let $v \in \mathbb{R}^n$ be one of the eigenvectors of R, and let $\mathrm{sign}_{H_2}(v)$ denote a function determining the choice of sign for the vector v given the convex hull H. Some examples of such a function are:

- Comparing the maximum and minimum values in the eigenvector directions:

$$\text{sign}_H(v) = \begin{cases} +1, \text{ if } |\max_i v^T \tilde{r}^{(i)}| > |\min_i v^T \tilde{r}^{(i)}| \\ -1, \text{ else,} \end{cases} \tag{10.7}$$

$$\text{where } \tilde{r}^{(i)} := r^{(i)} - c.$$

- Determining the direction of positive skewness (i.e., the third moment):

$$\text{sign}_H(v) = \begin{cases} +1, \text{ if } \frac{\int_{x \in H} \left(v^T (x-c)\right)^3 \mathrm{d}\mu}{\int_{x \in H} \mathrm{d}\mu} > 0 \\ -1, \text{ else.} \end{cases} \tag{10.8}$$

Second, in order for $\Lambda_1 = \Lambda_2 = \Lambda$ to hold in (10.6), we need to fix an ordering for the eigenvalues of the second moments. To define this ordering, we require the eigenvalues of Σ_1 and Σ_2 to be simple. These eigenvalues characterize the magnitude of variations along the principal axes of H_1 and H_2, so this assumption essentially requires the environment to be sufficiently *asymmetric*.

With these considerations, given two compact sets H_1 and H_2, we define the output of PASTA as:

$$(\hat{p}, \hat{R}) = \text{PASTA}(H_1, H_2) = \left(c_2 - V_2 V_1^T c_1, V_2 V_1^T\right), \tag{10.9}$$

whose operation is summarized in Algorithm 1.

10.3 Non-ideal case

As shown above, PASTA's output is exact when the sets H_1, H_2 are exactly related by a rigid transformation. We now explore what happens when trying to reconstruct (p, R) using (10.9) when this is no longer the case.

In practice, the sets H_1 and H_2 are constructed by computing the convex hulls of point clouds corresponding to LiDAR scans of some shape or environment. These scans typically do not sample the same points and are affected by noise, as shown in Fig. 10.2. Consequently, the relationship $H_2 = RH_1 + p$ will not hold. Instead we will have $H_2' = RH_1' + p$, where H_2' is the rigid

Algorithm 1 PASTA

Input: Point clouds $\{r_1^{(i)}\}_{i=1}^{m_1}$, $\{r_2^{(i)}\}_{i=1}^{m_2}$

Output: Transformation $\hat{R}, \hat{p}$

for each point cloud i **do**

$H_i \leftarrow$ convex hull of $\{r_i^{(j)}\}_{j=1}^{m_i}$

$c_i, \Sigma_i \leftarrow$ first and second moments of H_i

end for

$\hat{R} \leftarrow$ closed-form solution of $\Sigma_2 = R\Sigma_1 R^T$.

$\hat{p} \leftarrow$ closed-form solution of $c_2 = Rc_1 + p$

Summary of the PASTA algorithm. Note that the rotation equation may have multiple solutions, and we must choose one properly, as explained in Section 10.2.

transformation of the "perturbed" convex hull H_1', with the perturbation caused by the difference in sampled points and noise.

Given the convex hulls H_1, H_1', and H_2', we consider their respective second moments Σ_1, Σ_1', and Σ_2', with eigenvalue decompositions defined by:

$$\Sigma_1 = V_1 \Lambda_1 V_1^T, \quad \Sigma_1' = V_1' \Lambda' V_1'^T, \quad \Sigma_2' = V_2' \Lambda' V_2'.$$

Applying (10.9) to the sets H_1 and H_2', i.e. $(\hat{p}, \hat{R}) = \text{PASTA}(H_1, H_2')$, the estimated rotation $\hat{R}$ is:

$$\hat{R} = V_2' V_1^T. \tag{10.10}$$

Note that the following relationship holds:

$$\begin{aligned} \Sigma_2' &= R\Sigma_1' R^T \\ V_2' \Lambda' V_2'^T &= R V_1' \Lambda' V_1'^T R^T, \end{aligned} \tag{10.11}$$

implying that $V_2' = RV_1'$ and $\hat{R} = RV_1' V_1^T$.

If we define $V_1' = EV_1$ for some error rotation matrix $E \in \mathcal{SO}(n)$ (which always exists since

$V_1, V_1' \in \mathcal{SO}(n)$) , we arrive at

$$\hat{R} = REV_1V_1^T = RE. \tag{10.12}$$

Consequently, since R preserves the Frobenius norm, we can write the distance between $\hat{R}$ and the real rotation R as:

$$\left\|\hat{R} - R\right\| = \|RE - R\| = \|R(E - I)\| = \|E - I\|, \tag{10.13}$$

which captures how close the error rotation E is to the identity (the null rotation).

Similarly, under these non-ideal conditions, the translation vector estimated by PASTA is:

$$\hat{p} = c_2' - \hat{R}c_1, \tag{10.14}$$

and if we define $e = c_1' - c_1$ as the difference in the first moments of H_1' and H_1, we can bound the difference between the estimated and the real translation vector as:

$$\begin{aligned} \|\hat{p} - p\| &= \left\|c_2' - \hat{R}c_1 - c_2 + Rc_1\right\| \\ &= \|p + Rc_1' - REc_1 - p - Rc_1 + Rc_1\| \\ &= \|R(c_1 + e) - REc_1\| \\ &= \|R((I - E)c_1 + e)\| \\ &= \|(I - E)c_1 + e\| \leq \|I - E\| \, \|c_1\| + \|e\|, \end{aligned} \tag{10.15}$$

where the first equality holds due to the relationships $H_2 = RH_1 + p$ and $H_2' = RH_1' + p$, and the final equation holds because R is norm preserving.

In Chapter 11, we provide bounds on $\|e\| = \|c_1' - c_1\|$ and $\|I - E\| = \left\|I - V_1'V_1^T\right\|$, parameterized by how "close" H_1' is to H_1 in a manner described next. Note again that the "closeness" of H_1' and H_1 is determined by occlusions, measurement noise and non-uniform sampling effects in the LiDAR sensor.

10.4 Perturbation measure

The bounds that we will provide are expressed as functions of the size of H_1 and H_1', described by $\rho(H_1 \cup H_1')$, and an overlap parameter δ defined in (9.8) that describes how close H_1 is to H_1'.

Fig. 10.2 illustrates how point clouds generated from LiDAR measurements in different positions of a rectangular room sample different points, resulting in these different and non-overlapping convex hulls. Moreover, as the LiDAR resolution increases and its measurement noise lowers, there is greater overlap between H_1 and H_1'.

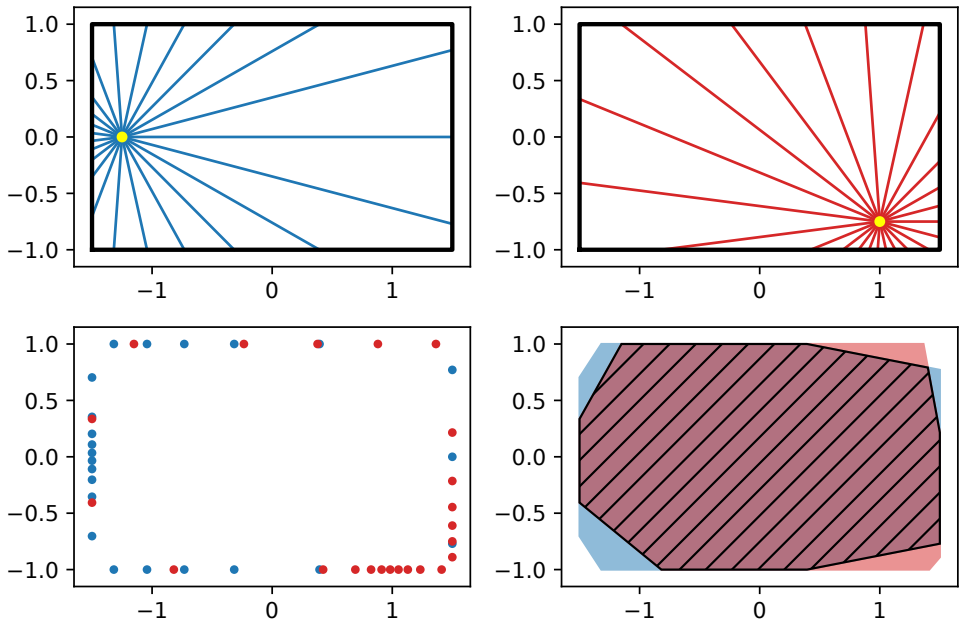

Figure 10.2: Top row: LiDAR rays from different positions in a 2D environment. Bottom left: point clouds corresponding to the two measurements (blue is the first measurement, red is the second). Bottom right: hulls of the two point clouds and their intersection (hatched region). Here, δ is the surface area of the hatched region divided by the greatest of the areas of the two hulls.

CHAPTER 11

Theoretical Analysis

PASTA uses the unit eigenvectors of the second moments to estimate the relative rotation between frames. We are therefore interested in bounding the change in these eigenvectors as a function of the perturbation of the second moment. We prove precisely this bound in the following result. In this section we present our main theoretical result, a new and tighter bound on the error in PASTA's estimates $(\hat{p}, \hat{R})$. This error is a function of how "closely" a compact set H and a perturbed version of H–here denoted by H'–are related by some true rotation and translation, as measured in variations of their second moments. In the following, we use $\|\cdot\|$ to refer to the ℓ_2 norm on $\mathbb{R}^n$ and its induced norm on $\mathbb{R}^{n\times n}$ and use $\lambda_i(A)$ to denote the ith eigenvalue of a symmetric matrix A. For compactness, we also define:

$$\begin{aligned}\overline{\Delta}_\lambda(A) &= \max_{i\neq j} |\lambda_i(A) - \lambda_j(A)| \\ \underline{\Delta}_\lambda(A) &= \min_{i\neq j} |\lambda_i(A) - \lambda_j(A)| .\end{aligned} \tag{11.1}$$

11.1 Eigenvector perturbation bounds

The second moment (9.6) corresponds to a symmetric and positive definite covariance matrix, and assuming distinct eigenvalues, this matrix has a set of orthogonal *eigenvectors*. If we additively perturb any symmetric matrix A with another symmetric matrix B, the eigenvectors of A will be perturbed by some angle. The following lemma bounds the magnitude of this angle as a function of the *norm* of B, and is a stepping stone for a stronger result to follow.

Theorem 11.1 (Symmetric Matrix Eigenvector Perturbation)**.** *Let $A, B \in \mathbb{R}^{n\times n}$ be symmetric matrices, where A has simple eigenvalues. Let λ_i and γ_i, $i = 1, ..., n$, be the ordered eigenvalues of*

A and $A+B$ respectively, with corresponding unit eigenvectors r_i and s_i. If $\|B\| < \frac{1}{2}\min_{i\neq j}|\lambda_i - \lambda_j|$, the angle θ_k between r_k and s_k satisfies:

$$|\theta_k| \leq \sin^{-1}\left(\frac{2\|B\|}{\min_{i\neq j}|\lambda_i - \lambda_j|}\right). \tag{11.2}$$

Proof: A and B are symmetric, thus their sum $C = A + B$ is also symmetric and we can write the eigenvalue decompositions $A = R\Lambda R^T$ and $C = S\Gamma S^T$. Note that Λ and Γ are diagonal matrices whose ith entries are the ith eigenvalues of A and C, respectively, in ascending order. Similarly, R and S are orthonormal matrices whose ith columns r_i and s_i are the eigenvectors associated to λ_i and γ_i, respectively.

Consider the following for any unit-eigenvector s_k of C:

$$\begin{aligned}
&s_k^T(A-C)^T(A-C)s_k \\
&\quad= s_k^T\left(R\Lambda R^T - S\Gamma S^T\right)^T\left(R\Lambda R^T - S\Gamma S^T\right)s_k \\
&\quad= \left(s_k^T R\Lambda R^T - \gamma_k s_k^T\right)\left(R\Lambda R^T s_k - \gamma_k s_k\right) \\
&\quad= s_k^T\left(\sum_{i=1}^n \lambda_i^2 r_i r_i^T\right)s_k - 2\gamma_k s_k^T\left(\sum_{i=1}^n \lambda_i r_i r_i^T\right)s_k + \gamma_k^2 \\
&\quad= \sum_{i=1}^n\left(\lambda_i^2 - 2\lambda_i\gamma_k\right)s_k^T r_i r_i^T s_k + \gamma_k^2\sum_{i=1}^n s_k^T r_i r_i^T s_k \\
&\quad= \sum_{i=1}^n\left(\lambda_i - \gamma_k\right)^2 s_k^T r_i r_i^T s_k,
\end{aligned}$$

where the fifth equality holds because s_k has unit norm, implying $\sum_i s_k^T r_i r_i^T s_k = \sum_i \left(r_i^T s_k\right)^2 = 1$. Further, note:

$$\begin{aligned}
\min_{i\neq k}(\lambda_i - \gamma_k)^2\sum_{i\neq k} s_k^T r_i r_i^T s_k &= \sum_{i\neq k}\min_{i\neq k}(\lambda_i - \gamma_k)^2 s_k^T r_i r_i^T s_k \\
&\leq \sum_{i\neq k}(\lambda_i - \gamma_k)^2 s_k^T r_i r_i^T s_k \leq s_k^T(A-C)^T(A-C)s_k \\
&= s_k^T B^T B s_k \leq \|B\|^2,
\end{aligned}$$

implying that as long as $\lambda_i - \gamma_k \neq 0$, we have that

$$\sum_{i\neq k} s_k^T r_i r_i^T s_k \leq \frac{\|B\|^2}{\min_{i\neq k}\left(\lambda_i - \gamma_k\right)^2}. \tag{11.3}$$

Because $\sum_i s_k^T r_i r_i^T s_k = 1$, we can rewrite (11.3) as:

$$\begin{aligned}\sum_{i \neq k} s_k^T r_i r_i^T s_k &= 1 - s_k^T r_k r_k^T s_k = 1 - \cos^2 \theta_k \\ &= \sin^2 \theta_k \leq \frac{\|B\|^2}{\min_{i \neq k} (\lambda_i - \gamma_k)^2}.\end{aligned} \tag{11.4}$$

Finally, by the triangle inequality,

$$|\lambda_i - \gamma_k| \geq |\lambda_i - \lambda_k| - |\lambda_k - \gamma_k|$$

and since $|\lambda_k - \gamma_k| \leq \|B\|$ and $\|B\| \leq \frac{1}{2} \min_{i \neq j} |\lambda_i - \lambda_j|$ by assumption, we conclude that:

$$|\lambda_i - \gamma_k| \geq \frac{1}{2} \min_{i \neq j} |\lambda_i - \lambda_j|.$$

Then, we can rewrite the bound (11.4) and find:

$$\begin{aligned}\sin^2 \theta_k &\leq \frac{4 \|B\|^2}{\min_{i \neq j} (\lambda_i - \lambda_j)^2}, \\ \Rightarrow \ |\theta_k| &\leq \sin^{-1} \left(\frac{2 \|B\|}{\min_{i \neq j} |\lambda_i - \lambda_j|} \right).\end{aligned}$$

■

The bound above is expressed in terms of the norm of the perturbation B and the difference in the eigenvalues of A. We can exploit this result to obtain a separate and strictly tighter bound expressed in terms of the maximum *eigenvalue separation* of B.

Corollary 1. *Under the same assumptions of Theorem 11.1, the angle θ_k between the kth eigenvector of A and $A + B$ is bounded by:*

$$|\theta_k| \leq \sin^{-1} \left(\frac{\max_{i \neq j} |\zeta_i - \zeta_j|}{\min_{i \neq j} |\lambda_i - \lambda_j|} \right), \tag{11.5}$$

where ζ_i is the ith eigenvalue of B.

Proof: First, note that adding a multiple of the identity kI to a matrix does not alter its eigenvectors, and shifts all its eigenvalues by k. Thus, the eigenvectors of $A + B$ are the same as those of the matrix

$$A + B - \frac{1}{2} \left(\overline{\lambda}(B) + \underline{\lambda}(B) \right) I$$

where $\overline{\lambda}(B)$ and $\underline{\lambda}(B)$, respectively, denote the maximum and minimum eigenvalues of B.

