University of California

Santa Barbara

High-Dimensional Polynomial Approximation with Applications in Imaging and Recognition

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy

in

Electrical and Computer Engineering

by

Abhejit Rajagopal

Committee in charge:

Professor Shivkumar Chandrasekaran, Chair

Professor Hua Lee

Professor Kenneth Rose

Professor Upamanyu Mhadow

Professor Fabien Scalzo

Professor Hrushikesh N. Mhaskar

September 2019

The dissertation of Abhejit Rajagopal is approved.

Professor Hua Lee

Professor Kenneth Rose

Professor Upamanyu Mhadow

Professor Fabien Scalzo

Professor Hrushikesh N. Mhaskar

Professor Shivkumar Chandrasekaran, Chair

September 2019

High-Dimensional Polynomial Approximation

with Applications in Imaging and Recognition

Abhejit Rajagopal

To my family.

Acknowledgements

Graduate school has been quite a journey for me, and without the support of my many mentors, family, and friends, I most certainly would not have made it this far.

I would first like to thank my advisor Shivkumar Chandrasekaran, who has shown great patience in exposing me to the language, art, and practice of mathematics. For almost any problem that I bring, Shiv has a knack of tearing away as many assumptions as possible to illuminate the underlying mathematical problem and its core difficulties. Whether it was about linear algebra, PDEs, radar, or pattern recognition, Shiv has always been an endless well of knowledge and a crucial part of my education. Without Shiv's creativity, guidance, and friendship, this dissertation would not have been possible.

I would similarly like to thank Hua Lee, who has been a mentor and sponsor for my interests in imaging and has exposed me to an exciting number of possibilities, starting with radar. His curiosity and wisdom is imparted by the great number of stories he tells, and I'm grateful for all the afternoons, evenings, meals, tea-times, and road trips he has spent sharing them with me.

I am also grateful for the support and mentorship of Andrew Brown, who believed in me early in my graduate career, and enabled new opportunities for me at Toyon. With his guidance, I was able to pursue funding to mature my interests in the analysis of various signal modalities.

My graduate experience would not have been the same without the support of the Scientific Computing Group: Kristen, Chris, Nithin, and Ethan, who have all provided guidance and input on various parts of my graduate work and have exposed me to new problem areas and approaches from their own research. Nithin, Kristen, and I spent hours working through linear algebra problems. Chris spent time teaching me many of the fundamentals of MSN interpolation. Ethan spent hundreds of hours collecting, translating,

explaining, and extending the many theorems of classical approximation theory, including the ones featured in this thesis concerning deep MSN networks. I am extremely grateful for his work and intellect, and it is important that his contribution to this work is stated.

There are many friends that have made my journey possible, including Oytun, Asutay, Isabella, Rohan, Itir, Camille, Rahul, Nate, Sean, Bryce, David, Connor, and Noah.

I would especially like to thank Vincent R. Radzicki, who has been a friend, collaborator, and brother to me throughout graduate school. The story of how we first met is still mysterious, but his camaraderie and enthusiasm to explore the unexplored has resulted in several enjoyable years exploring the theory of Fourier imaging, radar, PDEs, basketball, golf, computers, and everything in between. I am looking forward to continuing our collaborations and discussions for many years to come.

Finally I would like to thank my family, Amma, Dad, Ana, who have supported me and fostered an a strong sense of ambition in education. Looking back, the years spent working through Spectrum Mathematics books with Amma, learning physics and chemistry from Self Teaching Guides with Dad, and being inspired by the various research projects undertaken by Ana, have truly had a phenomenal impact on my curiosity and ambition as a scientist. I am forever grateful.

And last, and certainly not least, I want to thank Deeksha, who has been a true friend, confidant, and partner throughout my time at UC Santa Barabra. Whether it was nerding out over VScode setups, staying in the lab late and making spontaneous trips to NaanStop, or walking along the beach to decompress, Deeks has supported me and nurtured growth in all aspects of my life. With her love, I have developed a sense of awareness in both my academic and personal interactions that has resulted in a great number of new opportunities for me, including a postdoc position at UC San Francisco. I am very thankful, and I am looking forward to pursuing our dreams together.

Abhejit Rajagopal
abhejit@ece.ucsb.edu

EDUCATION

Ph.D. Electrical & Computer Engineering 2019
University of California, Santa Barbara

M.S. Electrical & Computer Engineering, 2016
University of California, Santa Barbara

B.S. Electrical Engineering 2014
University of California, Los Angeles

ACADEMIC EXPERIENCE

Graduate Student Researcher, June 2015 - September 2019
Scientific Computing Group, UC Santa Barbara
Advisor: Dr. Shivkumar Chandrasekaran

- ⊙ Design of numerical algorithms for neural information processing and imaging systems.

Teaching Assistant, September 2014 - December 2017
Department of Electrical & Computer Engineering, UC Santa Barbara

- ⊙ ECE 2a - Circuits, Devices, and Systems (Fall 14) – *w/ Prof. H. Lee*
- ⊙ ECE 137a - Circuits and Electronics I (Winter 15) – *w/ Prof. M. Rodwell*
- ⊙ ECE 137b - Circuits and Electronics II (Spring 15) – *w/ Prof. M. Rodwell*
- ⊙ ECE 130a - Signal Analysis & Processing I (Fall 15) – *w/ Prof. H. Lee*
- ⊙ ECE 130b - Signal Analysis & Processing II (Winter 16) – *w/ Prof. S. Chandrasekaran*
- ⊙ ECE 130c - Signal Analysis & Processing III (Spring 16, 17) – *w/ Prof. S. Chandrasekaran*
- ⊙ ECE 134 - Introduction to Fields and Waves (Fall 17) – *w/ Prof. B. York*
- ⊙ ECE 210a - Matrix Analysis (grader–Fall 15) – *w/ Prof. S. Chandrasekaran*
- ⊙ ECE 259a - Digital Speech Processing (grader–Winter 16, 17) – *w/ Prof. L. Rabiner*
- ⊙ ECE 258a - Advanced Digital Signal Processing (grader–Winter 17) – *w/ Prof. M. Liebling*

Collaborator, Staba Lab, UCLA August 2014 - June 2015
Advisor: Dr. Shennan Weiss, Dr. Richard Staba

- ⊙ Design and numerical optimization of epilepsy detection algorithms that process intracranial EEG to identify interictal discharges and coherent high-frequency neural oscillations.

Undergraduate Researcher, April 2013 - August 2014
Neurovascular Imaging Research Core, UCLA *Advisor: Dr. Fabien Scalzo*

- ⊙ Discrimination and estimation of intracranial pressure (ICP) signals via manifold learning.

Undergraduate Researcher, April 2013 - June 2014
Integrated Nanomaterials Core Lab, UCLA *Advisor: Dr. Dianna Huffaker*

- ⊙ Optical and electrical simulation of sub-wavelength nanostructures. Development of tools for characterization and design of 3D III-V nanowire solar cells, APDs, and modulators.

Research Volunteer, May 2008 - May 2010
Collin's Group, UC Irvine *Advisor: Dr. Philip G. Collins*

- ⊙ Investigated electrostatic properties of single-walled carbon nanotubes in FETs using AFM and SEM microscopy, E-beam lithography, and Raman spectroscopy.

PROFESSIONAL EXPERIENCE

Analyst, Toyon Research Corporation December 2015 - September 2019

⊙ Development of new signal analysis and learning paradigms for EO/IR, LiDAR, and SAR imagery.

Research Intern, Akela Inc. June 2015 - October 2015

⊙ R&D in radar signal processing, EM simulation, antenna design, VNA calibration.

Consultant, Neural Analytics Inc. December 2014 - December 2015

⊙ Signal processing, machine learning, and QA for traumatic brain injury (TBI) detection and prognosis using ICP, TCD and other measurements of CBFV.

Intern, Hardware Testing, Broadcom Corporation April 2012 - April 2013

⊙ Test and debug of reference designs in HWDSL group. Automated DSL/WiFi throughput testing, designed a temperature feedback controller for PVT, developed drivers and tools.

PUBLICATIONS

* A. Rajagopal, E. Epperly, H.N. Mhaskar, and S. Chandrasekaran, "Deep Polynomial Networks", *in preparation*, Sep 2019.

* A. Rajagopal, V.R Radzicki, H. Lee, S. Chandrasekaran, "DeepISAR: End-to-End Imaging and Recognition in Inverse Synthetic Aperture Radar (ISAR)", *under review*, Sep 2019.

* A. Rajagopal, W. Nelson, N. Stier, S. Chandrasekaran, A.P. Brown, "Deep OSM-3D: Scalable Recognition in Wide-Area LiDAR and RGBD Imagery", *under review*, Sep 2019.

11. A. Rajagopal, N. Stier, J. Dey, M. King, S. Chandrasekaran, "Towards deep iterative-reconstruction algorithms for single-photon emission computed tomography (SPECT)", *SPIE: Medical Imaging*, Feb 2019. DOI: 10.1117/12.2513005

10. A. Rajagopal, V.R. Radzicki, H. Lee, and S. Chandrasekaran, "Nonlinear electrocardiographic imaging using polynomial approximation networks", *APL Bioengineering*, Oct 2018. DOI: 10.1063/1.5038046

9. S. Chandrasekaran, N. Govindarajan, and A. Rajagopal, "Fast Algorithms for Displacement and Low-Rank Structured Matrices", *2018 ACM International Symposium on Symbolic and Algebraic Computation*, Jul 2018. DOI: 10.1145/3208976.3209025

8. A. Rajagopal, H.N. Mhaskar, and S. Chandrasekaran, "Deep Algorithms: designs for networks", *arXiv:1806.02003*, Jun 2018. arXiv:1806.02003

7. A. Rajagopal, V.R. Radzicki, H. Lee, and S. Chandrasekaran, "Towards non-invasive electrocardiographic imaging using regularized neural networks", *SPIE: Medical Imaging*, Feb 2018. **(Best Poster Award)** DOI: 10.1117/12.2294474

6. A. Rajagopal, V.R. Radzicki, S. Chandrasekaran, and H. Lee, "Tracking Information in RaDAR Image Formation and Classification Algorithms", *International Telemetry Conference (ITC)*, Oct 2017. ISSN: 0884-5123; 0074-9079

5. A. Rajagopal, K. Chellappan, S. Chandrasekaran, and A.P. Brown, "A machine learning pipeline for automated registration and classification of 3D LiDAR data", *SPIE: Defence and Security*, May 2017. DOI: 10.1117/12.2262872

4. S. Chandrasekaran and A. Rajagopal, "Fast indefinite multi-point (IMP) clustering", *Calcolo*, Apr 2016. DOI: 10.1007/s10092-016-0191-2

3. A. Rajagopal, R. Hamilton, and F. Scalzo, "Noise reduction in intracranial pressure signal using causal shape manifolds", *Biomedical Signal Processing and Control*, Mar 2016. DOI: 10.1016/j.bspc.2016.03.003

2. A. Farrell, P. Senanayake, CH. Hung, G. El-howayek, A. Rajagopal, M. Currie, M. Hayat, and D.L. Huffaker, "Plasmonic field confinement for separate absorption-multiplication in InGaAs nanopillar avalanche photodiodes", *Scientific Reports*, Dec 2015. DOI: 10.1038/srep17580

1. G. Mariani, M. Haddad, A. Rajagopal, D.L. Huffaker, "High-efficiency nanopillar solar cells employing wide-bandgap surface recombination barrier", *SPIE: Photonics West*, Invited Paper, Feb 2014. DOI: 10.1117/2.1201401.005303

WORKSHOPS & WHITE PAPERS

2. A. Rajagopal, V.R. Radzicki, H. Lee, S. Chandrasekaran, "Deep Learning for Inverse Problems", *2019 International Conference on Machine Learning: Workshop on Physics for Deep Learning*, Jun 2019. [link]

1. A. Rajagopal, A.C. Nguyen, and D.M. Briggs, "NeuroPass: A secure neural password based on EEG", *University of California, Los Angeles*, Dec 2013. perm: [link]

PATENTS

1. F. Scalzo and A. Rajagopal, "Machine-learning based denoising of doppler ultrasound blood flow and intracranial pressure signal", University of California, US201662279653, Jan 2016.

FUNDING

SBIR Phase I – SCO182-008, $225K Dec 2018 – Jun 2019
Principal Investigator, "Maritime Target Classification from ISAR Imagery Using Machine Learning"

STTR Phase I – N18B-T033, $125K Oct 2018 – Apr 2019
Principal Investigator, "Blending Classical Model-Based Classification with Data-Driven Artificial Intelligence"

SBIR Phase II – AF161-138, $750K Nov 2017 – Feb 2020
Principal Investigator, "Cognitive Processing and Exploitation of 3D LiDAR Imagery"

HONORS & AWARDS

ECE Dissertation Fellowship , UC Santa Barbara	May 2019
Outstanding ECE Teaching Assistant Award, UC Santa Barbara	May 2018
Best Poster Award, Physics of Medical Imaging, SPIE-MI 2018	Feb 2018
Best Poster Award, UC Bioengineering Symposium 2017	Jun 2017
ASEI Graduate Fellowship	Jun 2014

MENTORING

UCSB Research Mentorship Program, Santa Barbara, CA Summers 2015, 2016, 2017
Mentoring high school students in basic algorithms useful for image processing and automatic signal analysis. Guiding students to develop fundamental skills in programming and mathematics by engaging in relevant research. New projects every year. Students advised:

- Tiffany Huang, *now at M.I.T.* Jun 2017 - Aug 2017
- Michael B. Zhang, *now at UC Santa Barbara* Jun 2017 - Aug 2017
- Catherine Chi, *now at UC Berkeley* Jun 2016 - Aug 2016
- Anuva Mittal, *now at University of Southern California* Jun 2017 - Aug 2016
- Tejas Thvar, *now at UC Berkeley* Jun 2017 - Aug 2016
- Priyanka Multani, *now at Stanford University* Jun 2015 - Aug 2015
- Gitanjali Multani, *now at Stanford University* Jun 2015 - Aug 2015

UCLA NSF IGERT, Los Angeles, CA Jan 2014 – May 2014
Directed two students to develop a hub for clean-energy web-applications. Served as a software architect and managed the development in all stages. Facilitated conversations between engineers and developers, and established an inter-department collaboration between Dept. of Electrical Engineering and Dept. of Geography.

UCLA SEAS Mentor, Los Angeles, CA Sep 2012 – Jun 2014
Served as a mentor for freshman and sophomore students in Electrical Engineering, part of the Henry Samueli School of Engineering and Applied Science.

SERVICE

Reviewer

- Elsevier Biomedical Signal Processing and Control 2017 - 2019
- IEEE Journal of Biomedical and Health Informatic 2019
- IEEE WACV 2019

Graduate Simulation Seminar Series (GS^3), Santa Barbara, CA Summers 2016-2018
Seminar series on numerical simulation. Member, speaker, and volunteer.

PROFESSIONAL AFFILIATION

Institute of Electrical and Electronics Engineers (IEEE)
International Society for Optics and Photonics (SPIE)
Society for Industrial and Applied Mathematics (SIAM)
Association for Computing Machinery (ACM)

Abstract

High-Dimensional Polynomial Approximation
with Applications in Imaging and Recognition
by
Abhejit Rajagopal

Deep learning has demonstrated unreasonable effectiveness on several high dimensional regression and classification problems, far exceeding theoretical expectations. In this thesis, we analyze this phenomena from the perspective of approximation theory. Utilizing recent theoretical advances, we investigate whether and under what conditions deep networks can escape the curse of dimensionality, providing experimental evidence where the theory falls short. We use these insights to suggest new approaches to network design that is more in accordance with this theory, and give several examples of where such designs succeed.

Contents

Chapter 1

Introduction

In this thesis, we are interested in studying and developing scalable approaches to high dimensional imaging and image recognition problems. Imaging refers to the process of transforming raw sensor data into descriptive 1D, 2D, and 3D physical representations or "images" of objects. Image recognition refers to the process of understanding the objects in these images by computing a descriptive segmentation, clustering, classification, or sub-classification result. The task is to construct useful and accurate functions that perform imaging and/or image recognition. While previously physics, geometry, topology, statistics, and even computer vision have been used to construct simple linear and polynomial models for these problems, as the dimension of data has increased these techniques have been replaced by deep learning approaches. Today, deep learning is used pervasively for image recognition, and more recently for image reconstruction, in various signal modalities because of its unreasonable effectiveness in achieving and advancing the state-of-the-art for these tasks.

What is surprising and "unreasonable" about deep learning is that its performance is to-date not supported satisfactorily by a theory. In particular from the perspective of approximation theory, even though the input dimension is very high (often in the thousands), deep learning algorithms are able to uncover representations that seem to

generalize surprisingly well to previously unseen queries from relatively few data samples. While deep networks empirically perform better than shallow ones, the connection between structure and performance of deep networks is still poorly understood, often only vaguely motivated by notions in computer vision rather than by a rigorous theory. This has lead to an abundance of ad hoc network designs, often arrived upon by trial and error, frustrating the application of deep learning to critical imaging and image recognition systems. Thus, a major focus of this thesis is to understand and quantify the phenomena of deep learning to develop scalable techniques that can provide reliable and robust performance for these tasks.

In this thesis, we approach this problem in two ways. First, by appealing to approximation theory and recent theoretical work on using deep neural networks (DNNs) to approximate compositional functions, we investigate under what conditions deep networks can avoid the curse of dimensionality with respect to the degree of approximation (e.g. number of neurons) and the data rate (amount of data required) for a desired level of accuracy. Second, by generalizing the architecture-design strategies employed in today's DNNs, we investigate whether prior human knowledge about a problem can be used to accelerate the search for robust data-efficient architectures. Both these approaches are used to influence the design and application of high dimensional end-to-end imaging and recognition systems.

To speak to the first point, approximation theory is concerned with how functions can be approximated by simpler functions and in Chapter 2 we focus on polynomial approximation. Polynomials are chosen for this task not only because they match the language used in classical approximation theory, but because they are simple and reflect our descriptions of physics and other low-dimensional models. In electrodynamics, for example, a polynomial approximation (e.g. Taylor series) is used to describe the electric field density in terms of the electric field [103]. Note that polynomials form the basis for other popular means of approximation, such as via neural networks. More generally,

the use of polynomials is pervasive throughout physics and engineering, from the way we describe physical phenomena (fields and dielectric responses), to how we manipulate algebraic expressions and variables (i.e. indeterminates), and how we think about functions and their limits (e.g. calculus).

In fact, as we see in Chapter 3, traditional algorithms can also be viewed through the lens of polynomial approximation. Specifically, we show how sequential programs correspond to high order polynomial networks, and how these *deep* networks can be exploited for inference and regression tasks such as classification and imaging, sometimes with performance guarantees. This technique is called the approach of "Deep Algorithms".

In Chapter 4, we take a more pragmatic view for a few application problems. In these problems we show how exploiting the structure of the data, signal modality, or a problem-specific solution heuristic yields avenues for designing robust architectures for imaging and image recognition. Many of the insights that are derived from polynomial approximation theory and Deep Algorithms are utilized or showcased in these application domains.

Chapter 2

Polynomial Approximation

In this chapter, we review some of the relevant classical theory for using polynomials to approximate functions. We then ask how this theory extends to to high dimensional problems, touching on issues that arise such as the curse of dimensionality and severe undersampling with respect to the dimension. Building on some recent theoretical work, we prove that deep networks can avoid the curse of dimensionality with respect to the degree of approximation for compositional functions when the compositional graph is known, and we show empirically that this technique may also ameliorate data rate requirements.

2.1 Background

2.1.1 Classical Approximation

The central classical result of approximation theory is the well-known Weierstrass theorem.

Theorem 2.1.1. Let f be a continuous function on a compact set, say $[-1,1]^d$. Then there exists a sequence of polynomial functions p_n converging uniformly to f.

A constructive proof is given by Bernstein [153], stated here for $d = 1$. Specifically, for a function $f : [0,1] \to \mathbb{R}^1$ define the Bernstein polynomial as:

$$B_n(x) := \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} f\left(\frac{k}{n}\right).$$

Proposition 2.1.2. The Bernstein polynomials converge uniformly to f on $[0,1]$. Moreover, if f is Lipschitz continuous with Lipschitz constant L, then:

$$||f - B_n||_{L^\infty} \leq \frac{L}{2\sqrt{n}}. \tag{2.1}$$

A probabilistic interpretation for the Bernstein polynomials is as follows. For $n \geq 1$, define X_n to be the average of n independent Bernouli trials with success probability x. The Bernstein polynomial $B_n(x)$ is then the expectation of $f(X_n)$. In effect, f has been smoothed to a polynomial by a discrete random walk. The proof of Proposition 2.1.2 goes by the law of large numbers and the estimate (2.1) goes by Chebyshev's inequality. Where more sophisticated approximation bounds for the Bernstein polynomials are possible [46, 22], there are much better ways of approximating a given function.

As is usual in approximation theory, the degree of approximation depends on the smoothness of the function being approximated. To this end, we introduce the Sobolev space $W^{k,p}(\Omega)$ to consist of the functions f on a domain $\Omega \subseteq \mathbb{R}^q$ which are k times (weakly) differentiable such that the Sobolev norm $||f||_{W^{k,p}}$ is finite, where:

$$\|f\|_{W^{k,p}(\Omega)} := \sum_{|\alpha| \leq k} \|D^\alpha f\|_{L^p(\Omega)}, \quad \|f\|_{L^p(\Omega)} := \begin{cases} \left(\int_\Omega |f(x)|^p \, dx\right)^{1/p}, & p < \infty, \\ \operatorname{ess\,sup}_{x \in \Omega} |f(x)|, & p = \infty. \end{cases}$$

Here $\alpha = (\alpha_1, \ldots, \alpha_q)$ denotes a multiindex of nonnegative integers such that $|\alpha| = \alpha_1 + \cdots + \alpha_q \leq k$ and $D^\alpha f$ is defined to as:

$$D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_q^{\alpha_q}}.$$

When the domain Ω is clear from context, we shall omit it. The mathematical theory of Sobolev spaces is introduced in [66, Ch. 5]. Given a polynomial p, we define the degree

of p to be $\max |\alpha|_\infty$, where $|\alpha|_\infty = \max_{1 \leq i \leq q} |\alpha_i|$ and the maximum is taken over all monomials $x^\alpha = x_1^{\alpha_1} \cdots x_q^{\alpha_q}$ appearing in p. The following classic result states the degree of approximation by polynomials.

Theorem 2.1.3 ([204]). For $f \in W^{k,p}([-1,1]^q)$ and $n \geq 1$, there exists a constant $C = C(k,p)$ and a polynomial p_n of degree not exceeding n such that

$$\|f - p_n\|_{L^p} \leq C \|f\|_{W^{k,p}} n^{-k}.$$

Moreover, the theory of nonlinear widths says that for any means of approximating a function in $W^{k,p}$ to accuracy ϵ in L^p where the approximant continuously depends on the target function f, there exists a constant $C = C(k,p)$ such that the approximant requires $\geq C\epsilon^{-q/k}$ terms to describe [60, Thm. 4.2]. (A more simple statement is provided in [222, Thm. 3].) Thus, in terms of complexity, approximation by polynomials is asymptotically optimal with respect to the degree of approximation. In the same stroke, however, we see that approximation by polynomials suffers from the curse of dimensionality, as the number of terms grows exponentially in the dimension. Even for $d = 10$, this seems intractable for $\epsilon \leq 0.1$.

2.1.2 The Curse of Dimensionality

The curse of dimensionality refers to unsustainable growth in the runtime, storage complexity, or power consumption of an algorithm as the dimension of the input increases. In the context of approximation algorithms, this comes in two flavors: (1) exponential growth in the number of parameters or terms in an approximation, and (2) exponential growth in the amount of training data needed to even coarsely sample a domain (e.g the d-dimensional hypercube).

In the approximation theory literature, the growth in the number of terms required for provably achieving ϵ accuracy for some function class is studied as the "degree of approximation" of an approximation scheme. It can be shown that, for a given function

class, some schemes (algorithms) are more "optimal" than others with respect to the degree of approximation that they offer. Note that these are typically complexity estimates. The major task of the theorist, then, is to design a scheme that offers a high degree of accuracy for representing a large (useful) function class with as few terms as possible. If she succeeds, the next challenge is to outline a scheme for setting or learning the corresponding parameters or coefficients.

There have been many attempts in this vein. Each scheme has its own tradeoffs in the degree of approximation, functions it can approximate, learnability, and practical numerical feasibility (although not typically considered in theoretical work). An important question for such investigations is whether the function class is of practical importance, e.g. representing the solution to a family of partial differential equations (PDEs).

Unfortunately, most of the existing theory and relies on density of data in the input domain, and in many ways constructing approximations from a finite amount of data remains an open problem. This problem is exacerbated in high dimensions. Assuming the data lies in the unit d-dimensional hypercube $[-1, 1]^d$, the number of data points is μ^{-d}, where μ represents the data density or the mesh-norm (i.e. distance between samples). This is clearly infeasible when $d > 15$, even with a coarse sampling (e.g. 2 samples per axis yields 2^15 data points). Instead, and in some cases, we can show that an approximation scheme converges in the *neighborhood* of the available data. In some problems, even high dimensional imaging and recognition problems where the data is sampled from sufficiently close clusters, this can be sufficient to yield a useful approximation. On the other hand, uncovering and exploiting the structure of data is a challenging task (e.g. on curved domains).

2.1.3 Neural Networks

Single Layer Networks

Definition: An **activation function** σ is a continuous, nonlinear, non-polynomial function from $\mathbb{R}$ to $\mathbb{R}$. A **sigmoid** is an activation function which limits to 0 at $-\infty$ and 1 at ∞. A **single-layer feed-forward network** is a function $g : \mathbb{R}^d \to \mathbb{R}$ given by

$$g(x) = \sum_{i=1}^{M} a_i \sigma(y_i^T x + \theta_i)$$

The most important result of neural network approximation theory is that neural networks are so-called universal approximators; that is, they can approximate an arbitrary function to arbitrary accuracy.

Theorem 2.1.4 ([53], [97], among many others). Let σ be a bounded nonlinear non-polynomial activation function. Then for every $\epsilon > 0$ and any continuous function $f : [-1, 1]^d \to \mathbb{R}$, there exists a single layer feed-forward neural network g such that $\|f - g\|_\infty < \epsilon$.

Extensions such as [96] show these approximation properties extend to arbitrary measurable functions in L^p and Sobolev norms. The original proofs of universal approximation were non-constructive. Cybenkos proof [53] is function analytic and Hornik et al.s [97] uses the Stone-Weierstrass theorem. Additionally, these theorems provide no bound on the number of neurons M needed to obtain such an approximation.

A quantitative bound on the approximation error is given by Barron [11].

Theorem 2.1.5 ([11]). Let B be a ball in $\mathbb{R}^d$ of radius r, μ a probability measure, and $f : B \to \mathbb{R}$. Then for the constant $C := \int_{\mathbb{R}^d} |k||\hat{f}(k)|\, dk$ there exists a single-layer feed-forward network g of M neurons with sigmoid activation σ, such that:

$$\|f - g\|_{L^2(\mu)} \leq \sqrt{2}\, rCM^{-1/2}. \tag{2.2}$$

The proof can be made "constructive" based on a greedy algorithm where each individual neuron is added to minimize the error, although this depends upon finding the largest inner products among infinitely many. Still, Barron's result is noteworthy in that the approximation bound is independent of the dimension d. However, the dimension-dependence may be effectively hidden in the constant C, as the integral constraint effectively implies that the target function f must be very smooth. In fact, for such smooth functions there is nothing special about sigmoidal networks. As Candes shows in [27], the convergence rate in (2.2) may be improved to $O(M^{-1/2-1/d})$ by simply thresholding the Fourier series. Barron's result is not the only one that can be improved this way, as Candes's result is just an application of Hoeffding's inequality.

Mhaskar and others went on to study approximation by neural networks of functions depending on their smoothness properties.

Theorem 2.1.6 ([146]). Given a function $f \in W^{s,p}([-1,1]^d)$ and a smooth bounded activation function σ, there exists a constructive algorithm to produce a single-layer feedforward network of M neurons such that

$$\|f - g\|_{L^p} \leq O\left(\|f\|_{W^{s,p}} M^{-s/d}\right). \tag{2.3}$$

Moreover, the error bound (2.3) is optimal in the sense of n-widths theory (see [60]).

Sampled Data

Barron's result can be improved to include errors due to approximation error and estimation error.

Theorem 2.1.7 ([10]). Let σ be a sigmoid, μ a probability measure, and $f : [-1,1]^d \to \mathbb{R}$. Suppose the value of f is known only at N samples. Then for the Barron constant $C; = \int_{\mathbb{R}^d} |k||\hat{f}(k)|\, dk$, one can construct a single-layer feedforward network such that

$$\|f - g\|^2_{L^2(\mu)} = O\left(\frac{C^2}{M}\right) + O\left(\frac{Md}{n}\log n\right).$$

In particular, setting $M \sim C(N/(d\log N))^{1/2}$, then $\|f - g\|^2_{L^2(\mu)} = O(C((d/N)\log N)^{1/2})$.

Deep Networks

Definition: The ReLU function is defined to be $\mathrm{ReLU}(x) := \max(x, 0)$. A **deep feed-forward network** with k hidden layers and activation function σ is a function $g : \mathbb{R}^d \to \mathbb{R}$ given by

$$\begin{aligned} y_{0,j} &= x_j, && 0 \le j \le d \\ y_{i,j} &= \sigma\left(\sum_{\ell=1}^{M_{i,j}} a_{i,j,\ell} y_{i-j,I_{i,j\ell}} + b_{i,j}\right), && 1 \le i \le k, 1 \le j \le N_i \\ g(x) &= \sum_{j=1}^{N_k} c_k y_{i,k} + d. \end{aligned}$$

The number of nodes in the network is $N := \sum_{i=1}^{k} N_k$ and the number of weights is $M := \sum_{i=1}^{k}\sum_{j=1}^{N_i} M_{i,j}$. The **network architecture** is the collection $(k, \{N_i\}, \{M_{i,j}\}, \{I_{i,j,\ell}\})$ which characterizes the number of neurons and how they are connected. An architecture is said to **approximate** a class of functions to degree ϵ (with respect to some norm $\|\cdot\|$, if for every function f in that class, there exists a deep feed-forward network g of that architecture such that $\|f - g\| < \epsilon$.

While single-layer feed-forward networks are universal approximators optimal in the sense of n-width theory, empirical studies have continued to show that deep networks perform better than single layer networks. Mhaskar et. al show in [149, 144] that deep networks are effective at approximating functions possessing a special compositional structure.

Theorem 2.1.8 ([149, 144]). Let B_d^s denote the unit ball in $W^{s,\infty}$ with the norm $\|f\|_{W^{s,\infty}} = \sum_{|\alpha|\leq s} \|D^\alpha f\|$. Let F_d^s denote the space of functions which can be achieved by composition of functions in B_2^s. Then the complexity of single-layer feed-forward networks approximating B_d^s to degree ϵ in L^∞ is $O(\epsilon^{-d/s})$ and no better result is possible. The complexity of deep networks in approximating F_d^s to degree ϵ is $O((d-1)\epsilon^{-2/s})$.

An extension to incorporate training and generalization errors is provided in [145, Theorem 4.2].

Theorem 2.1.9 ([222]). For any d, s, and desired tolerance $\epsilon \in (0,1)$, there is a ReLU network architecture capable of approximating the unit ball of $W^{s,\infty}$ to degree ϵ in L^∞ norm such that, for some constant $c = c(s,d)$, $k \leq c(\ln(1/\epsilon+1)$ and $N+M \leq c\epsilon^{-d/s}(\ln(1/\epsilon)+1)$.

The proof goes by construction of an efficiently representatable partition of unity and polynomial approximation. An extension to general Sobolev spaces $W^{s,p}$ is provided by [86]. [222] also provides a lower bound

Theorem 2.1.10 ([222]). Let $f \in \mathbb{C}^2([-1,1]^d)$ be nonlinear. Then for any fixed k, a k-depth neural network approximating f in L^∞ to degree ϵ must have at least $N+M \geq c\epsilon^{-1/2k}$ for $c = c(f,k) > 0$.

Thus, combining Theorems 2.1.9 and 2.1.10, a neural network of unbounded depth performs strictly better in an asymptotic sense than a neural network with k levels at least when $s > 2kd$. Thus, as is highlighted in [86], for high dimensional problems, essentially bounded derivatives of order $s > 2d$ are required for the Yarotsky theory to prove asymptotic superiority of deep networks. Additionally, the dimension of the problem effects the constant c in the complexity bounds in Theorem 2.1.9.

In [201], it is shown that certain networks of depth $\Theta(L^3)$ and $\Theta(1)$ nodes per layer cannot be well-approximated by networks with $O(L)$ layers without $\Omega(2^L)$ complexity. This demonstrates by example the benefits of arbitrarily deep networks over shallow networks. [202] shows another example where shallow networks have an exponentially larger complexity than deep networks.

2.1.4 Interpolation

Often, we are interested in approximating a function f with knowledge of f only at a finite collection of N sample points $X = \{x_j\}_{j=1}^N$. It is well known that for N sample points, there exist a unique Lagrange polynomial $I_{f,X}$ of degree $N-1$ interpolating these points. However, for equispaced points, it is known that the sequence of interpolatory polynomials need not converge uniformly to f. In fact, much more is known.

Theorem 2.1.11 (Faber [67], translated in [207]). For any sequence of interpolation nodes $X_N = \{x_j^{(N)}\}_{j=1}^N$ for $N \geq 1$, there exists a continuous function f such that the interpolatory Lagrange polynomials I_{f,X_N} do not converge uniformly to f.

However, for functions possessing more regularity then mere continuity (even just Lipschitz continuity) there do exist interpolation points for which the interpolatory polynomials converge uniformly. The most famous and widely used of these interpolation points are the famous Chebyshev nodes:

$$x_j^{(N)} = \cos\left(\frac{2j-1}{2N}\right), \quad 1 \leq j \leq N,$$

which are the zeros the Chebyshev polynomials and the projection of equispaced points from the unit circle onto the x axis.

Theorem 2.1.12 ([206]). Let p_n denote the nth interpolatory polynomial on the Chebyshev nodes to a Lipschitz continuous function f. Then p_n converges uniformly to f. Moreover, if f is k-times differentiable with a kth derivative of bounded variation, then $\|f - p_n\|_\infty = O(n^{-k})$. If f is analytic, then $\|f - p_n\| = O(\rho^{-n})$ for $\rho > 1$.

Polynomial interpolation in higher dimensions is a more complicated problem algebraically. For a review, see [74]. A simple case is given by the Chebyshev grid C_N^d, where C_N denotes the Chebyshev points. In which case, there is a unique polynomial of termwise degree not exceeding $N-1$ interpolating f at C_N^d satisfying the error estimates in Theorem 2.1.12. Note that the total term of terms in a polynomial is $M = N^d$, so in terms of the

total complexity M, the error estimates have the form $||f - p_n||_\infty = O(M^{-k/d})$.

2.1.5 Minimum Sobolev Norm (MSN)

As we saw in the previous section, the interpolatory polynomial of minimal degree has nice convergence properties if the function being approximated is sufficiently well-behaved and the interpolatory nodes are chosen judiciously (e.g. the Chebyshev nodes). Unfortunately, in many practical applications, we must deal with samples of a function where they are given, and cannot insist, for example, that these sampling points be the Chebyshev nodes. To address this problem, we relax the condition that the interpolatory polynomial have minimal degree. Given N sample points $\{x_j\}_{j=1}^N$, there are infinitely many polynomials of degree $M > N - 1$ interpolating f at these points, so the problem of determining a polynomial interpolant becomes *underdetermined*. As usual when dealing with an underdetermined problem, we shall seek a solution of minimal-norm. In this case, we shall seek the solution of minimum Sobolev norm (MSN) in particular, which can be interpretted as finding the "smoothest" solution that interpolates the data.