Accordingly, we may consider the equivalent perturbation $\widehat{B} = B - \frac{1}{2}\left(\overline{\lambda}(B) + \underline{\lambda}(B)\right) I$. By construction, then, $\overline{\lambda}(\widehat{B})$ and $\underline{\lambda}(\widehat{B})$ are:

$$\overline{\lambda}(\widehat{B}) = \frac{1}{2}\left(\overline{\lambda}(B) - \underline{\lambda}(B)\right) = \frac{1}{2}\max_{i,j}|\zeta_i - \zeta_j|,$$
$$\underline{\lambda}(\widehat{B}) = -\frac{1}{2}\left(\overline{\lambda}(B) - \underline{\lambda}(B)\right) = -\frac{1}{2}\max_{i,j}|\zeta_i - \zeta_j|.$$

Therefore, the induced ℓ_2 norm of $\widehat{B}$ is:

$$\left\|\widehat{B}\right\| = \frac{1}{2}\max_{i,j}|\zeta_i - \zeta_j|, \tag{11.6}$$

and by Theorem 11.1, since perturbing by $\widehat{B}$ and B produces the same eigenvalue perturbation to A, we conclude that for the kth eigenvector angle:

$$|\theta_k| \leq \sin^{-1}\left(\frac{\max_{i\neq j}|\zeta_i - \zeta_j|}{\min_{i\neq j}|\lambda_i - \lambda_j|}\right). \qquad \blacksquare$$

For a symmetric matrix B it always holds that $\max_{i\neq j}|\zeta_i - \zeta_j| \leq 2\|B\|$, making this a tighter bound than that in Theorem 11.1. Intuitively, Corollary 1 *minimizes* the bound (11.2) over all matrices $\widehat{B}$ that produce the same eigenvectors as $A + B$.

11.2 From overlap to eigenvalue separation

First, we need a measure of how "close" two sets H and H' are. To characterize this, we recall the following notions of size ρ and overlap δ:

$$\rho(H, H') = \min_{q\in\mathbb{R}^n}\max_{x\in H\cup H'}\|x - q\|, \tag{11.7}$$
$$\delta(H, H') = \frac{|H \cap H'|}{\max\{|H|, |H'|\}}. \tag{11.8}$$

Note that $\rho(H, H')$ describes the radius of the smallest ℓ_2-ball containing both H and H', while $\delta(H, H')$ is a measure of geometric similarity.

Remark 1. *When using* PASTA*, the definitions of δ and ρ do not apply directly to the sets H_1 and $H_2 = RH_1 + p$, which will not overlap in general. Instead, we are interested in how similar the*

(noisy) observed *set H_2 is to the reference set H_1 under the* true *rotation R and translation p. This is equivalent to using δ and ρ to compare a "perturbed" version of the set H_1, which we call H_1', which* is defined by $H_2 = RH_1' + p$. *Note that $H_1 \neq H_1'$ in general because of occlusions, LiDAR resolution, and noise. In the following, δ quantifies the overlap between H_1 and H_1'.*

We first need to prove a bound for the perturbation of the first moment:

Theorem 11.2 (First moment perturbation). *Let $H, H' \subset \mathbb{R}^n$ be compact sets of non-zero measure, and let their first moments be $c, c' \in \mathbb{R}^n$ respectively. Further, let the overlap between H and H' be $\delta \in [0, 1]$. Then:*

$$\|c' - c\| \leq 2(1 - \delta)\rho(H \cup H'). \tag{11.9}$$

Proof. First, let the vector $b \in \mathbb{R}^n$ be such that:

$$b \in \operatorname*{argmin}_{b \in \mathbb{R}^n} \max_{x \in H \cup H'} \|x - b\|, \tag{11.10}$$

which exists by compactness of $H \cup H'$. Then, we write:

$$\begin{aligned} \|c' - c\| &= \|c' - b - (c - b)\| \\ &= \left\| \frac{1}{|H'|} \int_{x \in H'} (x - b) \, \mathrm{d}\mu - \frac{1}{|H|} \int_{x \in H} (x - b) \, \mathrm{d}\mu \right\|. \end{aligned} \tag{11.11}$$

Before computing the bound, we observe that the first moment of a compact set is the expected value of a uniform probability distribution with that set as a support. To simplify the following expressions we define the indicator function:

$$\mathbf{1}_H(x) = \begin{cases} 1, & x \in H \\ 0, & \text{else.} \end{cases} \tag{11.12}$$

Then, we can write the uniform distribution over H' evaluated at any point $x \in \mathbb{R}^n$ as:

$$\begin{aligned} \frac{\mathbf{1}_{H'}(x)}{|H'|} &= \frac{\mathbf{1}_{H'}(x)}{|H'|} + \frac{\mathbf{1}_H(x)}{|H|} - \frac{\mathbf{1}_H(x)}{|H|} + \frac{\mathbf{1}_{H \cap H'}(x)}{\max\{|H|, |H'|\}} - \frac{\mathbf{1}_{H \cap H'}(x)}{\max\{|H|, |H'|\}} \\ &= \frac{\mathbf{1}_H(x)}{|H|} + f_+(x) - f_-(x), \end{aligned} \tag{11.13}$$

where we have defined f_+ and f_- as:

$$f_+(x) = \frac{\mathbf{1}_{H'}(x)}{|H'|} - \frac{\mathbf{1}_{H\cap H'}(x)}{\max\{|H|,|H'|\}}$$
$$f_-(x) = \frac{\mathbf{1}_{H}(x)}{|H|} - \frac{\mathbf{1}_{H\cap H'}(x)}{\max\{|H|,|H'|\}}.$$

Observe that f_+ and f_- are non-negative, and by definition of δ enjoy the following property:

$$\int f_+(x)\,\mathrm{d}\mu = \int f_-(x)\,\mathrm{d}\mu \leq 1-\delta. \tag{11.14}$$

We can now directly compute:

$$\begin{aligned}
\|c'-c\| &= \left\| \int (x-b)\frac{\mathbf{1}_{H'}}{|H'|}\,\mathrm{d}\mu - \int (x-b)\frac{\mathbf{1}_{H}}{|H|}\,\mathrm{d}\mu \right\| \\
&= \left\| \int (x-b)\,(f_+ - f_-)\,\mathrm{d}\mu \right\| \\
&\leq \int \|x-b\|\,(|f_+| + |f_-|)\,\mathrm{d}\mu \\
&\leq 2(1-\delta)\rho(H\cup H').
\end{aligned} \tag{11.15}$$

□

Following, we prove a bound for the eigenvalue difference of the covariance perturbation:

Theorem 11.3. *Let $H, H' \subset \mathbb{R}^n$ be compact sets of non-zero measure, and let their second moments be $\Sigma, \Sigma' \in \mathbb{R}^{n\times n}$ respectively. If their size is $\rho(H,H')$ and their overlap is $\delta(H,H') \in [0,1]$, then:*

$$\overline{\lambda}(\Sigma'-\Sigma) - \underline{\lambda}(\Sigma'-\Sigma) \leq \left(2(1-\delta) + 4(1-\delta)^2\right)\rho^2, \tag{11.16}$$

where $\overline{\lambda}(\cdot)$ and $\underline{\lambda}(\cdot)$ again denote the maximum and minimum eigenvalues of a symmetric matrix.

Proof. Let c and c' be the first moments of H and H' respectively, and let us define $\Delta c = c' - c$. Then, the following holds:

$$\Sigma' - \Sigma = \int (x-c')(x-c')^T f_+(x)\,\mathrm{d}\mu - \int (x-c')(x-c')^T f_-(x)\,\mathrm{d}\mu + \Delta c \Delta c^T, \tag{11.17}$$

where f_+ and f_- are non-negative and satisfy:

$$\int f_+(x)\,\mathrm{d}\mu = \int f_-(x)\,\mathrm{d}\mu \leq 1-\delta. \tag{11.18}$$

Note that $\Sigma' - \Sigma$ is the sum of three terms such that:

1. $\int (x-c')(x-c')^T f_+(x)\,\mathrm{d}\mu$ is positive definite with minimum eigenvalue zero and maximum eigenvalue:

$$\begin{aligned}\overline{\lambda}\left(\int (x-c')(x-c')^T f_+(x)\,\mathrm{d}\mu\right) &= \left\|\int (x-c')(x-c')^T f_+(x)\,\mathrm{d}\mu\right\| \\ &\leq \int \left\|(x-c')(x-c')^T\right\| f_+(x)\,\mathrm{d}\mu \\ &\leq (1-\delta)\rho^2.\end{aligned}$$

2. Analogously, $-\int (x-c')(x-c')^T f_-(x)\,\mathrm{d}\mu$ is negative definite with maximum eigenvalue zero and minimum eigenvalue:

$$\underline{\lambda}\left(-\int (x-c')(x-c')^T f_+(x)\,\mathrm{d}\mu\right) \geq -(1-\delta)\rho^2.$$

3. The eigenvalues of $\Delta c \Delta c^T$ are all zero except for $\overline{\lambda}(\Delta c \Delta c^T) = \|\Delta c\|^2 \leq 4(1-\delta)^2\rho^2$. Where the inequality holds by Theorem 11.2.

Consequently, the eigenvalues of $\Sigma' - \Sigma$ are bounded as:

$$\begin{aligned}\overline{\lambda}(\Sigma' - \Sigma) &\leq (1-\delta)\rho^2 + 0 + 4(1-\delta)^2\rho^2, \\ \underline{\lambda}(\Sigma' - \Sigma) &\geq 0 - (1-\delta)\rho^2 + 0.\end{aligned}$$

Taking the difference, we recover:

$$\begin{aligned}\overline{\lambda}(\Sigma' - \Sigma) - \underline{\lambda}(\Sigma' - \Sigma) &\leq (1-\delta)\rho^2 + 4(1-\delta)^2\rho^2 - \left(-(1-\delta)\rho^2\right) \\ &= \left(2(1-\delta) + 4(1-\delta)^2\right)\rho^2.\end{aligned}$$

□

11.3 Pose estimate error bound

With Theorem 11.3 in hand, we can finally express an error bound on the pose estimate as a function of $\delta(H, H')$ and $\rho(H, H')$.

Theorem 11.4 (PASTA error bound). *Let $H, H' \subset \mathbb{R}^n$ be non-empty compact sets of non-zero measure with an overlap of $\delta(H, H') \in [0, 1]$. Let c, c' and Σ, Σ' be the first and second moments of H, H', and define the constants:*

$$e_c = 2(1-\delta)\rho(H, H'),$$
$$e_\Sigma = \left(2(1-\delta(H, H')) + 4(1-\delta(H, H'))^2\right)\rho^2(H, H').$$

Let (R, p) be the true transformation relating H to $RH' + p$, and $\left(\hat{R}, \hat{p}\right) = \text{PASTA}(H, RH' + p)$ be the transformation estimated by PASTA. *Then, if $\min_{i,j} |\lambda_i - \lambda_j| > 2e_\Sigma$, where λ_i is the ith eigenvalue of the second moment of H, the following holds:*

$$\begin{aligned} \left\|\hat{R} - R\right\| &\leq \sqrt{n}\frac{e_\Sigma}{\underline{\Delta}_\lambda(\Sigma)} \\ \|\hat{p} - p\| &\leq \sqrt{n}\frac{e_\Sigma}{\underline{\Delta}_\lambda(\Sigma)}\|c\| + e_c. \end{aligned} \tag{11.19}$$

Proof. By the analysis in Section 10.3, it holds that:

$$\|\hat{R} - R\| = \left\|V'V^T - I\right\| = \|V' - V\|, \tag{11.20}$$

where V and V' are the eigenvector matrices of the covariance matrices of H and H' respectively. Then, note that by simple trigonometry if a vector $v \in \mathbb{R}^n$ is rotated by an angle θ into the vector v', their distance is $\|v' - v\| = 2\|v\|\left|\sin\frac{\theta}{2}\right|$. Therefore, by Theorem 1 and Theorem 11.3:

$$\begin{aligned} \left\|\hat{R} - R\right\|^2 = \|V' - V\|^2 \leq \|V' - V\|_F^2 &\leq \sum_{i=1}^n \|v_i' - v_i\|^2 \\ &\leq \sum_{i=1}^n \left(2\sin\left(\frac{1}{2}\sin^{-1}\left(\frac{\overline{\Delta}_\lambda(\Sigma' - \Sigma)}{\underline{\Delta}_\lambda(\Sigma)}\right)\right)\right)^2 \\ &\leq \sum_{i=1}^n \left(2\sin\left(\sin^{-1}\frac{1}{2}\left(\frac{\overline{\Delta}_\lambda(\Sigma' - \Sigma)}{\underline{\Delta}_\lambda(\Sigma)}\right)\right)\right)^2 \\ &\leq n\left(\frac{\overline{\Delta}_\lambda(\Sigma' - \Sigma)}{\underline{\Delta}_\lambda(\Sigma)}\right)^2 \leq n\left(\frac{e_\Sigma}{\underline{\Delta}_\lambda(\Sigma)}\right)^2. \end{aligned}$$

As for the position error, also by the analysis in Section 10.3 and Theorem 11.2, it holds that:

$$\begin{aligned}\|\hat{p} - p\| &\leq \left\|\hat{R} - R\right\| \|c\| + \|c' - c\| \\ &\leq \sqrt{n}\frac{e_\Sigma}{\underline{\Delta}_\lambda(\Sigma)} \|c\| + \|c' - c\| \\ &\leq \sqrt{n}\frac{e_\Sigma}{\underline{\Delta}_\lambda(\Sigma)} \|c\| + e_c.\end{aligned}$$

□

The bounds in (11.19) depend on the size of the environment (described by ρ) and the achievable overlap (described by δ). While these bounds are always valid, it should be noted that in practice δ can be especially low in non-convex environments, because different parts of the environment are visible from different positions. Note that there is an additional dependence on the minimum separation in the eigenvalues of Σ. This quantity depends on how "asymmetric" the environment is. For example, a very long but thin room would have a high minimum separation.

Some additional remarks are in order. We note that explicitly computing the bounds requires knowledge of the parameter δ. One open question is if one can obtain a bound on the errors which only depends on knowledge of the size and shape of the environment, resolution of the LiDAR, and bounds on the measurement noise. In practice, an estimate for δ can easily be obtained via a calibration-like experiment, where the LiDAR sensor is used to collect point cloud data from multiple positions, and the overlaps of their hulls are evaluated. Empirically, we observe that the lowest overlaps – and therefore worst associated guarantees – occur at poses where the sensor is close to walls and corners.

Remark 2. PASTA*'s theoretical guarantees can be applied to other algorithms with a supervisory approach. First run* PASTA*, which produces an estimate with worst-case guarantees. Then, run any other algorithm to find a new estimate. The triangle inequality immediately provides a naive guarantee on the estimate provided by this new algorithm.*

Remark 3. *Theorem 11.4 depends on the similarity, δ, between the two convex hulls H_1 and H_2. Occlusions or non-convex geometry cause dramatic changes in these shapes, and the overlap δ*

naturally reduces and the estimation error increases according to (11.19). *Note that when the shapes are sufficiently different and* δ *is very small, the measurements have so little in common that it is unrealistic to expect* any algorithm *to correctly solve the localization problem.*

Remark 4. PASTA*'s error bound can be computed through either* $\overline{\Delta}_\lambda(B)$ *or the geometric parameters* (δ, ρ). *This flexibility raises the question of which quantity to use in a practical setting. Either set of parameters can be estimated by performing calibration experiments in an environment with access to the ground truth poses by computing the true* $\overline{\Delta}_\lambda(B)$ *or* δ *at these poses, then estimating their worst-case values outside. Note that the bound computed directly from an estimate of* $\overline{\Delta}_\lambda(B)$ *provides a significantly tighter bound (see Fig. 13.3).* δ *and* ρ*, however, have a clearer* physical *interpretation and are easier to reason about during the calibration and subsequent worst-case estimation.*

CHAPTER 12

Simulations

In this section, we illustrate the effectiveness of PASTA as a localization method in a simple simulated closed-loop trajectory tracking task. Together with the simulation results, we show a numerical evaluation of the pose estimate bounds according to the theoretical analysis above.