We are interested in approximating a function $f : [-1, 1]^d \to \mathbb{R}$ given a collection of samples $X_1, X_2, X_3, \ldots$, $|X_n| \to \infty$ as $n \to \infty$. The total collection of all samples is denoted by X, and is referred to as the *interpolation matrix*. For theoretical purposes, it is convenient to convert our function $f : [-1, 1]^d \to \mathbb{R}$ to a function f° on the torus $\mathbb{R}^d/(2\pi\mathbb{Z})^d$ by $f^\circ(\theta_1, \ldots, \theta_d) = f(\cos\theta_1, \ldots, \cos\theta_d)$.

Theorem 2.1.13 ([38]). Let $s > d/2$, $f \in H^s$, and let X be the interpolation matrix. Denote by $\eta_N := \inf_{x,y\in X_N, x\neq y} |x - y|$ the minimum intersample distance. Then there exists $M_N = O(\eta_n^{-1})$ such that the polynomial p_N of degree M_N interpolating the points function f at the interpolatory nodes X_N converges pointwise with error estimate

$$|p_M(x) - f(x)| \leq C\delta(x)^{\min(s-d/2,1)},$$

where $\delta(x) := \inf_{y\in X_N} |x - y|$.

Note that Theorem 2.1.13 refers to a particular construction, and that there are much stronger existence results that give the degree of approximation comparable to the best degree of approximation achievable by trigonometric polynomials.

As a practical note, the MSN scheme proceeds as follows (shown here for $d = 1$). Consider a function $f(x)$ with samples $f(x_i)$. Let the interpolant $p(x)$ to $f(x_i)$ be given by:

$$p(x) = \sum_{m=0}^{M-1} a_m \cos\left(m \cos^{-1} x\right) = \sum_{m=0}^{M-1} a_m T_m(x) \tag{2.4}$$

That is, we choose a basis composed of trignometric Chebyshev polynomials (for convenience and numerical reasons). In this basis, we let V be the Chebyshev-Vandermonde matrix:

$$V = \begin{bmatrix} \cos\left(0 \cos^{-1} x_0\right) & \cos\left(1 \cos^{-1} x_0\right) & \ldots & \cos\left((M-1) \cos^{-1} x_0\right) \\ \cos\left(0 \cos^{-1} x_1\right) & \cos\left(1 \cos^{-1} x_1\right) & \ldots & \cos\left((M-1) \cos^{-1} x_1\right) \\ \ldots & \ldots & \ldots & \ldots \\ \cos\left(0 \cos^{-1} x_{N-1}\right) & \cos\left(1 \cos^{-1} x_{N-1}\right) & \ldots & \cos\left((M-1) \cos^{-1} x_{N-1}\right) \end{bmatrix},$$

a the vector of coefficients a_m, D_s a diagonal matrix of the filter/scaling coefficients $D_s(m) = \frac{1}{(1+m)^s}$ used to penalize the derivative, and f be the vector of samples $f(x_i)$.

The MSN interpolant is given by solving the optimization problem:

$$\min_{Va=f} \|D_s a\|_2^2. \tag{2.5}$$

The solution to this problem is given by:

$$\begin{aligned} p(x) &= V(x) D_s^{-2} V^T (V D_s^{-2} V^T)^{-1} f \\ &= V(x) a \end{aligned} \tag{2.6}$$

where $V(x)$ is a row of the matrix V, computed at x. Note that direct computation of

Eq. 2.6 is likely to be numerically unstable, and solvers based on orthogonal factorizations (QR) are used in practice.

One challenge with the conventional MSN approach is that it seeks a perfect interpolation of the provided data. When dealing with undersampled representations of data, or real-world noisy data, this may cause the required polynomial order to be very high. Thus, a useful extension is the generalized minimum Sobolev norm (GMSN) solution, which can be formulating by introducing a parameter θ that controls the accuracy of the approximation, and a window W that weights the point-wise approximation error, as [35]:

$$\min_{a} \ \sin^2(\theta)\|D_s a\|_2^2 + \cos^2(\theta)\|W(Va - f)\|_2^2 \tag{2.7}$$

The first term in the above formulation corresponds to choosing a smooth interpolant and the second term corresponds to choosing a good approximant to the given set of samples. The parameter θ can be chosen such that $\tan\theta \approx \frac{\sigma}{\|p\|_s}$, where σ is the standard deviation of noise, or also set to zero as $\sin\theta \approx 0$. Eq. 2.7 can be posed as the least squares system:

$$\min_{a_M} \left|\left| \begin{bmatrix} \sin(\theta) I \\ \cos(\theta) V D_s^{-1} \end{bmatrix} D_s a_M - \begin{bmatrix} 0 \\ \cos(\theta) f \end{bmatrix} \right|\right|_2 \tag{2.8}$$

with the corresponding solution for the interpolant:

$$p(x) = \cos^2(\theta)(\sin^2(\theta) D_s^2 + \cos^2(\theta) V^T W^2 V)^{-1} V^T W^2 f \tag{2.9}$$

Again, a direct solution as above is numerically unstable, and a stable solver based on orthogonal factorizations should be constructed instead [35].

There are many practical implications of seeking the MSN solution. First, while it appears we are over-parameterizing the model via an underdetermined system, the Golumb-Weinburger principle tells us that the number of free parameters is determined by the normal equations which is always a $N \times N$ system for N training data points [77, 39].

Thus, we are robust to egregiously overfitting, and we gain additional regularization by controlling the smoothness of the interpolant. Second, the location of the data points is not a requisite for convergence, so the MSN solution will converge to the true solution at limit points of the training data. Thus, the method can be applied to sparse data (although approximation guarantees far away from the provided data are minimal, especially in high dimensions). Third, with numerically stable techniques to compute the MSN solution, the method is guaranteed to converge as we add more data points. On this last point, the MSN method has been shown as an effective solution for various approximation problems, including approximation of the Runge function and other challenging problems [35, 36, 38, 32, 37, 82].

2.2 MSN Polynomial Networks

With the requisite theoretical tools in place, we now outline the construction of networks representing high-dimensional polynomials. Many of the concerns here are practical in nature, because our ultimate goal is to develop set of techniques that allows us to extend and validate our theory on practical problems such as image reconstruction and image recognition. On this note, in Section 2.2.2 we define the various constituent functions we will use to construct approximations, with emphasis on frames amenable to direct and efficient Sobolev-norm penalization. In Section 2.2.2 we outline several structured sub-sampling techniques for generating polynomial terms in high-dimensions as a practical means to *avoid* (but not escape) the curse of dimensionality in a single-layer. In Section 2.2.3 we introduce deep (multi-layered) networks as a means to escape the curse of dimensionality when the compositional structure of the target function is known, including an extension of recent theory to objective functions with a Sobolev-norm penalty amenable to gradient-based optimization. In Section 2.3 we provide promising numerical results that indicate the benefit of depe networks in ameliorating the curse of

dimensionality, especially with respect to the data rate.

2.2.1 Choice of Basis and Frames

When constructing an interpolant, there are many choices one can make. In general, we prefer to pick basis functions that are well matched to the target function, so that a small number of coefficients can be used (i.e. $\arg\min_{Va=f} ||a||_0$). For a general class of target functions, it is typically considered optimal to pick a set of linearly independent functions (i.e. a basis), so that there is limited re-use or redundancy in the required number of non-zero coefficients. However, in some instances it is desirable to use a set of linearly *dependent* functions (a frame) to achieve more stable or efficient representation. In this work, we will not distinguish between these cases, but we will assume that the constructed interpolation matrix (e.g. Vandermonde matrix V) has full row rank. In addition, we will work primarily in finite-dimensional input and output spaces, and will only consider interpolates of finite degree. This is amenable to practical implementation via high-dimensional numerical linear algebra techniques.

In particular, we will first construct the interpolation matrix V based on the basis or frame of choice. In the case of interpolation, we look for the solution to the system:

$$V(x)a = f(x) \tag{2.10}$$

where a is the matrix of coefficients and f is the vector of function evaluations at the sample points x_i corresponding to the rows of V. For simplicity we drop the the notation $V(x), f(x)$ in favor of V, f.

The MSN solution can be expressed as:

$$p(x) = VD_s^{-2}V^T(VD_s^{-2}V^T)^\dagger f \tag{2.11}$$

where we have traded the inverse of $VD_s^{-2}V^T$ used in Eq. 2.6 for the pseudo-inverse, and

we define the D_s appropriately to penalize the derivative or the roughness of the resulting interpolant.

Similarly, the GMSN solution with parameters θ, W can be expressed as:

$$p(x) = \cos^2(\theta)(\sin^2(\theta)D_s^2 + \cos^2(\theta)V^T W^2 V)^\dagger V^T W^2 f \tag{2.12}$$

Note that we are not suggesting this as a viable computational formula. More stable algorithms can be derived using RRQR and similar orthogonal factorizations, in favor of spectral approaches (e.g. SVD) which generally have worse complexity when $M > N$.

We will now give a few formulas for V and D_s.

Monomial Basis

The monomial basis, described in 1D as the moments $\{1, x, x^2, x^3, ...\}$, is perhaps the most familiar basis used by engineers and mathematicians. In d dimensions, we can define the basis as the set:

$$\left\{ g_i = \prod_{j \in J_i}^{K} x_j \mid i \in \mathbb{N}_{[0,S]} \right\} \tag{2.13}$$

where g_i is the ith monomial polynomial term of maximum degree K, $J_i \in \mathbb{Z}^K_{[0,d]}$ is a multi-index indicating the multiplicity of each variable, $x \in \mathbb{R}^d$ is the variable indexed as $x_j \; \forall \; j \in [1, d]$, and $S = \binom{d+K}{K}$ is the total number of terms in a polynomial of maximum degree K. These terms are evaluated for the various scattered data, and the corresponding columns are stacked up to form the Vandermonde interpolation matrix V. Note that the ordering of the columns is non-unique, even though fast algorithms for construction and evaluation may want to exploit the structure of V.

We define D_s vector based on the degree of each term, as:

$$D_s = \Big\{ \frac{1}{(1 + \sum J_i)^s} | i \in [1, S] \Big\} \tag{2.14}$$

Unfortunately, the monomial basis is known to be numerically unstable [205, 111]. Instead, we will use an orthogonal basis.

Chebyshev Basis

Chebyshev polynomials of the first kind are a family of orthogonal functions on the interval $[-1, 1]$, expressed in 1D using the recurrance relation:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \end{aligned} \tag{2.15}$$

where n is the degree of the term. Applying the change of variables $x = \cos(\theta)$, $\theta \in [0, 2\pi]$, we see that the Chebyshev polynomials and trignometric functions are equivalent on the unit interval:

$$T_n(x) = \cos(n \arccos x) = \cos(n\theta) \tag{2.16}$$

We will use this representation for convenience. We can express d dimensional Chebyshev polynomials as the product of d 1D Chebyshev polynomials, as the set:

$$\Big\{ T_{J_i} = \prod_{k=1}^{d} T_{J_i(k)}(x_k) \mid i \in \mathbb{N}_{[0,S]} \Big\} \tag{2.17}$$

where $J_i \in \mathbb{Z}^d$ s.t. $||J_i||_1 <= K$ is a multi-index indicating the degree of the d 1D polynomials in the ith term, and K is the max degree of the multivariate polynomial term. Note that there are still $S = \binom{d+K}{k}$ terms, and the D_s vector is still defined as in Eq. 2.14.

Besides convenience in direct control over the derivative, as measured via the decay in Fourier coefficients, the trignometric Chebyshev basis has also been shown to provide superior numerical stability. We will use this basis for the majority of our experiments.

"Neural" Sigmoid/ReLU Basis

Following our discussion of neural networks, we note that one can also consider this to be a basis. In the original definition of neural networks, the network output is the simple sum of a set of neurons. Neurons are defined via a simple inner product between the input vector and a coefficient vector, followed by an activation function σ (see Definition 2.1.3). In [127] it is shown that this formulation is "universal" (i.e. that it can approximate any function) as long as σ is chosen to be non-polynomial.

However, this technique does not benefit from the direct solver for a system $Va = f$, since it is instead $\sigma(A, a) = f$. Instead, the task of finding the coefficients a is often posed as a convex problem and solved using gradient descent or similar scheme. In fact, because for some finite number of neurons (i.e. the number of columns in a) the coefficients depend on the training data, neural networks were touted as being unique for their ability to "adapt to the data" via so-called "data-dependent" basis functions [19, 20].

Of course, it is debatable whether this offers a fundementally different kind of representation compared to polynomials. As Candes points out in [27], a number of different functions can be used for approximation, sometimes with even better properties than neural networks for specific function classes. For example, in [26, 28] Candes introduces ridgelets as a competitive basis with some superior properties to wavelets in some cases. In all these cases, the basic idea is to take an inner product between the data point $x \in \mathbb{R}^d$ and a coefficient vector, and to generate moments of this sum via a so-called "squashing" function.

2.2.2 Single-layer Networks

Regardless of the choice of basis, a direct implementation of interpolation suffers from the curse of dimensionality because for general functions defined on the d dimensional hypercube an infeasible number of expansion terms and data points are needed. Therefore, some sensible techniques must be used to judiciously select which terms to include in practice, which undoubtedly depends on the morphology of the data and the target function.

Classically in low dimensions this problem was addressed by picking a suitable basis for each problem. However, in high dimensional problems (e.g. image recognition), it is difficult to understand and visualize the morphology of the input-output map. As a result, practitioners are left to picking an arbitrary basis with generic approximation properties, and carefully selecting or tuning terms within that set with a set of heuristics.

In the following, we outline a few simple choices or algorithmic tools for picking terms in a "sensible" way, understanding that this choice is highly problem dependent. In each case, we will assume the input is a vector $x \in \mathbb{R}^d$, referred to as the input pixels. Input pixels are processed in "layers" that generate terms and output one or several polynomials of the input pixels. Each polynomial that is computed is considered a "node" in the network. In single-layer polynomial networks, the output of the first set of nodes is concatenated and taken as the output of the network (i.e. it should match the dimension of the target function). Note that this grouping differs from some of the nomenclature used to describe single-layered and deep *neural* networks (DNNs), where an indefinite number of neurons can be linearly combined to form each dimension of the output node. To be clear, we now define a few types of layers that we will use to construct single-layer, multi-layer, and deep-compositional MSN polynomial networks used in our experiments (Sec. 2.3).

The "Dense" Layer of Degree K

The dense layer produces and sums all possible polynomials (i.e. combinations) of the input pixels. The number of terms depends on the desired degree K (as defined in Section 2.2.1). The number of polynomials computed by a dense layer is conformal with the output dimension of the target function. The feedforward computation can be expressed succinctly as $V(x)a \rightarrow f$, with the appropriate dimensions on a and f. When it is desirable for each polynomial to have a different degree, zeroes are inserted at appropriate locations in $V(x)$ for feed-forward evaluation, although a more complicated routine is required for training. As mentioned, this layer suffers from the curse of dimensionality with respect to the number of terms (rows in a).

The "Sparse" Layer of Max-Degree K

The sparse layer produces and sums a subset of the terms generated by the dense layer. This can be achieved in a variety of ways, including by random subsampling of the terms. The subsampling can also be structured, e.g. organized or prioritized by the degree of each term. There is no minimum number of terms, while the maximum is the same as the corresponding dense layer of the same maximum degree K. In our implementation, we use a Python generator to collect, shuffle, and sample terms for each desired degree. Although this is an implementation detail, we note that in very high dimensions designing a good sampling strategy for such terms is not trivial. For example, the naive technique does not produce well-spaced samples when the degree is large, since the required buffer-size is very large:

```
1 def random_combinations_with_replacement_v1(iterable, degree):
2     num2get = nCr_with_replacement(len(iterable),degree)
3     combos = itertools.combinations_with_replacement(iterable,degree)
4     for k in shuffle_generator(combos, bufsize=200000): yield k
```

where `iterable` is an vector of indices corresponding to the pixels of x, and `degree` is the desired polynomial degree.

A better technique, although extremely slow, is to create a set and sample at runtime, as:

```
def random_combinations_with_replacement_v2(x, degree):
    num2get = nCr_with_replacement(len(iterable),degree)
    unique = set()
    while len(unique) < num2get:
        tmp = random_combination_with_replacement(iterable,degree)
        tmp_tup = tuple([tuple(p) for p in tmp])
        if tmp_tup not in unique:
            unique.add(tmp_tup)
            yield tmp
        #
    #
```

In low-dimensions and for reasonably low-degree polynomials, of course, either technique can be used effectively to enumerate all combinations (in lexicographic order, if desired).

Normalization and Batch-Normalization

Without loss of generality, generated polynomial terms can be scaled with a diagonal matrix Y; its feedforward computation can be expressed as $VYa \to f$. Shifting is also possible as $(VY + B)a \to f$, but some care needs to be taken to preserve the column-rank of the LHS. These factors can be picked a priori, e.g. heuristically or via batch normalization on a subset of training data, and then frozen for all subsequent evaluation or training. In general, when there are free parameters in Y or B, we would consider the normalization as its own layer. Note that when the scale or shift factors are a function of the input data itself e.g. $Y = \phi_1(V)$, $B = \phi_2(V)$, we get more complicated higher-order

terms. In the general case, we may want to allow normalization of this form to yield a rational function of the input x, rather than a polynomial.

Dropouts and K-sparse Weights

The technique of dropouts can be applied by picking a sparse binary-valued diagonal matrix Y with $Y_{i,i} \in \mathbb{Z}_{\{0,1\}}$ as a simple normalization layer. Although this technique is typically coupled with gradient-based optimization, we can understand the operation as increasing the number of data points (number of rows in V) via data augmentation, when all possible dropout instances are realized. The added data matches the original data up to K of its M moments (polynomial terms). However, in this fixed-data setup we would require an intelligent training routine that would respect the information sparsity and allow the remaining moments to be free, rather than pinning them at zero.

Controlling the In-Degree

Not all the input pixels need to be used in each node of a layer. Instead, the pixels can also be grouped into smaller bins, prior to selecting terms via the aforementioned dense or sparse polynomial layers. This has the benefit of letting each node be low-dimensional. We refer to the number of pixels entering a node as the "in-degree". Interpolation of low-dimensional polynomials is a well studied subject, and the numerical tools developed for MSN approximation can be applied directly. However, because these choices are fixed at the time of construction, an arbitrary choice may not yield a good result for a general class of target functions without structure. Note that in conventional neural networks, even if each neuron selects a subset of the input pixels (e.g. encouraged via a $\mathcal{L}_0$ or $\mathcal{L}_1$ penalty on the weights), summing these nodes within a layer increases the in-degree of the output "node". In a non-polynomial network, the in-degree of any node i can be computed by the number of non-zero entries in the gradient of the node-output $\hat{f}_i$ with respect to pixel inputs x. As we will argue in Section 2.2.3, having low in-degree may

also ease the curse of dimensionality in scenarios with sufficiently *representative*, albeit sparse, training data.

The Local Neighborhood Layer

For data that is locally correlated (e.g. images), input pixels can be organized into bins based on their location in pixel-space. We call these groups, whether they are locally contiguous or disjoint, as local pixel "neighborhoods", and can leverage them as a strategy to sample relevant polynomial terms of high dimensional inputs using the aforementioned dense and sparse term generators. While thus far we have only considered a vectorized version of the input $x \in \mathbb{R}^d$, utilizing the intrinsic format of the sampled data x' can be beneficial for uncovering and leveraging its structure in approximation tasks. For N dimensional Euclidean data such as images or griddata $x' = \mathbb{R}^{\prod_{k=1}^{N+1} D_k}$ with $\sum_k D_k = N$, where the last dimension D_{N+1} represents the number of channels at a given voxel index. For 2D natural images with one or more intensity or color channels, $x' = \mathbb{R}^{D_1 \times D_2 \times \mathbb{N}_{\text{channels}}}$. We will use 2D images as an illustrative example.

The local neighborhood can be based on a fixed-size or location-dependent stencil, with fixed or location-dependent sparsity. Each neighborhood will generate polynomial terms with respect to the input pixels or the "support" defined by the stencil. It is left to the algorithm or network designer to decide how to use these terms. For example, in a single-layer network with one output node, the terms produced by each neighborhood can be linearly combined. In a single-layer network with several output nodes, the terms produced by each neighborhood can be further grouped and linearly combined into each output node (without going to a multi-layer network). In either case, the network retains the feedforward format $Va \to f$ without restriction on the number of unique stencils or number of output nodes. However, notice that the in-degree is likely to increase as the number of neighborhoods contributing to an output node increases.

To keep the in-degree small, we can enforce a sparsity condition on the stencil

corresponding to each neighborhood. However, this fixes the minimum number of output nodes. For some applications, single-layer networks multiple-output nodes are desirable and sufficient. However, in many applications we would like to uncover more complicated groupings of the input pixels or neighborhoods. In this case, we can take the output nodes of a layer to be the input nodes of the next layer. This technique helps define multi-layer deep compositional networks that retain low in-degree at each node (Sec. 2.3).

The "Convolutional" Layer

In the local neighborhood layer, when the same stencil is used throughout the image (e.g. by sweeping with some stride-length), and the resulting polynomial terms are identical for each neighborhood, one can share some or all of the polynomial parameters (e.g. linear-combination coefficients) across neighborhoods. Sharing all the parameters yields a "convolutional" layer, similar in structure to the initial layers of convolutional neural networks (CNNs). The feedforward computation can be expressed via the block matrix expression $[V \cdot \tilde{a}_1, V \cdot \tilde{a}_2, ..., V \cdot \tilde{a}_k]^T \rightarrow [f_1, f_2, ..., f_k]^T$ with $\tilde{a}_k = R_k \cdot a$, where a are the shared polynomial coefficients, f_k is the (possibly) vector-valued output of one neighborhood, and R_k is the k-th shift/displacement or rotation operator resulting from the stencil sweep. With appropriate reformatting, this layer can be efficiently vectorized for fast computation on CPUs and GPUs. Neighborhoods of different sizes may be used, but the encoding will be slightly more expensive. Notice that in this view, convolutional layers increase the data-rate by a multiple of the number of neighborhoods.

2.2.3 Deep Compositional Networks

Still, single-layer networks have limitations. In particular, they suffer from the curse of dimensionality when the value of an output node must depend on more than a small subset of the input pixels. This high in-degree poses undue burden on the network designer to enumerate or sample relevant terms from a potentially large pool. While single-layer

neural networks sought to ameliorate this issue by proving dimension-independent bounds for the degree of approximation for some functions (Theorem. 2.1.5), the constants are typically too large to be feasible in practical problems. Moreover, these early works fail to explain the practical success of deep, multi-layer networks over shallow networks.

Recent work has considerably improved our understanding of the benefits of deep networks over shallow networks. [222] showed that for functions possessing bounded derivatives of very high order relative to the dimension, very deep networks would asymptotically outperform networks of any fixed depth. [201] demonstrated the existence of deep neural networks with $\Theta(n^3)$ layers which could not be approximated by shallow networks of $\Theta(n)$ layers without exponentially many nodes.

In [149, 143, 144] it is argued that deep networks are better than shallow networks at approximating functions with a compositional structure—that is, functions such as $f(x, y, z, w) = h_3(h_1(x, y), h_2(z, w))$ and the like. They model a compositional function f as a directed acyclic graph (DAG) $\mathcal{G}$. Directed edges from one node to another represent the output of one function being used as an input to another function. Source nodes S represent inputs to the function and the (unique) sink node represents the output of the function. Letting V denote non-source nodes, for each $v \in V$, there exists a function f_v taking d_v inputs where d_v is the in-degree of f_v. The entire collection $(S, V, \{f_v\}_{v\in V}, \mathcal{G})$ is referred to as a $\mathcal{G}$-function of f. We shall refer to $d := \max_v d_v$ as the degree of the q-dimensional $\mathcal{G}$-function f. In this framework, a deep neural network is just a $\mathcal{G}$-function where each nodal function f_v is a linear combination of shifted, scaled activation functions. The total number of summands in all nodal functions is the complexity of the network.

Proposition 2.2.1 ([149, 143, 144]). Let f be a $\mathcal{G}$-function where $\|f_v\|_{W^{k,\infty}} \leq 1$. Then there exists a constant $c(\mathcal{G}) > 0$ such that for every $\epsilon > 0$, there exists a deep network g such that $\|f - g\|_{L^\infty} \leq \epsilon$ and complexity $\leq c(\mathcal{G})\epsilon^{-d/k}$.

In this way, deep networks avoid the curse of dimensionality for compositional functions in that the exponential dependence on the dimension q is removed. Instead, we have an

exponential dependence on the $\mathcal{G}$-function degree d, which we can take to be considerably smaller than q, $d \ll q$. Moreover, it is shown in [145] that it is possible to train such a deep network from noisy data by solving an appropriate optimization problem for a function with known compositional structure but unknown nodal functions $\{f_v\}$. For simplicity of analysis, the domain is taken to be the q-dimensional torus $\mathbb{T}^q$ and the activation function is taken to be $t \mapsto \cos t$. Thus, in this framework, approximation by neural networks is reduced to the well-studied subject of approximation by trigonometric polynomials. For a collection $N = \{N_v\}_{v \in V}$, we denote $\mathcal{G}\mathbb{H}_N$ the class of $\mathcal{G}$-functions g for which g_v is a trigonometric polynomial of degree not exceeding N_v.

Theorem 2.2.2. Suppose we are given the values y_i of a $\mathcal{G}$-function f with some error $y_i = f(x_i) + \epsilon_i$. Assume that each f_v satisfies the Lipschitz condition $\|\nabla f\|_{L^\infty} \leq L$ and $\epsilon := \max_i \epsilon_i \leq L$. Then there exists constants $C(\mathcal{G}), c(\mathcal{G}, L) > 0$ such that for $N_v \geq C(\mathcal{G})\eta(\mathcal{C}_v)^{-1}$, for the loss function:

$$\mathscr{L}[g] := \max_i |y_i - g(x_i)| + \sum_{v \in V} \frac{1}{N_v} ||\nabla g_v||_{L^\infty},$$

defined for every $\mathcal{G}$-function g, we have:

$$\min_{g \in \mathcal{G}\mathbb{H}_N} \mathscr{L}[g] \leq c(\mathcal{G}, L) \left[\epsilon + \sum_{v \in V} \frac{1}{N_v} ||\nabla f||_{L^\infty} \right]$$

and with $h := \arg\min \mathscr{L}[g]$,

$$|h(x) - f(x)| \leq c(\mathcal{G}, L) \left[\epsilon + \sum_{v \in V} \frac{1}{N_v} ||\nabla f||_{L^\infty} \right] + (c(\mathcal{G}, L) + \epsilon \max_v N_v)\delta(x),$$

where $\delta(x) = \min_i |x - x_i|$ is the distance to nearest sample.

In particular, this theorem shows that provably good generalization error can be obtained by minimizing the error on the training data with a gradient penalty.

Generalization Error for Deep Minimum Sobolev Networks

An important step in actualizing the ideas presented in Theorem 2.2.2 is replacing the L^∞ gradient penalization to a method more amenable to training with existing methologies such as stochastic gradient descent. To this end, we shall prove a modified version of Theorem 2.2.2 which uses the Sobolev norms $W^{2,s}$ rather than the gradient's L^∞ norm. This shall introduce a significant benefit in that, for a function f expressed as a Chebyshev series,

$$f = \sum_{k \in \mathbb{N}_0^q} a_k T_k,$$

the Sobolev norm of f can be defined to be $||f||_s^2 := \sum_{k \in \mathbb{N}_0^q} (1 + |k|)^{2s} a_k^2$. This norm differs from the one defined in the background but is equivalent in the sense that $C_1 \, ||f||_{W^{2,s}} \leq ||f||_s \leq C_2 \, ||f||_{W^{2,s}}$ for constants $C_1 = C_1(s, q)$ and $C_2 = C_2(s, q)$. For a function f on the torus, there is a simple relation between the Sobolev norm and its Fouerier coefficients,

$$f = \sum_{k \in \mathbb{Z}^q} a_k e^{ik\bullet} \implies ||f||_s = \sum_{k \in \mathbb{Z}} (1 + |k|)^{2s} |a_k|^2.$$

There is a natural equivalence between functions f on $[-1, 1]^q$ and functions f° on the torus $\mathbb{T}^q$ by $f^\circ(\theta_1, \ldots, \theta_q) = f(\cos\theta_1, \ldots, \cos\theta_q)$. Under this correspondence, Chebyshev expansions are equivalent to Fourier cosine expansions. For this reason, for the remainder of the section, we shall focus on functions defined on the torus.

We shall need the following Lemma.

Lemma 2.2.3 ([33, 38]). Let $f \in W^{2,s}(\mathbb{T}^d)$ and let Y be a collection of points in $\mathbb{T}^d$, $s > d/2$. Define $\eta := \min_{x,y \in Y} |x - y|$. Then there exist constants $c = c(d, s)$ and $C = C(d, s)$ such that there exists a trigonometric polynomial p of degree not exceeding $C\eta$ such that $p(y) = f(y)$ for every $y \in Y$ and $\|p\|_{W^{2,s}} \leq c\|f\|_{W^{2,s}}$.

For a $\mathcal{G}$-function g and a collection $\mathcal{C} \subseteq \mathbb{T}^q$ of points, denote by $\mathcal{C}_v$ the inputs to g_v as the collection of points $\mathcal{C}$ is fed through the $\mathcal{G}$-function for each $v \in V$,

$$\eta_v(\mathcal{C}) := \inf_{x,y\in\mathcal{C}_v} |x-y|.$$

When the input set $\mathcal{C}$ is clear from context, we shall simply write η_v. We also need the following result. Call a $\mathcal{G}$-function *streamlined* if the nodes of V can be broken into disjoint levels $1, 2, \ldots, \ell$ such that if there is a directed edge from u on level k to v on level j, then $j = k+1$.

Lemma 2.2.4. Given a $\mathcal{G}$-function g, g can be converted to an equivalent streamlined $\mathcal{G}$-function G with node set $\tilde{V} \supseteq V$ such that $G_v = g_v$ if $v \in V$ and $G_v = \mathrm{id}$ if $v \notin V$. G can be written as a composition $G = G_\ell \circ \cdots \circ G_2 \circ G_1$, where

$$||\nabla G_k||_{L^\infty} \leq c \left(1 + \sum_{v \text{ level } k} ||\nabla g_v||_{L^\infty}\right)$$

for some constant $c = c(\mathcal{G})$. If g is already streamlined and $g = G$ or $k = L$, this estimate can be improved to

$$||\nabla G_k||_{L^\infty} \leq c \sum_{v \text{ level } k} ||\nabla g_v||_{L^\infty}.$$

Theorem 2.2.5. Suppose we are given the values y_i of a $\mathcal{G}$-function f with some error $y_i = f(x_i) + \epsilon$ at M points $\mathcal{C} = \{x_1, \ldots, x_M\}$. Assume that each $f_v \in W^{2,s}$ for $s > d/2+1$. Then there exist a constant $C = C(\mathcal{G}, s)$ such that for $N_v \geq C\eta_v^{-1}$, the loss function

$$\mathscr{L}[g] := \frac{1}{M}\sum_i |y_i - g(x_i)|^2 + \alpha^2 ||g||_s^2,$$

defined for every $\mathcal{G}$-function g, we have

$$\min_{g\in\mathcal{G}\mathbb{H}_N} \mathscr{L}[g] \leq C\left[\epsilon^2 + \alpha^2 ||f||_s^2\right], \tag{2.18}$$

where $\epsilon := \max \epsilon_i$, and with $h := \arg\min \mathscr{L}[g]$,

$$|h(z) - f(z)| \leq C\left[\sqrt{M}(\epsilon + \alpha ||f||_s) + \left(1 + \frac{\epsilon}{\alpha} + ||f||_s\right)^\ell \delta(z)\right]. \tag{2.19}$$

Proof. Throughout, we shall use $C = C(\mathcal{G}, s)$ to denote arbitrary constants depending on $\mathcal{G}$ and s. We begin by showing (2.18). Lemma 2.2.3 says there exists a constant C and a polynomial p_v of degree not exceeding $C\eta_v^{-1}$ such that p_v agrees with f_v on $\mathcal{C}_v$ and $||p_v||_s \leq C\,||f_v||_s$. Then the $\mathcal{G}$-function p agrees with f on $\mathcal{C}$ and thus has

$$\mathscr{L}[p] = \frac{1}{M}\sum_i |y_i - p(x_i)|^2 + \alpha^2\,||p||_s^2 \leq \epsilon^2 + C^2\alpha^2\,||f||_s^2 \leq C\left[\epsilon^2 + \alpha^2\,||f||_s^2\right].$$

This establishes the estimate (2.18). Let $h := \arg\min_g \mathscr{L}[g]$. For any $z \in \mathbb{T}^q$ and $w = x_i := \arg\min_{w\in\mathcal{C}} |w - z|$, we have

$$|h(z) - f(z)| \leq \underbrace{|h(z) - h(w)|}_{E_1} + \underbrace{|h(w) - f(w)|}_{E_2} + \underbrace{|f(w) - f(z)|}_{E_3}. \tag{2.20}$$

We now seek to estimate $|h(z) - f(z)|$ by estimating the three terms E_1, E_2, E_3. We note that $|z - w| = \delta(z)$ by definition. Let g denote either f or h. Note by the general Sobolev inequality [66, Thm. 5.6.6], for each $v \in V$, g_v is a Lipschitz continuous function with $||\nabla g||_{L^\infty} \leq C\,||g_v||_s$. Let $g = g_\ell \circ \cdots g_2 \circ g_1$ be an expression of g as a composition as argued for by Lemma 2.2.4. Then by Lemma 2.2.4,

$$\begin{aligned}
||\nabla g_k||_{L^\infty} &\leq C\left(1 + \sum_{v \text{ level } k} ||\nabla g_v||_{L^\infty}\right) \\
&\leq C\left(1 + \sum_{v\in V} ||\nabla g_v||_{L^\infty}\right) \\
&\leq C\left(1 + C\sqrt{|V|}\sqrt{\sum_{v\in V} ||g_v||_s^2}\right) \\
&\leq C(1 + ||g||_s).
\end{aligned}$$

Taking $g = f$, we have $||\nabla f_k||_{L^\infty} \leq C(1 + ||f||_s) \leq C(1 + ||f||_s)$. Thus, by the chain rule,

$$||\nabla f||_{L^\infty} \leq \prod_{k=1}^{\ell} ||\nabla f||_{L^\infty} = C(1 + ||f||_s)^\ell.$$

Thus, by the mean value theorem,

$$E_3 = |f(z) - f(w)| \leq ||\nabla f||_{L^\infty} |z - w| \leq C(1 + ||f||_s)^\ell \delta(z). \tag{2.21}$$

Taking $g = h$, we have $||\nabla h_k||_{L^\infty} \leq C(1 + ||h||_s) \leq C(1 + \sqrt{\mathscr{L}[h]}/\alpha)$. Then, by (2.18),

$$||\nabla h_k||_{L^\infty} \leq C\left(1 + \frac{1}{\alpha}\sqrt{\epsilon^2 + \alpha^2 ||f||_s^2}\right) \leq C\left(1 + \frac{\epsilon}{\alpha} + ||f||_s\right)$$

Thus, $||\nabla h||_{L^\infty} \leq C(1 + \epsilon/\alpha + ||f||_s)^\ell$,

$$E_1 = |h(z) - h(w)| \leq ||\nabla h||_{L^\infty} |x - y| \leq C\left(1 + \frac{\epsilon}{\alpha} + ||f||_s\right)^\ell \delta(z). \tag{2.22}$$

For E_2,

$$E_2 = |h(w) - f(w)| \leq |f(w) - y_i| + |y_i - h(x_i)| \leq \epsilon + \sqrt{M\mathscr{L}[h]} \leq C\sqrt{M}(\epsilon + \alpha ||f||_s). \tag{2.23}$$

Combining (2.21), (2.22), and (2.23) with (2.20) gives (2.19).