Our simulation consists of a simple robot moving within a convex 2D environment that extends approximately 6m horizontally and 3m vertically. The robot possesses three degrees of freedom (horizontal position, vertical position, and angle) and obeys simple double integrator dynamics in each state,where $u_x, u_y, u_\omega \in \mathbb{R}$ are positional and angular acceleration control inputs. The system does not have access to the state, but instead receives a point cloud constructed from LiDAR measurements at the current pose.

In the experiment, the LiDAR data is produced with a resolution of 1° (360 rays), and the distance measurements are affected by zero mean Gaussian noise with a standard deviation of 1cm. The values are picked to be comparable to the performance of available real-world LiDAR sensors.

The system is simulated using zero-order hold control inputs generated at a frequency of 100Hz. The controller consists of a simple full state feedback linear controller acting on a state estimate provided by a Luenberger observer. The Luenberger observer operates using only the pose estimate generated by PASTA from the LiDAR data.

We then ask the system to track a sinusoidal trajectory within the environment. A visualization of the environment and the trajectories is shown in Fig. 12.1.

We initialized the state of the system at a position within 0.75m and 45° of the reference trajectory with zero velocity. Similarly, the initial state estimate is chosen within 0.25m and 15° from

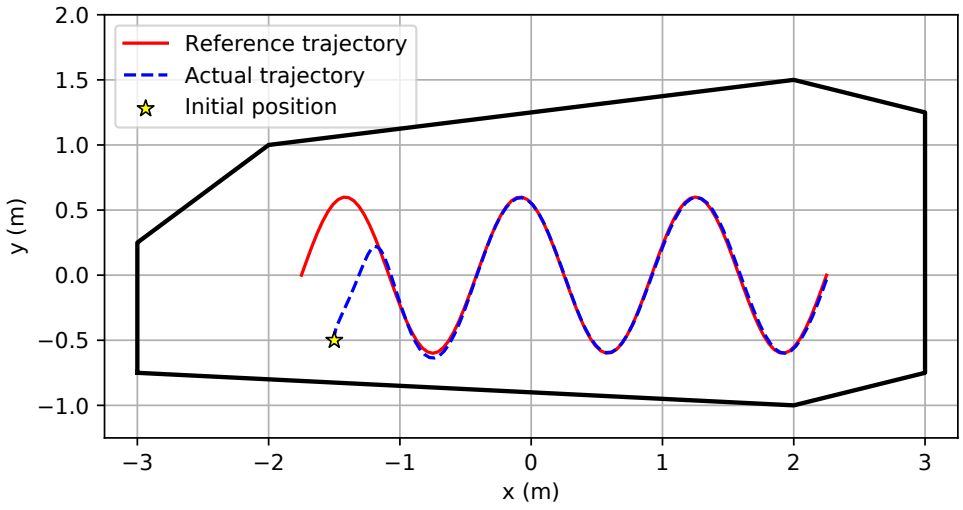

Figure 12.1: Visualization of the environment, the reference and actual position trajectories in the closed-loop simulation.

the real initial state.

In Fig. 12.2, we show the error in state estimation when the Luenberger observer uses the pose estimate provided by PASTA as opposed to the true pose of the system. Notably, the errors are almost identical, with steady state errors in the order of centimeters and fractions of a degree.

We also evaluate the numerical bounds for the pose estimate provided by PASTA along the trajectory, together with its actual incurred error, plotted using different scales, in Fig. 12.3. Note that the error bound on the norm of the rotation matrix is converted to a bound on the corresponding rotation angle for ease of interpretation. As expected, the worst case bound is conservative, with the actual error being at least an order of magnitude lower.

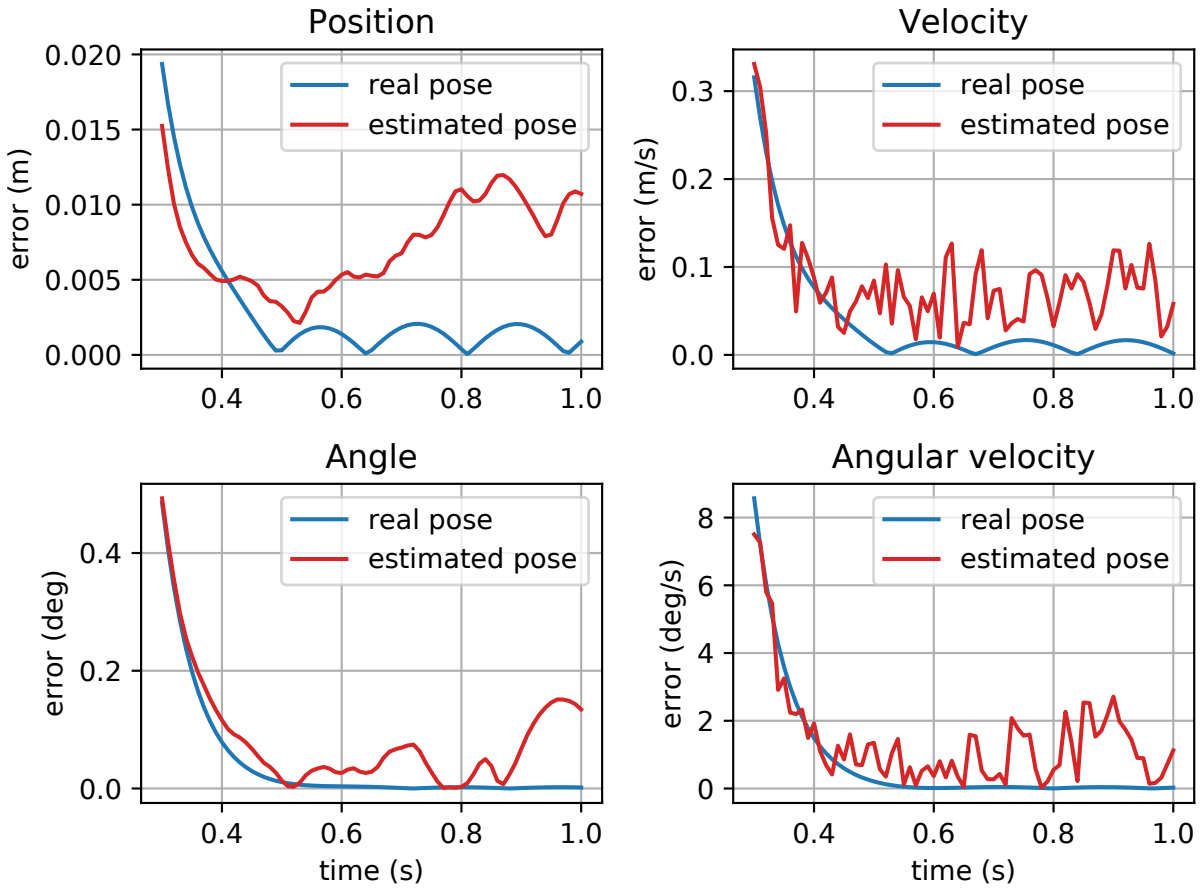

Figure 12.2: Norm of the trajectory tracking error over time for the closed-loop control task with the observer fed by the real pose and the pose estimated by PASTA. We do not plot the initial transient for ease of visual comparison.

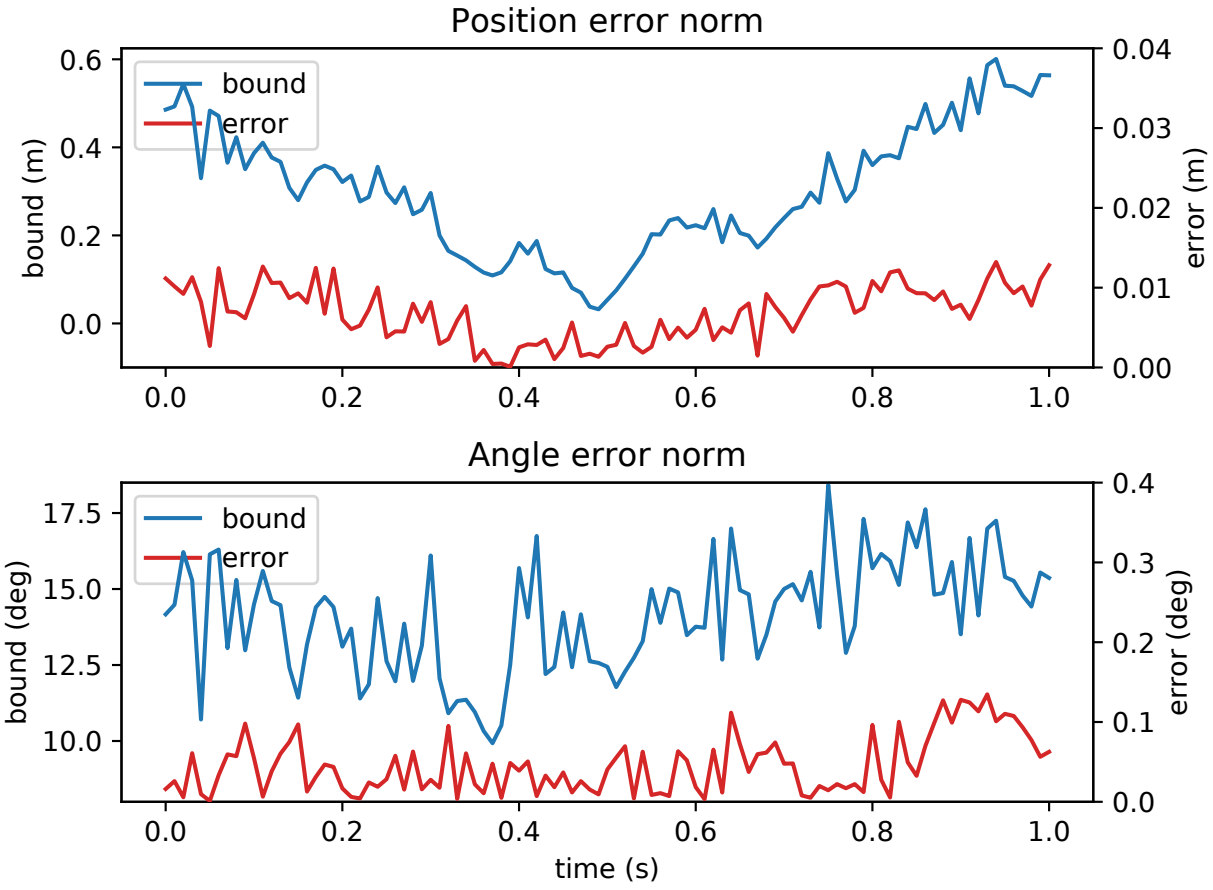

Figure 12.3: Norms of the actual error of the pose estimated by PASTA along the trajectory compared with the error bound guaranteed by Theorem 11.4. Note the bound scale (left-hand side) differs from the actual error scale (right-hand side).

CHAPTER 13

Experiments

While the bounds from Corollary 1 and Theorems 11.2 and 11.4 hold in theory, we now establish that they are reasonably tight and behave as expected in extreme settings.

We equipped a small wheeled robot with a 360° 2D LiDAR with 0.5° of angular resolution (i.e., each LiDAR point cloud consists of 720 points in the plane). We placed the robot in a closed indoor environment with several obstacles, measuring both ground truth poses and LiDAR sensor measurements as the robot constantly moved around the environment. See Fig. 13.1 for a sample LiDAR measurement and the robot's trajectory in the environment.

Remark 5. *We use a 2D LiDAR sensor in this regime since we are considering the indoor navigation setting, where 2D LiDAR is often sufficient. The theory developed for* PASTA *extends naturally to three dimensions.*

For a given pair of scans from the experiment, we compute the error bounds in two ways: i) We use the ground truth measurements to align the convex hulls of LiDAR point clouds in the two compared scans, then we compute the size and overlap parameters ρ and δ, and the error bound via Theorem 11.4; ii) Since both scans are now aligned with the true transformation, we can also directly compute the perturbations to the second moments $\overline{\Delta}_\lambda(\Sigma' - \Sigma) = \overline{\Delta}_\lambda(B)$, which characterizes the bound in Corollary 1. The first approach is equivalent to first computing an upper bound for $\overline{\Delta}_\lambda(\Sigma' - \Sigma)$ from δ and ρ, and then using this upper bound in Corollary 1.

For each LiDAR scan at index i, we align it with the scan at index $i + k$, and plot the computed error bounds along the trajectory for three different values of "lag" $k \in \mathbb{N}$, displayed in Fig. 13.3. Generally, with higher k the scans we compare are taken from poses further apart along the trajec-

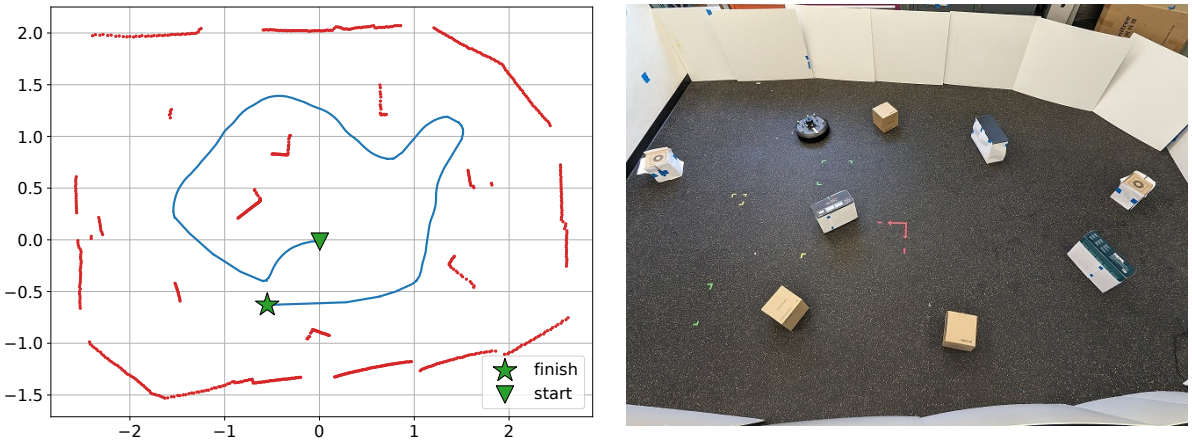

Figure 13.1: Sample LiDAR scan and trajectory (left) and image (right) of the robot from the experimental setup.

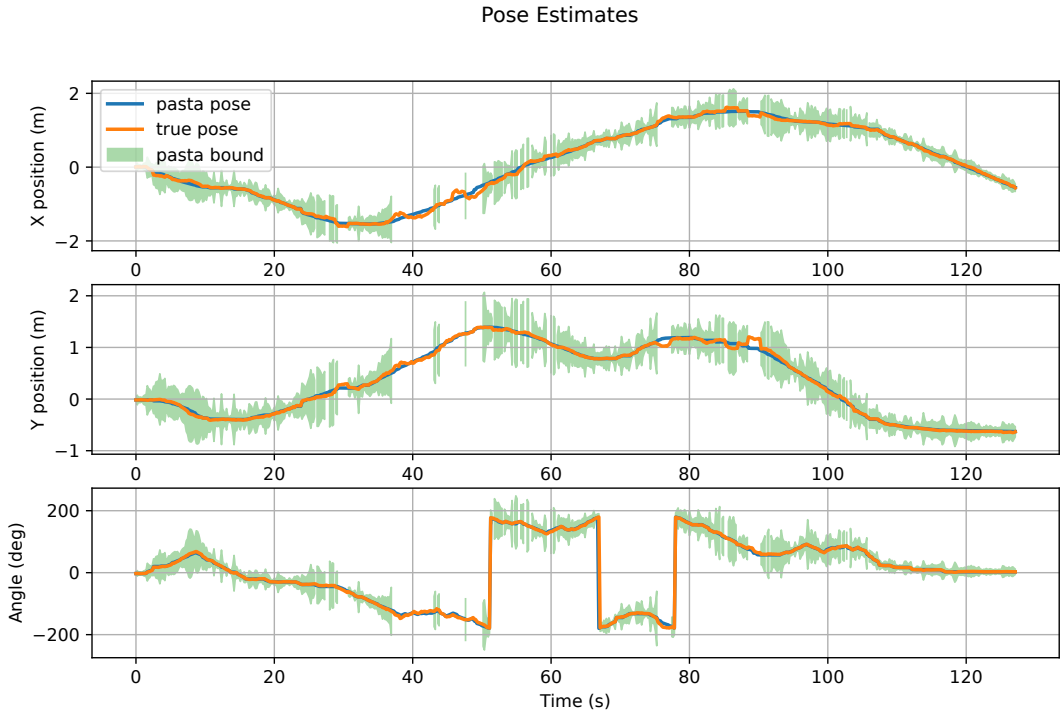

Figure 13.2: Comparison of the estimated pose vs the true pose of the robot. The green band shows the magnitude of the error bound in Corollary 1 (orange) for a lag value of 15.

EXPERIMENT SUMMARY

Lag k	Bound from δ, ρ	Bound from $\overline{\Delta}_\lambda(\Sigma' - \Sigma)$	PASTA error
1	$13.0°,\ 99.1\%$	$1.4°,\ 100\%$	$0.6°$, n/a
10	$27.9°,\ 80.5\%$	$4.4°,\ 100\%$	$1.9°$, n/a
50	$38.8°,\ 50.0\%$	$9.0°,\ 100\%$	$4.1°$, n/a

For each lag and error type, we list the mean error over the experiment and the fraction of data points that satisfy the necessary assumptions.

tory, thus more likely to be different, with the expectation of a worse bound and estimation error. Other variations naturally occur depending on the obstacles and their relative location to the robot, seen as small spikes in Fig. 13.3. We only show angular error bounds, as translation error bounds are a simple affine function of this value. For a given pair of scans, PASTA provides two candidate estimated angles separated by 180 degrees. We plot the error for the correct choice, as there are multiple ways to reliably select the correct one, and the bounds only apply to this "correct choice". Missing values in the plot correspond to scan pairs where $\min_{i,j} |\lambda_i - \lambda_j| > 2e_\Sigma$ fails to hold, violating the assumptions for our results.

We observe that the perturbation bound in Corollary 1 is very tight, and most of the difference between the bound in Theorem 11.4 and the actual estimation error comes from upper bounding $\overline{\Delta}_\lambda(\Sigma' - \Sigma)$ via knowledge of δ and ρ. As expected, larger distances between poses (as measured by the "lag" k) lead to higher error bounds. To counteract these effects, one may need a weak correspondence method that identifies common measured regions of both scans, which is a direction of current research.

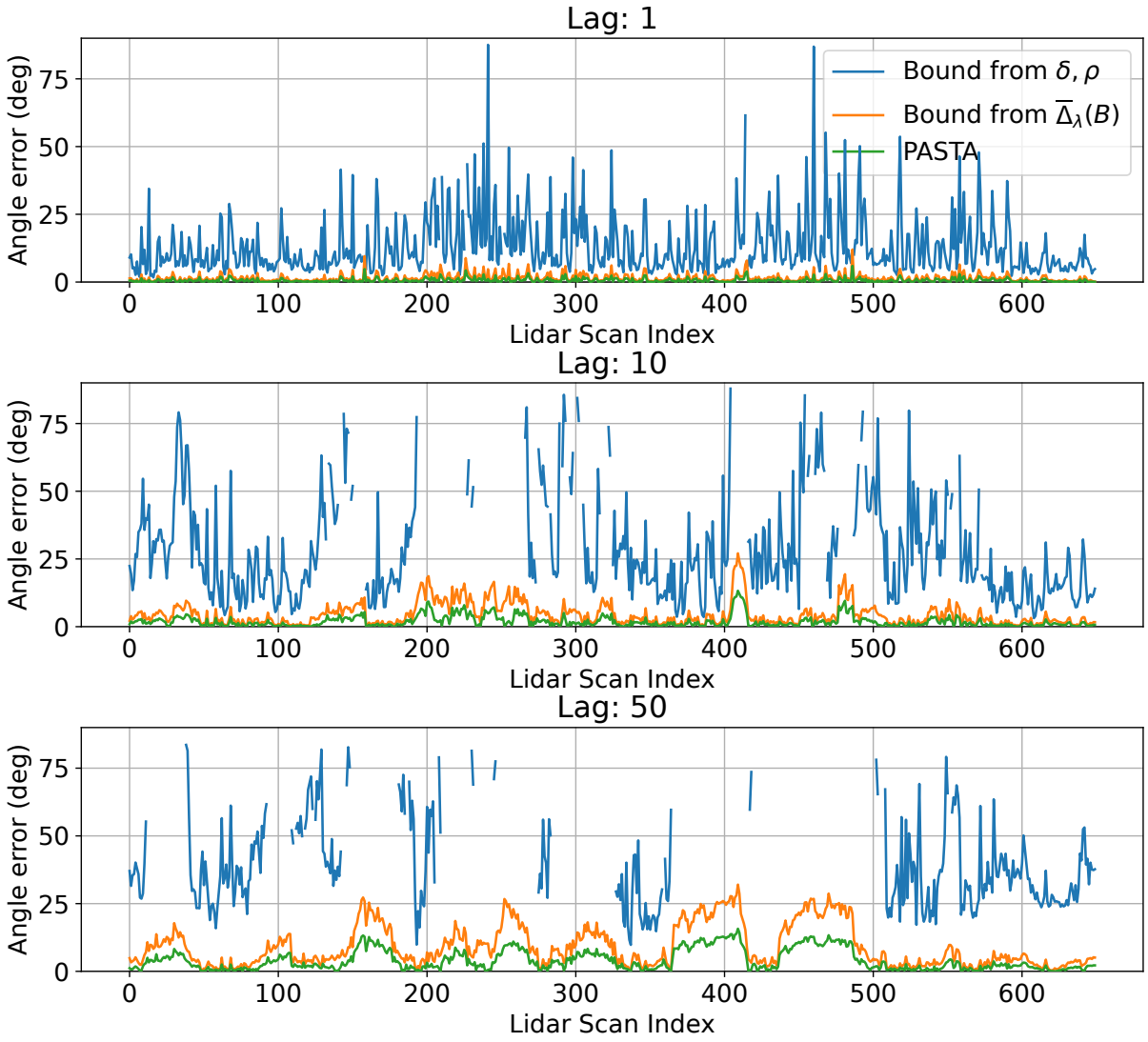

Figure 13.3: Comparison of the error bound in Corollary 1 (orange) and Theorem 11.4 (blue) and the empirical error of PASTA (green) on real LiDAR data. Increased "lag" implies larger times between compared LiDAR measurements.

CHAPTER 14

Conclusions

In this work, we presented a LiDAR localization algorithm called PASTA and derived new theoretical worst-case bounds on its estimation error. These worst-case bounds are crucial for interfacing robotic systems using LiDAR for localization with safety-critical control algorithms. We also provided experimental evidence highlighting the tightness of the bounds and where improvements could be made.

Part III

Generative Models in Closed-Loop Learning

Improvement and adoption of generative machine learning models is rapidly accelerating, as exemplified by the popularity of LLMs (Large Language Models) for text, and diffusion models for image generation. As generative models become widespread, data they generate is incorporated into shared content through the public web. This opens the question of what happens when data generated by a model is fed back to the model in subsequent training campaigns. This is a question about the stability of the training process, whether the distribution of publicly accessible content, which we refer to as "knowledge", remains stable or collapses.

Small scale empirical experiments reported in the literature show that this closed-loop training process is prone to degenerating. Models may start producing gibberish data, or sample from only a small subset of the desired data distribution (a phenomenon referred to as mode collapse). So far there has been only limited theoretical understanding of this process, in part due to the complexity of the deep networks underlying these generative models.

The aim of this work is to provide insights into this process (that we refer to as "generative closed-loop learning") by studying the learning dynamics of generative models that are fed back their own produced content in addition to their original training dataset. The sampling of many of these models can be controlled via a "temperature" parameter. Using dynamical systems tools, we show that, unless a sufficient amount of external data is introduced at each iteration, any non-trivial temperature leads the model to asymptotically degenerate. In fact, either the generative distribution collapses to a small set of outputs, or becomes uniform over a large set of outputs.

CHAPTER 15

Introduction

Generative models have exploded in popularity in recent years, primarily driven by the adoption of diffusion models [Cao24] for image generation, and so-called LLMs (Large Language Models) [Zha23] for textual generation. With this explosion, came renewed concerns about AI, especially tied to the *generative* nature of these models. Large scale neural networks underlie most of these models, including, for example, Llama 2 which is trained on 2 trillion tokens [Tou23]. As these models generate data that is published on the internet, they pollute their own training datasets with synthetic data, possibly leading to a spiraling decay of the quality of these models and of the internet by extension.

We are concerned with the setting where a generative model is iteratively trained, and the outcome of each iteration is dependent on the current data distribution encoded by the model (typically by including samples generated by the model in the training set). Serious concerns about decay of such a training process arose first in GANs (Generative Adversarial Networks), where the problem of "mode collapse" [TT20] was identified. Analogous issues seem to be a general feature of violating distributional assumptions about the training dataset, even for non-GAN models. One way of framing such violations is as "data poisoning" [BNL12], a problem that is likely to become more common, as models trained from public domain internet data are especially susceptible to data poisoning attacks [Car24]. This is also related to the notion of "distribution shift" [Koh21], although most existing work focuses on the distribution shift occurring at test time and coming from an external source. In our setting the shift occurs at *training time* and has an *internal* origin.

As this is such a new development, there is still only partial understanding of the phenomenon, and much published work is empirical in nature. In [Mar23a] and [Mar23b], the authors train image

diffusion models, iteratively including synthetic samples, and show significant degradation of the quality of the produced images. In [Shu23], it is shown, both theoretically and experimentally, that generative Gaussian models undergo degenerative collapse. A case of closed-loop learning when the sampling of the model is biased (samples may be taken closer to the mean) under a variety of synthetic data policies is studied in [Ale24]. In their results, non-degeneration could be ensured only by introducing a sufficient fraction of fresh data at each training iteration. This aligns with the results in [Ber24], where the authors establish (theoretically and experimentally) that maintaining a high enough fraction of fresh data is a sufficient condition to prevent degeneration.

Most generative models include a way to modulate their sampling probabilities through "temperature", typically as a way to make the outputs more or less random. In this work, we focus on the effect of temperature on the closed-loop learning dynamics of generative models, a perspective that received little attention so far. In particular: 1) We define a class of "generative closed-loop learning models with temperature" that captures many real-world scenarios. 2) We perform a theoretical analysis of the resulting closed-loop learning dynamics, and establish that modulating sampling with temperature leads to degeneration of the learning process. 3) We characterize the type of degeneration depending on one of three possible temperature regimes. As the models degenerate (for any amount of temperature modulation), so do their datasets, consequently losing any knowledge they originally contained, if not explicitly preserved and re-introduced. When applied to the internet, this predicts that unless a copy of the pre-generative-models Internet is preserved, eventually no model will be able to be trained effectively using the internet as a data source. Our results share some similarities with [Ale24, Ber24], and are compatible with their conclusions, but in contrast to those papers we use tools and techniques from dynamical and control systems for the analysis.

15.1 Notation

- We denote by e_i the i-th element of the standard basis of $\mathbb{R}^n$, i.e., the vector of all zeroes except for a one in its i-th entry.

- The symbol $\mathbf{1}$ denotes the vector $x \in \mathbb{R}^n$ with all elements equal to one.
- We define Δ^n as the n-dimensional probability simplex $\Delta^n = \{x \in \mathbb{R}^n \mid \sum_{i=1}^n x_i = 1, x_i \geq 0\}$, and its restriction to strictly positive probabilities as $\Delta^n_{>0}$. An element of Δ^n is called a "probability vector". The boundary of Δ^n is denoted by $\partial\Delta^n$.
- Given some $X \in \Delta^n$, we say that the random variable Y is sampled according to X, or $Y \sim X$, to mean that for all $i \in \{1, 2, \ldots, n\}$ we have $\mathrm{P}(Y = e_i) = X_i$.
- If $X(k)$ is a stochastic process, $\mathcal{F}_k$ denotes the filtration adapted to the stochastic process up to time k. We say an event happens a.s. to mean "almost surely", i.e., with probability 1 (w.p. 1).
- Unless otherwise noted, $\|\cdot\|$ denotes the usual vector 2-norm over $\mathbb{R}^n$, and $d(x, y) = \|x - y\|$ with $x, y \in \mathbb{R}^n$ is the distance between x and y. If one of the arguments is a set $\Omega \subseteq \mathbb{R}^n$, it denotes the distance from a point to that set $d(x, \Omega) = \inf_{y \in \Omega} d(x, y)$.
- The notation $f(x) \xrightarrow[x \to a]{} \Omega$, with Ω a set means $\lim_{x \to a} d(f(x), \Omega) = 0$.
- We normally use capitalized letters to denote random variables, and lower-case when they are deterministic, or when the randomness is not relevant (i.e., X vs x).
- A continuous function $\alpha : [0, a) \to \mathbb{R}_{\geq 0}$, with $a \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$, is said to be of class kappa ($\alpha \in \mathcal{K}$) if it is strictly increasing and $\alpha(0) = 0$.

CHAPTER 16

Closed-Loop Learning

We describe a generative model as a parameterized family of probability distributions over a finite set of $n \in \mathbb{N}$ possible elements $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_n\}$[1]. We denote such family as $\phi : \mathbb{R}^p \to \Delta^n$, a map from a parameter vector $w \in \mathbb{R}^p$ to a probability vector $\phi(w) \in \Delta^n$ for the elements $\mathcal{Y}$. These are the outputs that the model can generate when sampled. Without loss of generality, we identify each $\mathcal{Y}_i$ with the i-th vector of the standard basis of $\mathbb{R}^n$. These elements can be interpreted differently depending on the specific generative model, e.g., for a language model each $\mathcal{Y}_i$ could be a word, token, sentence, or sentence class from a large but finite set.

16.1 Model sampling with temperature control

For a trained generative model, letting Y be the output of the model when sampled, we denote by $\Theta = \phi(w)$ the "nominal" probability of generating each of the possible elements of $\mathcal{Y}$. Specifically, the probability of generating the i-th element corresponds to the i-th entry of the vector Θ. However, when sampled, the actual generation probabilities are filtered through a *temperature* function $\tau : \Delta^n \to \Delta^n$. Therefore, for $i \in \{1, 2, \ldots, n\}$, the sampled output $Y \in \mathcal{Y}$ satisfies:

$$P\left(Y = \mathcal{Y}_i\right) = \tau\left(\Theta\right)_i, \tag{16.1}$$

where a subscripted index i denotes the i-th vector element. For our results to hold, we require the temperature function to satisfy some assumptions (see Chapter 17). We show that these assump-

[1]This family is a subset of the set of categorical distributions over n categories. In general we do not require that the family of distributions expressible by the model is the full set of categorical distributions, which is unrealistic for very high n (for example, if the outcomes $\mathcal{Y}$ are rgb-images).

tions hold for the temperature function induced by the *softmax* operation, typically used in deep learning.

16.2 Learning process

We use the term "generative closed-loop learning", or just "closed-loop learning", to refer to a generative model trained on data that includes its own output from prior runs. When a generative model learns from its own output, the probability vector Θ becomes a stochastic process $\Theta(k)$ evolving over (discrete) time $k \in \mathbb{Z}_{\geq 0}$. We assume that a model is initially trained on some externally provided dataset of some size $\ell \in \mathbb{N}$:

$$D(\ell) = \{Y(1), Y(2), \ldots, Y(\ell)\},$$

where we use $Y(k)$ with $k \leq \ell$ to denote the externally provided data, i.e., training only starts at time $k = \ell$. Similarly, for each time $k \geq \ell$ we have a parameter vector $w(k)$ and its associated probability vector $\Theta(k) = \phi(w(k))$. Finally, let the training be represented by a (in general stochastic) function[2] f that maps a parameter vector w and training data D to a "retrained" parameter $f(w, D)$. Then, for each time $k \in \mathbb{Z}_{\geq 0}$, $k \geq \ell$, the closed-loop learning stochastic process unfolds as follows:

$$\begin{aligned} Y(k+1) &\sim \tau\left(\Theta(k)\right) \\ D(k+1) &= D(k) \cup \{Y(k+1)\} \\ w(k+1) &= f(w(k), D(k+1)) \\ \Theta(k+1) &= \phi(w(k+1)), \end{aligned} \tag{16.2}$$

where $D(k) \cup \{Y(k+1)\}$ models "adding" the generated output sample to the current set of training data[3]. For some initial dataset of ℓ samples, the process has the initial conditions $w(\ell) =$

[2]While the retraining function here takes the current parameters as an argument, it can also represent a form of retraining where the model is "reset" and trained from scratch over a new dataset by ignoring w.