□

Compared to Theorem 2.2.2, we pay a penalty of $\sqrt{M}$ for switching from the mean-square error to the maximum error, but in the noise-free case $\epsilon = 0$, this issue can be entirely mollified by taking $\alpha = M^{-(1/2+\gamma)}$ for $\gamma > 0$.

2.3 Numerical Experiments

With the theory and construction of single- and multi-layered polynomial networks in place, we now turn to numerical experiments that seek to validate or uncover important caveats. In particular, while the theory has established approaches for avoiding or even

escaping the curse of dimensionality with respect to the degree of approximation, it is generally silent about the problem of sampling sufficient data in the d dimensional hypercube (i.e. the data rate). Therefore, it is important to ask if the developed theory is useful for practical problems involving sparse data. Our experiments provide initial evidence that the curse of dimensionality with respect to the data rate may be ameliorated with the techniques described in the previous sections.

2.3.1 Image Classification

As a first attempt, we construct MSN networks with a single sparse layer. We train these networks with the GMSN objective using numerical linear algebra techniques, for simplicity. In particular, we set $\lambda = \sin(\theta)$ reasonably high, so that the resulting interpolant is not overly rough. Surprisingly, these single-layer networks work well for MNIST-like image recognition datasets, even with an extremely low data rate. Current results indicate upwards of 95% performance using just 400 samples for each of the 10 classes ($N = 4000$) on MNIST, although the required data rate for FMIST is more than triple that. Interestingly, the generalization performance *increases* as the number of parameters, provided we seek the GMSN solution, i.e. $s > 0$. However, the performance on CIFAR-10 plateaus around 45%, even as the number of parameters and amount of data are maximized within reasonable limits. With some local correlation and data augmentation tweaks, this can be boosted to 54%. Note that these tests were conducted in single-precision.

As shown in Table 2.1, experiments on MNIST-Digits indicate increased performance as the data rate and number of parameters increases. As shown in Table 2.2, experiments on Fashion-MNIST indicate more incremental improvements to performance as the data rate and number of parameters are increased. The accuracy is further improved when local neighborhood techniques are used. Note that the local neighborhood layers utilized here are not convolutional layers, since we are initially restricting ourselves to shallow

1-layer networks.

	# Parameters	# Data	Train Acc.	Val Acc.	Test Acc.
GMSN ($s = 1$)	10K	100/class	100%	89.8%	90.3%
GMSN ($s = 1$)	70K	100/class	100%	91.1%	91.5%
GMSN ($s = 1$)	150K	100/class	100%	**91.6%**	**91.6%**
GMSN ($s = 1$)	10K	400/class	100%	92.2%	92.3%
GMSN ($s = 1$)	150K	400/class	100%	**95.4%**	**95.7%**
GMSN ($s = 1$)	150K	800/class	100%	96.8%	96.7%
GMSN ($s = 1$)	296K	800/class	100%	**98.9%**	**98.6%**

Table 2.1: MNIST-Digits classification accuracy as a function of data and parameters for 1 sparse polynomial layer of max degree 10.

	# Parameters	# Data	Train Acc.	Val Acc.	Test Acc.
GMSN ($s = 1$)	10K	400/class	100%	79.0%	78.3%
GMSN ($s = 1$)	30K	400/class	100%	82.7%	82.2%
GMSN ($s = 1$)	70K	400/class	100%	83.5%	82.8%
GMSN ($s = 1$)	150K	400/class	100%	83.6%	83.0%
GMSN ($s = 1$)	600K	400/class	100%	83.6%	83.0%
GMSN ($s = 1$)	30K	1000/class	100%	82.8%	82.8%
GMSN ($s = 1$)	150K	1000/class	100%	86.0%	85.1%
GMSN ($s = 1$)	300K	1000/class	100%	86.3%	85.4%
GMSN (5x5 neighborhood, $s = 1$)	30K	400/class	100%	83.6%	82.5%
GMSN (5x5 neighborhood, $s = 1$)	296K	800/class	100%	85.8%	86.4%
GMSN (5x5 neighborhood, $s = 1$)	588K	800/class	100%	88.8%	88.6%
GMSN (5x5 neighborhood, $s = 1$)	800K	1000/class	100%	92.3%	93.4%
GMSN (5x5 neighborhood, $s = 1$)	800K	1400/class	100%	94.8%	95.1%

Table 2.2: Fashion-MNIST classification accuracy as a function of data and parameters for 1 sparse polynomial layer.

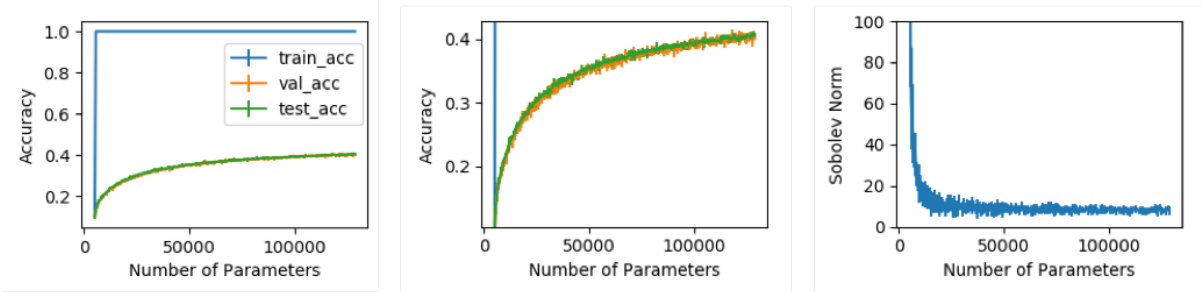


Figure 2.1: CIFAR-10 image classification accuracy as a function of number of parameters for 1 sparse polynomial layer.

Experiments on CIFAR-10 indicate incremental, but monotonic, improvements as the data rate and number of parameters are increased (Fig. 2.1). The performance eventually

plateaus, even after employing local neighborhood, sparsity, and sparse auto-encoder coefficient selection schemes, and is unable to effectively solve the problem with $\geq 95\%$ accuracy for reasonably-sized polynomials.

To deal with the undesirable plateauing effect for CIFAR-10, we switch to multi-layer networks. There are many architectures we can explore here, but in our initial experiments we explore deep compositional MSN networks composed of sparse layers *with low in-degree*. We provide some initial numerical evidence to highlight the benefit of this approach, as outlined in the following (Sec. 2.3.2), with the hope that future work will scale to addressing the data rate issue for image classification.

2.3.2 `max`-d

For our first multi-layer example, we will attempt to approximate the `maximum` or `max` function in d dimensions. We will use the Chebyshev basis to illustrate our points not only for convenience, but because polynomials are not particularly well-suited to the `max` function. The key insight we are exploiting is that the maximum function can be written as a compositional network. We will show that penalizing the Sobolev norm using the objective function in Theorem 2.2.5 not only enables the interpolant to converge to the true function as in the MSN theory, but that this technique works together with the compositional network to ameliorate the curse of dimensionality in deep networks.

In each experiment, we seek the d dimensional polynomial approximation to the function $\max(x_1, x_2, ..., x_d)$. For simplicity, we will switch to the notation $\max_d(x),\ x \in \mathbb{R}^d$. As described in the previous sections, the standard approach is to construct an interpolation matrix V in the basis of choice from the data points x corresponding to the samples $f(x)$. Using this point-data, we will seek the parameters of a polynomial f^* such that $|\hat{f}(x) - f(x)| < \epsilon\ \forall x$. In the following, we will denote number of training points is N, and the number of parameters is M. We will approach the problem of finding the polynomial coefficients as an optimization problem, solved using either using numerical linear algebra

techniques amenable to single-layer networks, or using a gradient descent-like algorithm. This latter technique is applicable to multi-layer networks, but comes at the cost of a hyperparameter on the Sobolev-norm penalty (α^2 as in Theorem 2.2.5) which will adittionally control the convergence rate in terms of the number of training iterations. The point-wise "training" data used to construct the polynomial will be sampled sparsely from the d dimensional hypercube, unless otherwise stated. The generalization error will be measured using the mean-square error and the infinity-norm (maximum) error on two denser samplings of the d dimensional hypercube (denoted "validation" and "test"), unless otherwise stated. Note that these tests were conducted in double-precision.

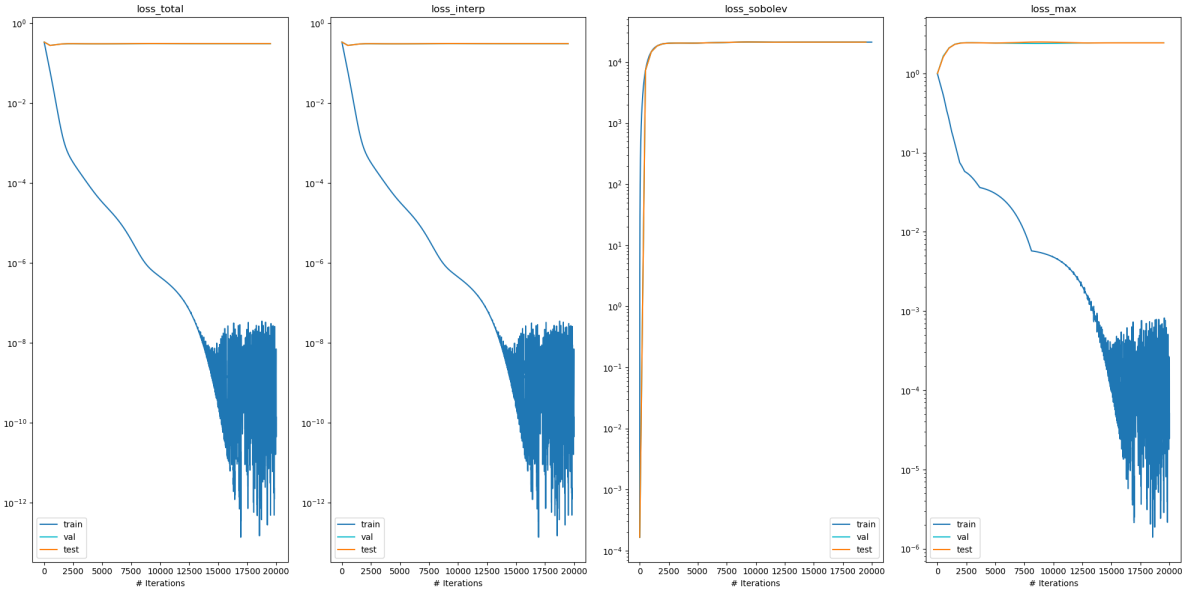
max-2

For $\max_2$ we compare the performance of a least-squares network (# parameters $\leq$ # data points), the underdetermined network (# parameters $>$ # data points), and the MSN solution to the underdetermined network, in comparison to the ground-truth function. All polynomials were computed using a gradient-descent algorithm (Adam optimizer), although they could equivalently be computed with numerical linear algebra techniques. The MSN solution has the best generalization performance.

	# Parameters	Max-Degree	Train MSE	Test MSE	Max Test Error
Least-Squares	100	13	1.3e-5	1.5e-1	2.8e0
No MSN	201	19	**7.5e-9**	3.0e-1	2.4e0
MSN	201	19	9.0e-6	**9.4e-5**	**8.8e-2**

Table 2.3: Approximation of $\max_2$ for different networks and objective functions.

(a) No MSN $\alpha^2 = 0.0$



(b) MSN ($\alpha^2 = 1e-6$)

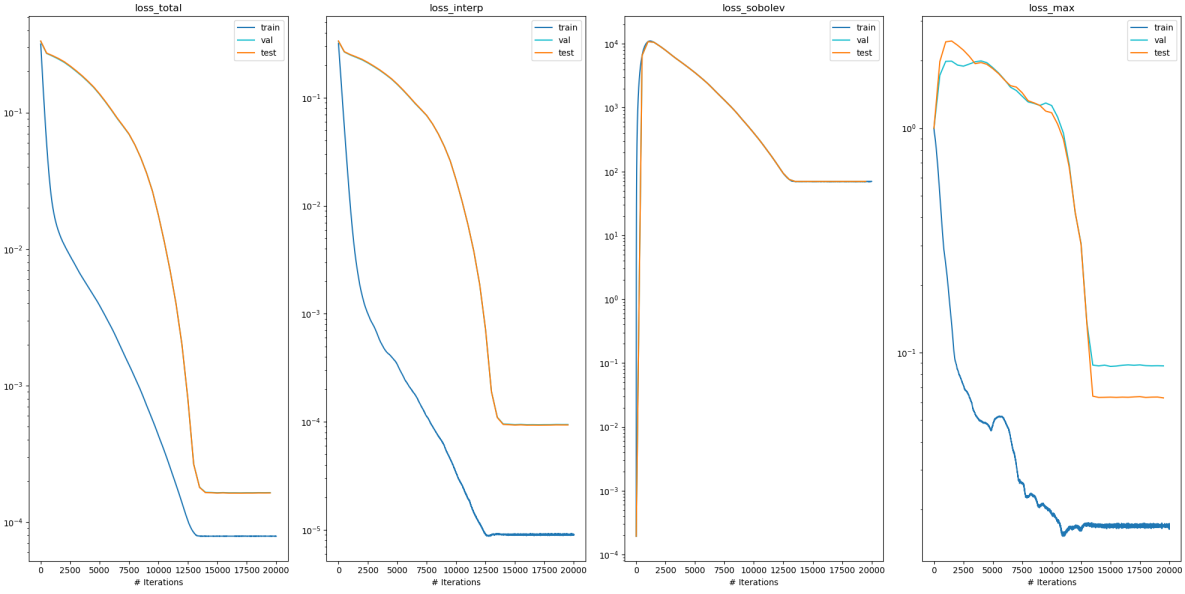
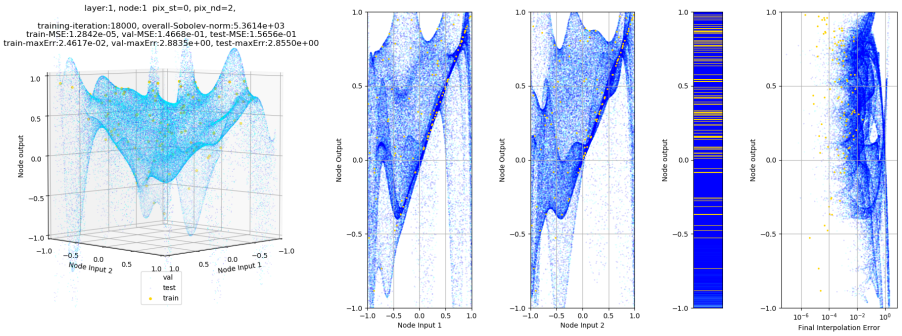
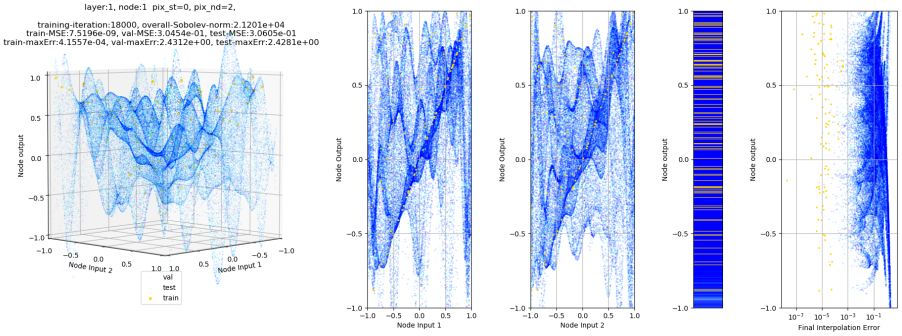


Figure 2.2: Performance vs training iteration for target function: $\max_2$.

(a) Least-Squares



(b) No MSN ($\alpha^2 = 0.0$)



(c) MSN ($\alpha^2 = 1e-6$)

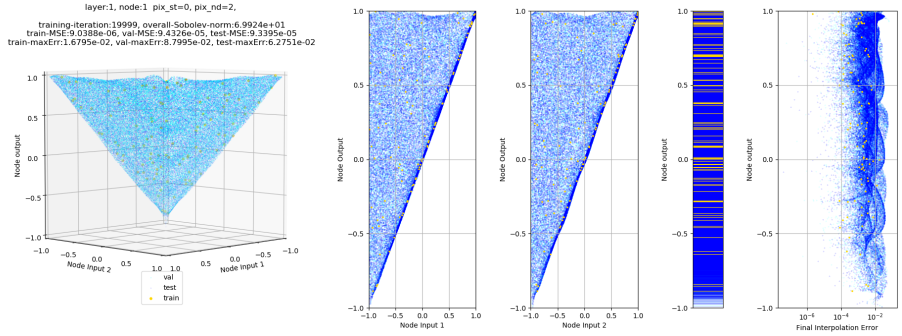


Figure 2.3: Approximation of $\max_2$ for different networks and objective functions, $N = 100$.

max-4

For $\max_4$, we notice that more data is required to get good generalization performance, since the sampling is much more sparse in the 4 dimensional hypercube. We find that the compositional network trained with the Sobolev-norm penalty helps the interior nodal functions to converge faster and with less data than the corresponding compositional network trained only on the mean-square error (MSE). In particular, adding more data allows does not significantly change the shape or value of the interior nodal functions of the deep MSN (DMSN) network (i.e. the nodal functions have converged). Interestingly, we observe that although we are regressing only on the final output of the network, letting the interior nodal functions be free, the nodes seem to converge to functions resembling $\alpha + \beta \max_2$. This is not necessarily predicted by the theory.

Graph	Objective	# Parameters	# Data	Train MSE	Test MSE	Max Test Error
1 Layer	MSE	603	100	**2.9e-16**	4.9e-1	2.0e0
1 Layer	GMSN	603	100	3.8e-5	**5.1e-2**	1.2e0
2 Layer	MSE	603	100	1.3e-8	2.0e-1	1.5e0
2 Layer	DMSN ($s = 1$)	603	100	1.1e-6	1.1e-1	**1.4e0**
1 Layer	MSE	603	1000	8e-3	2.2e0	4.2e1
2 Layer	MSE	603	1000	**3.3e-5**	1.8e-2	2.3e0
2 Layer	DMSN ($s = 1$)	603	1000	6.4e-4	**1.9e-3**	**5.5e-1**
1 Layer	MSE	603	2000	1.6e-2	1.5e-1	3.0e1
2 Layer	MSE	603	2000	**8.1e-5**	4.3e-3	2.5e0
2 Layer	DMSN ($s = 1$)	603	2000	6.2e-5	**8.0e-5**	**1.0e-1**

Table 2.4: Approximation of $\max_4$ for different objective functions and # training data.

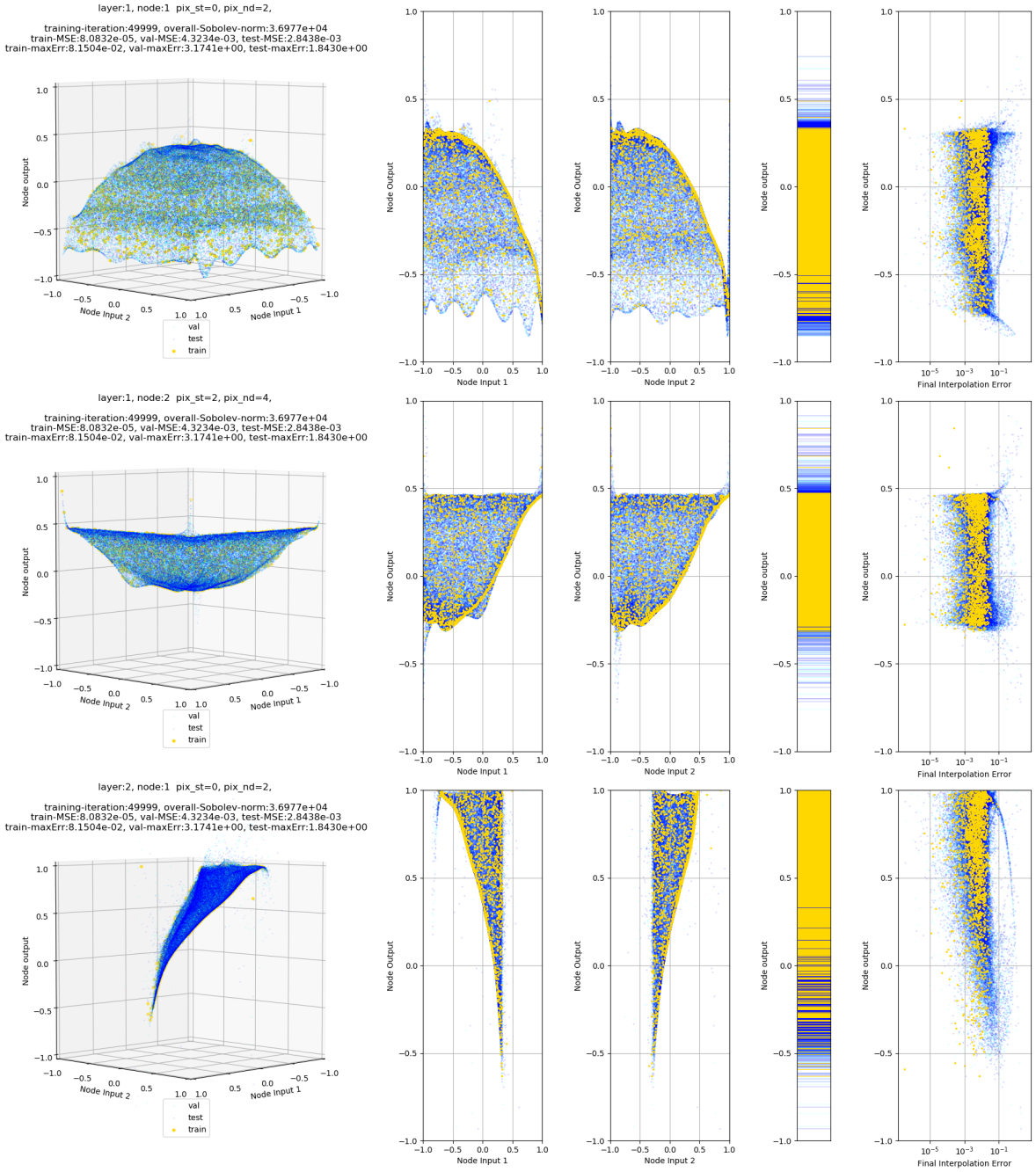


Figure 2.4: Interior nodes of a compositional $\max_4$ without MSN ($\alpha^2 = 0.0$), for $N = 2000$.

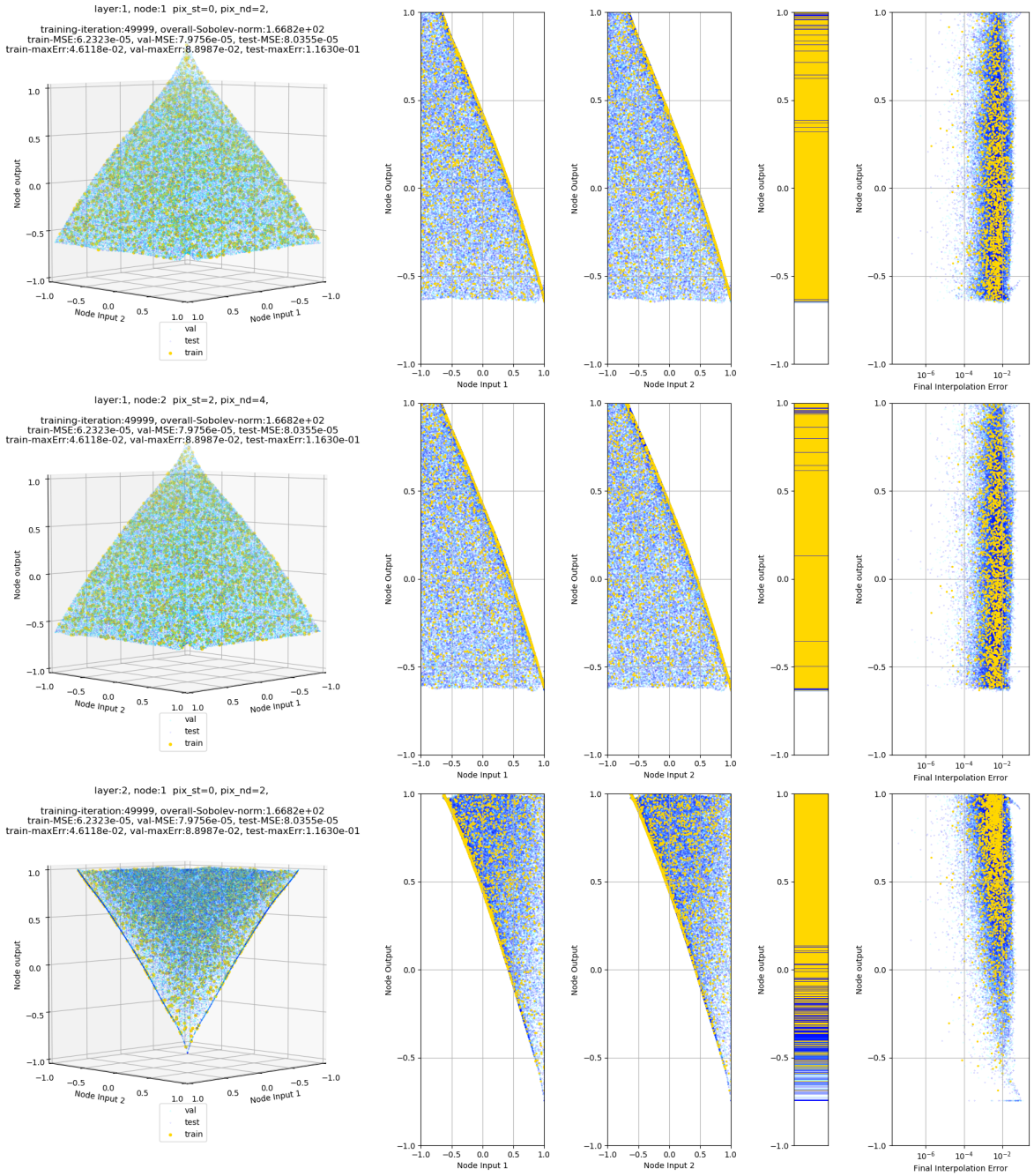
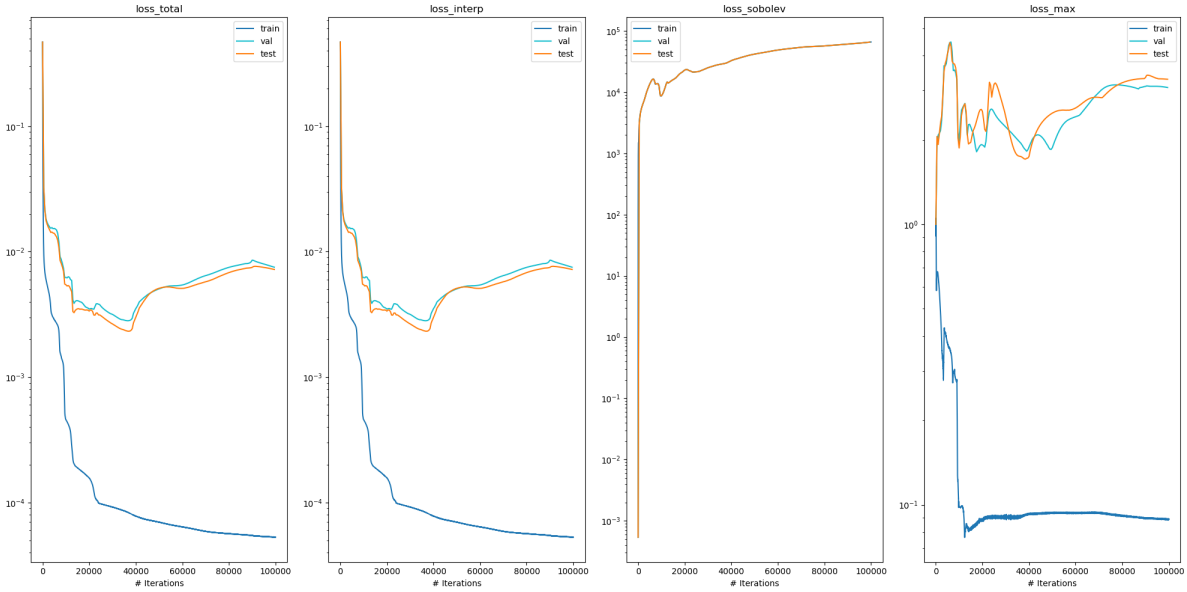


Figure 2.5: Interior nodes of a compositional $\max_4$ with MSN ($s = 1, \alpha^2 = 1e-6$), for $N = 2000$.

(a) No MSN $\alpha^2 = 0.0$



(b) MSN ($\alpha^2 = 1e-6$

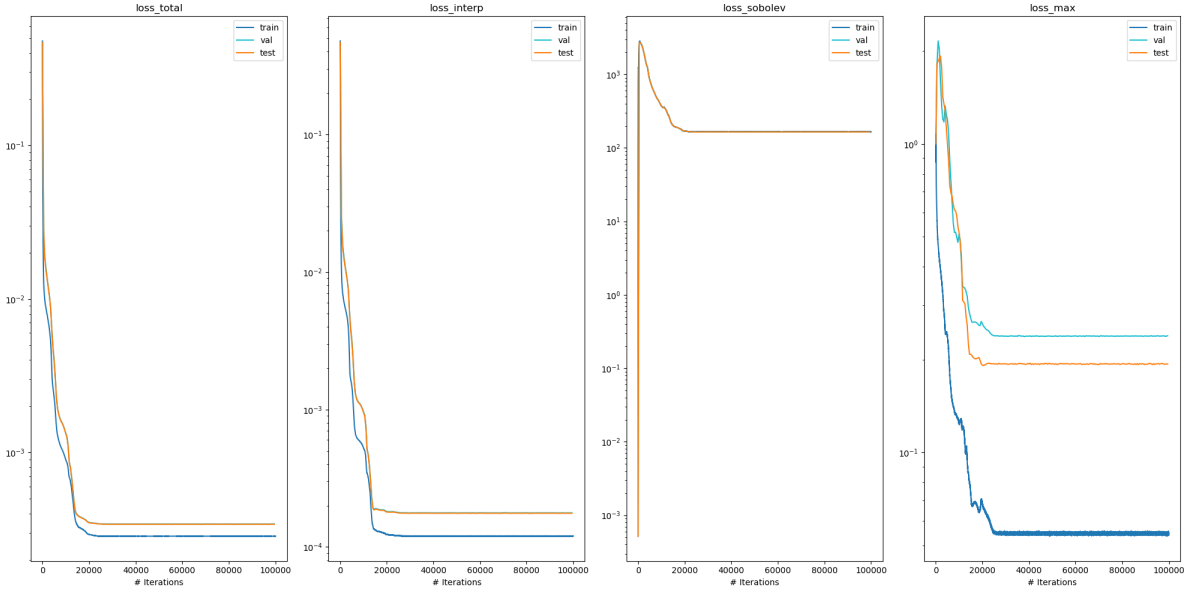


Figure 2.6: Performance vs training iteration for target function: $\max_4$, for $N = 2000$.

max-8**, max-**16**, max-**32

The trend continues for $\max_8$, $\max_{16}$, and $\max_{32}$, where more data is required to get good generalization accuracy, but not exponentially more. We find that increasing the data rate does encourage MSE-trained networks to converge to the true function, but this convergence is not guaranteed and is much slower than DMSN-trained networks. The disparity between the convergence of MSE-trained and DMSN-trained networks is made apparent by observing the smoothness of the interior nodal functions of $\max_8$ (Fig. 2.7-2.8). The same trend continues for $\max_{16}$ and $\max_{32}$. Again, the nodes need not converge to shifted and scaled versions of $\max_2$, but we do observe this in practice for $\max_d$, when $1 < log_2(d) \in \mathbb{Z}$.

Graph	Objective	# Parameters	# Data	Train MSE	Test MSE	Max Test Error
1 Layer	MSE	1407	2000	3.5e-3	4.2e-1	2.3e1
3 Layer	MSE	1407	2000	2.7e-3	6.9e-2	4.9e0
3 Layer	DMSN ($s = 1$)	1407	2000	**1.6e-3**	**8.0e-3**	**1.4e0**
1 Layer	MSE	1407	6000	1.1e-2	2.3e-2	3.7e0
3 Layer	MSE	1407	6000	1.2e-3	1.5e-2	2.8e0
3 Layer	DMSN ($s = 1$)	1407	6000	**1.2e-4**	**1.5e-4**	**1.5e-1**

Table 2.5: Approximation of $\max_8$ for different objective functions and # training data.

max-16

Graph	Objective	# Parameters	# Data	Train MSE	Test MSE	Max Test Error
1 Layer	MSE	3015	16K	4.3e-3	6.7e-3	1.4e0
4 Layer	MSE	3015	16K	4.4e-4	8.66e-4	2.1e0
4 Layer	DMSN ($s = 1$)	3015	16K	**9.1e-5**	**4.1e-4**	**5.9e-1**

Table 2.6: Approximation of $\max_{16}$ for different objective functions and # training data.

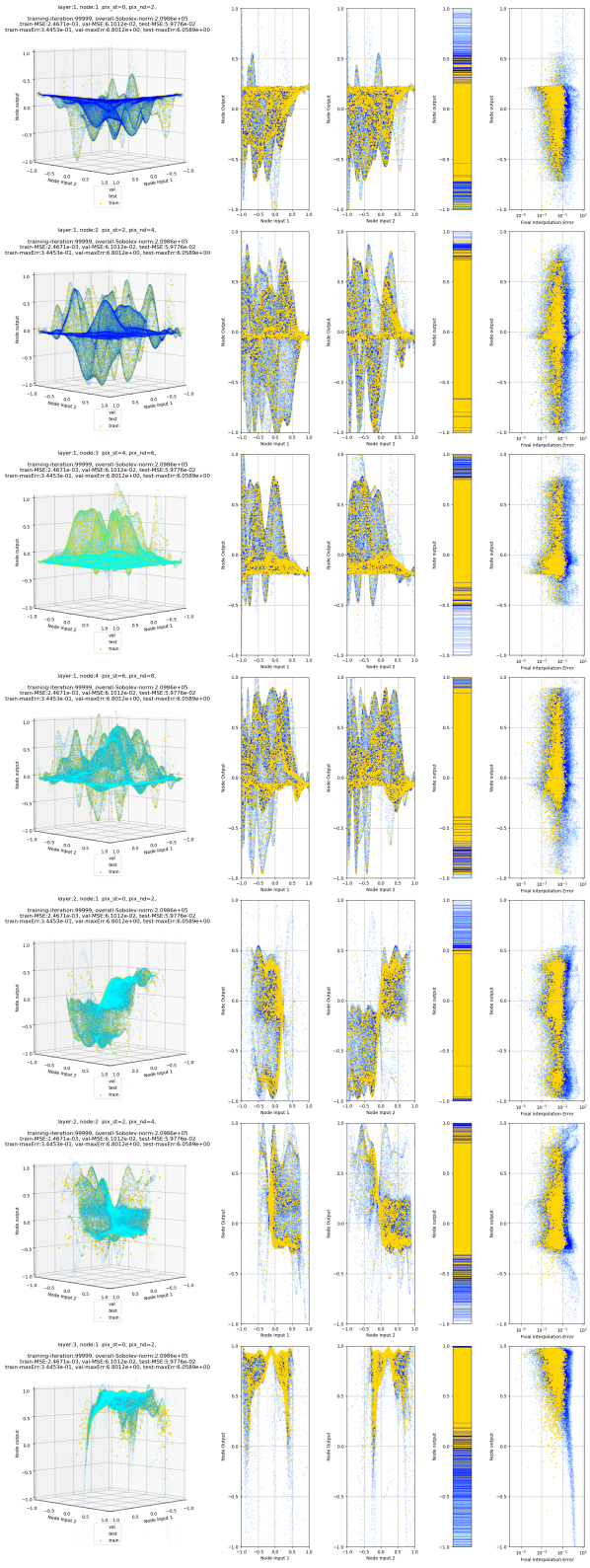


Figure 2.7: Interior nodes of a compositional $\max_8$ without MSN ($\alpha^2 = 0.0$), for $N = 6000$.