[3]For notational simplicity, in (16.2), the process retrains the model after each generated sample, however our results hold even in the case where some variable but bounded number of samples $N \geq 1$ is generated and added to the dataset before retraining.

$f(w_0, D(k))$, and $\Theta(\ell) = \phi(w(\ell))$ for some $w_0 \in \mathbb{R}^p$. The recursive process (16.2) induces a probability distribution for each $\Theta(k)$. Like for the temperature function τ, in Chapter 17 we will require that the process (16.2) satisfies some general properties.

16.3 Problem statement

Given the closed-loop learning process (16.2), we want to know what are the long term properties of the probability vector $\Theta(k)$, i.e., what is the asymptotic behavior of $\Theta(k)$ as time increases? Since $\Theta(k)$ describes the probability of generated data, the asymptotic behavior of $\Theta(k)$ determines the ultimate composition of the dataset $D(k)$ as well. For example, if $\Theta(k)$ were to converge to a point independent of the initial dataset $D(\ell)$, any initial knowledge encoded by the dataset is eventually lost.

CHAPTER 17

A common class of models

In the previous section we presented an abstracted notion of closed-loop learning. We now give specific conditions on the temperature function τ and the behavior of the training algorithm represented by f and ϕ in (16.2), and show that they are realistic for common closed-loop learning models.

17.1 Temperature

We assume that the class of temperature functions τ as defined in Chapter 16 satisfies a few properties:

Assumption 17.1. *The temperature function $\tau : \Delta^n \to \Delta^n$ in* (16.2) *is assumed to satisfy the following properties:*

1. *It is continuous and strictly element-wise order preserving, i.e., for any $\theta \in \Delta^n$ and $i, j \in \{1, 2, \ldots, n\}$:*

$$\begin{aligned} \theta_i < \theta_j &\implies \tau(\theta_i) < \tau(\theta)_j \\ \theta_i = \theta_j &\implies \tau(\theta_i) = \tau(\theta_j). \end{aligned} \tag{17.1}$$

2. *Given an index set $I \subseteq \{1, 2, \ldots, n\}$, let $V_I : \Delta^n \to \mathbb{R}_{\geq 0}$ be defined as*[1]*:*

$$V_I(\theta) = \max_{i \in I} \left\{ \frac{\theta_i}{\sum_{j \in I} \theta_j} \right\} - \min_{i \in I} \left\{ \frac{\theta_i}{\sum_{j \in I} \theta_j} \right\}. \tag{17.2}$$

Then, τ satisfies exactly one of the properties:

[1] *V_I will be used as a Lyapunov function later in the analysis.*

(a) It is the identity: $\tau(\theta) = \theta$.

(b) For any index set $I \subseteq \{1, 2, \ldots, n\}$ *where* $\min_{i \in I} \theta_i > 0$ *and* $\max_{i \in I} \theta_i > \min_{i \in I} \theta_i$*:*

$$V_I(\tau(\theta)) - V_I(\theta) < 0.$$

(c) For any index set $I \subseteq \{1, 2, \ldots, n\}$ *where* $\min_{i \in I} \theta_i > 0$ *and* $\max_{i \in I} \theta_i > \min_{i \in I} \theta_i$*:*

$$V_I(\tau(\theta)) - V_I(\theta) > 0.$$

Intuitively, case 2.b represents a "contracting" τ, and case 2.c an "expanding" τ. The function V_I quantifies how close a probability vector $\theta \in \Delta^n$ is to uniform when conditioned to a specific subset of variables. Note that when an index set I includes all non-zero elements of θ, the renormalizing term $\sum_{j \in I} \theta_j$ in (17.2) is equal to one, and (17.2) reduces to $V_I(\theta) = \max_{i \in I} \theta_i - \min_{i \in I} \theta_i$.

The notion of temperature typically used in generative models satisfies the requirements listed above. In fact, this is the case for the *softmax* temperature, as we now show. Many machine learning models do not directly output a set of probabilities, but a vector of so-called *logits* $z \in \mathbb{R}^n$ that is converted into a probability vector via the $\mathrm{softmax}$ function and a positive temperature parameter $T > 0$ as follows:

$$\begin{aligned} \theta_T &= \mathrm{softmax}\left(zT^{-1}\right) \\ &= \frac{1}{\sum_{i=1}^{n} \exp\left(\frac{z_i}{T}\right)} \begin{bmatrix} \exp\left(\frac{z_1}{T}\right) & \ldots & \exp\left(\frac{z_n}{T}\right) \end{bmatrix}^\top. \end{aligned} \tag{17.3}$$

Note that τ, as defined in Chapter 16, is a map between probability vectors, but (17.3) maps logits to probabilities. Consider a logit vector $z \in \mathbb{R}^n$. While (17.3) is not a valid τ, it induces a unique map transforming $\theta = \mathrm{softmax}(z)$ (the "nominal" probabilities associated to z) to $\theta_T = \mathrm{softmax}\left(T^{-1}z\right)$ (the "temperature filtered" probabilities associated to the same z). Defining $Z =$

$\sum_{i=1}^{n} \exp(z_i)$ we have:

$$\begin{aligned}
\theta_T &= \operatorname{softmax}\left(zT^{-1}\right) \\
&= \operatorname{softmax}\left(T^{-1}\left(\begin{bmatrix}\ln(\theta_1) & \dots & \ln(\theta_n)\end{bmatrix}^\top + \mathbf{1}\ln(Z)\right)\right) \\
&= \operatorname{softmax}\left(T^{-1}\begin{bmatrix}\ln(\theta_1) & \dots & \ln(\theta_n)\end{bmatrix}^\top\right) \\
&= \frac{1}{\sum_{i=1}^{n}\theta^{\frac{1}{T}}}\begin{bmatrix}\theta_1^{\frac{1}{T}} & \dots & \theta_n^{\frac{1}{T}}\end{bmatrix}^\top,
\end{aligned} \tag{17.4}$$

where the third equality holds because $\operatorname{softmax}$ is invariant to addition of the same constant ($T^{-1}\ln(Z)$) to all input entries. This induced map $\tau(\theta) = \theta_T$ satisfies exactly one of the three requirements previously introduced. We formalize this in the following Lemma:

Lemma 17.1. *Consider the function $\tau : \Delta^n \to \Delta^n$ defined by $\tau(\theta) = \operatorname{softmax}(T^{-1}\ln(\theta))$, with $T \in \mathbb{R}_{>0}$. The function τ satisfies Assumption 17.1. In particular, it satisfies properties 2.a, 2.b, and 2.c for $T = 1$, $T > 1$, and $T < 1$ respectively.*

Proof. Since τ takes the form (17.4), and $x \mapsto x^{\frac{1}{T}}$ is a continuous strictly monotone increasing function, it is immediate that τ is also continuous (the denominator in (17.4) is always bounded away from zero) and preserves the order of the elements of θ. Further, note that if $\theta_i = 0$, then $\tau(\theta)_i = 0$.

For the case $T = 1$, it is immediate to see that τ becomes the identity function (Note that $\sum_{i=1}^{n}\theta_i = 1$).

For the cases where $T \neq 1$, let $I \subseteq \{1, 2, \dots, n\}$ be as in Assumption 17.1, then, $V_I(\tau(\theta)) < V_I(\theta)$. To see that this is true, let $M, m \in I$ be respectively the (not necessarily unique) indices of the greatest and smallest non-zero elements of θ, and consider the derivative with respect to the

temperature parameter T of $V_I(\tau(\theta))$:

$$\begin{aligned}\frac{\partial}{\partial T}V_I(\tau(\theta)) &= \frac{\partial}{\partial T}\left\{\frac{\theta_M^{\frac{1}{T}}}{\sum_{j\in I}\theta_j^{\frac{1}{T}}} - \frac{\theta_m^{\frac{1}{T}}}{\sum_{j\in I}\theta_j^{\frac{1}{T}}}\right\} \\ &= -T^{-2}\left[\frac{\theta_M^{\frac{1}{T}}}{\sum_{j\in I}\theta_j^{\frac{1}{T}}}\sum_{i\in I}\left(\frac{\theta_i^{\frac{1}{T}}}{\sum_{j\in I}\theta_j^{\frac{1}{T}}}\left(\log(\theta_M)-\log(\theta_i)\right)\right)\right. \\ &\qquad \left. - \frac{\theta_m^{\frac{1}{T}}}{\sum_{j\in I}\theta_j^{\frac{1}{T}}}\sum_{i\in I}\left(\frac{\theta_i^{\frac{1}{T}}}{\sum_{j\in I}\theta_j^{\frac{1}{T}}}\left(\log(\theta_m)-\log(\theta_i)\right)\right)\right].\end{aligned}$$

The sums are convex combinations of non-negative terms for the first and non-positive for the second summation, as $\log(\theta_M) \geq \log(\theta_i)$ and $\log(\theta_m) \leq \log(\theta_i)$ for all $i \in I$. Further, by Assumption 17.1 $\max_{i\in I}\theta_i > \min_{i\in I}\theta_i$, therefore the sums are strictly positive and strictly negative respectively, and $\frac{\partial}{\partial T}V_I(\tau(\theta)) < 0$. Then, fixing some specific temperature $\overline{T}$:

$$V_I(\tau(\theta)) - V_I(\theta) = \int_{T=1}^{\overline{T}} \frac{\partial}{\partial T}V_I(\tau(\theta))\,\mathrm{d}T, \tag{17.5}$$

where (17.5) is negative for $\overline{T} > 1$, and positive for $\overline{T} < 1$.

□

17.2 Closed-loop learning dynamics

For a training dataset $D(\ell) = \{Y(1), Y(2), \ldots, Y(\ell)\}$, "learning" a generative model is usually framed in a maximum-likelihood sense, i.e., we want to find the set of parameters $w \in \mathbb{R}^p$ whose associated probability vector $\Theta = \phi(w)$ maximizes the log-probability of the observed data:

$$\begin{aligned}w^* &= \arg\max_{w\in\mathbb{R}^p}\ \frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{j=1}^{n}\mathbb{1}_{Y(i)=\mathcal{Y}_j}\log\left(\phi(w)_j\right) \\ &= \arg\min_{w\in\mathbb{R}^p}\ -\sum_{j=1}^{n}\left(\frac{1}{\ell}\sum_{i=1}^{\ell}\mathbb{1}_{Y(i)=\mathcal{Y}_j}\right)\log\left(\phi(w)_j\right) \\ &= \arg\min_{w\in\mathbb{R}^p}\ -\sum_{j=1}^{n}\Theta_j^*\log\left(\phi(w)_j\right) \\ &= \arg\min_{w\in\mathbb{R}^p}\ H\left(\Theta^*, \phi(w)\right),\end{aligned} \tag{17.6}$$

where $\mathbb{1}_{(\cdot)}$ is the indicator function (takes value 1 if the subscript expression is true and 0 otherwise), H is the cross-entropy between probability vectors, and $\Theta^* = \frac{1}{\ell}\sum_{i=1}^{\ell} Y(i)$ is the "empirical" probability vector associated to the data D (remember that with no loss of generality we take $\mathcal{Y}_j$ to be the j-th standard basis element of $\mathbb{R}^n$). Therefore, although usually not expressed in this way, learning the model is also equivalent to minimizing the cross entropy with respect to Θ^*. Note that if ϕ is surjective (where its codomain is Δ^n), there always exists a w^* such that $\phi(w^*) = \Theta^*$ and is thus the optimal solution of (17.6).

When ϕ is surjective, $\Theta = \Theta^*$, thus we study the behavior of Θ^* under the closed-loop learning dynamics. Consider the process (16.2), and let $\Theta^*(k) = \frac{1}{k}\sum_{i=1}^{k} Y(i)$ be the empirical probability vector corresponding to the dataset at time k, $D(k)$. When a new sample $Y(k+1)$ is generated by the model, Θ^* evolves as:

$$\begin{aligned}\Theta^*(k+1) &= \frac{1}{k+1}\sum_{i=1}^{k+1} Y(i) \\ &= \frac{k}{k+1}\Theta^*(k) + \frac{1}{k+1}Y(k+1) \\ &= \Theta^*(k) + \frac{1}{k+1}\left(Y(k+1) - \Theta^*(k)\right).\end{aligned} \tag{17.7}$$

Since $Y(k+1)$ is a random variable, we perform a Martingale decomposition [LS89]:

$$\begin{aligned}\Theta^*(k+1) &= \Theta^*(k) + \frac{1}{k+1}\big(\mathbb{E}\left[Y(k+1)|\mathcal{F}_k\right] - \Theta^*(k) + Y(k+1) - \mathbb{E}\left[Y(k+1)|\mathcal{F}_k\right]\big) \\ &= \Theta^*(k) + \frac{1}{k+1}\big(\tau(\Theta(k)) - \Theta^*(k) + U(k+1)\big),\end{aligned}$$

where $U(k+1) = Y(k+1) - \mathbb{E}\left[Y(k+1)|\mathcal{F}_k\right]$ is a bounded Martingale difference sequence, and the second equality holds because $\mathbb{E}\left[Y(k+1)|\mathcal{F}_k\right] = \tau(\Theta)$.

Then, if the update function f in (16.2) is the maximum-likelihood optimization (17.6):

$$\begin{aligned}w(k+1) &= f(w(k), D(k+1)) \\ &= \arg\max_{w\in\mathbb{R}^p} H\left(\Theta^*(k+1), \phi(w)\right),\end{aligned} \tag{17.8}$$

and ϕ is surjective on Δ^n, at each step $\Theta(k) = \phi(w(k)) = \Theta^*(k)$, leading to the following dynamics:

$$\Theta(k+1) = \Theta(k) + \frac{1}{k+1}\left(\tau(\Theta(k)) - \Theta(k) + U(k+1)\right).$$

An actual model is unlikely to have enough expressivity as to represent *any* element of Δ^n, especially for very high dimension n. However, we assume the model is able to approximate a probability vector with some small finite accuracy:

Assumption 17.2. *There exists some $\delta \in \mathbb{R}_{\geq 0}$ such that for all $k \in \mathbb{Z}_{\geq 0}$ the process* (16.2) *satisfies:*

$$\|\Theta^*(k) - \Theta(k)\| = \|\Theta^*(k) - \phi(w^*(k))\| \leq \delta. \tag{17.9}$$

Under Assumption 17.2, the dynamics of $\Theta^*(k)$ obey:

$$\Theta^*(k+1) = \Theta^*(k) + \frac{1}{k+1}\left(\tau(\Theta(k)) - \Theta^*(k) + U(k+1)\right). \tag{17.10}$$

If we now define the perturbation $\varepsilon(k) = \tau(\Theta) - \tau(\Theta^*)$, recalling that a continuous function on a compact set is uniformly continuous, we have the following inequality, where η is the modulus of continuity of τ:

$$\|\varepsilon(k)\| = \|\tau(\Theta) - \tau(\Theta^*)\| \leq \eta(\delta). \tag{17.11}$$

Then, the dynamics of $\Theta^*(k)$ become a stochastic approximation (see [Bor23]) of the form:

$$\Theta^*(k+1) = \Theta^*(k) + \frac{1}{k+1}(\tau\left(\Theta^*(k)\right) - \Theta^*(k) + \varepsilon(k) + U(k+1)). \tag{17.12}$$

If we understand the behavior of (17.12), we automatically understand the behaviour of $\Theta(k)$, since by (17.9) $\Theta(k)$ is always within a distance δ from $\Theta^*(k)$.

CHAPTER 18

Asymptotic Dynamics

We now present the main results describing the asymptotic behavior of (17.12) (and hence of $\Theta(k)$ up to error δ). This asymptotic behavior is important, as it determines the long-term composition of the dataset $D(k)$, and how much it may diverge from its initial distribution. The results are different depending on which of the three conditions (2.a, 2.b, 2.c) in Assumption 17.1 is satisfied by τ, therefore we split the analysis in three different cases.