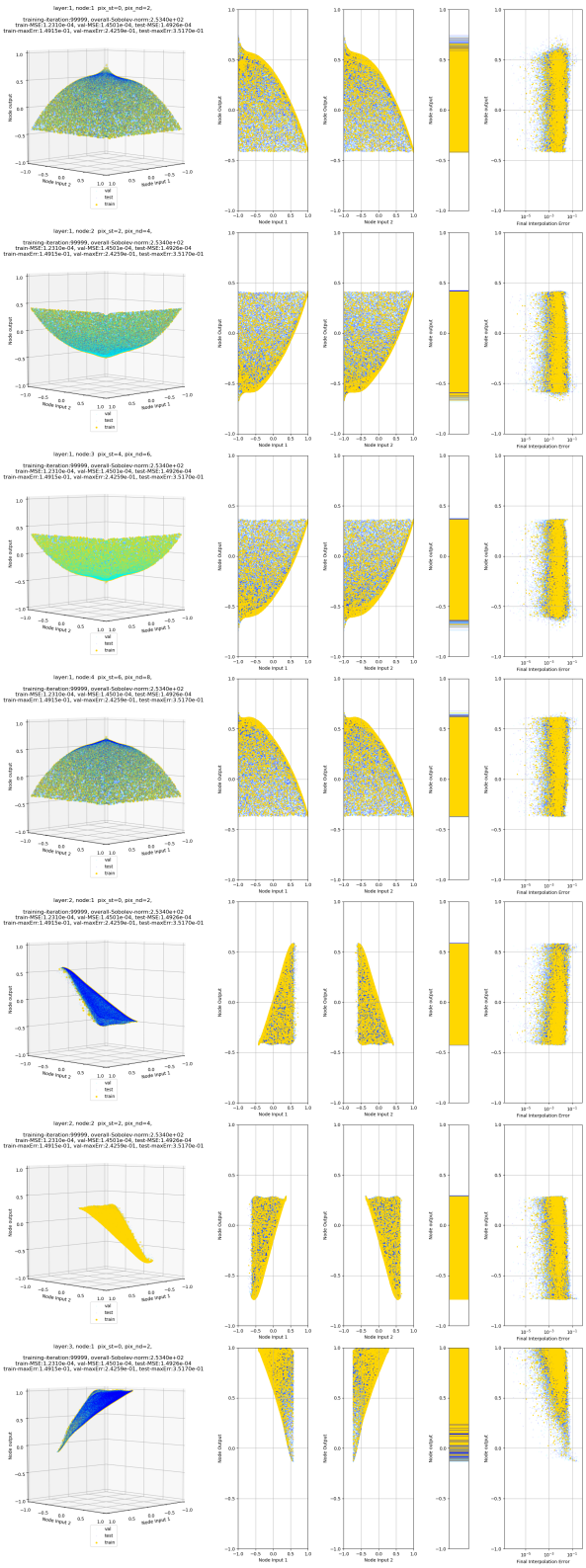


Figure 2.8: Interior nodes of a compositional $\max_8$ with MSN ($s = 1, \alpha^2 = 1e-6$), for $N = 6000$.

The experiments show that using the deep minimum Sobolev norm (DMSN) objective (Theorem 2.2.5) not only results in better generalization error, but allows the interior nodal functions to converge as more training data is added (i.e. they are stable). Interestingly, the nodal functions for $\max_d$ appear to be shifted and scaled copies of $\max_2$, although this is not necessitated or predicted by the theory. Specifically, the numerical results show that generalization performance at $\epsilon \leq 0.001$ can be achieved for $\max_d$ without requiring the number of data samples or polynomial terms to scale as ϵ^{-d} when the compositional structure of the target $\mathcal{G}$-function is utilized, hence breaking the curse of dimensionality for this problem.

2.4 Conclusion

To summarize, in this chapter we saw that there are natural ways to implement theoretically-sound approximation schemes, provided a constructive proof and stable numerical algorithms. However, the theory generally requires density of the available input training data, which is both unlikely and unfeasible for modern high dimensional problems. Moreover, even if the available training data is dense, the number of "units" or terms in the approximation is often intractable since it grows exponentially as $O(K^D)$ or worse, where D is the dimension of the input data and K represents a required polynomial order (degree) determined by the mesh norm, the roughness of the target function, and/or the required ϵ of accuracy. To this end, the solutions presented in this chapter centered around two techniques: (1) judiciously picking polynomial terms via randomized sampling and search-space constraints, and (2) utilizing the compositional structure of the target function to provably remove exponential dependence in the number of terms and empirically ameliorate exponential dependence on the amount of data required to train the network. Both of these techniques yield significant empirical benefits in generalization accuracy when combined with minimum Sobolev-norm approximation schemes, indicating

the existence of fruitful avenues for effectively avoiding the curse of dimensionality in the unasymptotic regime of modern high-dimensional classification and regression problems. In the next chapter, we will see an alternative way to construct high-dimensional polynomial approximations by carefully selecting the terms used in the approximation based on the "physics" of the problem, a technique that is termed "Deep Algorithms". In many cases, these sparse and highly-structured polynomials can possess powerful approximation properties for a large class of target functions *without any training*.

Chapter 3

Deep Algorithms

In this chapter, we introduce alternative technique for constructing approximations in the unasymptotic setting when the compositional structure and constituent functions of the target function may be unknown. In particular, when we are given only input-output pairs of a high dimensional function without an analytic or graphical description of their relationship (e.g. DAG), we will show how the intuition of the human algorithm designer can be used to extract a powerful, "deep" network representation that provides a useful starting point as a regression model along with a natural weight initialization prior to any numerical optimization. These network representations can be subsequently improved in two ways: (1) via numerical optimization schemes such as random search, evolutionary algorithms, or stochastic gradient descent using a corpus of training data (however sparse in the input domain), and (2) via a process termed "tensorization" that utilizes abstract concepts of algorithmic generalization to yield additional polynomial terms useful for the regression problem. While we will primarily demonstrate the construction of deep polynomial and deep rational function networks, this design principle can be naturally combined with a number of other primitives, including traditional DNN-type operations such as `max`, `argmax`, `ReLU` and `sigmoid`. In fact, in a broad stroke, this perspective of "Deep Algorithms" serves as a theoretical and practical link between the previously

presented compositional polynomial approximation technique and contemporary work in designing DNN architectures for computer vision applications.

Despite an abundance of recent advancements in neural architecture design and their contributions to improving the state-of-the-art in computer vision and pattern recognition systems (including via automated discovery techniques such as AutoML), to-date the science of DNN design is poorly understood, leading to frustrated opportunities in theory and application. One of the primary reasons for this is that, in our view, conventional DNN implementations utilize flexible, albeit *generic* computation units (e.g. neurons) whose parameters are initialized generically, necessitating data-hungry non-linear optimization routines (e.g. SGD). So, while DNN designers may place these units carefully using a set of sensible meta-heuristics, concretely evaluating the efficacy of specific designs becomes a challenge because it depends intimately on the dataset and training hyper-parameters.

Our key insight in this chapter is that traditional algorithms, such as those designed by a human algorithm developer, represent high dimensional polynomial and rational function networks with *natural weight initializations*, and that these highly-structured networks can be tuned as differential programs when parameters of the corresponding algorithm are set free for optimization. Similar to how we saw in the previous chapter that the use of heuristics for sampling polynomial terms can gently introduce structure into the regression model while maintaining a great amount of flexibility in how the terms are ultimately used, in this chapter we will see how algorithmic heuristics correspond to a particular choice of polynomial terms (and corresponding DAG structure) that can be flexibly interpreted as both a polynomial network and as a traditional algorithm. This structured representation is useful because it reflects the human algorithm designer's view of the problem i.e. an underlying compositional structure, even if this is not specified mathematically by the problem or does not match the DAG of the target function exactly. Moreover, it helps us *quantify* heuristic-based network designs via the polynomial terms and weight initializations that are used, providing insight into the current wave of heuristic-based

DNN designs used for computer vision problems. As we will see, the fluid interpretation of algorithms as deep networks (and vice-versa) not only provides a convenient framework for construction that is supported by basic approximation theory (e.g. universality), but offers a promising alternative for efficient inference and new opportunities for network design that leverage richness in the class of algorithms.

Why Deep Networks?

The original interest in neural networks arose from their connection with biology. Even though the input and output of biological networks are continuous, much of the current interest lies in static inputs and outputs. For such feed-forward networks there has existed a strong theoretical foundation based on approximation theory since the pioneering work of Cybenko, Hornik, Mhaskar, and others [53, 147, 96]. The recent resurgence of interest arises from the practical success of deep feed-forward networks on image classification problems, which in turn seems to be tied to several factors: the availability of massive training datasets, the availability of large amounts of cheap computing power, and the arrival of practical training algorithms [185, 200]. While the approximation theorists have shown the existence of good, but relatively large and shallow networks, practitioners have found success primarily with deep networks composed of several standard layers (convolutional, max-pooling, etc.) [197, 121]. In particular, practitioners have primarily worked with a *lego-block style approach* to design, where they successively add and remove standard layers of varying tensor dimensions, until a sufficiently good design is arrived upon. Each iteration requires careful tweaking of the learning parameters and a carefully calibrated sense of when to pull the plug on a slowly converging network and try a new design. There are many papers devoted to this art with many case studies [93].

But what is it, *exactly*, about deep networks that gives them an edge over traditional approximation and machine learning methods? For instance, is it their width, their depth, and/or chosen non-linearity? Or, are there finer features of their compositional structure

that are not captured by traditional algorithms? Understanding these questions could yield substantial improvements to the way modern networks are designed and tuned. Some of these questions have been answered, while several more remain open:

- It is well known that wide single-layer networks offer universality but can be impractical for modern problems due to requirements on sufficient width and size of the training set [124].

- In single-layer networks, the specific non-linearity (e.g. squashing) used is not important as long as it is non-polynomial [127]. The jury is still out for multi-layer networks [176, 209].

- Increasing depth empirically increases performance [198], but only yields universal approximation with exponentially many layers and only under some specific conditions [196, 165].

- As we saw in the Chapter 2, when the target function has a known compositional structure that is mimiced exactly by a deep network, the deep network can avoid the can avoid the curse of dimensionality. Although for practical image classification problems, for example, the compositional structure is generally not known.

What is unsatisfactory about these findings is that, while they provide both empirical success stories as well as failure examples, they do not yield *constructive* nor practical (i.e. implementable) algorithms on modern high-dimensional data-defined problems [27]. On the other hand, several fruitful "heuristics" have been developed by practitioners without a generalized theory to-date, e.g.:

- *Going deeper:* Stacking multiple layers to form what is now-called a deep neural network (DNN) empirically yields enhanced performance [121, 81]. What is surprising is that, despite having millions of parameters more than the number of data points, these networks seem to offer excellent generalization performance even for high-dimensional problems [227, 64, 108].

- *ConvNets:* Constraining deep networks by weight-sharing reduces the total number of free parameters, but yields highly structured (e.g. Toeplitz or block-Toeplitz) networks. What is interesting is that, while not in an asymptotically-converging framework, this structured representation can reduce training data requirements (e.g. translation equivariance) and offer better efficiency in parameters [51], yielding better performance in some problems [198, 114].

- *ResNets:* Simple modifications of the sequential stacking architecture can yield significant gains in computational feasibility, ease of optimization, and generalization [199]. Although ResNets were originally motivated as a technique to improve the efficiency of deep layers [94], several works have shown wide-residual networks are just as powerful [225], suggesting the introduced structure is more important than the deep vs. wide interpretation [217].

There are of course many other innovations—too many to name—that have been integral in improving the state of the art. The purpose of this chapter is not to compete with such methods, but rather to position them in the context of the original viewpoint of approximation theory. In doing so, we realized that there is a deeper connection between the theory and prior efforts in computer vision.

In particular, we propose an extension and generalization of today's neural network design principles to include a larger class of heuristics, namely those defined by programs. These programs bring the flexibility of finite state machines to neural network designers while maintaining expressivity and universality, as justified by approximation theory. Consistent with the other approaches to approximation (Chapter 2), the justification for this approximation scheme goes through polynomials and the Stone-Weierstrass theorem, which in turn opens a window to high dimensional approximation by the way of polynomials rather than by sigmoidal basis functions. By representing programs as polynomials, we not only have a theoretical basis on which to judge the efficacy of heuristic algorithms, but natural and well-studied techniques to optimize them [78, 14]. Thus,

this viewpoint also provides a formal basis for approaching the design and training of differentiable programs [163]. To be clear, these sentiments are not necessarily new; the original proofs of universal approximation go through polynomials [53, 147, 96], and more recently [132] points out that Turing machines can be approximated by polynomials. However, where these works stopped short—and where the contribution of this chapter lies—is that they do not provide examples of practical algorithms for problems of interest[1].

On the other hand, we observe that heuristics are used pervasively in the design of modern DNNs [152]. The heuristics here have generally been architectural and are thus applied at a high level, relying on gradient-descent to fill in the low-level details [185, 183]. In contrast, the paradigm described in this work shows how to *fully-initialize* these heuristic-inspired networks designs with sensible parameters that yield a significant baseline-level of performance *prior to any training*. What is surprising is that for many problems a relatively small amount of data is sufficient to construct a performant network. Moreover, when the human algorithm designer is stuck, the resulting networks can still be optimized using additional training examples (e.g. when the chosen heuristic program is differentiable, gradient descent can be used). The connection of these heuristic-based polynomial networks with approximation theory provides an avenue for formalizing this procedure, and provides insights about how to improve network performance on semantic tasks (e.g. via architectural search). In fact, in this viewpoint, current network designs are a special case of this general framework.

In short, our contributions are as follows:

- We introduce and formalize the use of heuristic programs as a network design principle.
- We demonstrate several algorithms that exemplify this principle on both classical and modern problems, utilizing both gradient and *non-gradient* optimization for

[1]There is a growing movement that embraces these changes with applications ranging from physics-based simulation and imaging to control and robotics [210, 13, 80]. We use these case studies to motivate our work.

improved performance.

- We argue and demonstrate that problem-specific algorithms not only yield structured networks, but provide a scalable technique for embedding prior knowledge into the solution.

To provide some clarity to the broad range of questions we have raised, this chapter is broken into two parts. In Section 3.1, we briefly describe the idea and theoretical justification for representing programs as polynomial networks, an approach we term "Deep Algorithms". In Section 3.2, we provide a few *deliberately simple* examples of networks that demonstrate these principles. In Section 3.3 we discuss performance in the context of contemporary methods and suggest a path for blending the two.

3.1 Algorithms as Networks

Machine learning is normally used when the problem specification is so complicated as to defy a compact mathematical presentation. Typical in this area are image and video classification problems, where no precise mathematical formalism exists, but rather the problem is presented as a large corpus of ground-truth data. However, machine learning is also a valid approach when the mathematical problem is so difficult as to defeat the best effort of human algorithm designers to come up with a well-performing algorithm (both in terms of accuracy and speed). There are many classical problems that easily fall into the latter category. A simple example that has deep roots in functional analysis, are root finders for systems of polynomial equations. The literature on this area is classical and vast, and yet one can safely say that there exists no reliable practical algorithm that can be used in a black-box manner [16, 18]. So, whether the problem is specified by data or mathematical formulas, one thing that is common is that the human algorithm designer has reason to believe that the problem is solvable and even has ideas on how to do so. We refer to these ideas as "heuristics" and will assume that they are presented as algorithms

(or programs) that work reasonably well.

Our contention is that in many cases these heuristic algorithms can be viewed as special cases of a very large family of algorithms that can be parameterized by many real numbers. The initial heuristic itself can be viewed as a particular choice of these numerical parameters.

For instance, when evaluating the similarity of two feature vectors $x, y \in \mathcal{R}^d$, it is natural to compare their distance in some norm. In the absence of other information, the algorithm designer may pick a familiar norm like the Euclidean distance:

$$d(x, y) = \|x - y\|_2.$$

Being aware that this might not be the best choice, designers may generalize to the Mahalanobis distance instead:

$$d_2^2(x, y) = (x - y)^T A(x - y)$$

where $A \in \mathcal{R}^{d \times d}$ [150]. Notice, that the original distance measure is recovered for the choice $A = I_d$. We first observe that this is not the only such generalization. For example, one might embed this computation into an even larger computation graph, as:

$$d_k(x, y) = f\big((I_k \otimes (x - y))^T A_k (I_k \otimes (x - y))^T\big)$$

where the original distance could be recovered for some suitable choice of f (e.g. average of the trace) and $A_k = I_k \otimes A$.

In particular, note that the original heuristic distance is being recovered *in every case* by carefully selecting the numbers in a larger sparse matrix. That is, this process corresponds to the insertion of additional edges in the computational graph of the original heuristic with trivial weights. Thus, one can generalize this observation in another direction too: by structure. For example by picking A to be a Toeplitz matrix we get a *convolutional* layer, and if we pick A to be a Toeplitz-block-Toeplitz matrix we get a *2D*

convolutional layer. We can also choose A to be the product of Toeplitz matrices in which case we would get several convolutional layers, and so on. Of course choosing A to be a fully dense matrix would give us a full-connected layer at that stage of the computational graph. In this case, all these expressions are naturally polynomials in x, y, and are thus differentiable representations.

Furthermore, we observe that when these heuristic networks (differentiable or not) are parametrized by real numbers, they can be *tuned* by special training algorithms. Popular frameworks such as Tensorflow and PyTorch accomplish this via automatic differentiation through standard neural network layers (`dense`, `conv2D`, `maxpool`), although there are numerous other frameworks that attempt to accelerate this procedure for standard *programs* instead (e.g. JAX [73], Flux [101]). In this sense, we can understand conventional neural network architectures as a small subset in the class of programs.

3.1.1 Programs as Polynomials

The use and extension of heuristic programs as polynomial networks is justified by approximation theory. Here, we summarize a line of reasoning that offers universality in these representations.

The first observation, is that polynomial networks of polynomial size can express all functions that can be implemented efficiently using a Turing machine [132].

Theorem 3.1.1 (Polynomial networks can express Turing Machines). Let $N_{t,n,\sigma_2,L}$ represent the class of functions that can be implemented using a neural network of depth t, size n, squared activation function (i.e. polynomial), with a bound L on the l_1 norm of the input weights of each neuron. Let $T : \mathbb{N} \to \mathbb{N}$, and let F_d be the set of functions that can be implemented by a Turing machine using at most $T(d)$ operations. Then there exists constants $b, c \in \mathbb{R}_+$, such that for every d, the class $N_{t,n,\sigma_2,L}$, with $t = cT(d)log(T(d)) + b$, $n = t2$, and $L = b$, contains F_d.

The proof relies on the result of [162], and can be derived by constructing an approxi-

mation of each component primitive (e.g. Boolean gates), as shown in [132].The basic idea is that we can implement standard Boolean gates (`AND, OR, NEG, Identity`) using polynomial networks of fixed depth and size. A shorter proof is to note that all Boolean functions (and thus programs) can be expressed using a finite-depth circuit using a `NAND` gate (i.e. they are functionally complete for Boolean functions). These gates can be expressed exactly in a polynomial network by relaxing the Bools to real numbers and performing the substitutions:

- $\texttt{NEG}(x_1) = 1 - x_1$
- $\texttt{AND}(x_1, x_2) = \frac{1}{4}\big((x_2 + x_1)^2 - (x_2 - x_1)^2\big)$

Then by [162], any Turing machine with runtime T can be simulated by an *oblivious* Turing machine, where the position of the head at time t does not depend on the input to the machine (i.e. the input can be embedded into the memory). We can simulate such a machine by a network of depth $O(T \log T)$, where the nodes at each layer contain the state of the Turing machine, and the transition from layer to layer depends on a constant sized circuit. Finally, this circuit can be implemented using a constant depth polynomial network.

The second observation is that polynomials, themselves, are universal approximators (Theorem 2.1.1).

Therefore, we can approximate or learn any function with a sequence of polynomials. Heuristic programs are represented exactly as polynomial networks, and therefore correspond to one polynomial in this sequence. The rest of the sequence can be generated by successively building on or expanding the previous polynomial within a converging framework that includes all the necessary terms. For example, if we have an approximation of $f(x)$ for $x \in [a, b]^d$, denoted $p_K(x)$ of maximum degree K, then a sequence can be generated as p_0, p_1, ... p_M, such that p_M approximates f uniformly. However, the conditions under which our approximation converges to the true function (e.g. in value)

also depends on the practical scheme we use for construction and learning from a finite number of samples. Clearly, the ResNet-like refinement $p_{k_3} = (1 + p_{k_1}) \cdot p_{k_2}$ is a heuristic to construct deeper representations [199], albeit one that skips many low-order terms.

3.2 Designs for Networks

3.2.1 A Newton heuristic for systems of polynomial equations

Our first example is a root finder for a system of polynomial equations in d variables:

$$\sum_{i \in R^d, \|i\|_1 < N} a_{j,i} x^i = 0, \qquad j = 1, \ldots, d, \tag{3.1}$$

where $x \in \mathcal{R}^d$ is the unknown. It is well-known that this is an extremely difficult problem in floating-point arithmetic [16]. One approach is to convert it into a polynomial equation in a single variable, but the price to pay is an exponential growth in the degree of the polynomial and potentially in the size of the coefficients. Another popular approach is to use a continuation technique [4]. However, the latter is notoriously difficult to implement well in floating-point arithmetic and slow to boot. Therefore the most popular approach is to use a locally convergent method like Newton, perhaps reinforced with some kind of line search or back-stepping technique[2]. All of these methods fall into the category of what we call heuristics, and they may be specified as "algorithms" or "programs".

If we represent the polynomial system as $F(x) = 0$, then the simple Newton *heuristic* could be expressed as follows. Pick an initial guess x_0 and then, until $\|F(x_n)\| \leq \epsilon$, apply the update:

$$x_{n+1} \leftarrow x_n + \alpha (F'(x_n))^{-1} F(x_n) \tag{3.2}$$

where α is a step length to be chosen and ϵ is a number to be provided by the user.

[2]It is difficult to get algorithms of this form to compute more than one root reliably due to deflation [129].

First note that there are some obvious numerical parameters that have to be chosen, namely x_0 and α. However, we note that we can easily make many more. For example we could make α a matrix that depends on the iteration number: α_n. Another possibility is to borrow from accelerated gradient methods and other higher order methods and look for an iteration of the form:

$$x_{n+1} \leftarrow \sum_{0 \le l < K} \beta_{n,l} x_{n-l} + \alpha_{n,l} (F'(x_{n-l}))^{-1} F(x_{n-l}), \tag{3.3}$$

where $\beta_{n,l}, \alpha_{n,l} \in \mathcal{R}^{d \times d}$. We can go further and allow multiple choices per step and take the best, as:

$$\begin{aligned} x_{n+1} &\leftarrow \arg\min_{x_{n+1,r}} |F(x_{n+1,r})| \\ \text{with} \qquad x_{n+1,r} &\leftarrow \sum_{0 \le l < K} \beta_{n,l,r} x_{n-l} + \alpha_{n,l,r} (F'(x_{n-l}))^{-1} F(x_{n-l}), \qquad 1 \le r \le N. \end{aligned} \tag{3.4}$$

This version of the algorithm can be thought of as a natural generalization of Newton's method with line search [85], and we notice it has a natural interpretation as a deep rational function network [3] that resembles modern DNNs (Fig. 3.1).

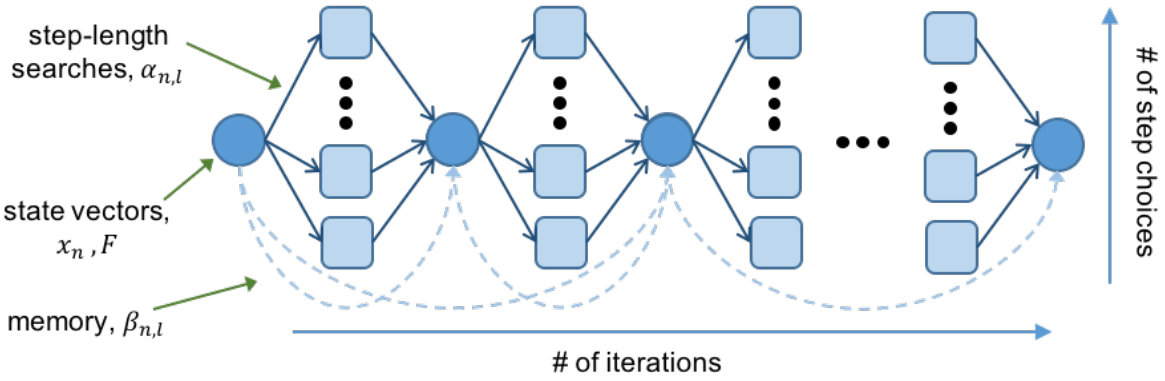


Figure 3.1: The unrolled graph of Newton's method with line search resembles a compositional deep neural newtwork (DNN) or recurrent neural network (RNN) with pooling.

When we generalize so much it is good to observe that the original trusted Newton

[3] Direct computation of F^{-1} necessitates incorporation of a `divide` primitive, which is not present in purely polynomial networks.

method can be recovered for special choices of the new weight matrices $\alpha_{n,l,r}$ and $\beta_{n,l,r}$. This is an important observation as when this heuristic is unrolled and trained via one's favorite machine learning framework we are *assured* of good starting weights with a known performance threshold [4].

We now turn to our numerical experiments for this problem. For simplicity in our initial tests, we consider the following classes of polynomials for $F(x)$: $F_1(x) = x^2 - a_0 = 0$, $F_2(x) = x^5 - a_0 = 0$, $F_3(x) = \sum_{k=0}^{5} a_k x^k$, and $F_4(x) = \sum_{i+j\geq 0}^{2} a_{j,i} x^i y^j$, where $k, a_k, a_{j,i}$ are sampled over a large range to create large, disjoint training and testing sets. For these classes, we compare the performance of three different network/algorithms fixed at $n = 3$. The algorithms are all variants of Newtons method, except with different update rules: f_1, "Newton" – Newton's method with no modification (Eq. 3.2), f_2, "Newton-LS" – Newton's method ($K = 2$) with line-search (Eq. 3.3), andf_3, "DeepNewton-LS" – a parameterized Newton's method ($K = 2$) with line-search (Eq. 3.4), with 3-tunable step-sizes (initialized at $\alpha_{n,l,r} \in \{0.5, 1.0, 1.5\} \cdot \mathcal{I}_d$) and memory-term (initialized $\beta_{n,l} = \mathcal{I}_d$) per iteration. We empirically initialized the step-sizes of the DeepNewton algorithm as $\alpha_* = \{-0.5, -1.0, -1, 5\} \cdot \mathcal{I}_d$, because these correspond to "undershooting", vanilla, and "overshooting" Newton's method on the canonical convex examples that are typically studied when using the method. For clarity, note here $\mathcal{I}_d$ just indicates that we are initially picking the same sized-step for each dimension of the unknown variable x.

Dataset

A dataset was generated as follows:

- In a given basis (we chose monomial for simplicity), for a given system size (i.e. number of equations p, and degree of each polynomial in that system D), randomly select coefficients for polynomials of one less degree $D - 1$. For the monomial

[4]Note that rather than arbitrarily setting $x_0 = 0$ we could use a more complicated expression like $x_0 = \sum_{i \in R^d, j} a_{j,i} \gamma_{j,i}$, or a intelligent class of heuristics based on root localization theorems [17].

basis, the coefficients are chosen from the unit interval $[-1, 1]$, and (without loss of generality) the leading-coefficient was always picked as 1.

- In the unit interval $[-1, 1]$, randomly pick a number as a root.
- Increase the degree of each of the previously generated polynomials by updating the coefficients generated in Step (1) as necessary. For example, in the 1D monomial basis, we would multiply a polynomial of degree $D-1$ by the new root x_0, as:

$$f(x) = p^{D-1}(x)(x - x_0) \tag{3.5}$$

We did this repeatedly to generate many such systems ($\sim 1000 - 2000$), with various choices for d. For the presented experiments in the work, we picked $D = 3$, and $P = 5$ for 2-D polynomials. For the 1D-polynomials, we picked $D = 6$. We ran this twice, as to generate both a "training" set ($n = 1000$), and a "testing" set ($n = 2000$).

For some special 1-D polynomials in the table (square-root and fifth-root), we generated these polynomials by randomly picking several roots on the interval $[-1, 1]$, and using these to generate polynomials of the form $x^\alpha - S = 0$. Notice that this is just a special case of the previous 1D systems for $\alpha = \{2, 5\}$ with all but the leading and lowest-order coefficients zero-ed out [5].

Experiment

For each f_*, the performance of the network algorithm is measured by the mean magnitude of the residual $r = ||F_s(f_p(x, \alpha, \beta))||$ over the given dataset, where f_p represents the aforementioned feed-forward Newton heuristic networks. Notice that, while the problem is specified entirely by the given $F_s(x)$, *there is no explicit ground-truth data* (roots). Furthermore, f_1, f_2, f_3 are all differentiable, since they have entirely rational polynomial

[5] While we are guaranteed that there exists at least one root in the unit interval (by construction), there may actually be many. And, in particular, we are not interested in *which* root the method converges to, just that it does converge.

representations. This allowed us (for f_3 in particular) to utilize machine learning frameworks to optimize the specified tunable parameters via a gradient descent-like method, although non-gradient methods are also an option. Specifically, we experimented with both stochastic gradient descent (SGD) and random search, also sometimes referred to as random descent. Our experiments indicate that these two methods often yielded models with comparable performance for these class of target functions, although the random search method seemed to be able to escape local minima more often. The results presented in this chapter are for networks trained with random search, which was implemented using custom optimizer class in `Tensorflow`.

Results

In our experiments, summarized in Table 3.1 below, we observed a positive result; the parameterization f_2 significantly outperforms f_1 on most tasks, and f_3 consistently performs better than f_2.

Table 3.1: Relative testing performance (MSE) of selected methods on some polynomial systems.

	F_1	F_2	F_3	F_4
Newton	0.7658	2.2511	0.0588	-
Newton-LS	0.0195	0.1286	0.0699	0.0331
DeepNewton-LS	**0.0070**	**0.0560**	**0.0287**	**0.0225**

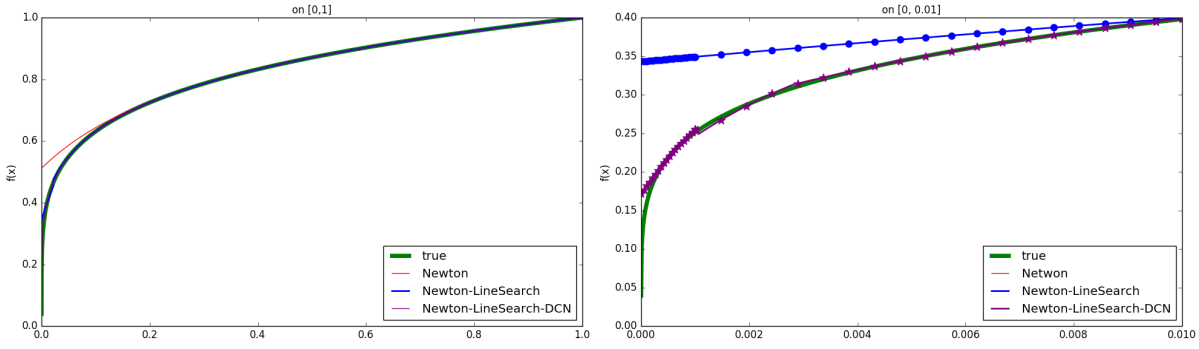


Figure 3.2: Performance of selected methods on the class of 1D polynomials: $F_2(x) = x^5 - a_0 = 0$.

While the computation and metric calculations in Table 3.1 were computed point-wise over a large family of polynomials, it is often interesting and educational to choose a small

number of polynomials for visualization. This is visually depicted in Figure 3.2, where we observe the improved accuracy for the family of 1D polynomials $f(x) = x^5 - a_0$ (this family also has the benefit of having a known, analytic solution). This can be repeated for many such families, although the plots increase in dimension according with the number of non-zero a_k in the expression.

In Figure 3.3, we visually depict the performance of the trained line-search algorithm (DeepNewton) in comparison with the other methods for the family $f(x) = x^3 - a_0$. In the bottom portion of the figure, depicted "effective domain partition", we plot the branch of the network that was used to compute the approximation (the y-axis corresponds to a unique, indexed branch number). What this plot shows is, that training enhances the line-search algorithm by utilizing a larger number of active branches, especially when as approximation becomes more difficult (near $x = 0$). This agrees well with what we would expect from approximation theory: that more oscillatory functions (i.e. with high-valued derivatives) will require wider networks.

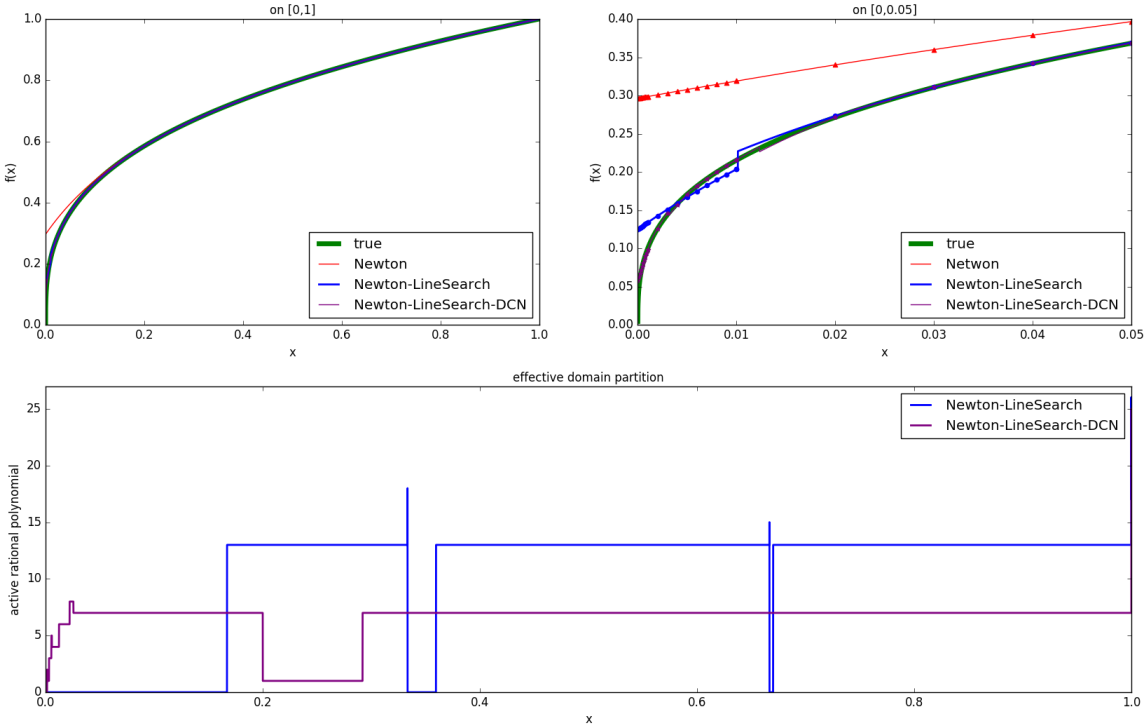


Figure 3.3: Performance of selected methods on the class of 1D polynomials: $x^3 - a_0 = 0$.