18.1 Identity temperature leads to Martingale-like behavior

This case is the most straightforward and does not require any machinery beyond standard analysis tools of stochastic processes. If condition 2.a of Assumption 17.1 is satisfied, the stochastic process (17.12) reduces to:

$$\Theta^*(k+1) = \Theta^*(k) + \frac{1}{k+1}\left(U(k+1) + \varepsilon(k)\right), \tag{18.1}$$

and we state the following formal result.

Theorem 18.1. *Consider the closed-loop learning stochastic process* (16.2)*, where τ satisfies Assumption 17.1 with property 2.a (τ is the identity), and Assumption 17.2. Then it holds that:*

$$\mathbb{E}\left[\Theta(k+1) \mid \mathcal{F}_\ell\right] = \Theta(\ell) + \sum_{i=\ell}^{k} \frac{1}{i+1}\mathbb{E}\left[\varepsilon(i) \mid \mathcal{F}_\ell\right] + e_\delta(k+1),$$

where $e_\delta : \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$ is such that $\|e_\delta(k)\| \leq \delta$. In addition, if ε is a Martingale difference sequence, there is a constant $b \in \mathbb{R}_{\geq 0}$ such that the asymptotic variance is bounded as:

$$\lim_{k\to\infty} \operatorname{var}\left(\Theta(k) - \Theta(\ell)\right) \leq b\left(\sum_{i=\ell}^{\infty}\left(\frac{1}{i+1}\right)^2 + \delta^2\right). \tag{18.2}$$

Proof. If τ is the identity function, $\Theta^* : \mathbb{Z}_{\geq 0} \to \Delta^n$ satisfies (18.1). Then, because U is a Martingale difference sequence:

$$\begin{aligned}\mathbb{E}\left[\Theta^*(k+1) \mid \mathcal{F}_k\right] &= \mathbb{E}\left[\Theta^*(k) + \frac{1}{k+1}\left(U(k+1) + \varepsilon(k)\right) \;\middle|\; \mathcal{F}_k\right] \\ &= \Theta^*(k) + \frac{1}{k+1}\varepsilon(k),\end{aligned}$$

and by the tower property of expectation:

$$\begin{aligned}\mathbb{E}\left[\Theta^*(k+1) \mid \mathcal{F}_{k-1}\right] &= \mathbb{E}\left[\mathbb{E}\left[\Theta^*(k+1) \mid \mathcal{F}_k\right] \mid \mathcal{F}_{k-1}\right] \\ &= \mathbb{E}\left[\Theta^*(k) + \frac{1}{k+1}\varepsilon(k) \;\middle|\; \mathcal{F}_{k-1}\right] \\ &= \Theta^*(k-1) + \frac{1}{k}\varepsilon(k-1) + \frac{1}{k+1}\mathbb{E}\left[\varepsilon(k) \mid \mathcal{F}_{k-1}\right].\end{aligned}$$

Finally, by recursion we arrive at:

$$\mathbb{E}\left[\Theta^*(k+1) \mid \mathcal{F}_\ell\right] = \Theta^*(\ell) + \sum_{i=\ell}^{k} \frac{1}{i+1}\mathbb{E}\left[\varepsilon(i) \mid \mathcal{F}_\ell\right],$$

and the statement is obtained once we take into account that $\Theta(k)$ is always within a distance δ from $\Theta^*(k)$.

In addition, if ε is a Martingale difference sequence, the whole $\Theta^*(k)$ process reduces to a sum of bounded Martingale differences, and it immediately follows that its variance is bounded by a term of the order of the converging sum $\sum_{i=\ell}^{\infty}(i+1)^{-2}$. □

Theorem 18.1 states that in this case $\Theta(k)$ is essentially a Martingale biased by the perturbation ε. In general the asymptotic behavior can be arbitrary as it is dominated by the behavior of $\varepsilon(k)$. However, if ε is also a Martingale difference sequence, with probability one $\Theta(k)$ will only drift a finite amount from its initial value. The magnitude of this drift depends on the converging sum $\sum_{i=\ell}^{\infty}(i+1)^{-2}$, which is smaller the greater the initial dataset size ℓ is. This shows that in this case the initial data distribution of the dataset is not necessarily lost. However, this condition requires hard to verify assumptions on the training behavior (captured by $\varepsilon(k)$) and, if the training dataset is shared by multiple generative models, that no model is biasing their own sampling via temperature. We consider this especially unlikely for data on the public web. From a control

perspective, the behavior with identity temperature is similar to that of a marginally stable system, and any arbitrarily small perturbation $\varepsilon(k)$ can destabilize it.

18.2 High temperature leads to uniformly generated data

While the analysis of the previous case is relatively straightforward, we need to introduce additional machinery for the remaining two. Let us define a vector field $F : \Delta^n \to T\Delta^n$ over the probability simplex as $F(\theta) = \tau(\theta) - \theta$. The behavior of stochastic approximations in the long-term approaches that of a continuous-time ODE (ordinary differential equation) or differential inclusion (see [Bor23]). Thus, under our assumptions, the limit sets of $\Theta^*(k)$ in (17.12) are determined by the attractors of:

$$\dot{\theta}(t) = F(\theta(t)) + \varepsilon(t). \tag{18.3}$$

Then, we introduce two families of sets, parameterized by $a \in \mathbb{R}_{\geq 0}$ and an index set $I \subseteq \{1, 2, \ldots, n\}$, that will be used to characterize the attractors and basins of attraction of (18.3):

$$\begin{aligned} \underline{\Omega}_I(a) &= \left\{\theta \in \Delta^n \;\middle|\; V_I(\theta) \leq a\right\} \\ \overline{\Omega}_I(a) &= \left\{\theta \in \Delta^n \mid \min_{i \in I} \theta_i \leq a\right\}. \end{aligned} \tag{18.4}$$

With respect to the subset of variables indexed by I, the set $\underline{\Omega}_I(a)$ is a compact neighborhood of the uniform probability vector (all θ_i are equal), while $\overline{\Omega}_I(a)$ is a compact neighborhood of the boundary of the probability simplex (at least one θ_i is zero). We can now state the following lemma:

Lemma 18.1. *Consider the continuous-time ODE:*

$$\dot{\theta}(t) = F(\theta(t)) + \varepsilon(t), \tag{18.5}$$

where $\theta \in \Delta^n$, $\|\varepsilon\| \leq \eta \in \mathbb{R}_{\geq 0}$, and $F(\theta) = \tau(\theta) - \theta$. Let $\tau : \Delta^n \to \Delta^n$ satisfy Assumption 17.1 and property 2.b (contractive τ). Then, for any index set $I \subseteq \{1, 2, \ldots, n\}$ there exists $\kappa \in \mathcal{K}$ such that for any $t_0 \in \mathbb{R}$:

$$\theta(t_0) \notin \overline{\Omega}_I(\kappa(\eta)) \tag{18.6}$$

implies:

$$\theta(t) \xrightarrow[t\to\infty]{} \underline{\Omega}_I(\kappa(\eta)). \tag{18.7}$$

Proof. Consider a point $\theta \in \Delta^n$, and an index set $I \subseteq \{1, 2, \ldots, n\}$ where $\min_{i\in I} \theta_i > 0$ (note that the lemma only makes a claim for index sets such that $\theta(t_0) \notin \overline{\Omega}_I(\kappa(\eta))$, which satisfy this condition). Let $M, m \in I$ be the (not necessarily unique) indices of the greatest and smallest elements of $\{\theta_i \mid i \in I\}$. The time derivative of V_I at some point θ is:

$$\begin{aligned}
\dot{V}_I(\theta) &= \frac{\mathrm{d}}{\mathrm{d}t}\left\{\left(\sum_{i\in I}\theta_i\right)^{-1}\left(\max_{i\in I}\theta_i - \min_{i\in I}\theta_i\right)\right\} \\
&= -\left(\sum_{i\in I}\theta_i\right)^{-2}\left(\sum_{i\in I}\dot{\theta}_i\right)(\theta_M - \theta_m) + \left(\sum_{i\in I}\theta_i\right)^{-1}\left(\dot{\theta}_M - \dot{\theta}_m\right) \\
&\leq \frac{\sum_{i\in I}\tau(\theta)_i}{\sum_{i\in I}\theta_i}\left(\frac{\tau(\theta)_M - \tau(\theta)_m}{\sum_{i\in I}\tau(\theta)_i} - \frac{\theta_M - \theta_m}{\sum_{i\in I}\theta_i}\right) + b\eta \\
&\leq c\left(V_I(\tau(\theta)) - V_I(\theta)\right) + b\eta,
\end{aligned}$$

where the second equality holds because τ preserves the indices of the maximum and minimum elements of θ by its order preserving property, guaranteeing that the derivative of V_I is well-defined. In the last two inequalities $b, c \in \mathbb{R}_{>0}$ are finite positive constants, as the sums that appear are always positive and bounded from above and below.

We now seek to establish that for η small enough, there is some $a \in \mathbb{R}_{\geq 0}$ such that the set difference $\mathcal{B}(a) = \Delta^n \setminus (\underline{\Omega}_I(a) \cup \overline{\Omega}_I(a))$ is a basin of attraction for $\underline{\Omega}_I(a)$. Remember that by property 2.b in Assumption 17.1, $V_I(\tau(\theta)) - V_I(\theta) < 0$ over points that lie in $\mathcal{B}(0)$ (i.e., points that are not the uniform probability vector over I and with no zero elements). Consider the closure of the set $\mathcal{B}(a)$ for some $a > 0$, and let us define the "worst-case" decrease of V_I for points in $\mathrm{clo}(\mathcal{B}(a))$ as:

$$\beta(a) = \sup_{\theta\in\mathrm{clo}(\mathcal{B}(a))}\left\{V_I(\tau(\theta)) - V_I(\theta)\right\}. \tag{18.8}$$

By continuity, and because for $a > 0$, the set $\mathcal{B}(a)$ always excludes an open neighborhood of the region where $V_I(\tau(\theta)) - V_I(\theta) = 0$, we have that $\beta(a) < 0$. Then, for any $\mathcal{B}(a) \neq \varnothing$, as long as $\eta < -cb^{-1}\beta(a)$, the term $\dot{V}_I(\theta)$ is strictly negative for all $\theta \in \mathcal{B}(a)$.

Observe that $\beta(\cdot)$ is decreasing, since $\mathcal{B}(a_2) \subseteq \mathcal{B}(a_1)$ for $a_2 > a_1$, and $\beta(0) = 0$. Then, we can define $\kappa \in \mathcal{K}$ as any strictly increasing continuous function such that:

$$\kappa(\eta) > \inf \left\{ a \in \mathbb{R}_{\geq 0} \;\middle|\; \eta < -cb^{-1}\beta(a) \right\}, \tag{18.9}$$

for any η where this $\inf$ is finite, which is guaranteed for any η smaller than some finite threshold. Then, any initial condition in the basin $\theta(t_0) \in \mathcal{B}(\kappa(\eta))$ guarantees that $\dot{V}_I(\theta) < 0$, and $\theta(t)$ will converge to $\underline{\Omega}(\kappa(\eta))$.

□

This lemma essentially claims that the solutions of the limiting ODE (18.3) over a subset of indices whose elements are sufficiently away from zero converge to a neighborhood of the uniform probability vector conditioned over that subset of indices. This key result allows us to then claim the following theorem:

Theorem 18.2. *Consider the closed-loop learning stochastic process* (16.2)*, where τ satisfies Assumption 17.1 with property 2.b (τ is contractive), and Assumption 17.2. Then, for any index set $I \subseteq \{1, 2, \ldots, n\}$ there exists $\sigma \in \mathcal{K}$ such that:*

$$\Theta(k) \xrightarrow[k\to\infty]{} \underline{\Omega}_I(\sigma(\delta)) \cup \overline{\Omega}_I(\sigma(\delta)) \qquad \textit{w.p. 1.} \tag{18.10}$$

Further, for a given time $k_0 \in \mathbb{Z}_{\geq 0}$:

$$\mathrm{P}\left(\Theta(k) \xrightarrow[k\to\infty]{} \underline{\Omega}_I(\sigma(\delta)) \;\middle|\; \Theta(k_0) \notin \overline{\Omega}_I(\sigma(\delta)) \right) \xrightarrow[k_0\to\infty]{} 1.$$

Proof. Under these conditions, $\Theta^*(k)$ satisfies the stochastic approximation (17.10). Then, by Corollary 4 (Chapter 5) of [Bor23], the iterates of $\Theta^*(k)$ converge a.s. to a closed connected internally chain transitive invariant set of the continuous-time differential inclusion:

$$\dot{\theta}(t) \in \{ x \in \Delta^n \mid \|x - F(\theta(t))\| \leq \eta(\delta) \}, \tag{18.11}$$

where $F(\theta) = \tau(\theta) - \theta$.

Lemma 18.1 characterizes the solutions of (18.11), thus, letting $a = \kappa(\eta(\delta))$, any solution that enters $\mathcal{B}(a) = \Delta^n \setminus (\underline{\Omega}_I(a) \cup \overline{\Omega}_I(a))$ will converge to $\underline{\Omega}_I(a)$. Then, the set $\mathcal{L}$ of limit points of (18.11) is contained in:

$$\mathcal{L} \subseteq \underline{\Omega}_I(a) \cup \overline{\Omega}_I(a), \tag{18.12}$$

In turn, any internally chain transitive set $\mathcal{A}$ of (18.11) is contained in the closure of the limit set $\mathcal{L}$:

$$\mathcal{A} \subseteq \text{clo}(\mathcal{L}) \subseteq \text{clo}\left(\underline{\Omega}_I(a) \cup \overline{\Omega}_I(a)\right) = \underline{\Omega}_I(a) \cup \overline{\Omega}_I(a),$$

so that w.p. 1: $\Theta^*(k) \xrightarrow[k\to\infty]{} \mathcal{A} \subseteq \underline{\Omega}_I(a) \cup \overline{\Omega}_I(a)$. Finally, because $\|\Theta(k) - \Theta^*(k)\| \leq \delta$, we know that

$$\Theta(k) \xrightarrow[k\to\infty]{} \underline{\Omega}_I(\sigma(\delta)) \cup \overline{\Omega}_I(\sigma(\delta)),$$

with $\sigma(\delta) = a + 2\delta = \kappa(\eta(\delta)) + 2\delta$.

Because any point $\theta \notin \overline{\Omega}_I(a)$ either is in $\underline{\Omega}_I(a)$, or is in $\mathcal{B}(a)$, which is an open basin of attraction for $\underline{\Omega}_I(a)$, the last result holds by [YB19, Theorem III.2], noting that the stochastic recursion (17.10) satisfies assumptions A1-3. In fact, the theorem gives explicit bounds for this probability. □

The result essentially states that as long as the learning model is sufficiently powerful, and the training sufficiently good (low δ), every set of elements of Θ will either converge to a neighborhood of the uniform probability vector conditioned over that subset, or at least one of its elements remains trapped close to zero. The reason this second possibility can occur, is that the vector field induced by the temperature function may vanish at the boundary $\partial\Delta^n$, so that some small perturbation ε over the process can keep it trapped. However, the greater the size of the initial dataset, the higher the probability of the process converging towards the uniform probability.

Either way, regardless of the size of the initial dataset (for small δ, and iterating the result over all sets I), any information it originally contained is lost as $k \to \infty$. Some subset of output probabilities will approach zero, and the rest will approach their (conditioned) uniform distribution.

In summary, in the limit, as k increases, the set of possible outputs is partitioned into the outcomes that will (almost) never be generated, and the outcomes that will be (almost) uniformly generated.

Remark 6. *While we are considering the setting where there is only a fixed amount of external initial data $D(\ell)$, our results hold even when some limited amount of external data is introduced at each training iteration.*[1] *To see this, let $\lambda \in [0, 1]$ be the fraction of external data we introduce at each time step. Then* (17.12) *becomes:*

$$\Theta^*(k+1) = \Theta^*(k) + \frac{1}{k+1}\Big(\tau\left(\Theta^*(k)\right) - \Theta^*(k) + \\ + \lambda\left(\widetilde{Y}(k) - \tau\left(\Theta^*(k)\right) - \Theta^*(k)\right) + \varepsilon(k) + U(k+1)\Big),$$

where $\widetilde{Y}(k)$ is the external data point at time k. Because $\widetilde{Y}$ is bounded, the term $\lambda(\widetilde{Y}(k) - \tau\left(\Theta^*(k)\right) - \Theta^*(k))$ is bounded and can be absorbed into $\varepsilon(k)$.

Remark 7. *It may be the case that for very high dimensional outputs ($n >> 1$), the assumption that δ is sufficiently small for every output probability is unrealistic. However, the assumption may still hold over a "coarse-grained" model, where we group outputs $\{\mathcal{Y}_1, \ldots, \mathcal{Y}_n\}$ into a set of $m < n$ categories $\left\{\widehat{\mathcal{Y}}_1, \ldots, \widehat{\mathcal{Y}}_m\right\}$. In this case the result would reduce to some categories disappearing, and others appearing uniformly randomly as $k \to \infty$.*

18.3 Low temperature leads to mode collapse

The low temperature case is identical to the high temperature one, but with the roles of $\underline{\Omega}$ and $\overline{\Omega}$ swapped, so we only state the corresponding theorem. In the proof, the direction of the Lyapunov inequalities is swapped and the sign inverted.

Theorem 18.3. *Consider the closed-loop learning stochastic process* (16.2)*, where τ satisfies Assumption 17.1 with property 2.c (τ is expanding), and Assumption 17.2. Then, for any index set*

[1] *These two scenarios are analogous to the "synthetic augmentation loop" and "fresh data loop" in [Ale24]. In the first one, the amount of external data is fixed at the start, so that over time the proportion of synthetic data dominates the dataset. In the second one, some proportion λ of external data is introduced at each step (this may be additional copies of samples from the initial dataset), guaranteeing that the proportion of synthetic data is always less than $1 - \lambda$.*

$I \subseteq \{1, 2, \ldots, n\}$ *there exists* $\sigma \in \mathcal{K}$ *such that:*

$$\Theta(k) \xrightarrow[k\to\infty]{} \underline{\Omega}_I(\sigma(\delta)) \cup \overline{\Omega}_I(\sigma(\delta)) \qquad \textit{w.p. 1.} \tag{18.13}$$

Further, for a given time $k_0 \in \mathbb{Z}_{\geq 0}$*:*

$$\mathrm{P}\left(\Theta(k) \xrightarrow[k\to\infty]{} \overline{\Omega}_I(\sigma(\delta)) \;\middle|\; \Theta(k_0) \notin \underline{\Omega}_I(\sigma(\delta))\right) \xrightarrow[k_0\to\infty]{} 1.$$

Just like for the high temperature case, any information in the original dataset is lost, with data generated by the asymptotic behavior of $\Theta(k)$ dominating the dataset. Unlike the high temperature case, with high probability Θ will converge to a region where most outputs have very low probability mass, and only a few outputs are likely to be sampled. In the limit of $\delta \to 0$, for almost every initial condition the generative probabilities of every output approach zero except for a single output element, that will completely dominate the dataset.

CHAPTER 19

Conclusions

We have shown that when a generative model is trained on the data it generates, and this generation is biased by temperature (no matter how small the biasing), there is a dichotomy between the accuracy of the learning model and preserving the initial distribution of the dataset unless that initial dataset is preserved and re-injected purposefully. A model capable of accurately reproducing the distribution of a training dataset (low δ in (17.9)) will inevitably degenerate into never producing some outputs and producing the rest uniformly randomly.

Our theoretical analysis adds to the increasing concern about data self-ingestion, especially in the current age where large scale deep networks are trained on data scraped from the internet, and data generated by these models inevitably finds its way back to their training processes.

Part IV

Appendix

APPENDIX A

Fast Computation of Simplex Moments

The pose estimation method presented in Part 8 relies on the computation of first and second moments as *integral* quantities over a set, rather than a summation over a set of points. In the case of sets H_1 and H_2 defined as convex hulls of finite collections of points, the resulting sets are *polytopes*, and as such can be partitioned into a set of simplices. The first and second moments can then be computed via weighted summation of the moments of the separate simplices. The first and second moments of a simplex can be easily computed as linear and quadratic functions, respectively, of the vertex coordinates, making fast and exact computation of them feasible.

A.1 Auxiliary results

We begin by introducing an auxiliary result that is necessary for the following theorem.

Lemma A.1. *For all $k, a, b \in \mathbb{N}$ and $x_1, x_2, \ldots, x_{k-1} \in \mathbb{R}$ the following holds:*

$$\int_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} x_k^a \left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^b \mathrm{d}x_k = \begin{cases} \frac{a!b!}{(b+a+1)!} \left(1 - \sum_{i=1}^{k-2} x_i - x_{k-1}\right)^{b+a+1}, & \text{if } k > 1 \\ \frac{a!b!}{(b+a+1)!}, & \text{if } k = 1. \end{cases} \tag{A.1}$$

Proof. We first prove the case for $k > 1$. The proof follows directly from integration by parts.

First, note that we can write:

$$\begin{aligned}
&\int_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} x_k^a \left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^b \mathrm{d}x_k \\
&= \left[-\frac{x_k^a \left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^{b+1}}{b+1}\right]_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} + \frac{a}{b+1} \int_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} x_k^{a-1} \left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^{b+1} \mathrm{d}x_k \\
&\quad = \frac{a}{b+1} \int_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} x_k^{a-1} \left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^{b+1} \mathrm{d}x_k.
\end{aligned} \tag{A.2}$$

Then, repeatedly integrating by parts until the exponent on x_k is zero, we arrive at:

$$\begin{aligned}
&\int_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} x_k^a \left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^b \mathrm{d}x_k \\
&\quad = \frac{a(a-1)\dots 1}{(b+1)(b+2)\dots(b+a)} \int_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} \left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^{b+a} \mathrm{d}x_k \\
&\quad = \frac{a(a-1)\dots 1}{(b+1)(b+2)\dots(b+a)} \left[-\frac{\left(1 - \sum_{i=1}^{k-1} x_i - x_k\right)^{b+a+1}}{b+a+1}\right]_{x_k=0}^{1-\sum_{i=1}^{k-1} x_i} \\
&\quad = \frac{a(a-1)\dots 1}{(b+1)(b+2)\dots(b+a+1)} \left(1 - \sum_{i=1}^{k-1} x_i\right)^{b+a+1} \\
&\quad = \frac{a!b!}{(b+a+1)!} \left(1 - \sum_{i=1}^{k-2} x_i - x_{k-1}\right)^{b+a+1}.
\end{aligned} \tag{A.3}$$

Following the same computations, we can prove the case for $k = 1$. □

We now prove a closed form expression for a family of multidimensional integrals. This expression will be used to calculate a set of coefficients used in the computation of the moments of a simplex.

Theorem A.1. *For all $k \in \mathbb{N}$, and for all $n_1, n_2, \dots, n_k \in \mathbb{N}$, the following holds:*

$$\int_{x_1=0}^{1} \int_{x_2=0}^{1-x_1} \cdots \int_{x_k=0}^{1-\sum_{i=0}^{k-1} x_i} x_1^{n_1} x_2^{n_2} \dots x_k^{n_k} \, \mathrm{d}x_k \, \mathrm{d}x_{k-1} \dots \mathrm{d}x_1 = \frac{\prod_{i=0}^{k} n_i!}{(k + \sum_{i=0}^{k} n_i)!}. \tag{A.4}$$

Proof. We first note that we can rewrite the integral as:

$$\int_{x_1=0}^{1} x_1^{n_1} \int_{x_2=0}^{1-x_1} x_2^{n_2} \cdots \int_{x_k=0}^{1-\sum_{i=0}^{k-1} x_i} x_k^{n_k} \, \mathrm{d}x_k \, \mathrm{d}x_{k-1} \ldots \mathrm{d}x_1. \tag{A.5}$$

Then, we apply Lemma A.1 to the inner integral [1] and obtain:

$$\begin{aligned}
&\int_{x_1=0}^{1} x_1^{n_1} \int_{x_2=0}^{1-x_1} x_2^{n_2} \cdots \int_{x_k=0}^{1-\sum_{i=0}^{k-1} x_i} x_k^{n_k} \, \mathrm{d}x_k \, \mathrm{d}x_{k-1} \ldots \mathrm{d}x_1 \\
&= \int_{x_1=0}^{1} x_1^{n_1} \int_{x_2=0}^{1-x_1} x_2^{n_2} \cdots \int_{x_k=0}^{1-\sum_{i=0}^{k-2} x_i} x_{k-1}^{n_{k-1}} \frac{n_k!}{(n_k+1)!} \left(1 - \sum_{i=1}^{k-2} x_i - x_{k-1}\right)^{n_k+1} \mathrm{d}x_{k-1} \ldots \mathrm{d}x_1 \\
&= \frac{n_k!}{(n_k+1)!} \int_{x_1=0}^{1} x_1^{n_1} \int_{x_2=0}^{1-x_1} x_2^{n_2} \cdots \int_{x_k=0}^{1-\sum_{i=0}^{k-2} x_i} x_{k-1}^{n_{k-1}} \left(1 - \sum_{i=1}^{k-2} x_i - x_{k-1}\right)^{n_k+1} \mathrm{d}x_{k-1} \ldots \mathrm{d}x_1.
\end{aligned} \tag{A.6}$$

Applying the lemma repeatedly until we exhaust the integrals we arrive at:

$$\begin{aligned}
&\int_{x_1=0}^{1} x_1^{n_1} \int_{x_2=0}^{1-x_1} x_2^{n_2} \cdots \int_{x_k=0}^{1-\sum_{i=0}^{k-1} x_i} x_k^{n_k} \, \mathrm{d}x_k \, \mathrm{d}x_{k-1} \ldots \mathrm{d}x_1 \\
&\quad = \frac{n_k!}{(n_k+1)!} \times \frac{n_{k-1}!(n_k+1)!}{(n_{k-1}+n_k+2)!} \times \cdots \times \frac{n_1!(n_2+\cdots+n_k+(k-1))!}{(n_1+n_2+\cdots+n_k+k)!} \\
&\quad = \frac{n_1! n_2! \ldots n_k!}{(n_1+n_2+\cdots+n_k+k)!} \\
&\quad = \frac{\prod_{i=1}^{k} n_i!}{(k+\sum_{i=1}^{k} n_i)!}.
\end{aligned} \tag{A.7}$$

□

In addition, to simplify some of the following notation, we define:

Definition A.1. *For any $k \in \mathbb{N}$, and $n_1, n_2, \ldots, n_k \in \mathbb{N}$ we define $M_{n_1,n_2,\ldots,n_k}$ as:*

$$M(n_1, n_2, \ldots, n_k) = \frac{\prod_{i=1}^{k} n_i!}{(k+\sum_{i=1}^{k} n_i)!}. \tag{A.8}$$

[1] Note that $x_k^{n_k} = x_k^{n_k} \left(1 - \sum_{i=0}^{k-1} x_i - x_k\right)^0$.

A.2 Closed-form expressions for moments

We can now proceed with the computation of the moments of a simplex. The computation relies on a change of coordinates that will be used to compute the necessary integrals. Consider a simplex $S \subset \mathbb{R}^n$, determined by $n+1$ vertices $v_0, v_1 \ldots, v_n \in \mathbb{R}^n$. Let us introduce the change of variables $g : \mathbb{R}^n \to \mathbb{R}^n$:

$$\begin{aligned} g(x) &= g(x_1, x_2, \ldots, x_n) \\ &= v_0 + x_1(v_1 - v_0) + x_2(v_2 - v_0) + \cdots + x_n(v_n - v_0) \\ &= \begin{bmatrix} z_1 & z_2 & \ldots z_n \end{bmatrix}^T \\ &= z. \end{aligned} \tag{A.9}$$

Then, defining $V = \begin{bmatrix} v_1 - v_0 & v_2 - v_0 & \ldots & v_n - v_0 \end{bmatrix}$, the Jacobian of g is:

$$\frac{\partial g}{\partial x}(x) = V. \tag{A.10}$$

Then, any integral over S of some function f can be rewritten as:

$$\begin{aligned} \int_{z \in S} f(z)\, \mathrm{d}\mu &= \int_{x_1=0}^{1} \int_{x_2=0}^{1-x_1} \cdots \int_{x_n}^{1-\sum_{i=1}^{n-1} x_i} f(v_0 + Vx) \det(V)\, \mathrm{d}x_n\, \mathrm{d}x_{n-1} \ldots \mathrm{d}x_1 \\ &= \det(V) \int_{x_1=0}^{1} \int_{x_2=0}^{1-x_1} \cdots \int_{x_n}^{1-\sum_{i=1}^{n-1} x_i} f(v_0 + Vx)\, \mathrm{d}x_n\, \mathrm{d}x_{n-1} \ldots \mathrm{d}x_1. \end{aligned} \tag{A.11}$$

A.2.1 Volume

The volume of a simplex is:

$$\int_{z \in S} \mathrm{d}\mu = \det(V) \int_{x_1=0}^{1} \int_{x_2=0}^{1-x_1} \cdots \int_{x_n}^{1-\sum_{i=1}^{n-1} x_i} \mathrm{d}x_n\, \mathrm{d}x_{n-1} \ldots \mathrm{d}x_1. \tag{A.12}$$

Then, we can apply Theorem A.1:

$$\begin{aligned} \det(V) \int_{x_1=0}^{1} \int_{x_2=0}^{1-x_1} \cdots \int_{x_n}^{1-\sum_{i=1}^{n-1} x_i} \mathrm{d}x_n\, \mathrm{d}x_{n-1} \ldots \mathrm{d}x_1 &= \det(V) M(0, 0, \ldots, 0) \\ &= \frac{\prod_{i=1}^{n} 0!}{(n + \sum_{i=1}^{n} 0)!} \\ &= \frac{1}{n!} \det(V). \end{aligned} \tag{A.13}$$

A.2.2 First moment

The first moment of a simplex is:

$$\begin{aligned}
\frac{\int_{z\in S} z\,\mathrm{d}\mu}{\int_{z\in S}\mathrm{d}\mu} &= \frac{n!}{\det(V)}\det(V)\int_{x_1=0}^{1}\int_{x_2=0}^{1-x_1}\cdots\int_{x_n}^{1-\sum_{i=1}^{n-1}x_i}(v_0+Vx)\,\mathrm{d}x_n\,\mathrm{d}x_{n-1}\ldots\mathrm{d}x_1 \\
&= n!\big(v_0M(0,\ldots,0)+(v_1-v_0)M(1,\ldots,0) \\
&\qquad +(v_2-v_0)M(0,1,\ldots,0)+\cdots+(v_n-v_0)M(0,\ldots,1)\big) \\
&= n!\left(\frac{1}{n!}v_0+\frac{1}{(n+1)!}\sum_{i=1}^{n}(v_i-v_0)\right) \\
&= v_0+\frac{1}{n+1}\sum_{i=1}^{n}(v_i-v_0) \\
&= \frac{1}{n+1}\sum_{i=0}^{n}v_i.
\end{aligned} \tag{A.14}$$