3.2.2 A simple matching-based heuristic for image classification

Our second example, applicable to modern deep learning tasks, is focused on image recognition. We begin by considering the MNIST-like image classification problem, where target objects have been pre-segmented and centered on a uniformly illuminated and textured canvas, and the goal of the problem is to identify which mutually-exclusive class each target belongs to. We contrast this with more natural settings for objects, such as in the CIFAR and STL datasets, where targets appear unsegmented from unlabeled and diverse background textures. The idea here is to establish the proposed design principle on a simple example, to lay the foundation for more complicated experiments on trickier problems that may require different or an expanded set of heuristics.

Figure 3.4: Example queries for 10-class image recognition datasets: (a) MNIST Digits, (b) Fashion-MNIST, and (c) CIFAR-10.

As can be seen, there is no precise mathematical formulation of the problem other than what is specifiable via the ground-truth in each dataset. Nevertheless, as humans we have some notions of how renderings of natural objects can be recognized. For example, we can use the common heuristic:

1. Choose a representative number of samples from the training data for each target class.

2. Given a query image, compare it to the representatives using some appropriate metric.

3. Based on the query's distance to the representatives, make a decision on the predicted class.

The performance of this heuristic depends crucially on the image metric that is used. It is well-known that classical norm-based distance functions tend to perform poorly and many alternatives have been proposed in the literature [191]. We can capture many of these variations by carefully parameterizing various choices in the above heuristic, as follows. Let $X, Y : \mathbb{R}^2 \to \mathbb{R}^d$ and

$$\sigma(X, Y) = \min_{\theta_1 \leq \theta \leq \theta_2} \|W_1(X - Y \circ R_\theta)\|_p, \tag{3.6}$$

where R_θ denotes rotation by an angle θ, W_1 denotes a linear operator, and $\|\cdot\|_p$ denotes the standard p-norm. Note that σ measures the distance between two images of the same size by considering the minimum over all rotations in the range from θ_1 to θ_2. Next let

$$\phi(X, Y) = \min_{|i|<w, |j|<h} \sigma(X, Y \circ \tau_{i,j}), \tag{3.7}$$

where $\tau_{i,j}$ denotes translation by the vector (i, j). Let $\mu(X, Y) \in \mathbb{R}^2$ be defined such that

$$\phi(X, Y) = \sigma(X, Y \circ \mu(X, Y)). \tag{3.8}$$

Let $\nu(X, i, j, s)$ denote the sub-image of X of size $s \times s$ centered at (i, j) with pixels outside the region set to 0. Then we define the distance between image X and Y as $\gamma(X, Y) =$

$$\int_{R^2} \phi\left(\nu(X, x, y, s), \nu(Y, x, y, s)\right) w_2(x, y)(1 + \|\nabla^2(\mu\left(\nu(X, x, y, s), \nu(Y, x, y, s)\right))\|)\mathrm{d}x\mathrm{d}y \tag{3.9}$$

where w_2 is a weight function, σ and ϕ are pooling layers, and W_1 is trivially initialized as $\mathbb{I}_d$.

In spite of its messy appearance, the image metric in Equation 3.9 is quite simple: it is computing the distance between the pixels in X and Y by comparing patches of size $s \times s$, and when it compares patches it allows some translation and rotation, picking the closest match in each case. Then it looks at the induced optical flow and further penalizes those distances where the Laplacian of the flow is large. This type of heuristic is not uncommon in the computer vision literature [221]. Moreover, we find that this heuristic is already quite performant (Table 3.2) *without any training* and using an extremely small number of training samples (10-25 per class). Here, prediction $\hat{y} \leftarrow \arg\min_k \gamma(X, Y_k)$.

Table 3.2: Performance of a ClusterNet prior to optimization on using the training dataset.

	Predicted Label									
	0	1	2	3	4	5	6	7	8	9
0	**85.3**	0.0	0.5	0.4	0.0	5.3	6.0	0.0	2.4	0.0
1	0.0	**95.8**	3.2	0.3	0.2	0.1	0.2	0.2	0.0	0.0
2	1.7	0.6	**86.4**	2.5	0.6	0.4	4.2	1.4	2.2	0.0
3	0.0	1.3	2.6	**78.0**	0.8	12.9	1.1	1.1	1.8	0.4
4	0.1	2.0	0.8	0.1	**66.1**	0.5	8.9	4.2	2.2	15.1
5	1.3	2.0	1.5	8.4	1.9	**75.3**	6.6	0.2	1.6	1.1
6	1.6	1.6	1.0	0.7	0.3	3.0	**91.6**	0.0	0.1	0.0
7	0.5	3.7	4.4	0.4	8.7	2.3	0.1	**68.2**	1.6	10.1
8	0.4	4.2	3.4	11.7	1.6	6.9	3.3	1.8	**66.1**	0.5
9	1.2	1.5	0.3	1.2	19.8	0.5	1.5	10.4	2.8	**60.8**

(a) MNIST, Pre-training Confusion Matrix, 10-samples/class, 77.4% accuracy.

	Predicted Label									
	0	1	2	3	4	5	6	7	8	9
0	**77.1**	1.5	3.1	6.6	1.6	0.4	8.4	0.0	1.3	0.0
1	1.2	**92.7**	0.5	4.2	0.7	0.1	0.5	0.0	0.1	0.0
2	2.2	1.4	**54.5**	0.6	19.8	0.2	20.6	0.0	0.7	0.0
3	8.8	5.0	1.1	**73.6**	6.6	0.1	3.5	0.0	1.3	0.0
4	0.7	1.4	19.0	4.3	**58.1**	0.2	15.4	0.0	0.9	0.0
5	0.0	0.0	0.0	0.1	0.0	**88.3**	0.3	6.8	1.0	3.5
6	23.5	1.2	17.9	3.5	14.0	1.0	**37.2**	0.0	1.7	0.0
7	0.0	0.0	0.0	0.0	0.0	8.7	0.0	**82.0**	0.3	9.0
8	0.6	0.1	3.6	0.6	1.3	2.9	1.8	0.4	**88.6**	0.1
9	0.0	0.0	0.0	0.0	0.0	3.6	0.0	6.7	0.1	**89.6**

(b) FMNIST, Pre-training Confusion Matrix, 25-samples/class, 74.1% accuracy.

Furthermore, this heuristic can now be unrolled into a network, tuned, and more parameters introduced as needed. In particular, by a careful selection of weights in Step 3 of the common heuristic, we uncover a natural multi-layered representation of the algorithm. For example, we can compute $\hat{y}$ as:

$$\hat{y} = \arg\max_{c} \sum_{k} b_k \cdot (Q \cdot g)_k \tag{3.10}$$

where $d_k = \gamma(X, Y_k)$ is computed for each of the established K representatives Y_k ("clusters-centers") with 1-hot-encoded labels b_k, g represents a soft weighting produced by a softmax layer, and $\hat{y}$ is taken to be the maximum class-index of the weighted average of the cluster labels after multiplication with Q (i.e. a fully-connected layer with linear activation). Notice, Q can be initialized trivially as identity, or with the eigenvectors of the generalized Gram matrix of the cluster "centers"; this latter initialization corresponds to an interpretation of the algorithm as a version of spectral clustering, or nonlinear PCA, with respect to the heuristic distance function defined by γ [9, 100, 47].

Dataset

As described, here we use the free and publicly available MNIST [122] and Fashion-MNIST datasets [218].

Experiment

We constructed the previously described network, making discrete choices for the various parameters and operations (fixed-θ rotation matrices, sums instead of integrals, etc). From high level, the network architecture is depicted in Figure 3.5. Here, "tensorization" refers to the practice of generalizing the computation graph. In the case of ClusterNet, for example, this could refer to trivially expanding the width at the first layer using more prototype examples, but also to evaluating multiple choices for parameters.

We evaluated a number of different strategies for choosing cluster centers from the

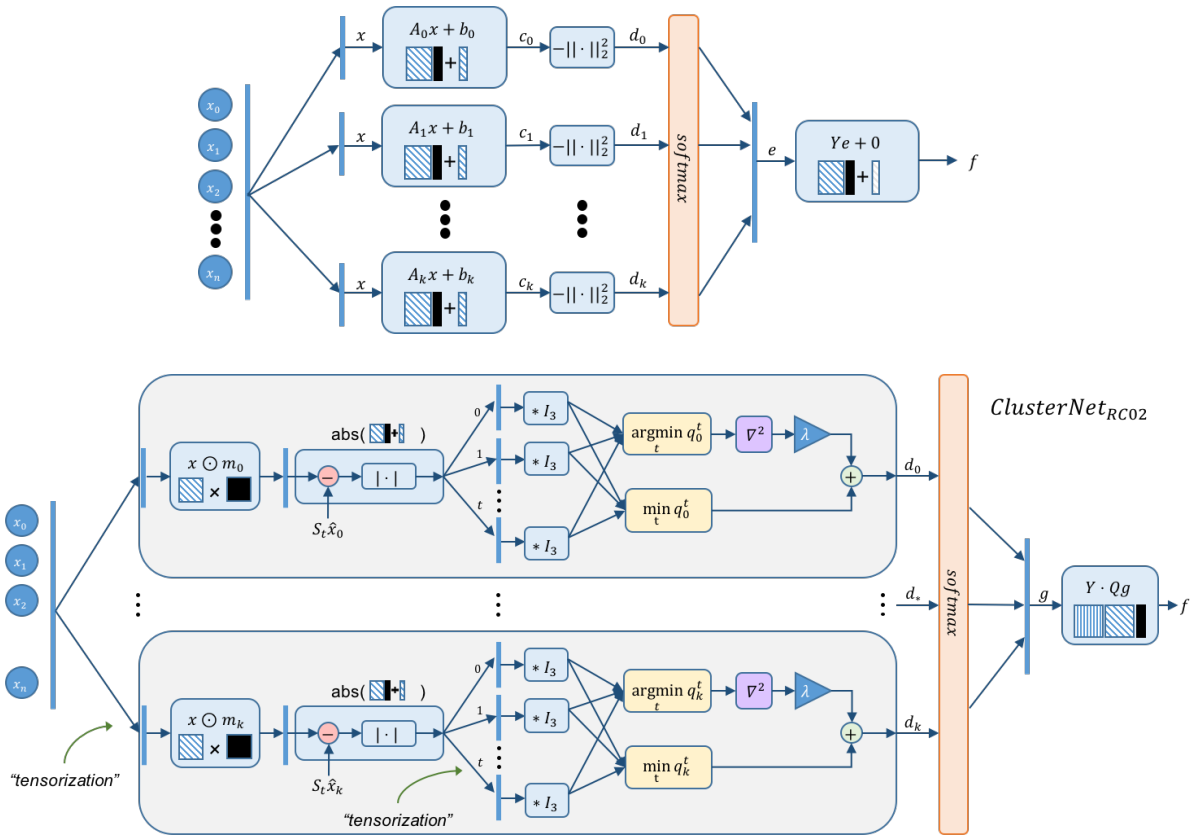


Figure 3.5: The computation (inference) graph of (a) a traditional weighted k-NN algorithm, and (b) a rendition of the "tensorized" ClusterNet deep algorithm.

training data, such as k-means with random / Lloyd-type initialization. In the end, we chose to initialize the cluster centers of our model with random samples from each class. This was much quicker than the other methods, and often produced good/better results than the k-means style methods, which looks at large chunk of the training data. The λ factor on the Laplacian of the "matched" image was empirically initialized at 0.3.

Upon encoding networks with these parameters, we used a version of stochastic gradient descent (Adam optimizer [109]) to optimize the various free parameters identified in the network. We also experimented with using random search, as in the previous Deep Newton example, but this was ultimately found to be computationally expensive within the optimization framework developed using the Tensorflow API.

The total number of free parameters in our network can be computed as follows:

- 28×28 parameters $\hat{x}_k$ for every N cluster center that was used as a template (we initialized with p cluster centers *per class*).
- 10×1 parameters $\hat{b}_k$ for each of these N cluster centers, which was the one-hot-encoded vector describing the cluster center's class membership.
- $N \times N$ parameters for matrix Q, which was initialized as the eigenvectors of the Grammian (for MNIST), or as identity. For Fashion-MNIST we froze this weight as identity, so it was not a free parameter (although it could have been).
- 28×28 parameters m_k every N cluster center that was used as an image mask (only for Fashion-MNIST).
- 1 parameter λ that was used for the the weight on the Laplacian (only for Fashion-MNIST)

Therefore,

- for the MNIST dataset, our (10x10) ClusterNet implementation (shown in Table 3.3), used $89,400$ parameters in total.
- for the Fashion-MNIST dataset, our (10x10) ClusterNet implementation (shown in Table 3.3) used $157,801$ parameters in total.

All of these were initialized using the heuristic.

Note that the initial (untrained) performance of these networks is an example of low-shot transfer learning, which is naturally made possible by the network. In this sense, the heuristic-based design is favorable for application areas without a lot of initial training data. As additional samples are made available, the network can be tuned (either via expansion or via training) to accommodate additional variance in the input space.

Results

We observed another positive result with increased performance in our ClusterNet implementation despite the extremely small number of parameters used (Table 3.3). There

are two prominent features of our ClusterNet implementation that are of significance: (1) The network has a non-trivial baseline initialized performance with extremely less training data and *no training flops*. (2): There are natural *zero-flop* ways to expand the coverage and accuracy[6] without departing from the understandability of the initial heuristic (e.g. higher K, $\theta_2 - \theta_1$, w, h), a feature not common in DNN architectures. Finally, the zero-flop algorithm can be embedded within a larger graph, and further improved via optimization (gradient-based or otherwise). Thus, our experiment provides further evidence that high accuracy is possible in relatively shallow networks, given the right parameterization and initialization of weights [186].

Table 3.3: Relative testing performance of a ClusterNet implementation on some datasets.

	Pre-Training (0 training `flops`)	Post-Training (10 `epochs`)	# Param.
MNIST [122]	77.4%	97.1%	89.4K
FMNIST [218]	74.1%	90.1%	157.8K

As a point of comparison, this heuristic is not yet optimized for natural images such as from the CIFAR or STL datasets. In our initial tests on STL-10, the initialized performance with 25 items per cluster was 26.1%, and the corresponding trained algorithm achieved 46.9%. While this accuracy is not groundbreaking, we note that we are not claiming the use of superior heuristics. We are simply showing that learning-based optimization of existing heuristics can yield significant improvements in performance, similar to the phenomena we see in deep neural networks. Thus, it is likely that an alternate heuristic (e.g. based on segmentation and region-of-interest localization) may perform much better on STL-10 and similar datasets. This is left as future work.

3.2.3 Targeting through fog

Our third example is an application of adaptive contrast enhancement for dim and unregistered grayscale electro-optic (EO) and short-wave infrared (SWIR) imagery. Unlike in

[6] Accuracy gains from increasing K often plateau, but this can be mitigated via cluster-repulsion as in [34].

the previous cases, there is no precise mathematical test, nor is there explicit ground-truth data, on which to evaluate the performance of the algorithm. Instead, we show how human intuition about the problem can translate to a tunable heuristic with outputs that agree with our perception.

The dataset we are using to demonstrate this task was collected over several hours in the Santa Barbara Harbor (Goleta, USA) between October 2016 and March 2017, and consists of still EO and SWIR imagery of a portion of the harbor that includes an off-shore oil rig. Due to the local climate, a human-perceptible view of the harbor and the rig is obscured to varying degrees by transient fog. The problem is then: given an EO/IR image-chip without a specified heading, identify and locate features of off-shore structures. Thus, in lieu of manually labeling each image to identify the location and outline of the structures that are present, there is no good prescription of ground-truth data. The resulting system could, for example, be used to identify bearing angles with respect to various landmarks (or seamarks).

This type of problem is typically approached via adaptive contrast enhancement, where the desired level of contrast is estimated automatically for different regions of the image based on features from the image itself (e.g. no side-information on the level of fog) [220]. A common heuristic in this vein is to use an adaptive phase-screen, which is based on a simple lightfield-based model of optics [49]. A simple version of the algorithm proceeds by computing:

$$t(x) = 1 - \lambda * \frac{d(x)}{A(x)} \tag{3.11}$$

$$J(x) = \frac{I(x) - A(x)}{t(x)} + A(x) \tag{3.12}$$

where d represents the local "dark-channel" of input image $I(x)$, $A(x)$ is estimated atmospheric light intensity, $t(x)$ is the transmission map, $J(x)$ is the defogged output, and the parameter λ is either fixed or estimated by processing $I(x)$. Clearly, this algorithm

is a heuristic whose performance depends on a simple model of how fog degrades the image. As such, considering simple extensions of this architecture are likely to yield fruitful algorithms, especially coupled with an optimization strategy. For this application, we push the this heuristic further via a simple algorithmic generalization, depicted in Figure 3.6, which utilizes optimization to find a better sensor-fusion strategy. In essence, the architecture composes two layers of the aforementioned heuristic, with a parallel branch added during the training-phase to control the level of overfitting. To elucidate, in addition to training the network using sub-images (chips) and with standard boosting techniques (e.g. translation), we found that training an auxiliary network $j(X)$ to identify whether is a structure present improves the level clarify in the resulting $J(X)$.

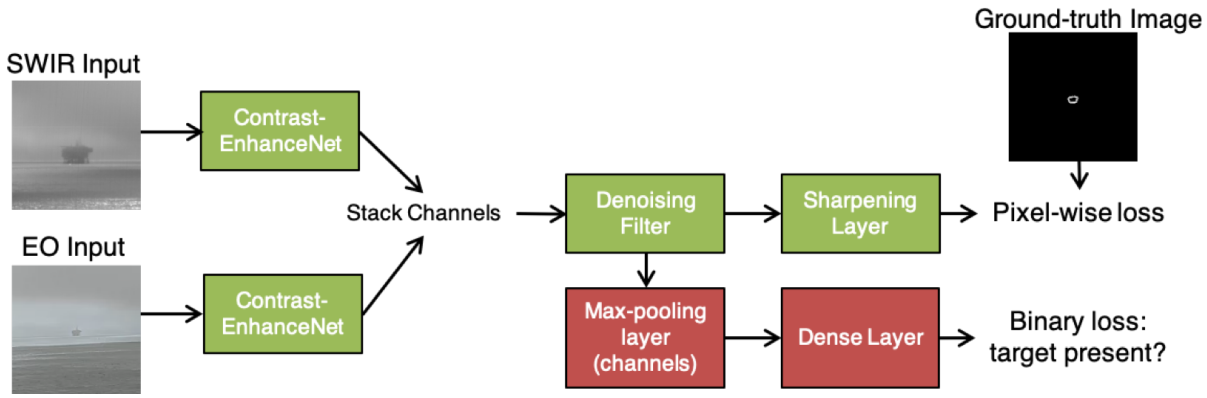


Figure 3.6: High-level architecture of the fused detection algorithm.

To construct a meaningful functional amenable to optimization routines e.g. gradient descent, we manually selected a portion of the EO training data corresponding to clear (no-fog) weather conditions to generate templates z of the structural features in the harbor via spatial-domain filtering. By mapping chips of the remaining training and testing data to the different templates, we generated a pseudo ground-truth binary-valued "label" images Y for every image X. To account for the change in modality, from visible features to edge-features, we evaluate the loss functional as follows:

$$||Y - \nabla_{x,y} J(X)||_F + \lambda ||y - j(x)||_2 \tag{3.13}$$

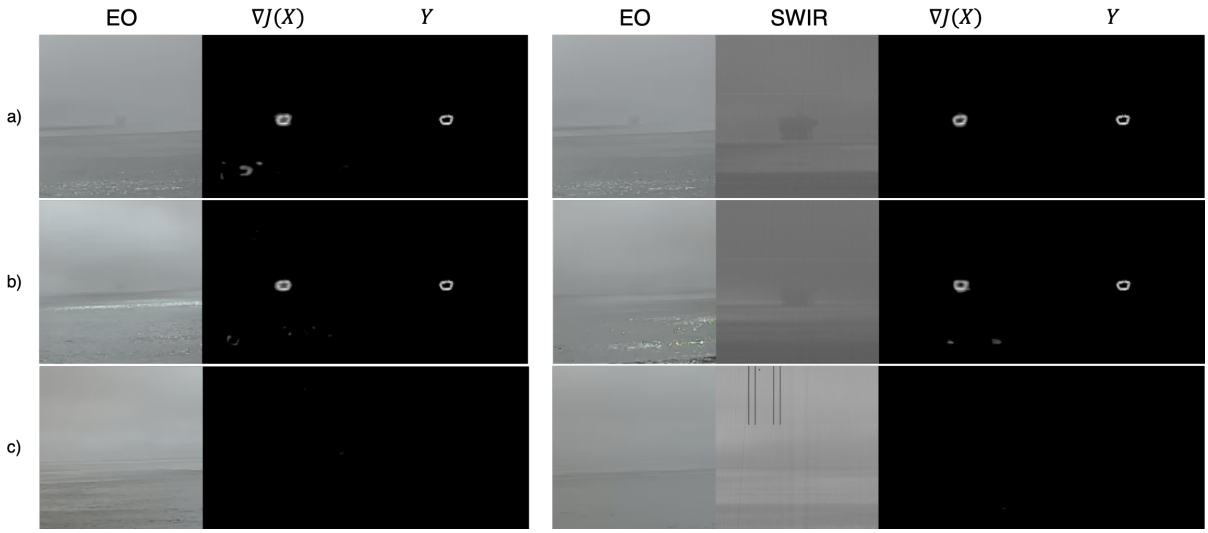


Figure 3.7: (a-b) Examples of edge-detection from dim EO and fused EO-SWIR imagery, taken from different geometric perspectives at various levels of fog. (c) A less exciting, but important, test-case that is used to validate that network is not overfit.

where y takes binary values representing if a sub-image should contains any structure pixels, and $j(x)$ is the aforementioned (generic) auxiliary detection network.

Thus, we seek to match the gradient of the defogged EO/IR images to the gradient of the clear EO image, while ensuring the intermediate representation can be used to identify the presence of an object. This form of weak supervision has also been referred to as "distant" supervision in the literature [151]. In typical scenarios it is important to ask if $J(X)$ produces meaningful information; considering the range to objects of interest (1-2 miles) and the density of the fog, for this dataset we evaluate this by monitoring the agreement of the gradients (Eq. 3.13) with respect to perceptual similarity between $J(X)$ and Y.

As depicted in Figure 3.7, the resulting algorithm produces surprisingly well-confined object signatures when objects (however dim) are in the scene. Moreover, the addition of SWIR information enhances detection in most cases and reduces the amount of background speckle. While the result itself is somewhat unsurprising, given the demonstrated capabilities and approximation power of modern DNNs, it is interesting that we can generate these networks from an algorithmic point-of-view rather than via blind-application of generic

prototypes. In particular, this application provides evidence that fully-convolutional networks can also be interpreted via composition of various "sub"-algorithms that can be parameterized (and initialized) in a sensible way [164]. Furthermore, the interpretability of the trained algorithm can be controlled via constraints in the optimization strategy.

3.3 Discussion

As we have shown in this chapter, traditional algorithms and programs can be used to construct highly-structured and performant interpolants for high-dimensional problems. The curse of dimensionality is effectively broken in these problems when the algorithm designer can identify and initialize the deep network with a suitable surrogate structure for the approximation task. Initializing models with known algorithms has the benefit of affording relatively simple architectures that can be further trained, or embedded within a converging framework (e.g. a larger polynomial network or DNN), which is useful when the algorithm designer does not have the exact graph of the target function or knowledge of all the interior nodal functions.

From the perspective of computer vision, the main attractiveness of the Deep Algorithms approach are: (1) natural 0-flop initialization, (2) potentially low data requirements, and (3) interpretable parameters. In the defogging problem, for example, a λ parameter generated inside the network can be directly interpreted as the inferred "level of fog". In the iEEG SoZ example, several parameters can be extracted indicating the sporadicity and correlation between different neural firing patterns, which currently serve as clinical diagnostic tools. Thus, while DNNs generally offer excellent performance in regression, they are often require significant analysis post-training to extract additional (e.g. physically relevant) insights, whereas Deep Algorithms offer a way to extract these directly by utilizing structure form the start.

As mentioned, this construction technique often yields an overall polynomial or rational-

function form that is conveniently differentiable, but also a good way of understanding the predictive power of models that is tied directly to the original approximation theoretic results [214, 125, 158]. Although not explored in this chapter, one intrinsic benefit of the polynomial representation is that the derivative of the approximation can be easily computed and controlled, which can make using regularized learning rules more feasible [131].

Connection to Related Work

The present work connects to and is motivated by the recent and renewed interest in differentiable programming [15, 211]. The literature on this subject is vast, extending back to the early work on automatic differentiation (AD) [12, 70] and learning via gradient updates [185, 123, 84]. Early work focused on its use in accelerating numerical computation in scientific applications [95, 52, 2], as well as for solving nonlinear problems [175, 7]. From the perspective of approximation, it was well known that AD could be used to accelerate the search for optimal fitting parameters given a forward model [113, 120], with applications in system identification and model-based control [79, 210, 13]. These early works laid the groundwork by establishing the machinery for using AD and optimization for neural networks and pattern recognition problems in general [160, 54, 128, 187], although it wasn't until recently that large problems were able to be handled efficiently by commodity computing hardware [3]. To this end, early computer vision techniques focused heavily on hand-tuned features [228, 56] and borrowed judiciously from the optimization literature [14, 116, 5, 1]. In this context, it is not surprising that modern DNNs utilize similar optimization and AD machinery for learning of semantic tasks.

What is surprising, however, is that the community has found success with relatively simple and abstract network designs (e.g. DNNs, RNNs, CNNs), with limited attention given to bolstering previous computer vision techniques with modern computational

firepower. Clearly, not every problem needs to be solved by a black-box DNN architecture. In fact, for many problems DNNs are not optimal, and utilizing the true compositional structure can provably yield more-accurate, faster-learning, and more-robust networks with optimal rates of convergence [149, 144].

This sets the stage for our work, which elucidates how old hand-designed algorithms are intrinsically connected to modern deep learning designs (e.g. ConvNets, ResNets), and argues that these paradigms can work in harmony as an effective and scalable design principle. From an systems perspective, the techniques presented here are in-line with the modern views on differential programming and network design [137, 224], and in the same spirit of more applications-focused works such as [45, 40, 140] that provide ground-up visual interpretability. Previous works have also tried to formalize the idea of incorporating Turing machines [83], or utilizing a programmer's knowledge [179, 90, 138], and even incorporating polynomials [21], but they have stopped short of realizing that conventional human-designed algorithms often serve as capable "deep" architectures *with natural parameter initializations* prior to any machine learning or numerical optimization.

Conclusion

Any sequential program in any programming language that has ever or will ever be invented can be encoded as a polynomial network. While this is also true of other computing frameworks (Turing machines, lambda calculus) and via universal approximators such as neural networks, polynomial networks offer a direct, efficient, and scalable way to encode that requires 0 training flops (e.g. to match network output to original function output). This paradigm does not currently exist in the context of DNNs, and moreover has (to-date) not been utilized in the context of machine learning despite nearly all neural network "universal approximation" proofs relying directly on polynomials. Our work points this out, and gives practical examples of how to construct/expand this encoding on modern high-dimensional problems.

There are numerous ways this idea of Deep Algorithms will influence the future of deep learning, besides casting an umbrella over current trends in DNN design. From the theoretical side, an open question is why deep networks are superior to shallow networks in generalization performance, with a recent finding being that deep networks are able to capture the compositional structure of the target function yielding better approximation properties. Deep Algorithms offers a practical avenue for connecting these ideas with the practice of deep learning, since for practical problems the true structure of the target function is usually unknown, although algorithms often offer a surrogate compositional representation that can be exploited in a similar way. On the algorithmic side, Deep Algorithms can be used to expand the expressivity of current designs by exploiting the richness of algorithms, including via internal and run-time objectives, as well as by accelerating encoding of capsules.

Chapter 4

Applications in Imaging & Recognition

In this chapter, we detail several problem domains where we apply lessons from polynomial approximation, deep minimum Sobolev norm (MSN) networks, and Deep Algorithms to modern imaging and recognition tasks.

4.1 Wide-Area Aerial 3D LiDAR Recognition

Annotated models of natural and urban environments are desirable for a number of applications, including geographical surveys, urban planning, and even civilian search-and-rescue operations. As transistor switching times have decreased, light-detection-and-ranging (LiDAR) has emerged in recent years as a viable option for surveying large landscapes, as opposed to via optical RGB-images alone. LiDAR introduces new information into surveys, primarily a "depth" channel that represents the distance from the LiDAR camera/platform to optical scatterers in the scene (this can be re-referenced to alternatively produce an earth-normalized height map). While such information could be derived from stereographic 3D-reconstruction, LiDAR provides this information to a much higher accuracy and is typically regarded as the gold-standard in structural modeling and mapping applications. Today we find LiDAR systems nearly ubiquitously, on ground-based, airborne, and–most captivatingly–on real-time autonomous vehicle platforms.

Despite the increasing availability of wide-area LiDAR imagery, automated annotation and scene understanding of this modality remains an open problem in computer vision. While structural and geometric segmentation codes have proven useful for recognition in small curated databases of 3D objects such as [226] and [115], these have recently been out-performed by modern deep learning approaches both in terms of accuracy and run-time performance [215, 65, 168]. However, demonstration of deep neural networks (DNNs) on real-world 3D LiDAR remains in its infancy, perhaps due to the insatiable amount of data required for these methods to be successfully validated and, more-generally, the high cost of generating high-quality labels for classes of interest.

In an attempt to bridge this gap, we begin by outlining an automated and scalable method for labelling wide-area geospatial data such as 3D LiDAR. We demonstrate that, where available, public ledgers such as OpenStreetMap or GoogleMaps can be leveraged to generate high-quality semantic segmentation masks for road, building, and natural

features (among other interesting classes), that can be used to train and validate detection, localization, and classification algorithms that we design. We show, that the method is also flexible when imagery and/or labels exist in non-geographic coordinate systems (e.g. the common case where small objects such as vehicles have labels in pixel-coordinate space), and that having flexibility between both representations allows for interesting real-time applications.

While this method can be used to automatically generate annotations of large wide-area imagery datasets, it is inevitable that some portions of the data are mislabelled. For example, in the case where labels are derived from a crowd-sourced database such as OpenStreetMap (OSM), some annotations may be missing from the ledger. Thus, in this scenario if a given segmentation or classification algorithm is "trained" with respect to an objective function that matches the network output to a naively presumed target value, the network is not only overfit, but will ultimately exhibit non-desirable performance in the wild. We attempt to address this issue in wide-area urban LiDAR and RGBD imagery in a two ways: (1) by collecting labels that geographically canvas the entirety of the dataset, and (2) by carefully designing a loss function that balances positive labels with dependence on the rendered depth channel. We show that incorporation of these techniques results in excellent pixel-wise, intersection-over-union, and instance detection scores for the geospatially-relevant object classes considered (`roads`, `vegetation`, `buildings`, and `vehicles`).

The rest of this section is organized as follows. In Section 4.1.1 we briefly summarize prior work related to 3D LiDAR segmentation in the context of geospatial annotation, including some of our own work on fast clustering [34] and voxel-based classification of 3D PCD [170]. In Section 4.1.2 we describe our method for automatic labelling of geospatial data such as 3D LiDAR and co-incident 2D electro-optic (EO) imagery, including the ability to manually stitch custom 3D models. In Section 4.1.3 we discuss some algorithms for processing 3D LiDAR data and associated learning objectives and algorithms. Finally,

in Section 4.1.3 we discuss the performance of the prototype system, and applications for future research, including incorporation of these techniques in existing EO and LiDAR map datasets.

4.1.1 Point-based and Voxel-based Algorithms

Processing and segmentation of 3D LiDAR point-cloud data (PCD) has been an active research area for more than a decade. However, until recently, most methods of acquiring point-clouds involved reconstructions based on stereo-images [62]. With increased availability of platform-mountable LiDAR systems, there has been a renewed interest in semantically segmenting and visualizing these point-clouds in real-time, e.g. for autonomous vehicle navigation.

Initially, most methods for processing LiDAR point-cloud data were focused on the 2D topographical problem for applications such as urban and farmland surveys [99], or even understanding and inference of 3D shape information [136, 135]. As consumer computer hardware improved, and spurred by the development of data visualization and filtering tools such as LAStools [102], direct processing of 3D PCD became a viable option, albeit for time-insensitive renderings. Numerous methods for recognition of PCD have been proposed, including 3D cluster-based approaches based on hierarchical point-selection, density-estimation, and local neighborhoods [29, 212], as well as more-abstract segmentation criteria such as graph-cuts, shape-priors, and rule-based segmentation [139]. These methods have demonstrated incredible advancements in 3D PCD processing, but have been limited to small example databases due to the lack of large annotated LiDAR PCD datasets and suitable fast algorithms for processing them.

In the wake of deep learning, modern learning-based approaches have come to match the performance of these meticulously designed, classical, vision-based image-processing approaches, both in terms of semantic segmentation accuracy and run-time performance [215, 65]. Instead of rendering 3D voxel grids [229, 170], recent methods have focused on

direct processing of 3D PCD using point-set neural networks [168, 167]. These approaches have been validated on a large number of datasets of small segmented objects [216], indoor scenes [55], and ground-based outdoor scenes [75, 193], while the application to large real-world datasets remains an open challenge, in terms of both the required processing resources and the size of the validation set. Some datasets do exist in attempt to address this scenario, namely various versions of 2D/3D ISPRS challenge data [155] and more-recently the (non-LiDAR) 2018 DeepGlobe challenge [58], but even these are limited to small areas of a few city blocks, or limited in availability and usage. In addition, prior work does exist in leveraging crowd-sourced OSM-labels for wide-area datasets [190, 57, 189], although these applications have largely been limited to classification tasks rather than detailed semantic segmentation and analysis of a wide-area scene.

At this point, it is worth mentioning some of our prior contributions to the LiDAR point-clustering and recognition problems.

Fast Indefinite Multi-Point Clustering (FIMP)

In [34], we introduced a new class of objective functions and an associated fast descent algorithm that generalizes the K-means algorithm. Here, "fast" means that the algorithm scales linearly with the number of points and also with dimension. The algorithm represents clusters as unions of Voronoi cells and an explicit, but efficient, combinatorial search phase enables the algorithm to escape many local minima with guaranteed descent. The objective function has explicit penalties for gaps between clusters, which leads to an indefinite objective function [112]. A brief literature review is included in [34], which highlights the differences from our contribution to prior approaches.

Before defining our objective function and algorithm, we must first introduce some notation.

Let $\mathbb{R}$ denote the set of reals, $\mathbb{N}$ the set of non-negative integers, and $\mathbb{N}_+$ the set of positive integers. Let $\mathbb{N}_p = \{0, 1, \ldots, p-1\}$, for $p \in \mathbb{N}$, with $\mathbb{N}_0 = \{\}$. Let $\|\cdot\|$ denote

the standard Euclidean 2–norm. Let $\|\cdot\|_F$ denote the Frobenius norm. Let e denote the column vector of all ones; the dimension will be apparent from the context.

Breaking from custom, we will place row indices on the left. For example, ${}_iA_j$ will denote the (i,j)-th entry of the matrix A. We will also use A_j to denote the j-th column of A, while ${}_iA$ will denote the i-th row of A. We will use a double index notation for block matrices. So ${}_{p;}A_{k;}$ will denote the (p,k)-th block sub-matrix of A, and ${}_{p;i}A_{k;j}$ will denote the (i,j)-th entry of the block ${}_{p;}A_{k;}$. Frequently our row and column indices will start with 0 rather than 1.

Let $N, L, K \in \mathbb{N}_+$. Let $Y \in \mathbb{R}^{N\times L}$. Block partition the columns of Y into K blocks:

$$Y = \begin{pmatrix} Y_{0;} & Y_{1;} & \cdots & Y_{K-1;} \end{pmatrix}.$$

Let $\lambda \in \mathbb{N}_+^K$ for $K \in \mathbb{N}_+$, and let $Y_{k;} \in \mathbb{R}^{N\times\lambda_k}$ for $k \in \mathbb{N}_K$; that is, λ_k denotes the number of columns in $Y_{k;}$ and $\sum_{k\in\mathbb{N}_K}\lambda_k = L$. Using Y and λ, partition $\mathbb{R}^N$ into K mutually disjoint subsets $\mathcal{S}_k$ according to the following membership rule: $x \in \mathbb{R}^N$ is assigned to $\mathcal{S}_k$ if k is the smallest integer for which

$$\min_{l\in\mathbb{N}_{\lambda_k}} \|x - Y_{k;l}\| = \min_{p\in\mathbb{N}_K}\min_{j\in\mathbb{N}_{\lambda_p}} \|x - Y_{p;j}\|.$$

Let $\mathcal{S}_{k;l}$, for $l \in \mathbb{N}_{\lambda_k}$, denote λ_k mutually disjoint subsets of $\mathcal{S}_k$. The membership rule for $\mathcal{S}_{k;l}$ is as follows: $x \in \mathcal{S}_k$ is assigned to $\mathcal{S}_{k;l}$ if l is the smallest integer for which

$$\|x - Y_{k;l}\| = \min_{n\in\mathbb{N}_{\lambda_k}} \|x - Y_{k;n}\|.$$

We will call $\mathcal{S}_{k;l}$ as a sub-cluster and $\mathcal{S}_k$ as a cluster. Let $X_{k;l}$ denote the sub-matrix of X that contains all the columns of X that lie in $\mathcal{S}_{k;l}$.

Then, the problem can be defined as follows.

Let $M, N \in \mathbb{N}_+$. Let $1 < L_1 \in \mathbb{N}_+$. Let $X \in \mathbb{R}^{N\times M}$ and $\Omega \in \mathbb{R}^N$ be given. Let $\alpha, \beta \geqslant 0$ and $\varsigma, \gamma > 0$ be given. Let $Y \in \mathbb{R}^{N\times L}$ for some $0 < L \leqslant L_1$. Let $K \in \mathbb{N}_+$ and

N	–	Dimensionality of vectors to be clustered.
M	–	Number of vectors to be clustered.
X	–	$N \times M$ matrix of vectors to be clustered.
L_1	–	Maximum number of sub-clusters allowed.

Table 4.1: FIMP Inputs.

K	–	Number of clusters found.
L	–	Total number of sub-clusters found.
Y	–	$N \times K$ matrix of sub-cluster centers.
λ	–	K dimensional vector of number of sub-clusters in each cluster.
$Y_{k;}$	–	$N \times \lambda_k$ matrix of sub-cluster centers of cluster k.
$Y_{k;l}$	–	Center of sub-cluster l in cluster k.

Table 4.2: FIMP Outputs.

$\lambda \in \mathbb{N}_+^K$ such that $\sum_{k \in \mathbb{N}_K} \lambda_k = L$. Let

$$\begin{aligned} F(Y,\lambda) &= \sum_{k\in\mathbb{N}_K} \sum_{l\in\mathbb{N}_{\lambda_k}} \|X_{k;l} - Y_{k;l}e^T\|_F^2 + \alpha \sum_{k\in\mathbb{N}_K} \sum_{l<n\in\mathbb{N}_{\lambda_k}} \|Y_{k;l} - Y_{k;n}\|^2 \\ &+ \frac{\beta}{\gamma} \sum_{k<p\in\mathbb{N}_K} \sum_{l\in\mathbb{N}_{\lambda_k}, j\in\mathbb{N}_{\lambda_p}} (1 - \gamma\|Y_{k;l} - Y_{p;j}\|^2) + \varsigma\|Y - \Omega e^T\|_F^2. \end{aligned} \tag{4.1}$$

Given X, Ω, α, β, ς, γ, L_1, find L, Y, K and λ, which solves the minimization problem

$$\min_{Y,\lambda} F(Y,\lambda), \qquad \text{when} \qquad \beta < \frac{\varsigma}{2(L_1 - 1)}.$$

For clarity, we have summarized the notation in Tables 4.1, 4.2 and 4.3. Note that the term $\|X_{k;l} - Y_{k;l}e^T\|_F^2$ is the usual penalty on the distance of a column of X from its assigned sub-cluster center. The term $\|Y_{k;l} - Y_{k;n}\|^2$ is a penalty on the distance between the centers of two sub-clusters that belong to the same cluster. The term $-\|Y_{k;l} - Y_{p;j}\|^2$ is a penalty on the distance between sub-cluster centers belonging to two different clusters. Note the negative sign as we want this term to be large in absolute value. This is also the term that makes the objective function indefinite. The term $\|Y - \Omega e^T\|_F^2$ is a penalty on the distance of the sub-cluster centers from Ω and is there to prevent the objective function from becoming unbounded from below.

Ω	–	Best chosen to be the mean of the columns of X.
α	–	Larger values forces sub-clusters of a single cluster to lie closer together.
β	–	Larger values forces clusters apart.
γ	–	Larger values increases minimum gap between clusters.
ς	–	Small number that prevents the Y's from drifting too far from Ω.

Table 4.3: FIMP algorithm parameters.

The global optimum is hard to find, so we settle for a "local" minima, though the word "local" is dubious in a discrete setting. The bound on β is needed to ensure that F is bounded from below. We recommend choosing γ to be reasonably small so as to discourage the formation of empty sub-clusters. A good default choice for Ω is the global mean $\Omega = \frac{Xe}{M}$. The role of α is to encourage sub-clusters belonging to a single cluster to be close together, while the role of β and γ is to encourage clusters to be well-separated. The role of ς is purely technical at this point; it keeps F bounded from below when some sub-clusters become empty.

There are several components to this problem and it is difficult to find a linear presentation. Assuming that the reader is familiar with the K-means algorithm, we provide a rough outline of the algorithm and then present the details. Our goal is a guaranteed descent algorithm to a local minimum. Additional details can be found in the main text of [34].

The algorithm is a form of block coordinate descent, complicated by the presence of a combinatorial part. The algorithm proceeds in multiple stages, and in each stage we guarantee that F is non-increasing.

1. Initialize Y (essentially randomly from columns of X) with $K = L_1$.

2. Assign columns of X by a nearest center rule.

3. **Repeat**:

   (A) Compute C, T and R.

   (B) **For each** column $Y_{k;l}$:

(a) **If** sub-cluster $\mathcal{S}_{k;l}$ is empty delete if descent is possible.

(b) **Else** among the following choices, **pick** the **one** with maximum descent:

(i) Split off sub-cluster into its own cluster if descent is possible.

(ii) Transfer sub-cluster to another cluster if descent is possible.

(iii) Swap with another sub-cluster if descent is possible.

(c) Update λ, T, R and other variables as needed.

(C) Freeze all partitions $\mathcal{S}_{k;l}$ and move Y to the nearest critical point.

(D) **For each** column X_j:

(a) **If** $L < L_1$ and if $X_j = Y_{K;0}$ would lead to descent take this path.

(b) **Else** assign to nearest $Y_{k;l}$ (guarantee descent).

(c) **If** X_j changed membership, freeze all the partitions $\mathcal{S}_{k;l}$ and modify Y to reach nearest local minimum (guarantee descent).

4. **Until** no (significant) descent

As mentioned earlier, the K-means objective can be attained by forcing all clusters to have only one sub-cluster, thereby skipping step 3B and effectively eliminating the first penalty term in the objective function F. We point out a key difference with what most people call Lloyd's [133] or Forgy's [72] version of the K-means algorithm: we update the centers every time X_j is re-assigned. This second version of K-means is known to be more efficient in the Euclidean case [119, 194]. Furthermore, it guarantees (modulo floating-point errors) that no empty clusters will be produced by K-means, which is a frequent problem in Forgy's version.

Proposition 4.1.1. The cost of one loop, steps 3(B), 3(C) and 3(D), is $O(NML + L^2)$ flops. Each step of the loop is guaranteed not to increase the objective function.

Proof. Established in the propositions in the main text of [34]. □

Despite this fast algorithm, the performance can be limited when the density of 3D LiDAR points is very high (e.g. 100M points). Moreover, to process real-world LiDAR data even semi-automatically requires a fair bit of tuning of the objective function and algorithm parameters, e.g. to learn data-dependent patterns of targets of interest (e.g. `buildings`, `vehicles`, etc.). To this end, we began exploring other representations of the data that are more amenable to fast processing over large geographic regions.

Object Recognition from 3D-Voxelized Grids

An alternative to point-data processing, is to process voxelized grids. In [170], we were one of the first to employ 3D CNNs to classify real-world point-cloud imagery. In this work, small geographic cut outs (20m $\times$ 20m) were extracted from the 3D PCD and gridded in X, Y, and Z dimensions at a resolution roughly equal the mesh norm of an interpolated point-cloud. The networks we built were built were relatively small ($\sim 150K$ parameters), but employed 8-10 layers to achieve robust performance ($\geq 90\%$) over the variety of different target-class variations we tested on. The algorithms were trained and validated using distinct geographic regions in the large 2007 OGRIP [159] dataset, which was initially labelled. Using a technique based on OpenStreetMap (OSM), we labeled a number of areas as `highway` or `building`, and wrote in-house 3D stitching tools to insert rendered (simulated) 3D LiDAR models of various real-world `vehicles` into a variety of natural locations in the LiDAR data. Overall, the scheme worked extremely well at classifying and localizing targets within imagery, including by investigating the 3D activation maps generated by the convolutional layers.

However, the scheme was ultimately too expensive for searching a wide-area scene covering e.g. 10-20 km^2 or larger in a reasonable amount of time. This complexity only increases with higher-resolution LiDAR datasets. While the initial idea was to queue regions of interest that were roughly clustered by the previous FIMP algorithm to the 3D CNN networks, this route was ultimately abandoned. As mentioned, FIMP is a "fast" algorithm, but it requires looking at all the data point before even one iteration of the algorithm completes. As such, even rudimentary heuristics, such as via octree sorting often yielded much better results in terms of the overall run-time latency to detect targets in large previously un-segemented and un-curated geographic point-clouds. Notice that such structured search algorithms exploit locality in the data, and thus are extremely simple to implement directly at the edge device or sensor (e.g. LiDAR platform) since data is collected locally. In this vein, we developed a more scalable technique to quickly analyze large area point-clouds via relatively inexpensive 2.5D representations prior to full 3D exploitation, as discussed in the next section.

4.1.2 Developing Wide-Area LiDAR Datasets

One of the primary goals of this work is to enable the full exploitation of publicly-available wide-area LiDAR datasets such as 2013 Vancouver [50], 2015 Dublin [117], 2017 OGRIP [159], and 2018 District of Colombia [157], which are all high resolution scans of rural and urban terrain, but with no associated labels for semantic classes of interest. We focus on the 2015 Dublin dataset here, because the structure of the dataset is typical of many geospatial imagery-collections, and thus these processing algorithms are highly applicable to various other datasets, including other modalities such as EO, synthetic aperture radar (SAR), and ground-based imaging (when camera, orientation, and position information are readily available).

Imagery Dataset

In this work we use the 2015 Dublin LiDAR data collection, which is freely available at https://geo.nyu.edu [117]. The dataset was collected via an airborne platform containing both a full-waveform LiDAR sensor (TopEye system S/N 443), as well as a Phase One imaging system that captured multiple modalities, including geo-referenced RGB imagery, color-infrared imagery, and video data. In this work, we choose to work exclusively with the LiDAR and RGB optical measurements, since these are the most commonly available modalities in geographical surveying and search-and-rescue operations, and have the clearest applicability to modern scenarios, e.g. autonomous vehicles.

Data was collected along 41 different flight paths over two sessions at an average altitude of 300m, canvasing a 3km x 3km region around Dublin, Ireland, as shown in Figure 4.1. EO images were recorded at regular intervals along each flight-path, and LiDAR from each path was integrated, referenced, transformed, and merged into a single LAS point-cloud. Thus, for each of the 41 flight paths, the dataset consists of an large number of EO images (50+ images of resolution 9000 x 6732 pixels), and one LAS point-cloud. In post-processing, these flight-path (FP) point-clouds were also merged into a single "composite" LAS point cloud, that was subsequently tiled by geographic area for processing efficiency. This merged format is the most common in mapping applications, since it provides the densest representation of large stationary objects, such as buildings, roads, and natural features. We are able to process these formats, but we note that for real-time applications, the flight-path LiDAR is more representative in terms of data-sparsity. For comparison, the density of the flight-path LiDAR is roughly 100 points/m^2, whereas the density of the composite PCD varies from 100-335 points/m^2 depending on the number of flight paths covering that region. Thus, each flight path image of 60.5M pixels corresponds to upwards of 12M LiDAR points. To make these numbers more manageable for real-time processing, we downsampled the RGB imagery by a factor of 4 prior to registration with the 3D LiDAR PCD.

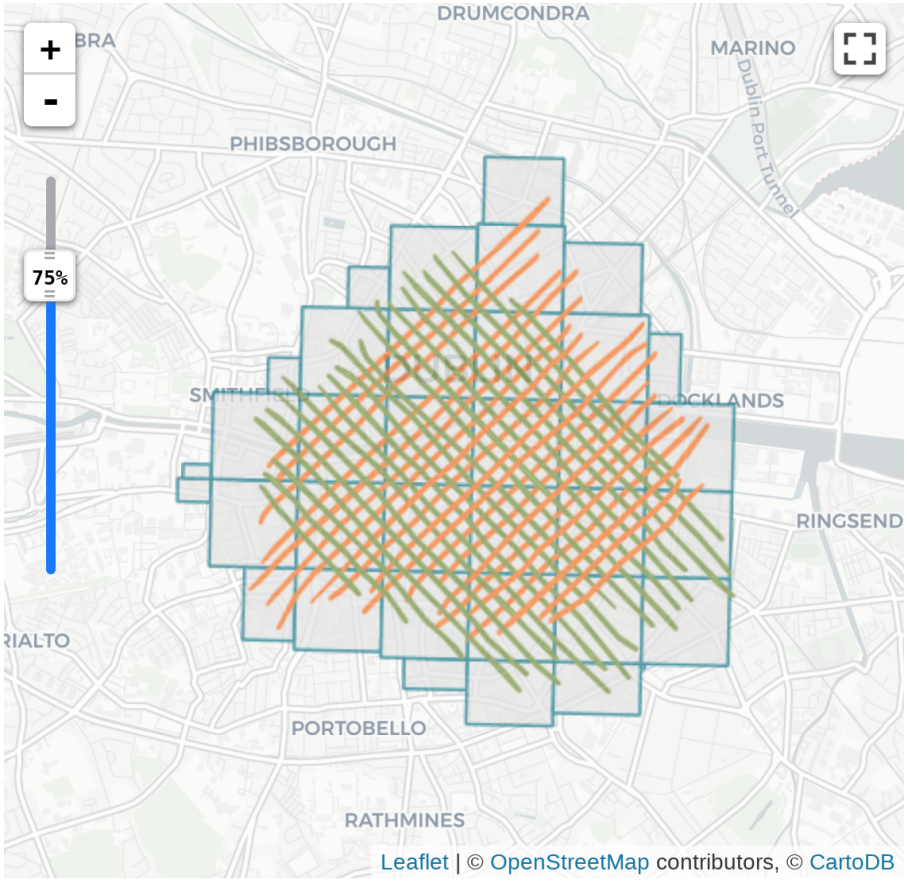


Figure 4.1: Flight-paths of the data-collection from the 2015 Dublin dataset. Boxes indicate the aforementioned geographic tiles, while the green and orange lines represent different flight-paths.

Mining OpenStreetMap

As mentioned, the biggest issue in exploiting such wide-area imagery datasets is the lack of labels for relevant classes of interest. To alleviate this issue to a significant degree, we developed an automated technique for mining labels from the public ledgers, such as crowd-sourced OpenStreetMap (OSM) databases. For the purposes of this work, we use freely available annotation data provided by (German) Geofabrik servers in the OSM XML and ESRI Shapefile formats, though it is obvious how the presented method can be extended to other annotation formats.

OpenStreetMap is conceptually organized as a set of elements of type `node`, `way`, or `relation`. The distinction between these is fairly obvious: `nodes` represent objects at some scale (e.g. a building vs a neighborhood), `ways` represent roads and trafficked

regions, and `relations` represent logical associations between `nodes`. Each element is additionally assigned several keyword attributes (such as feature-type, location, polygon information, etc.), which can be further qualified with a predicate. In the data that we collected from OSM, these elements are typically represented as GeoJSON objects amenable to filtering, which are accessed as simple Python dictionaries (e.g. Fig. 4.2).

```
'OSM_id': 393531070,
'OSM_label_1': {'bikepaths': False,
 'buildings': False,
 'clutter': False,
 'footpaths': False,
 'highways': True,
 'parks': False,
 'rails': False,
 'rivers': False,
 'roads': False,
 'structures': False,
 'walls': False},
'OSM_properties': {'highway': 'bridleway'}}
```

Figure 4.2: A collected OSM-element (GeoJSON object) is augmented with binary-classification vector $y \in \mathbb{B}^k$.

To extract useful information automatically from such data, we first collected and queried all the objects existing within the extents of the 2015 Dublin dataset (specified via WGS-84 bounding-box coordinates), and sorted them to identify unique class-types. In the case of the Geofabrik data that we used, many classes were already adequately separated (e.g. as `building`, `road`, `water`) into separate shapefiles, though some classes needed custom sorting rules (e.g. extracting man-made `structures` from the points-of-interest shapefile, etc.). While a range of natural language processing (NLP) tools can be used to automatically sort these elements in the general case, it is often more convenient and accurate to leverage the various tags that are associated with different classes of objects, and to write simple rules that match keywords and their associated predicates. For a more general example, see the approach used in [170]. In this work, we used a set of simple rules to match elements with mutli-class labels.

Once the relevant label-extraction rules are identified, we may begin the process of mining the labels. The algorithm begins by collecting geospatially-referenced imagery,

identifying its geographic extents (e.g. as a polygon), and querying an OSM-API for elements in the region specified by the extents. The OSM-API can be implemented in various ways, e.g. in [170] it is implemented via the `overpass` Python-API, similar to the use of Overpass-Turbo in [71]. As implied, in this work we created custom functions to read (offline) the data provided by Geofabrik servers. Once these elements are collected, they are semantically classified, and archived as polygon objects for subsequent use in the labelling steps. Unfortunately, not all OSM annotations have shape-information; as described in the next section, in some cases we can employ simple heuristics to still leverage portions of these annotations.

Labelling 3D LiDAR Data and EO Imagery

With extracted and classified polygon shape-information, it is relatively straight-forward to label 3D point-cloud data. When the labels are to be applied to stationary, optically-opaque ground-attached objects (e.g. buildings, structures) it is usually reasonable to assume all points at a geographic index correspond to the same object. Thus, the algorithm should look at each polygon, query the point-cloud for indices that exist within the polygon, and "mark" each point as belonging to that class. To implement this in the general multi-class setting, for example, one could mark points with a binary vector ($y \in \mathcal{B}^k$) representing assignment to each of k pre-identified classes. In this representation, labelling does not need to include a gridding step (which is computationally expensive, and ultimately prohibitive for wide-area imagery datasets).

Unfortunately, the shape information of OSM `ways` (which represent roads, walkways, bikepaths, etc.) currently do not contain polygon-type information, and instead include only `LineString` information that represents the start and end of a particular road-segment, along with some auxiliary information on the type of road. In our application, recognition and segmentation of roads is immensely useful, so we chose to infer this geometric information from the OSM tag (details are provided in the Appendix). However,

in estimating these geometries, care must be taken in the labelling algorithm to not replace well-trusted labels (e.g. building annotations) with those that are only estimated. To this end, we opted for non-mutually-exclusive labels that are eventually handled by our loss function and prediction rendering routines (Sec. 4.1.3).

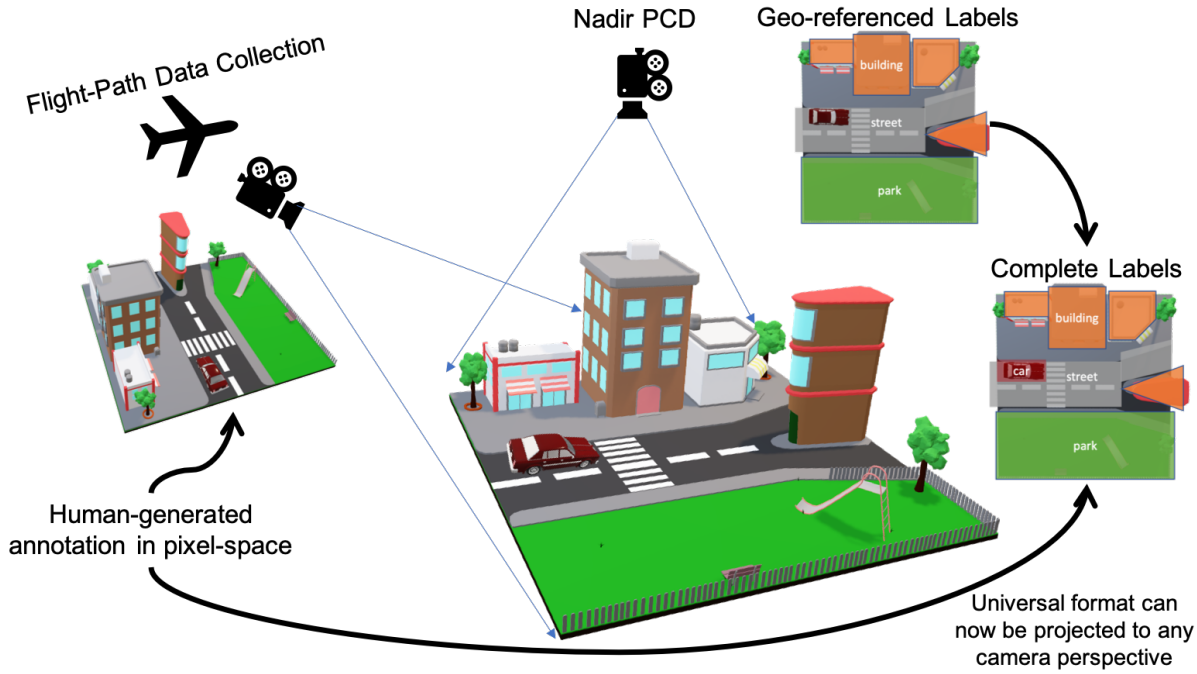


Figure 4.3: Co-registration between LiDAR PCD and depth images allows sharing of geographic and pixel-space annotations.

While identification of stationary objects is incredibly useful for geographical and urban planning applications, for some applications identification of ephemeral features such as vehicles is also of interest. These would not be stored accurately in a database such as OSM, but luckily can be manually identified with relative ease using conventional EO / RGB imagery. To this end, we labelled a subset of the 2015 Dublin data in the native RGB pixel-coordinate system (`row-col`), and transferred these labels into the flight-path PCD representation after applying a semi-automated geo-registration process on the whole dataset [63, 23]. Similarly, using this registration information, we were able to also project earth-referenced PCD information to the perspective of the camera frame, to generate a co-registered "depth" channel for the EO images (i.e. derived from the LiDAR PCD), along with corresponding OSM-derived label information (Fig. 4.3). In this

way, we were able to generate a high-quality semantically-labelled RGB-D dataset that covers a large portion of Dublin, Ireland. In the following, we emphasize the construction of this 2.5D representation of sparse LiDAR from flight path imagery because, in addition to mapping directly to the morphology of data in a real-time application, it provides a powerful representation that efficiently captures correlations across different spectral bands, as described in the next section.

Synthetic Models

As in [170], we maintain the ability to augment both the conglomerate and flight-path PCD, as well as the camera-rendered depth-channel imagery, with synthetic models of real-world objects. This is useful when processing datasets where no objects of a particular class exist, but it is still of interest to train algorithms to detect them. While realistic augmentation in the EO/RGB modality can be a challenging problem, the problem is simplified in the LiDAR/PCD modality and can greatly augment datasets (Fig. 4.4). Stitching can be implemented, for example, by identifying or generating canonical chips of the background class and using direct, patch-based replacement of portions of the point-cloud, as opposed to more computationally expensive techniques such as solving Poisson's equation in 3D [59]. This was used extensively in our work in [170], although in [171] we preferred manual vehicle annotations.

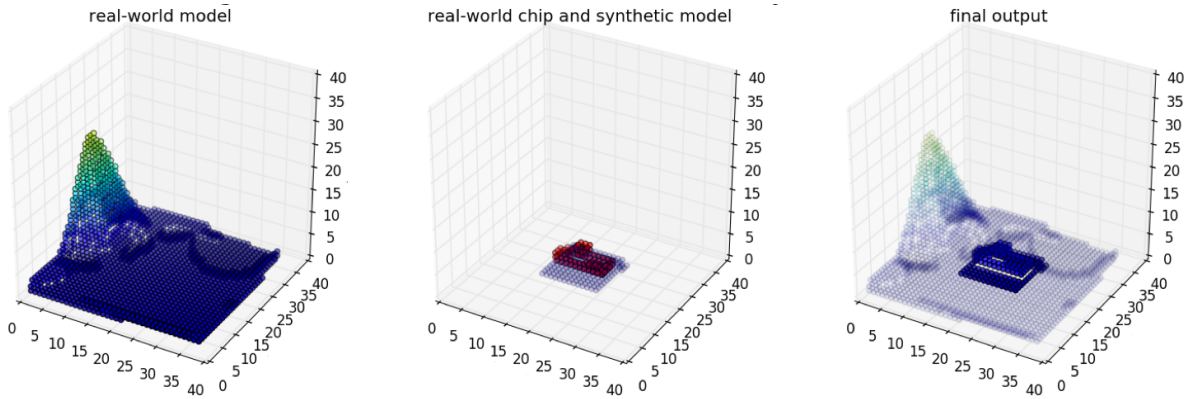


Figure 4.4: Synthetic models can be naturally stitched into LiDAR. Shown here is a simple example using a coarse voxel grid.

4.1.3 Recognition in Sparse 3D and 2.5D Imagery

With the dataset in place, we now turn to algorithms for detecting, localizing, and classifying objects in wide-area LiDAR scenes. There are various approaches this problem, some casting it as a problem in cluster-based segmentation, but for the purposes of this work we will focus on a formulation as regression problem. In this case, given a target function $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ specified only by data (i.e. input-output pairs $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$), the task is to design a function $\hat{f}$ which closely approximates f w.r.t. some measure, say $||f(x) - \hat{f}(x)||_\infty$, where $f : x \rightarrow y$. In practice, the target function f is unknown analytically, so the problem of finding a suitable approximation $\hat{f}$ consists of two steps: (1) finding an adequate model (classically, the most minimal model) for $\hat{f}$, and (2) finding its parameters. In this work we do not tackle the open design problem, rather we present techniques for refining the parameters of a given model function $\hat{f}$ using wide-area 2D and 3D imagery. To do so requires first defining a suitable model for x and y.

Data Model

A modern approach to the function-approximation-based segmentation and recognition of imagery is to use a convolutional neural network (CNN) [215, 65, 168]. The basic idea is that a spatially-invariant non-linear filter-bank can be used to detect hierarchies of shapes, or portions of shapes, in 2D and 3D imagery, and these detections can be used to classify input images by producing a continuous-valued output $y \in \mathcal{R}^k$, that can be thresholded to a multi-class vector $y_b \in \mathcal{B}^k$ indicating class-membership. However, in many instances, especially involving wide-area imagery, this type of chip-level classification can be an expensive approach to finding or localizing objects of interest (e.g. for the task of finding all the green cars in a M-km^2 region) [170, 188].

Instead, a variant of CNNs termed fully-convolutional neural networks (FCNNs) have emerged as a promising approach to the segmentation problem, that is able to produce pixel-level multi-class membership predictions while operating on regions larger than a

typical image chip. In truth, the convolutional-structure is not essential to the generation of pixel-level predictions, but utilizing convolution to guide segmentation is generally viewed as a rational choice (see [192] for details regarding the size of the receptive field), especially considering that in the space of possible designs convolutional structures have efficient CPU/GPU implementations. The application of FCNNs has a strong connection with conditional random fields (CRF) and other techniques that try to predict class-membership densities at each pixel location [43]. These techniques can be used, for example, in conjunction with a chip-level classifier or a subsequent instance segmentation routine [156].

However, one issue with these FCNN (and similar) codes is that they are typically only defined for input signals that appear on a regular grid, e.g. $x \in \mathcal{R}^n$. While this does not pose a problem for typical 2D imagery, it does for the task of segmenting and classifying 3D LiDAR point-clouds, since in a given region there are typically an indefinite number of points, whose value is generally not as meaningful as its location. That is, in LiDAR PCD, data-points are given as 3-tuples of the form $\{(x_k, y_k, z_k)|k = 1, ..., K\}$, where K depends on the density of the point-cloud collected by the LiDAR platform in a particular viewing geometry.

The typical and most common approach to this problem is to just grid the LiDAR PCD, by constructing a mapping from the space where the points lie to voxels, or cells that represent the extent of the space. To reduce the problem complexity for topographical mapping applications, where multi-bounce LiDAR responses are less useful, typically the 3rd dimension is collapsed and a 2D grid is written as $x \in R^{n_1 \times n_2}$, where the value at each grid-cell is a surrogate for a measure of the "height" (e.g. average, median) of the corresponding region in the point-cloud. Note that in cluster-based approaches, this type of gridding is not essential, and 3D-points can be directly segmented, albeit expensively. Unfortunately, to date such algorithms do not have efficient software implementations amenable to processing large point-clouds in near-real-time. Even point-set neural

networks such as [168, 167], for example, are relatively untested on sets $>$ 20K points. For comparison, in our high-resolution dataset, this would correspond to an approximate area of 60m^2, which is much smaller than the areas of $\sim$ 0.08km^2 that we consider at one-time in this work.

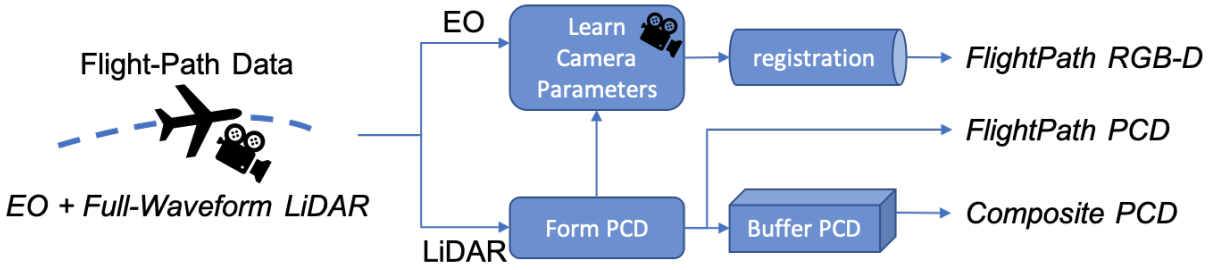


Figure 4.5: Overview of the data model.

Instead, we propose a hybrid algorithm that features a "gridded" 2D representation of the LiDAR PCD to first localize objects quickly, followed by a utilization of the full 3D model. In many cases, the 2D representation is sufficient to detect the super-class of objects (e.g. `vehicle`, `building`, `road`, etc.), whereas the 3D representation is more telling in the intra-class discrimination task (e.g. `sedan` vs `truck`), as evidenced by our low false-positive rates. In this work we will focus on recognition and localization from the initial 2D representation, since a more in-depth analysis of the cost of modern 3D extensions is required (see [170, 30] for a practical implementation of recognition in using the full 3D representation, albeit using voxel-grids), and because for many applications a 2D representation can be sufficient if collected from a non-nadir perspective resulting in a "2.5D" representation of LiDAR as a depth-channel image. Furthermore, when co-incident co-registered RGB imagery is available, as in the 2015 Dublin dataset, the 2.5D representation is actually preferable since we can directly form RGBD images that enhance scene understanding capabilities via a structured representation of color, texture, shape, and intensity information [167, 142, 25].

U-net Network Architecture

As an initial demonstration, we use an adapted (smaller) version of the U-Net architecture that is popular for segmenting 2D RGB images [181]. In short, we realized an implementation of a fully-convolutional neural network that operates on 2D input images of arbitrary size and depth, and produces output predictions on a grid of equal size to the input. In our implementation for the 2015 Dublin dataset, the size of the receptive field, measured by considering how many input pixels potentially influence an output pixel's value, is at most 26×26 pixels, which corresponds on average to a ground-sample area of 60m^2 in the nadir imagery (Fig. 4.6). In training, we use tiles of 512×512 pixels, each covering an area of roughly 0.01km^2, but this number is flexible and can be increased dramatically for inference (e.g. 2-4x larger).

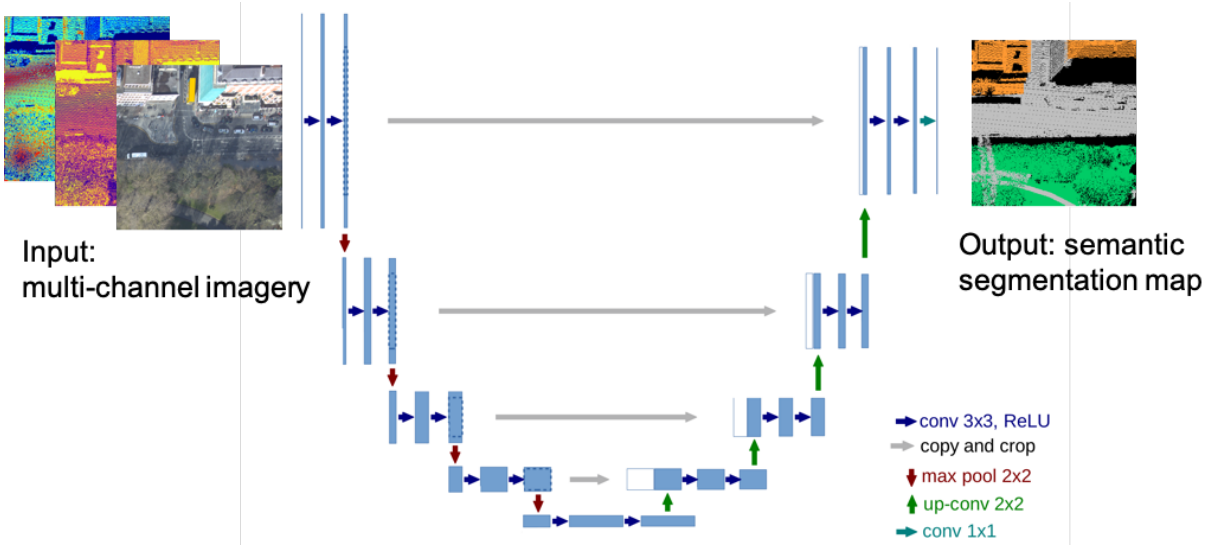


Figure 4.6: Example operation of a multi-channel extension of the U-Net FCNN architecture [181] for recognition in LiDAR depth- (D) and LiDAR intensity- (I) fused RGBD/I imagery.