A.2.3 Second moment

The second moment of a simplex, relative to a point $q\in\mathbb{R}^n$ is:

$$\begin{aligned}
&\frac{\int_{z\in S}(z-q)(z-q)^T\,\mathrm{d}\mu}{\int_{z\in S}\mathrm{d}\mu} \\
&\quad= \frac{n!}{\det(V)}\det(V)\int_{x_1=0}^{1}\int_{x_2=0}^{1-x_1}\cdots\int_{x_n}^{1-\sum_{i=1}^{n-1}x_i}(v_0+Vx-q)(v_0+Vx-q)^T\,\mathrm{d}x_n\,\mathrm{d}x_{n-1}\ldots\mathrm{d}x_1 \\
&\quad= n!\int_{x_1=0}^{1}\int_{x_2=0}^{1-x_1}\cdots\int_{x_n}^{1-\sum_{i=1}^{n-1}x_i}(v_0+Vx-q)(v_0+Vx-q)^T\,\mathrm{d}x_n\,\mathrm{d}x_{n-1}\ldots\mathrm{d}x_1 \\
&\quad= n!\int_{x_1=0}^{1}\int_{x_2=0}^{1-x_1}\cdots\int_{x_n}^{1-\sum_{i=1}^{n-1}x_i}\begin{bmatrix}v_0-q & V\end{bmatrix}\begin{bmatrix}1\\ x\end{bmatrix}\begin{bmatrix}1 & x^T\end{bmatrix}\begin{bmatrix}(v_0-q)^T\\ V^T\end{bmatrix}\mathrm{d}x_n\,\mathrm{d}x_{n-1}\ldots\mathrm{d}x_1 \\
&\quad= n!\begin{bmatrix}v_0-q & V\end{bmatrix}\int_{x_1=0}^{1}\int_{x_2=0}^{1-x_1}\cdots\int_{x_n}^{1-\sum_{i=1}^{n-1}x_i}\begin{bmatrix}1 & x^T\\ x & xx^T\end{bmatrix}\mathrm{d}x_n\,\mathrm{d}x_{n-1}\ldots\mathrm{d}x_1\begin{bmatrix}(v_0-q)^T\\ V^T\end{bmatrix} \\
&\quad= \begin{bmatrix}v_0-q & V\end{bmatrix}K\begin{bmatrix}(v_0-q)^T\\ V^T\end{bmatrix},
\end{aligned} \tag{A.15}$$

where the matrix $K \in \mathbb{R}^{(n+1)\times(n+1)}$ is defined as:

$$K = n! \int_{x_1=0}^{1} \int_{x_2=0}^{1-x_1} \cdots \int_{x_n}^{1-\sum_{i=1}^{n-1} x_i} \begin{bmatrix} 1 & x^T \\ x & xx^T \end{bmatrix} \mathrm{d}x_n \, \mathrm{d}x_{n-1} \ldots \mathrm{d}x_1. \tag{A.16}$$

Exploiting Theorem A.1, the entries of K follow the pattern:

$$\begin{aligned}
k_{0,0} &= n! M(0, \ldots, 0) = n! \frac{1}{n!} = 1 \\
k_{i,0} &= k_{0,i} = n! M(0, \ldots, 1, \ldots, 0) = n! \frac{1}{(n+1)!} = \frac{1}{n+1}, \qquad i > 0 \\
k_{i,j} &= n! M(0, \ldots, 1, \ldots, 1, \ldots, 0) = n! \frac{1}{(n+2)!} = \frac{1}{(n+1)(n+2)}, \qquad i, j > 1,\ i \neq j \\
k_{i,i} &= n! M(0, \ldots, 2, \ldots, 0) = n! \frac{2}{(n+2)!} = \frac{2}{(n+1)(n+2)}, \qquad i > 1.
\end{aligned} \tag{A.17}$$

REFERENCES

[AB09] M. Anthony and P. L. Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

[ABS13] H. Attouch, J. Bolte, and B. F. Svaiter. "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods." *Mathematical Programming*, **137**(1-2):91–129, 2013.

[Ack17] E. Ackerman. "How Drive.ai is Mastering Autonomous Driving with Deep Learning." https://spectrum.ieee.org/cars-that-think/transportation/self-driving/how-driveai-is-mastering-autonomous-driving-with-deep-learning, 2017. IEEE. Accessed: March 2021.

[AGS13] A. Aswani, H. Gonzalez, S. S. Shankar, and C. Tomlin. "Provably safe and robust learning-based model predictive control." *Automatica*, **49**(5):1216–1226, 2013.

[Ale24] S. Alemohammad et al. "Self-consuming generative models go mad." *International Conference on Learning Representations (ICLR)*, 2024.

[ALS19] Y. Abbasi-Yadkori, N. Lazic, and C. Szepesvari. "Model-Free Linear Quadratic Control via Reduction to Expert Prediction." In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 3108–3117. PMLR, 2019.

[AS20] A. Agrachev and A. Sarychev. "Control in the Spaces of Ensembles of Points." *SIAM Journal on Control and Optimization*, **58**(3):1579–1596, 2020.

[Atk08] K. E. Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 2008.

[Ber24] Q. Bertrand et al. "On the stability of iterative retraining of generative models on their own data." *International Conference on Learning Representations (ICLR)*, 2024.

[BM92] P. J. Besl and N. D. McKay. "Method for registration of 3-D shapes." In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pp. 586–606. Spie, 1992.

[BNL12] B. Biggio, B. Nelson, and P. Laskov. "Poisoning attacks against support vector machines." In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, p. 1467–1474, 2012.

[Bor23] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*, volume 48 of *Texts and Readings in Mathematics*. Springer, 2023.

[Bro07] R. W. Brockett. "Optimal control of the Liouville equation." *AMS IP Studies in Advanced Mathematics*, **39**:23, 2007.

[Cao24] H. Cao et al. "A survey on generative diffusion models." *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[Car24] N. Carlini et al. "Poisoning Web-Scale Training Datasets is Practical." In *IEEE Symposium on Security and Privacy (SP)*, 2024.

[CCC16] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age." *IEEE Transactions on Robotics*, **32**(6):1309–1332, 2016.

[CCL18] L. Cheng, S. Chen, X. Liu, H. Xu, Y. Wu, M. Li, and Y. Chen. "Registration of Laser scanning point clouds: A review." *Sensors*, **18**(5):1641, 2018.

[COM19] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3387–3395, 2019.

[CZ14] J. Chorowski and J. M. Zurada. "Learning understandable neural networks with nonnegative weight constraints." *IEEE Transactions on Neural Networks and Learning Systems*, **26**(1):62–69, 2014.

[DJS18] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. "Learning and verification of feedback control systems using feedforward neural networks." *IFAC-PapersOnLine*, **51**(16):151–156, 2018.

[DMM19] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. "On the sample complexity of the linear quadratic regulator." *Foundations of Computational Mathematics*, pp. 1–47, 2019.

[DMR20] S. Dean, N. Matni, B. Recht, and V. Ye. "Robust Guarantees for Perception-Based Control." In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 350–360. PMLR, 2020.

[DTC21] S. Dean, A. Taylor, R. Cosner, B. Recht, and A. Ames. "Guaranteeing Safety of Learned Perception Modules via Measurement-Robust Control Barrier Functions." In *Proc. of the 2020 Conference on Robot Learning*, volume 155 of *PMLR*, pp. 654–670, 16–18 Nov 2021.

[DV10] H. Daniels and M. Velikova. "Monotone and partially monotone neural networks." *IEEE Transactions on Neural Networks*, **21**(6):906–917, 2010.

[DV20] C. Debeunne and D. Vivet. "A review of visual-LiDAR fusion based simultaneous localization and mapping." *Sensors*, **20**(7):2068, 2020.

[Fre95] R. Freeman. "Global internal stabilizability does not imply global external stabilizability for small sensor disturbances." *IEEE Transactions on Automatic Control*, **40**(12):2119–2122, 1995.

[FRH19] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. J. Pappas. "Efficient and accurate estimation of Lipschitz constants for deep neural networks." In *Advances in Neural Information Processing Systems*, pp. 11423–11434, 2019.

[GZW19] Z. Gojcic, C. Zhou, J.D. Wegner, and A. Wieser. "The Perfect Match: 3D Point Cloud Matching With Smoothed Densities." In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Han09] S. Hanba. "On the "Uniform" Observability of Discrete-Time Nonlinear Systems." *IEEE Transactions on Automatic Control*, **54**(8):1925–1928, 2009.

[HKR16] 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)*, pp. 1271–1278. IEEE, 2016.

[Hor87] B. Horn. "Closed-form solution of absolute orientation using unit quaternions." *Journal of the Optical Society of America*, **4**(4):629–642, 1987.

[HR17] E. Haber and L. Ruthotto. "Stable architectures for deep neural networks." *Inverse Problems*, **34**(1), 2017.

[HS06] Morris W Hirsch and Hal Smith. "Monotone dynamical systems." In *Handbook of differential equations: ordinary differential equations*, volume 2, pp. 239–357. Elsevier, 2006.

[HS14] Uwe Helmke and Michael Schönlein. "Uniform ensemble controllability for one-parameter families of time-invariant linear systems." *Systems & Control Letters*, **71**:69–77, 2014.

[JL20] M. Jin and J. Lavaei. "Stability-certified reinforcement learning: A control-theoretic perspective." *IEEE Access*, **8**:229086–229100, 2020.

[KA19] S. Kolathaya and A. D. Ames. "Input-to-State Safety With Control Barrier Functions." *IEEE Control Systems Letters*, **3**(1):108–113, 2019.

[KL20] B. Karg and S. Lucia. "Stability and feasibility of neural network-based controllers via output range analysis." In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 4947–4954. IEEE, 2020.

[Koh21] P. W. Koh et al. "Wilds: A benchmark of in-the-wild distribution shifts." In *Proceedings of the 29th International Conference on Machine Learning*, ICML'21, pp. 5637–5664, 2021.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." *Nature*, **521**(7553):436–444, 2015.

[LCT17] Q. Li, L. Chen, C. Tai, and E. Weinan. "Maximum principle based algorithms for deep learning." *The Journal of Machine Learning Research*, **18**(1):5998–6026, 2017.

[LHA20] M. Lechner, R. Hasani, A. Amini, T. A. Henzinger, D. Rus, and R. Grosu. "Neural circuit policies enabling auditable autonomy." *Nature Machine Intelligence*, **2**(10):642–652, 2020.

[LJY99] F. W. Lewis, S. Jagannathan, and A. Yesildirak. *Neural network control of robot manipulators and non-linear systems*. CRC press, 1999.

[LK06] J. S. Li and N. Khaneja. "Control of inhomogeneous quantum ensembles." *Physical Review A*, **73**(3):030302, 2006.

[LKB17] G. Litjens, T. Kooi, B.E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. "A survey on deep learning in medical image analysis." *Medical image analysis*, **42**:60–88, 2017.

[LS89] R. Liptser and A. N. Shiryayev. *Theory of Martingales*, volume 49 of *Mathematics and its Applications*. Springer, 1989.

[LZL18] Y. Lu, A. Zhong, Q. Li, and B. Dong. "Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations." In *International Conference on Machine Learning*, pp. 3276–3285, 2018.

[Mar23a] G. Martínez et al. "Combining generative artificial intelligence (AI) and the internet: heading towards evolution or degradation?" *arXiv preprint arXiv:2303.01255*, 2023.

[Mar23b] G. Martínez et al. "Towards understanding the interplay of generative artificial intelligence and the internet." *arXiv preprint arXiv:2306.06130*, 2023.

[NT04] D. Nesic and A. R. Teel. "A framework for stabilization of nonlinear sampled-data systems based on their approximate discrete-time models." *IEEE Transactions on Automatic Control*, **49**(7):1103–1122, 2004.

[OK17] O. K. Oyedotun and A. Khashman. "Deep learning in vision-based static hand gesture recognition." *Neural Computing and Applications*, **28**(12):3941–3951, 2017.

[PB17] J. Park and S. Boyd. "General heuristics for nonconvex quadratically constrained quadratic programming." *arXiv preprint arXiv:1703.07870*, 2017.

[PCS15] F. Pomerleau, F. Colas, and R. Siegwart. "A review of point cloud registration algorithms for mobile robotics." *Foundations and Trends® in Robotics*, **4**(1):1–104, 2015.

[QSM17] C. Qi, H. Su, K. Mo, and L. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[RBB09] R. Rusu, N. Blodow, and M. Beetz. "Fast Point Feature Histograms (FPFH) for 3D registration." In *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[RGP17] S. Ramos, S. Gehrig, P. Pinggera, U. Franke, and C. Rother. "Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling." In *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1025–1032. IEEE, 2017.

[RJ16] M. Romdlony and B. Jayawardhana. "On the new notion of input-to-state safety." In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 6403–6409. IEEE, 2016.

[RL18] H. Rehman and S. Lee. "Automatic image alignment using principal component analysis." *IEEE Access*, **6**:72063–72072, 2018.

[SB11] C. Solomon and T. Breckon. *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*, pp. 247–262. John Wiley & Sons, 2011.

[SB14] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[SFP20] J.ob H. Seidman, M. Fazlyab, V. M. Preciado, and G. J. Pappas. "Robust Deep Learning as Optimal Control: Insights and Convergence Guarantees." In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 884–893. PMLR, 2020.

[SHT09] A. Segal, D. Haehnel, and S. Thrun. "Generalized ICP." In *Robotics: science and systems*, volume 2, p. 435. Seattle, WA, 2009.

[Shu23] I. Shumailov et al. "The curse of recursion: Training on generated data makes models forget." *arXiv preprint arXiv:2305.17493*, 2023.

[SMO04] J. T. Spooner, M. Maggiore, R. Ordonez, and K. M. Passino. *Stable adaptive control and estimation for nonlinear systems: neural and fuzzy approximator techniques*, volume 43. John Wiley & Sons, 2004.

[Son89] E. D. Sontag et al. "Smooth stabilization implies coprime factorization." *IEEE transactions on automatic control*, **34**(4):435–443, 1989.

[Son93] E. D. Sontag. "Neural networks for control." In *Essays on Control*, pp. 339–380. Boston, MA, USA: Birkhäuser, 1993.

[SP99] E. N. Sanchez and J. P. Perez. "Input-to-state stability (ISS) analysis for dynamic neural networks." *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **46**(11):1395–1398, 1999.

[SW96] E. D. Sontag and Y. Wang. "New characterizations of input-to-state stability." *IEEE transactions on automatic control*, **41**(9):1283–1294, 1996.

[TG20] P. Tabuada and B. Gharesifard. "Universal Approximation Power of Deep Residual Neural Networks via Nonlinear Control Theory." *arXiv preprint arXiv:2007.06007v3*, 2020. To appear in *9th International Conference on Learning Representations*.

[TMP20] A. Tsiamis, N. Matni, and G. Pappas. "Sample Complexity of Kalman Filtering for Unknown Systems." In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 435–444. PMLR, 2020.

[Tou23] H. Touvron et al. "Llama 2: Open foundation and fine-tuned chat models." *arXiv preprint arXiv:2307.09288*, 2023.

[TT20] H. Thanh-Tung and T. Tran. "Catastrophic forgetting and mode collapse in GANs." In *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[VDD18] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. "Deep learning for computer vision: A brief review." *Computational Intelligence and Neuroscience*, 2018.

[Wei17] E. Weinan. "A Proposal on Machine Learning via Dynamical Systems." *Communications in Mathematics and Statistics*, **5**, 2017.

[XTG15] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames. "Robustness of Control Barrier Functions for Safety Critical Control." *IFAC-PapersOnLine*, **48**(27):54–61, 2015. Analysis and Design of Hybrid Systems.

[XTR18] W. Xiang, H. Tran, J. A. Rosenfeld, and T. T. Johnson. "Reachable set estimation and verification for a class of piecewise linear systems with neural network controllers." In *American Control Conference (ACC), invited session on Formal Methods in Controller Synthesis*, 2018.

[YB19] V. G. Yaji and S. Bhatnagar. "Analysis of stochastic approximation schemes with set-valued maps in the absence of a stability guarantee and their stabilization." *IEEE Transactions on Automatic Control*, **65**(3):1100–1115, 2019.

[YHP18] T. Young, D. Hazarika, S. Poria, and E. Cambria. "Recent trends in deep learning based natural language processing." *IEEE Computational Intelligence Magazine*, **13**(3):55–75, 2018.

[YLJ13] J. Yang, H. Li, and Y. Jia. "Go-ICP: Solving 3D Registration Efficiently and Globally Optimally." In *2013 IEEE International Conference on Computer Vision*, pp. 1457–1464, 2013.

[YSA20] H. Yin, P. Seiler, and M. Arcak. "Stability analysis using quadratic constraints for systems with neural network controllers." *arXiv preprint arXiv:2006.07579*, 2020.

[YSC20] H. Yang, J. Shi, and L. Carlone. "Teaser: Fast and certifiable point cloud registration." *IEEE Transactions on Robotics*, **37**(2):314–333, 2020.

[YSC21] H. Yang, J. Shi, and L. Carlone. "TEASER: Fast and Certifiable Point Cloud Registration." *IEEE Transactions on Robotics*, **37**(2):314–333, 2021.

[Zha23] W. X. Zhao et al. "A survey of large language models." *arXiv preprint arXiv:2303.18223*, 2023.

[ZPZ00] Y. Zhang, P. Peng, and Z.Jiang. "Stable neural controller design for unknown nonlinear systems using backstepping." *IEEE Transactions on Neural Networks*, **11**(6):1347–1360, 2000.