In our implementation, we noticed that several factors influence the quality of the output segmentation maps. First, the size of the receptive field, which is adjusted by controlling the size and stride of the filter-windows, should be sufficiently large to provide context into the segmentations but also small enough that the network will not learn

biased, dataset-specific representations of where objects may lie (e.g. cars are always on labelled roads, next to buildings). Empirically, we found a good range to try is $15 \times 15 \leq R \leq 30 \times 30$. Second, the depth of the U-net should be adjusted such that for the smallest input image, a spatially-meaningful "encoded" pattern should be available. In this case, we found that having at least 16 pixels in the "middle" layer, produced desirable results for more-than-reasonable training times. Finally, when training this architecture, care must be taken to select examples images at a sufficiently large scale to provide relevance to the output detections (e.g. multiple different classes and boundaries present in the image), while providing sufficient attention to any given object, as to properly penalize any gaps in the output detections.

Note that other neural network architectures, such as DeepLab-v3 and InceptionResnet, were also considered. In our testing, these did not result in sufficient quantitative or perceptible improvements over U-net for this dataset and modality, and are thus not included in this work. Incorporating an enhanced multi-objective architecture such as Mask-RCNN or similar [156], on the other hand, is expected to improve localization accuracy beyond the semantic segmentation and grid-based instance detection methods we use in this work. However, significant changes to the operation and training of these architectures are required to accommodate issues of data and label sparsity, making this a candidate for future work.

Balanced Loss for Aerial LiDAR Imagery

There are several special considerations for training pattern recognition algorithms in sparse wide-area LiDAR imagery. In our application, the first consideration is that LiDAR returns may not be dense everywhere, resulting in gaps or holes in the imagery even though generated semantic labels are typically expected to be geographically dense. There are several possibilities to address this issue, such as (a) regressing on dense polygonal labels independent of the point-cloud density, and (b) regressing on point-data alone

and ignoring labels outside the labelled pixels/voxels. The issue with these approaches is that they trade specificity (dependence on LiDAR features to indicate a class) for sensitivity (point-wise accuracy), or vice-versa. Said another way, naively regressing on point-data alone encourages improvements in the true positive rate at the expense of the true negative rate, and potentially the false positive rate.

The second consideration is the availability of accurate and dense labels. In this case, shapefile information derived from OSM is generally quite accurate, but may not include annotations for *all* objects in the area (e.g. for `buildings` or `vegetation`) [170]. This prevents an accurate measure of the false positive rate for some classes. On the other hand, the OSM-derived `road` labels are spatially overestimated, providing an overly-pessimistic measure of the (pixel-wise) false negative rate. Luckily, manual annotations such as those for `vehicles` are fairly complete and accurate, albeit for a small set of images (typically smaller than the size of available training data). Thus, a good training strategy will use these priors to balance the precision and recall of the multi-class semantic segmentation and instance detection algorithms.

To address both of these issues we realized a scheme for training sparse LiDAR depth-rendered RGBD imagery via careful design of the loss function. The loss function that we use is based on the categorical cross-entropy loss that is commonly used to train DNNs and FCNNs except with some modality and data-specific modifications. In particular, we explored two different versions of this technique. In both cases, we define loss for each example as:

$$g(x, z) = -\sum_{i}^{P} \left(z_i \log(\hat{z}_i) \cdot C(z) \cdot \texttt{BOOL}(\min x_i) \right) + \lambda_s g_d \tag{4.2}$$

where $\hat{z} \leftarrow f(x)$ and z respectively represent multi-channel softmaxed-prediction and label images of P pixels, $\texttt{BOOL}(\min x_i)$ indicates the presence of a LiDAR return at pixel i, and $C(z)$ represents a user-defined mask with the same dimensions as $\hat{z}$ indicating which

classes or voxels of the output prediction to consider for a given training image. Despite its messy appearance, the logic of this objective is fairly simple. It first discounts the loss from pixels that lack an underlying LiDAR point. Then, it discounts the negative labels for some classes at some pixel locations that are considered to be misleading for the objective function. As we will see in the following, when the target function is a multi-class semantic segmentation of fairly diverse and spatially-broad object classes, this loss function with $\lambda_d = 0$ is a sufficient surrogate for wide-area detection metrics of interest w, provided that the pixel mask $C(z)$ is selected properly. Note that non-mutually-exclusive labels are necessary in this case, e.g. for detection of vehicles in parking lots atop buildings.

In our first version, we took $C(z) = 1 \ \forall \ z$, meaning that we accept and penalize incorrect predictions at any location where there is a LiDAR point. As mentioned previously, this presents an issue for OSM-labelled regions since some annotations are inevitably missing from the public ledger, yielding a significant number of false negative ground-truth labels. Since typical neural network training algorithms currently do not accommodate this kind of label-data uncertainty, we propose the use of a new term in the objective which encourages agreement between input imagery and the output predictions. In LiDAR-derived image channels (e.g. depth and height maps), we notice that the object boundaries often lie where the gradient changes; in fact, this was previously a popular heuristic used in computer vision to aid in segmentation. To this end, we tuned $\lambda_d > 0$ and defined g_d as:

$$g_d(x, \hat{z}) = ||p_1(\nabla_d(x)) - p_2(\nabla_d(\hat{z}))|| \tag{4.3}$$

where p_* can be taken to be a simple non-linear function that conforms the number of dimensions (e.g. ReLU with max-pooling), and ∇_d denotes the 2D spatial derivative of its argument. In essence, the g_d term seeks to encourage matching not-only the target-function's value, but also its agreement with the spatial-derivative of the original image

as to promote desirable network performance when some ground-truth labels are missing. Notice that with the right choice of p_*, when the predicted labels are accurate with respect to the *input* signal, g_d is zero; further, when the label information is correct we do not expect g and g_d to present competing objectives. In cases where RGB-D images are the inputs, the definition of g_d is modified to only consider the depth channel, since it is known this is not an ideal heuristic in color-space [89, 88]. We will see that while the addition of the g_d does not strictly improve performance with respect to the conformally defined groundtruth, it does improve the robustness of the network both perceptually and with respect to the Sobolev norm. Note that directly penalizing the Sobolev norm of U-net, or similar DNN structures, is extremely expensive in modern neural network APIs and is thus only used as a measurement tool.

In the second version, we took $C(z) = p_z \cdot \max z_i$, meaning that we accept and penalize incorrect predictions only at points with both a LiDAR point and at least one *positive* label. The idea here is to focus on positively labelled examples that we have much higher confidence in, although in many cases the polygons for some classes (e.g. `buildings`) are grossly overestimated. As we will see, this simple definition for $C(z)$ significantly improves the performance of the system when training over large geographic areas with labels that more-or-less densely canvas each image. The insight here is that having confident multi-class true positive labels effectively regularizes against overfitting to false negative labels. In particular, we are able to pick $\lambda_d = 0$ without significant consequence.

Performance Metrics

Due to the disparate nature of our labels, we use the following metrics to benchmark the performance of our system, defined here in brief in terms of pixel-wise (PW-) and instance-wise (IW-) true positive (TP), false positive (FP), true negative (TN), and false negative (FN) rates:

- Pixel-wise Accuracy

$$\text{PW-Acc} = \frac{TP + TN}{\#\text{ valid positive or negative voxels}} \tag{4.4}$$

- Pixel-wise Sensitivity or True Positive Rate

$$\text{PW-TPR} = \frac{TP}{\#\text{ valid positive voxels}} \tag{4.5}$$

- Pixel-wise Intersection-Over-Union (IoU)

$$\text{PW-IoU} = \frac{TP}{\#\text{ valid positive voxels} + FP} \tag{4.6}$$

- Instance Detection Sensitivity or True Positive Rate

$$\text{IW-TPR} = \frac{TP}{TP + FN} = \frac{TP}{\#\text{ positive instances}} \tag{4.7}$$

- Instance Detection Precision

$$\text{IW-Prec} = \frac{TP}{TP + FP} \tag{4.8}$$

Note that these metrics are being defined for the non-mutually-exclusive multi-class problem, so true positives and true negatives are pixels that are being predicted correctly above or below some threshold. Similarly, the instance detection rates are computed based on a threshold on the percentage of positive detections in a small detection area (empirically chosen to be 30×30 pixels). Note that our crowd-sourced labels do not provide a good measure of the IW-FN rate.

Of the 41 available flight paths in the Dublin datset, we selected 33 training and the remaining for testing. The training and testing sets were split geographically over

the extent of the dataset, such that ground-points (latitude, longitude) that are in a geographic bounding box corresponding to the optical (EO) or LiDAR field of view during a flight path appear mutually-exclusively in either the train or test set. We manually annotated 9 flight paths with `vehicle` polygons using a custom in-house software labelling tool, 7 of which were used for training, and the remaining used for testing. A majority of these polygons are bounding boxes, while a smaller number are free polygon shapes. Thus, many of the training flight paths did not have vehicle labels. This mismatch was handled by effectively picking aforementioned p_z in $C(z)$ to ignore the corresponding `vehicle` channel for those images.

Results and Discussion

The pixel-wise IOU scores for the first objective ($\lambda_d \geq 0$, $C(z) = 1$) on the various versions of the Dublin LiDAR dataset are summarized in Table 4.4. These are broken down by class and modality for the flight path data in Table 4.5. The effect of including the heuristic term g_d in the objective via $\lambda_d > 0$ is depicted in Table 4.6. A illustrative example is in Figure 4.7.

	Water	Park	Road	Building
Composite LiDAR – nadir-view (C-N)	.62	.52	.63	.61
Flight Path LiDAR – nadir-view (FP-N)	.61	.45	.57	.57
Flight Path RGBDI – camera-view (FP-C)	-	.39	.51	.49

Table 4.4: Per-class IOU ratio for various semantically-labelled OSM-annotated classes, trained using the first LIDAR objective (corresponding to $\lambda_d > 0$, $C(z) = 1$).

	D	DI	RGB-DI
Composite (nadir)	0.45	0.59	x
FlightPath (nadir)	0.41	0.53	x
FlightPath (camera-view)	0.38	0.42	0.46

Table 4.5: Comparison of mean IOU performance with different data models and input modalities. Values in the table represent average test IOU for `building`, `road`, and `vegetation` for networks trained using the first LIDAR objective (corresponding to $\lambda_d > 0$, $C(z) = 1$).

	D	DI	RGB-DI
$\lambda_d = 0$	0.41 — 1.0	0.43 — 1.0	0.45 — 1.0
$\lambda_d > 0$	0.42 — 0.09	0.42 — 0.18	0.46 — 0.11

Table 4.6: Effect of the LiDAR edge-based regularization term g_d on mean IOU performance in the FlightPath RGBDI dataset, when $C(z) = 1$. For a given performance level, the relative Sobolev norm is much lower for networks that include a contribution from g_d.

Clearly, the dense representation of the data yields the best performance for the OSM classes, and the inclusion of the edge-based heuristic in the objective function improves the "robustness" of the network with respect to the Sobolev norm. Note that the observation of performance increase, as measured using this format of the data and ground-truth labels can be a little misleading and pessimistic, since pixels with false negative groundtruths are affecting this mean IOU score. This issue was remedied in the next experiment using the second objective.

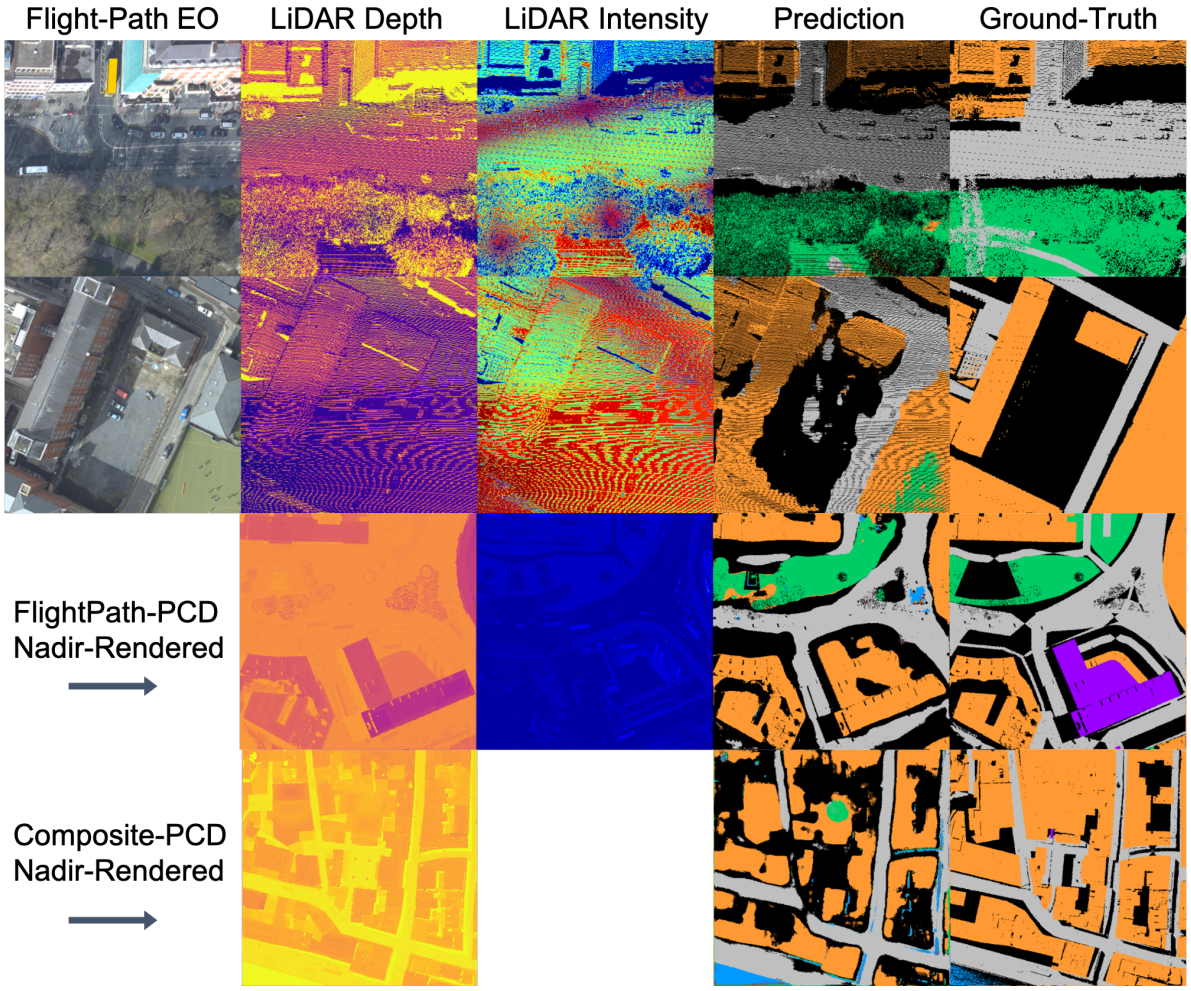


Figure 4.7: Examples of test-set detections on the various input-modalities in the OSM-labelled 2015 Dublin dataset using the first objective ($\lambda_d > 0$, $C(z) = 1$). This rendering uses the following color legend for Prediction and Ground-Truth columns: Grey–`roads`, Blue–`water`, Green–`vegetation`, Orange–`buildings`, Purple–`structures / other buildings`.

The pixel-wise and instance detection results for the second objective ($\lambda_d = 0$, $C(z) = p_z \cdot \max z_i$) are summarized in Table 4.7 and Table 4.8 below. These results are both visually and quantitatively better than those of achieved with the previous objective. Due to the strength in these numbers, we take this version of the network as our prototype system. Of course, in vacuum these metrics are not telling of the full performance of our system. In general, we visually observe excellent detection accuracy with a very low false positive rate and a strong correspondence with the underlying RGB imagery, as seen in Figures 4.8-4.9. The semantic segmentation is disassembled in Figure 4.11 in order to visualize the multi-class and pixel-wise performance.

	Road	*Vegetation*	*Building*	*Vehicle*
PW-Acc	0.796	0.975	0.845	0.932
PW-TPR	0.721	0.900	0.833	0.700
PW-IoU	0.644	0.8259	0.7034	0.674

Table 4.7: Pixel-wise performance metrics on "Test" data.

	Road	*Vegetation*	*Building*	*Vehicle*
IW-TPR	0.869	0.927	0.932	0.918
IW-Prec	-	-	-	0.980

Table 4.8: Instance-wise performance metrics on "Test" data.

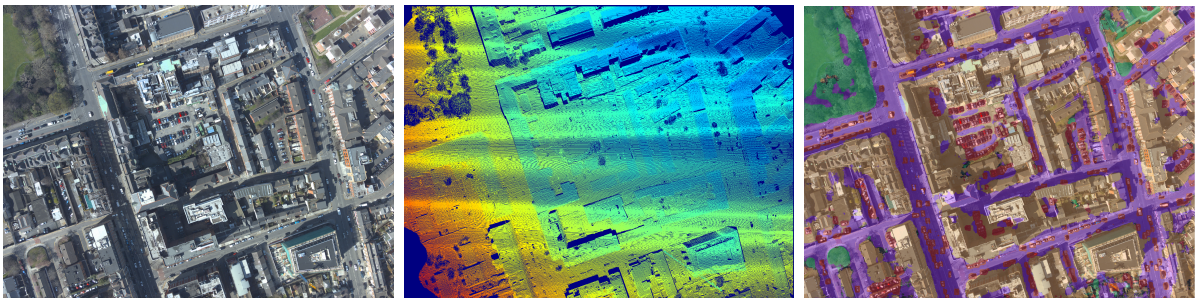


Figure 4.8: Aerial RGBD imagery (9000px × 6732px native) is rendered along a flight path via automated registration of electro-optic RGB and sparse 3D LiDAR PCD, and fed to a fully-convolutional neural network for semantic and instance segmentation of classes of interest.

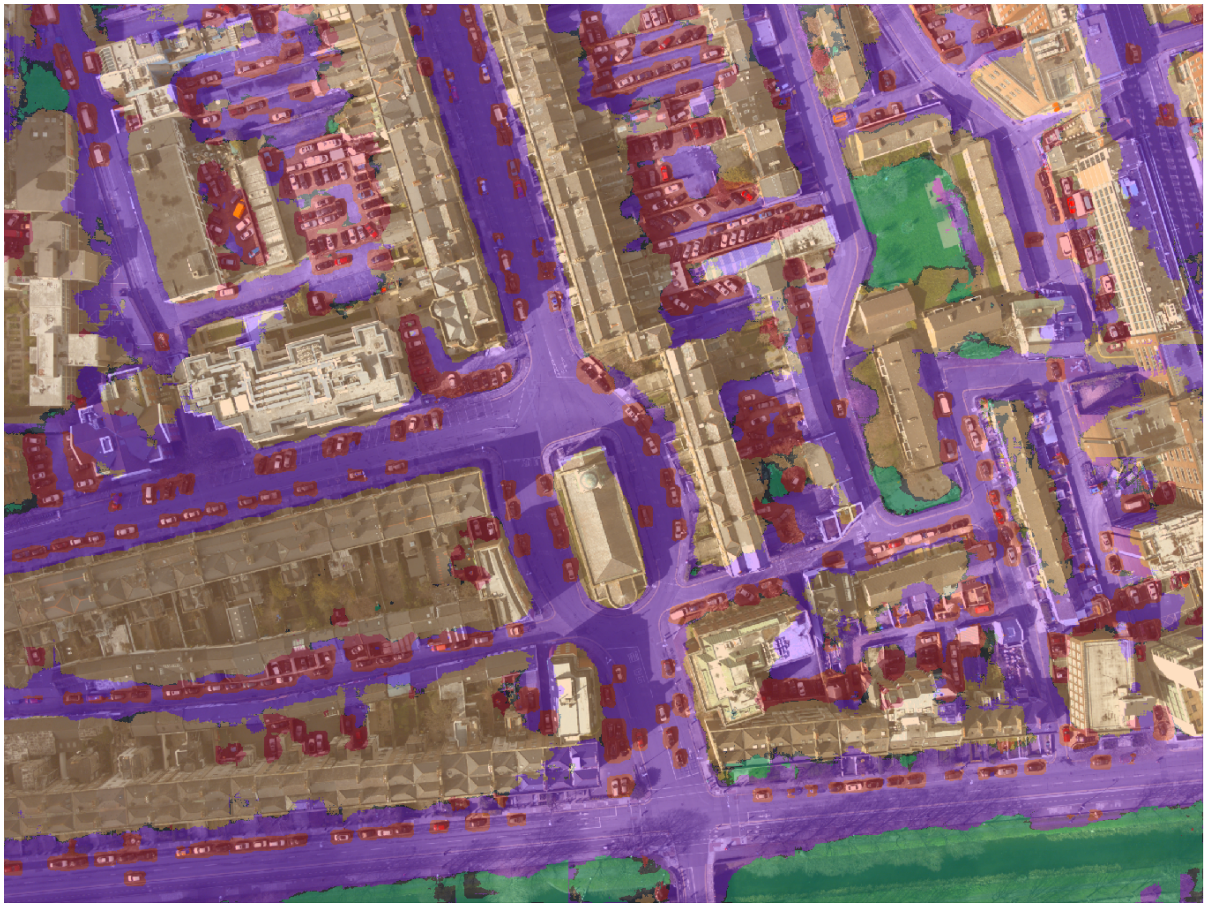


Figure 4.9: Semantic segmentation results are visualized atop native resolution RGB imagery. At each pixel, the color corresponding to the class with the highest activation is visualized. Colors: Purple–`roads`, Green–`vegetation`, Orange–`buildings`, Red–`vehicles`.

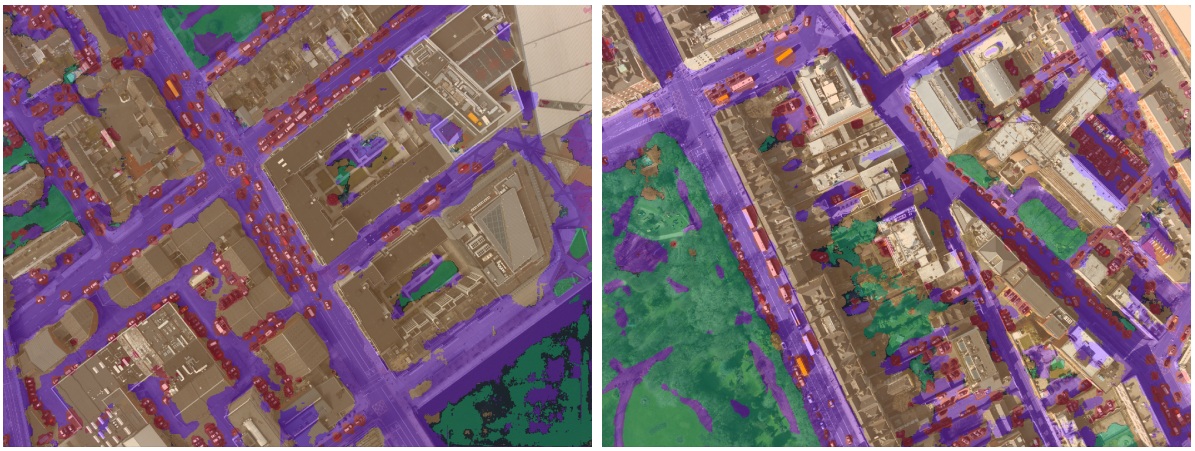


Figure 4.10: Additional wide-aperture visualization of semantic segmentation outputs.

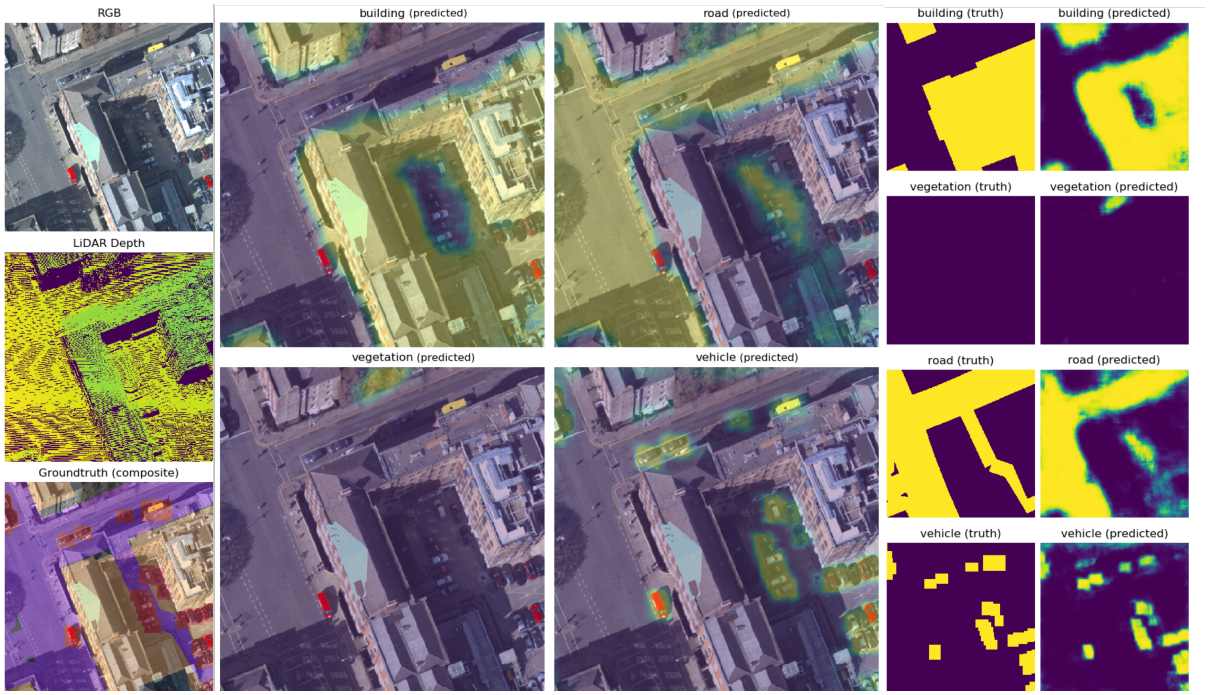


Figure 4.11: The prototype system exhibits excellent localization performance simultaneously on several classes of interest. Notice that our network correctly identifies trees as `vegetation` and ignores gaps in `buildings`, highlighting its resilience to imprecise groundtruth.

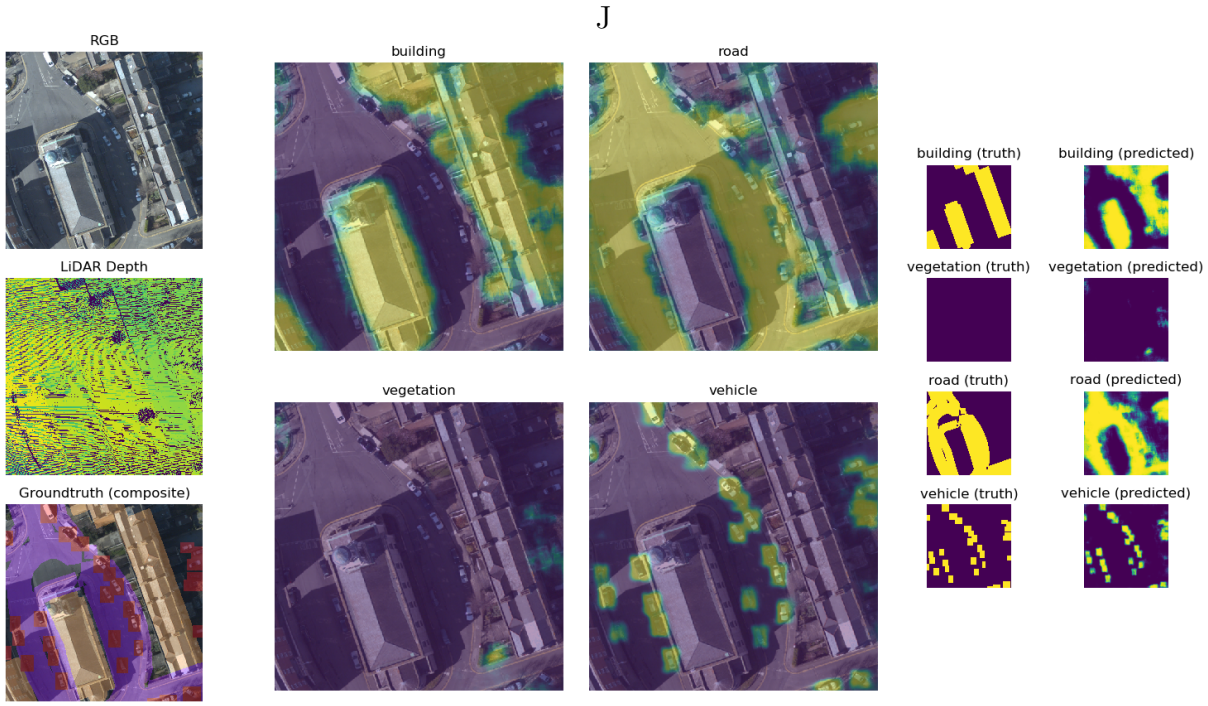


Figure 4.12: Additional example of RGBDI semantic segmentation results.

In addition to the IoU segmentation and instance precision scores, the performance of our system can be measured by how well it localizes targets in wide-area imagery at minimal latency. Our simple prototype system is able to process 1km^2 of imagery in under 3 seconds, albeit without exploiting the full 3D morphology of LiDAR PCD. The idea here is that highly-performant and relatively-inexpensive initial 2D semantic segmentation can queue subsequent fine 3D semantic and instance segmentation algorithms. However, as seen in Figure 4.13, the proposed 2.5D representation already serves as a good, fast, and cheap surrogate for a full 3D segmentation. In this vein, future work is focused on jointly refining 2D and 3D instance segmentations, disentangling the dependence on the EO and LiDAR modalities, and validating the robustness to changes in the aspect angle.

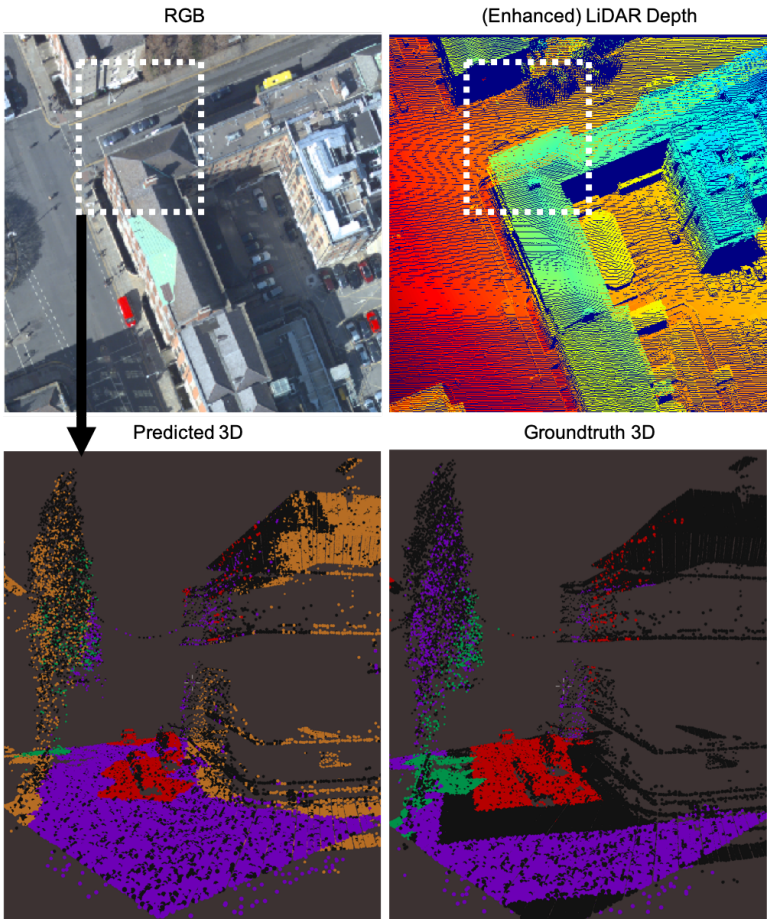


Figure 4.13: 2D semantic segmentation is projected onto LiDAR PCD, indicating a close correspondence with the 3D groundtruth. Black points indicate unlabelled (shadowed) PCD. Despite some obvious errors, the 2D-to-3D approach performs surprisingly well.

As we have seen, the ability to leverage new modalities such as wide-area aerial LiDAR are intimately dependent on the availability of high quality ground truth annotations and clever schemes to fully exploit them. Therefore, in this work we explored a technique to label wide-area imagery and carefully optimize around the quirks of the data modality and label statistics. Current and future work is focused on extending our models with enhanced 2D and 3D instance segmentation architectures by incorporating point-set neural networks.

More generally, the presented techniques suggest interesting extensions to pre-existing datasets, such as those currently used to validate for autonomous-vehicle platforms (e.g. KITTI, comma.ai), since OSM labels can be readily projected into these scenes, augmenting manually-generated segmentation labels. Moreover, with increased interest in image-fusion technologies, generating large wide-area datasets for RGBD imagery is crucial to developing robust and generalizable architectures for real-time scene understanding on these platforms [141, 98]. In this spirit, the presented technique and results on an airborne platform can be easily applied to ground-based platforms.

4.2 Electrocardiographic Imaging

Mapping of electronic cardiac potentials remains an important tool in electrophysiology, particularly for diagnosing and treating various types of cardiac arrhythmia including premature ventricular contractions (PVC), ventricular tachycardia (VT), atrial flutter, and atrial fibrilation (AF). While a majority of arrhythmia cases can be identified from multi-electrode body-surface electrocardiogram (ECG) measurements, clinicians performing interventions often rely on more local electronic mappings of the cardiac tissue (e.g. using multi-probe catheters) to classify the type of arrthymia, localize its source, and determine whether an intervention procedure such as an ablation is recommended for the patient [91, 24]. Unfortunately, physical contact between the catheter probes and the epicardial or endocardial tissue is typically a requisite to build these descriptive 3D cardiac potential maps, forcing clinicians to perform these procedures during planned surgical interventions. From a diagnostic perspective, studying the time-resolved 3D cardiac map of a patient *prior* to surgery can improve patient outcomes by helping clinicians identify and localize dominant AF or PVC sources, determine whether an ablation procedure would be an effective treatment (e.g. for long-standing AF, or when AF sources cannot be localized), and monitor patients' electrophysiology condition over time and during regular physical activity [195, 177]. In this vein, the goal of this work is to outline a new *noninvasive* technique that can aid in the construction of these cardiac potential maps.

This problem is known in the field as electrocardiographic imaging (ECGi), and has been studied extensively by a number of research groups [184, 110, 161]. The core mathematical problem of ECGi can be thought of as a special case of the more-general inverse scattering problem of electromagnetics, which involves determining characteristics of an object (in this case, the potentials of the heart), based on data of how it scatters incoming radiation.

Even in linear homogeneous media however, the problem is often ill-posed because the number of measurements is small relative to the number of unknown physical or

geometric parameters required in the model. To this end, various groups have approached this problem by (1) modeling the forward electromagnetic problem using computationally efficient linear formulations, such as the boundary-element method, (2) "inverting" the transformation to represent the inverse solution, and (3) adding a regularization constraint on the solution to both mitigate the ill-posed nature of the problem and help find a unique solution [134, 31]. Although such methods have shown promise in a number of imaging applications, including ECGi, the reconstruction accuracy has (empirically) been limited by the complexity of the forward and inverse models. Said another way, the reconstruction accuracy in these applications is typically reflective of the chosen forward and inverse models' ability to capture the intricate relationship between cardiac source-potentials and body-surface measurements, and is thus degraded when assumptions (e.g. of linearity, homogeneity, and source-free regions) are violated. In particular, in patient-specific applications, where it is common for a number of the materials and geometries to be estimated with high uncertainty or entirely unknown, simple forward models of linear homogeneous media and corresponding inverse models are often only sufficient as a first-order reconstruction heuristic [69, 180], and largely insufficient for accurately reconstructing higher-order spatial and temporal harmonics.

While these simplifying assumptions in the forward model have historically been integral to the formulation of classical imaging techniques and the field of Fourier optics, in this section we demonstrate how a direct, non-linear parameterization of the inverse problem can lead to more accurate reconstructions of cardiac potentials from torso measurements. That is, while the majority of previous ECGi studies have focused on developing the forward and inverse map from idealized material geometries, we instead cast the problem of inverse imaging as a task in function approximation, where the material parameters can be either postulated or entirely unknown. In line with techniques used in non-linear optics, the described technique relies on an approximation of the inverse map using a high-degree polynomial, but whose gradient is bounded. The parameters of this

model are found by "training" or optimization using historical data of cardiac potentials (measured using catheter probes), corresponding body-surface potentials, and their relative 3D location on the surface of real patients who underwent surgical interventions. In general, this historical data can include reconstruction parameters extracted from past measurements of the same patient, or from a database of different patients with varying torso and cardiac geometries. The idea here, is to replace the classically simple, but rigid geometrical models of the torso with more flexible parametrizations that can adapt to more realistic patient geometries involving multiple dielectric media, even when these are not explicitly known in the forward model.

In this vein, we note that the presented inversion technique can be used in two modes: (1) for initialization and fine-tuning of the inverse map from BS potentials to endocardial potentials, and (2) for non-invasive electronic imaging of cardiac tissue. Mode (1) can be used, for example, to study properties of the interstitial tissue between the torso and the endocardium (e.g. density, permetivity, etc). Whereas, mode (2) would used at a later time, when such catheters are removed from the patient, and only BS potentials are available to study cardiac activity (e.g. accurate localization of the site of origin of PVC or focal VT) and for personalized procedure planning (e.g. ablation, or other surgical interventions). While current ECGi methods provide this capability in a number of ways [213], they have been limited in their reconstruction accuracy, limiting their effectiveness in understanding and non-invasively localizing the source of AF, PVC, and VT [76]. To this end, the presented algorithms attempt to increase the accuracy of reconstructed endocardial potentials by incorporating a learning framework with a naturally-parametrized nonlinear reconstruction model. The resulting system is capable of using an array of electrocardiogram (ECG) signals (with corresponding electrode locations), target mesh locations, and a parametric reconstruction model (summarized as a polynomial network with variable coefficients), to produce an accurate 3D cardiac potential map.

From a practical viewpoint, we believe our approach can improve the analysis capabilities of cardiologists and electrophysiology specialists who are interested in studying cardiac events in live patients. Compared with linear formulations, the presented approach provides enhanced spatio-temporal resolution and reconstruction accuracy, which can help in identifying, localizing, and characterizing dominant sources of AF and PVC including higher-order temporal harmonics from high-quality localized maps of cardiac potentials, rather than simple projections of this information [87, 161, 213].

4.2.1 Experimental Dataset

The experimental data in this section was provided through the Consortium for Electrocardiographic Imaging (CEI), a group of engineers, scientists and clinicians who develop clinically and physiologically meaningful tools for simulation, modeling, statistical, and comparison studies of electrocardiographic imaging. The experiments were conducted at the Hospital General Universitario Gregor Maranon in Madrid, Spain in collaboration with the Universitat Politecnica de Valencia in Valencia, Spain. In this work, we examine electrical measurements of one male patient (aged between 40-50 years old), who was admitted for drug-refractory paroxsymal Atria-Fibrilation. The voltage measurements were comprised of both body surface (BS) potentials mappings and endocardial surface potentials. The body surface potential maps were obtained on the surface of the patients torso with a custom made electrocardiographic vest, and the endocardial potentials were measured through an electrical catheter probe in contact with the surface of the atria [161, 87].

Non-invasive reconstruction of the potential maps on the atria's surface requires mapping of the body surface potential across the subject. To measure the Body Surface Potential Maps (BSPMs), multiple electrocardiograms (ECGs) were measured at discrete positions on the patient's torso. A total of 54 ECGs were measured with Body Surface (BS) leads placed on the torso surface and electrically referenced to the Wilson Central

Terminal (WCT). To determine the locations of the BS leads, a 3D model of the patient was constructed from multiple optical images captured prior to the data collection. Additionally, the surface geometry of the patient's torso and atria were obtained from pre-scanned XRCT images with a resolution of $5mm$. These CT images were then co-registered to the optically generated 3D models to ensure the BS lead locations, torso surface, and atria surface geometries were also referenced to the same coordinate system [68, 92]. It can be noted that the proposed reconstruction method, does not explicitly require high accuracy positional information.

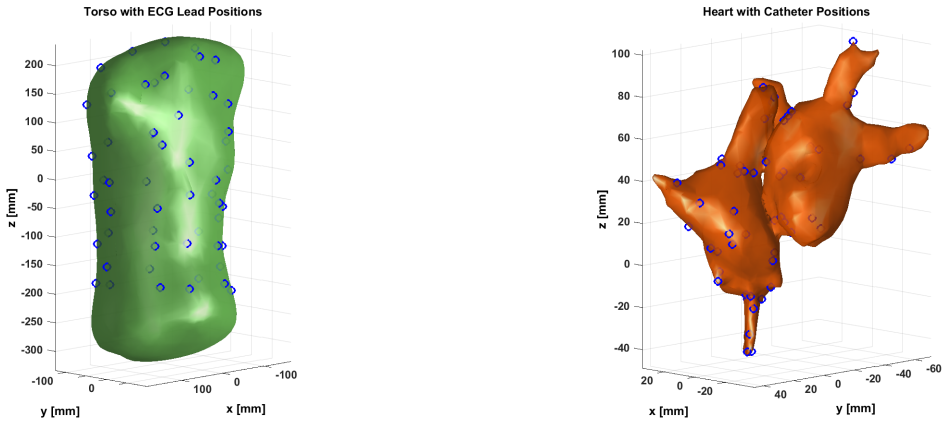


Figure 4.14: The surface mesh of the torso (a) and heart (b) are shown here, along with the corresponding locations of the electrical leads.

Similar to the BSPMs, multiple endocardial potentials were also recorded internally at different spatial positions. These electrical signals were acquired with a 64-pole basket catheter (Constellation, Boston Scientific, Natick, MA), surgically placed in contact with the surface of the atria. The position of the catheter leads were recorded through the catheter's internal body navigation system. The positional accuracy of the leads was reported to be $< 10mm$ with respect to the geometry of the atria's surface. A rendering of the torso's surface mesh and heart surface mesh along with the locations of the electrical leads can be seen in Fig. 4.14. The ECG recordings for both the endocardial leads and BS leads were acquired simultaneously at a fixed sampling rate of $2035.5Hz$. A total length of 7.4 seconds of data was recorded from all channels, which was used to help construct

our inversion model as discussed in more detail in Section 4.2.2.

4.2.2 Inverse Function Modelling

As mentioned, conventional approaches to ECGi have considered inversion techniques based on simplifications of the more general inverse scattering problem, e.g. by assuming linearity, the number and geometry of dielectrics, and the distribution of excitation sources within the endocardium. To a large extent, these approaches to the linear inverse problem can be summarized as modeling the relationship between the potentials on the atrial surface U_A and potentials on the torso U_T as:

$$U_T = M \cdot U_A \tag{4.9}$$

where M represents a linear operator that can be constructed in a number of ways from the assumed geometry, material parameters (e.g. permittivities and permeabilities), governing equations of the medium (e.g. Maxwell's equations, Coulumb's Law), and the chosen discretization and associated approximation scheme (e.g. diffeo-integral equation formulation as boundary-elements, finite-volumes, or finite-differences). While in general M represents a infinite-dimensional operator, in practice it is realized as a finite-dimensional operator M_k representing a subset of the corresponding rows of M. However, for clarity in our presentation, we will use M to refer to M_k.

Of course, to a large extent the materials and geometries of the torso can be estimated with considerable accuracy, using noninvasive techniques such as x-ray computed tomography (XRCT) and magnetic resonance imaging (MRI) techniques, as well as historical anatomical measurements from cadavers. In fact, these are often used to initialize discretized models of M used in the forward formulation, even though the condition-number of this matrix is typically large, resulting in large changes to the unknown U_A with respect to relatively small perturbations of the known U_T. This property is reflective of the

inherent ill-posed nature of the inverse problem, and is typically handled by addition of a regularization parameter that is computed as a part of the reconstruction. For example, in medical imaging the zero-order Tikhonov method, which is a type of $\mathcal{L}^2$ regularization, is used as:

$$U_A = (M^T \cdot M + \lambda I)^{-1} M^T U_T \tag{4.10}$$

where the regularization parameter λ is chosen heuristically during the image formation process, and the choice $\lambda = 0$ corresponds to the usual least-squares solution to a discrete form of Eq. 4.9.

In this work, we consider a simple modification to this formulation that allows us to optimize and reduce uncertainty in the material parameters and geometries that arise in discrete matrix M based on historical data of patients, thereby resulting in a lower overall reconstruction error. In short, we consider the replacement of the Moore-Penrose operator used in the least-squares solution, by a power-series expansion. For example, if $A = M$ is a finite-dimensional square invertible matrix, its inverse can be written down exactly as:

$$p(A) = A^n + c_{n-1}A^{n-1} + \cdots + c_1 A + (-1)^n \det(A) I_n \tag{4.11}$$

where the coefficients c_i are given by elementary symmetric polynomials of the eigenvalues of A, and these polynomials can be re-written using Newton identities in terms of the power sum symmetric polynomials of the eigenvalues $s_k = \sum_{i=1}^{n} \lambda_i^k = \mathrm{tr}(A^k)$, as:

$$\begin{aligned} A^{-1} &= \frac{(-1)^{n-1}}{\det A}(A^{n-1} + c_{n-1}A^{n-2} + \cdots + c_1 I_n), \\ &= \frac{1}{\det A}\sum_{k=0}^{n-1}(-1)^{n+k-1}\frac{A^{n-k-1}}{k!}B_k(s_1, -1!s_2, 2!s_3, \ldots, (-1)^{k-1}(k-1)!s_k) \end{aligned} \tag{4.12}$$

where the B_k represent the Bell polynomials of order-k, and n is the dimension of A.

Besides allowing for better numerical control over the inverse, using this expansion also enables us to directly optimize parametrizations of the matrix M, i.e. by finding more suitable dielectric parameters, attenuation or absorption coefficients, that minimize the maximum residual error between ground truth measurements of U_A and transformations of the input U_T, denoted as $\hat{U}_A$. In practice, we truncate the series at a smaller number $p < n$ for run-time efficiency, even though a slightly higher number is typically used during the optimization phase for optimal results.

Polynomial Neural Network

We generalize this scheme to scenarios where M is slightly non-square [106, 105], by introducing additional weighting parameters that are used as a surrogate for the determinant computation (i.e. scaling), and empirically mitigate the effects of series truncation. In particular, by noticing that the expansion in Eq. 4.12 represents an $(n-1)$-th order polynomial expression in A, we can re-interpret the reconstruction algorithm as a high-dimensional polynomial approximation algorithm, or neural network [148, 149]. That is, we can generalize Eq. 4.12 as:

$$A^{-1} \approx \sum_{k=0}^{n-1} \omega_k(A) \cdot A^{n-k-1} \tag{4.13}$$

$$\approx \sum_{k=0}^{p} \hat{\omega}_k(A) \cdot A^k \tag{4.14}$$

where we have intentionally absorbed both the geometry dependent and independent coefficients of the summation into the parameters $\hat{\omega}_k$, and truncated the series to a summation of the first p powers of A. The benefit of this formulation is that it can be easily expanded to mimic even more general polynomial approximation algorithms. For example, in our experiments we naturally expanded this formulation (Eq. 4.14) by composition, as:

$$\hat{U}_A = \sum_{k=0}^{p} \omega_k \odot (M \cdot U_T)^k \tag{4.15}$$

where $\odot$ represents the element-wise Schur-product, $\hat{U}_A$ is the reconstructed $\mathcal{R}^{m\times 1}$ potential map, we allow $\omega_k \in \mathcal{R}^m$ for $M \in \mathcal{R}^{m\times n}$, and we empirically chose $p = 3$ in our experiments. We note that when $\omega_k = 0 \ \forall \ k \neq 0$ and $\omega_0 = e_m$, the reconstruction algorithm is exactly equivalent to the linear case. From a practical point of view, we believe this initialization yields desirable improvements in the overall accuracy even when trained with only simple descent strategies, as described in the the following.

Regularized Gradient Descent with Line Search

As mentioned, in our experiments the parameters of M were found via optimization using instantaneously-corresponding pairs of body-surface (torso) potential U_T and endocardial potential U_A that were measured *in vivo*. Starting from an initial estimate of the parameters (based on simple forward/inverse models of the problem), gradient descent with line search was used to optimize our inverse model with respect to the available patient training data. That is, the "optimal" parameters of M can be found by minimizing the objective:

$$\hat{M} = \arg\min_M E = \arg\min_M ||\hat{U}_A - U_A||_2^2 \tag{4.16}$$

where $\hat{U}_A$ was computed from Eq.4.15, and U_A represents the ground truth atrial voltage signal. To mitigate the effects of limited data (e.g. small apeture size), discretization, and modeling errors during reconstruction, we augment this objective with a Sobolev-norm regularization term, as:

$$\hat{M} = \arg\min_M E = \arg\min_M ||\hat{U}_A - U_A||_2^2 + \lambda||\nabla_{U_T}\hat{U}_A||_s^s \tag{4.17}$$

which provides a bound on the roughness of the reconstruction algorithm with respect to the input data. The Sobolev-norm is known in approximation theory to be a natural surrogate for conventional $\mathcal{L}^2$ regularization such as total-variation (TV) or Tikhonov regularization, which are often sensitive to the choice of λ [203, 182]. Instead, weighting the objective function by the Sobolev-norm of the differentiable inverse map allows practitioners to analytically tune the sensitivity of the reconstruction algorithm to the *input data*, rather than using perceptual image-quality metrics on the output reconstruction (e.g. graphical L-curve method) [39, 208]. It is worth noting that in this formulation regularization is applied only during the "training phase" and *not* during reconstruction.

In our experiments, different values of M were found and evaluated via the update strategy:

$$M_{t+1} = M_t + \gamma_t \nabla_M E \tag{4.18}$$

where $\nabla_M E$ represents the normalized direction of the gradient of the objective function with respect to the parameters of M, and $\gamma_t \in [-1, 1]$ is the step-size that was selected to minimize the objective at training iteration t. In practice, γ_t is selected from a discrete set of r trials, by applying updates along the gradient direction (with different magnitudes), evaluating the objective at these various points as $E_{t,r}(M)$, and selecting the update that yields the lowest error.

Measurement of Errors

In the described experiments, we used the raw BS and endocardial measurements available in the described dataset, with little to no pre- or post-processing. The reported absolute and relative errors were computed directly from the output of the polynomial network at each discrete time-point k and averaged as follows:

$$e[k] = |U[k] - \hat{U}[k]| \tag{4.19}$$

$$\text{Mean Absolute Error [mV]} = \frac{1}{T}\sum_{k}^{T} e[k] \tag{4.20}$$

$$r[k] = \frac{|U[k] - \hat{U}[k]|^2}{U[k]} \quad \forall\, U[k] \neq 0 \tag{4.21}$$

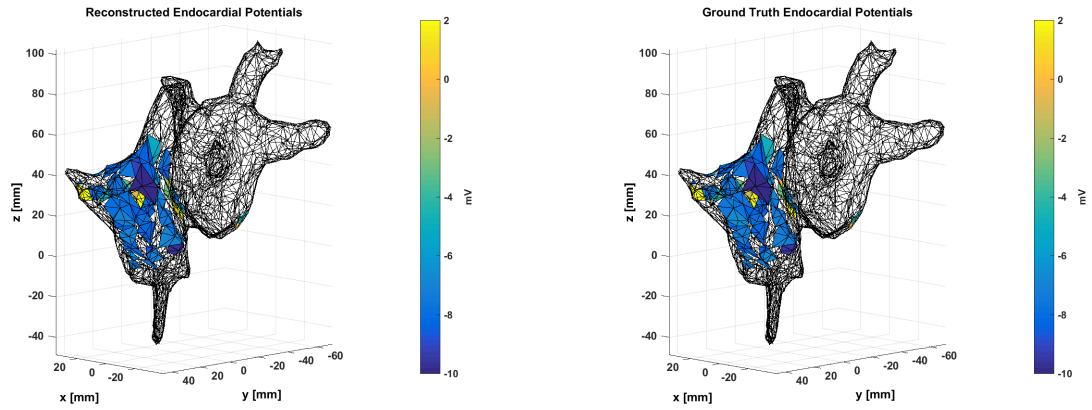
$$\text{Mean Relative Error [\%]} = \frac{1}{T'}\sum_{k}^{T'} r[k] \quad \forall\, k \in T' \; s.t.\; U[k] \neq 0 \tag{4.22}$$

In particular, we note some measurements (a total of 6 time-points over the entire dataset) were excluded from the relative error computation (i.e. when the recorded endocardial potential value was effectively 0) for interpretability. Measuring both the absolute and maximum error ameliorates the missing data in these cases.

4.2.3 Results and Discussion

To evaluate the proposed approach, a set of medical data was analyzed to determine the effectiveness of the presented reconstruction technique with respect to other contemporary methods, such as formulations based on the Boundary Element Method (BEM). The accuracy of our inverse solution is strongly dependent on the precision of historical measurements that are used to optimize our inverse operator. Due to this dependency, we identified a curated open-source data archive of high-quality *in vivo* measurements collected for physiological and medical study of the cardiovascular system. For this experimentation, the electrical measurements were acquired with precision medical instrumentation to ensure high accuracy, and other medical imaging data was also collected to supplement the ECG data [161]. In this section, we highlight our results through visualizations of imaging and reconstruction results, as well as the accuracy of the reconstructions.

The inverse model was optimized on the set of body-surface potential measurements (BSPM) and invasive endocardial measurements that were used to reconstruct the endocardial potential map, as described in Section 4.2.1. Given that both these measurements were recorded simultaneously for the given data set, sufficient information was available to train a model to estimate this inverse function. To achieve this with the limited data source, the body surface and endocardial measurements were separated into two disjoint sets. From the original data, time samples were randomly chosen to serve as the training data. After the training data was used to optimize the inverse network, the remaining data was used to test the reconstruction accuracy. Because the time samples were chosen at random sample instances, the learned inverse model is implicitly time invariant.



(a) (b)

Figure 4.15: Reconstruction of the endocardial potentials (a) and measured ground truth endocardial potentials (b) on the heart.

With the catheter endocardial measurements available, we were able to compare our reconstruction estimate to a "ground truth" measurement. Furthermore, with the geometry of the atria obtained from XRCT imagery, the reconstructed endocardial potentials were mapped to spatial positions on the endocardial surface for an accurate visualization of the voltage spatial profile as in Fig. 4.15. The reconstructed potential is compared to the ground truth measured potentials again, and close correspondence

between the two mappings is clear. The spatial-potential information here is a useful tool to determine and isolate concentrations of electrical activity in the atria, to further aid in noninvasive study of the electrophysiology of the cardiovascular system.

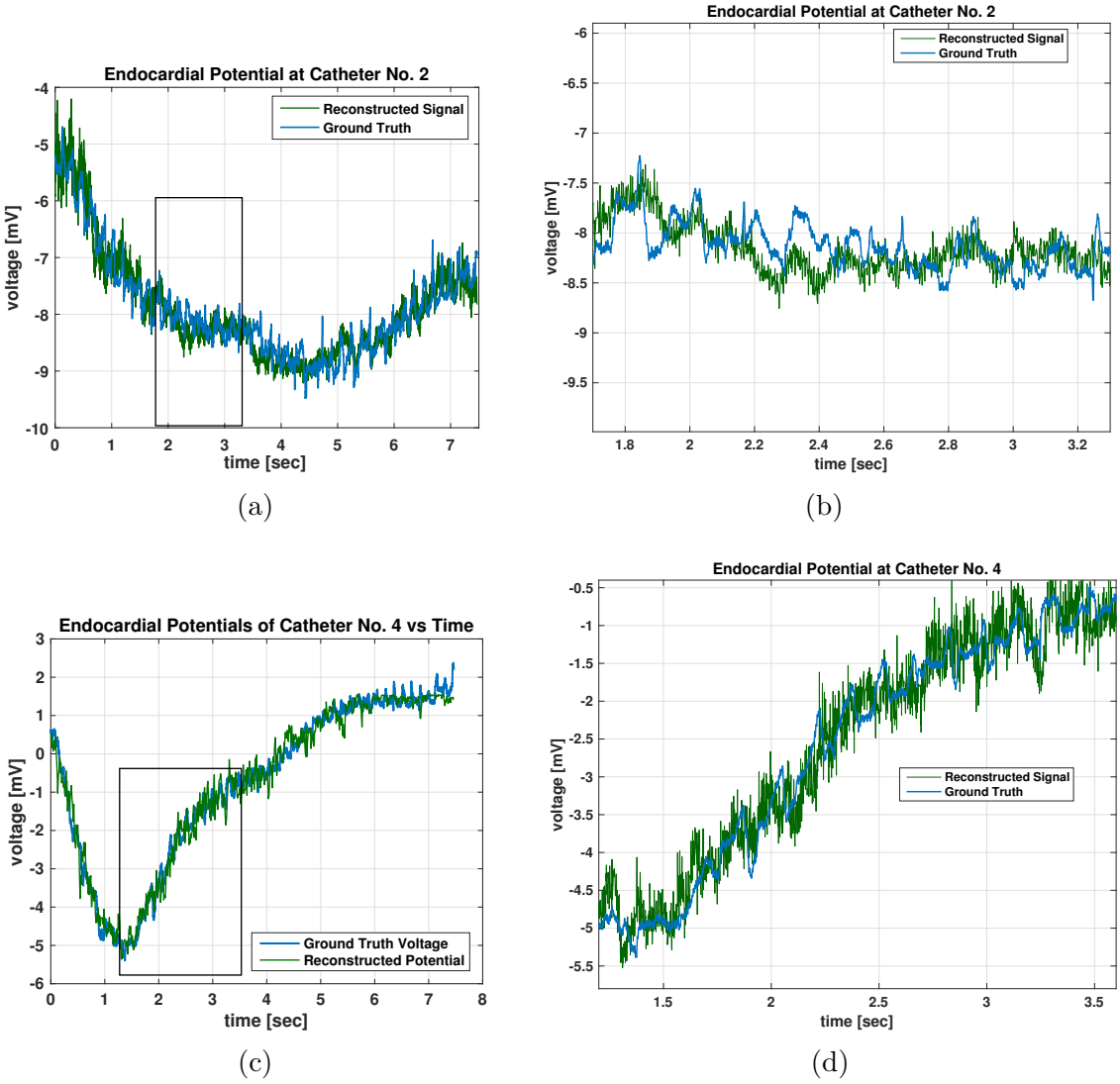


Figure 4.16: Shown here are small-window time-domain reconstructions of endocardial potentials from body-surface potentials via the presented nonlinear imaging technique.

Two important factors under consideration when interpreting inverse mappings of BSPM to endocardial potentials are preservation of temporal features like harmonic content and waveform shape, and also the smoothness of the spatial profile of the voltage pattern of the heart's surface. A temporal plot of the endocardial reconstructed voltage signal at multiple points on the heart's surface can seen in Fig. 4.16, which can be used to

develop intuition into how the inversion model reconstructs the signal. The reconstructed signal accurately tracks slowly changing features in ground truth signals over the 7-second measurement interval. The low-frequency content indicates accurate preservation of lower order harmonics by the inversion model, which is an important feature for spatio-temporal understanding of the physiology of AF. Furthermore, the relative magnitude and scale is also preserved in the reconstruction without the need for estimated scaling parameters, so as to provide interpretable results for users not familiar with the specific inversion method. The reconstructed signals in Fig. 4.17 are also representative of this qualitative analysis.

Accuracy of Reconstructed Endocardial Voltage Potentials

The performance of the reconstruction model was evaluated using several statistics. The mean absolute error, mean relative error, and their standard deviation of both are displayed in Table 4.9. The average absolute error over the entire test data set was found to be 0.327mV, which yielded a 12.47% average relative error with respect to the recorded data. Additionally, the training results are of similar accuracy to the reconstruction results, indicating that the model is robust to new test data and has not been over-fitted to the training data, which is an important quality for any inversion approach.

These results indicate that a high degree of confidence may be placed in the reconstruction of the endocardial potential distribution using the proposed methodology. As a comparison in Table 4.9, we list the accuracy of the inverse computed BEM results reported in [161]. Here, it should be noted that relative accuracy was reported for the normalized voltage signals of the endocardium due to numerical scaling issues when computing BEM inverse solutions. This metric describes how well the inversion procedure preserves the overall shape of the endocardial waveform, if not the true scale. Here, our proposed reconstruction method noticeably outperforms the BEM solution on average over the entire data set.

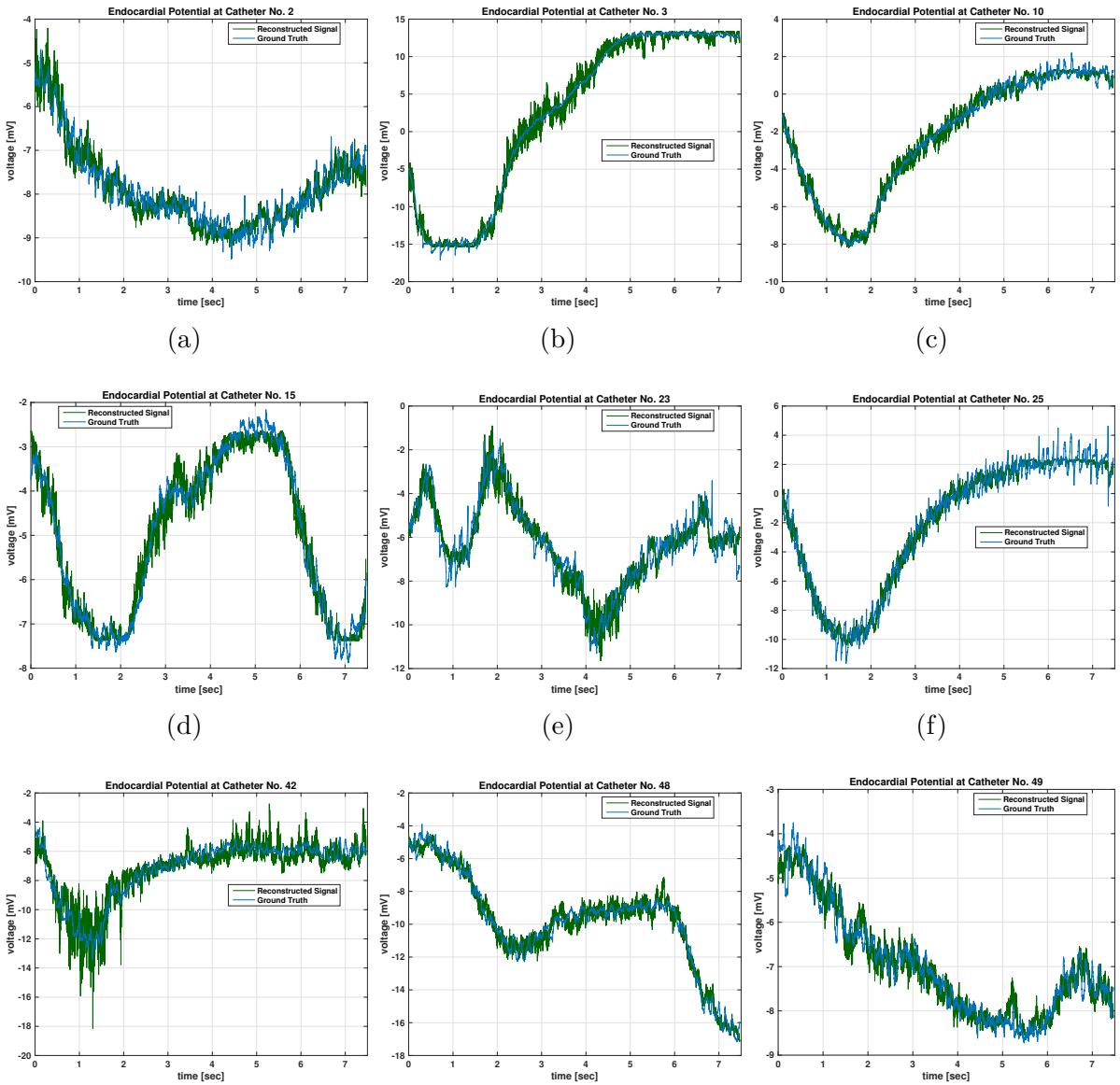
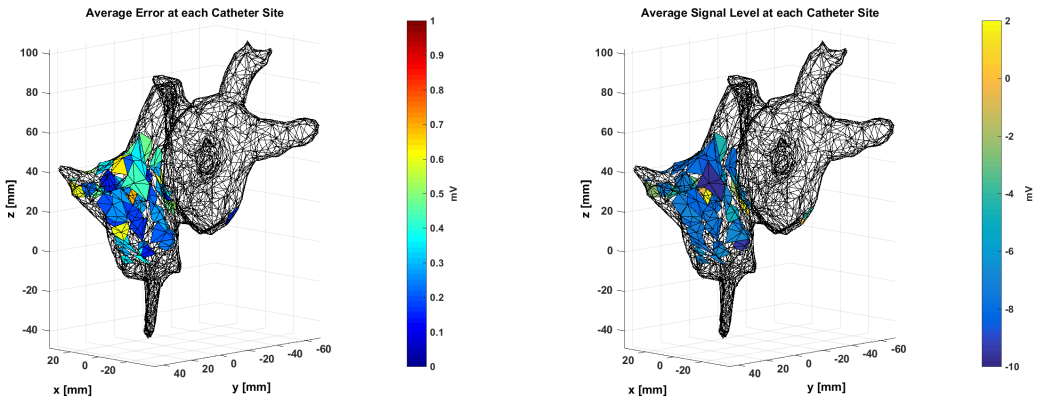


Figure 4.17: The presented nonlinear imaging technique produces high-quality time-domain reconstructions of endocardial potentials at various catheter positions.

	Absolute Error [mV]	**Relative Error [%]**
Training Results	0.310 ± 0.321 mV	12.07%
Reconstruction Results	0.327 ± 0.221 mV	12.47 %
Normalized Reconstruction Results	0.327 ± 0.146 mV	12.43 %
Normalized BEM Results [161]	–	35.8%

Table 4.9: Reconstruction Performance

The average reconstruction error for each sensor was also averaged across the entire time series, as can be seen in Fig. 4.18. Here the average error does not exceed 0.7mV over the various sensor positions, indicating that the reconstruction model does not suffer from any spatial bias. Therefore at any point in time, given body surface potential measurements, this model could be used to accurately reconstruct endocardial voltage potentials of the heart to within a prescribed tolerance.



(a) (b)

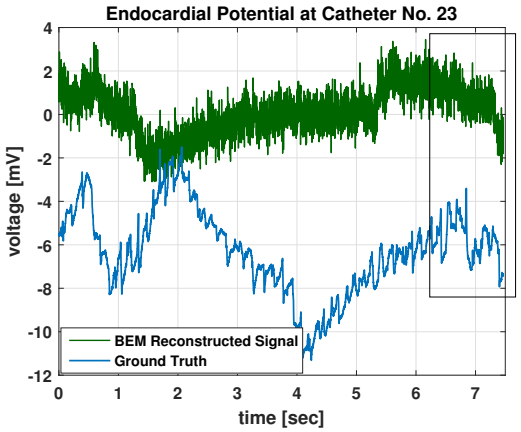
Figure 4.18: The average reconstruction error (a) is small over the spatial area where ground truth data was collected, with a maximum absolute error less than 0.70 mV. This figure is compared in the context of the average signal level (b), which varies by as much as 12 mV over the different catheter sites.

Discussion

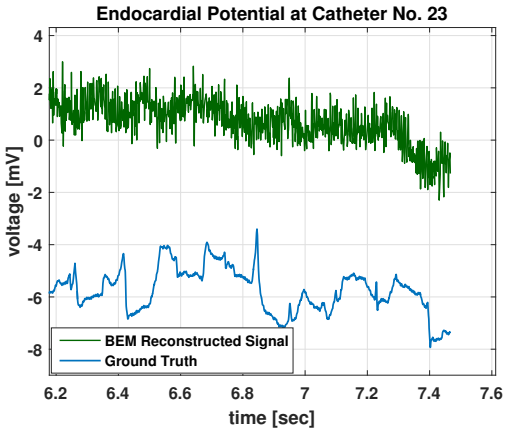
The results summarized here demonstrate the potential of a flexible inverse model that can be optimized with real, high-quality measured data. The optimization can help mitigate the effects of unknown physical quantities and parameters typically associated with inverse problems. The accuracy of the reconstructed cardiac potentials in the spatial-temporal domain is indicative of the validity of our optimized inverse model in comparison to other techniques. This is an important finding given that our inversion method differs from direct inversion of the forward model problem, and suggests that improvements can be made on the simplified physical models that are typically used.

Boundary-element method (BEM) solutions to the inverse problem have shown some

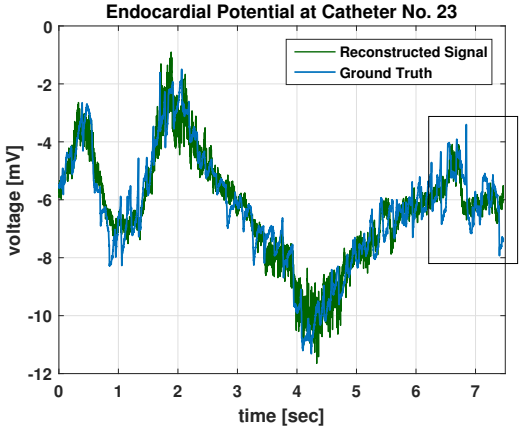
success in the reconstruction of relatively smooth cardiac potential maps. However, if the potential map is highly varying over time, the regularization introduced in the inversion will tend to smooth out complex features that are exhibited during AF events and thus decrease accuracy. Given that our inversion model was optimized over BPSMs and endocardial data acquired from a patient being treated for AF, these complex patterns in the potential maps are properly reproduced by our model (Fig. 4.19c).



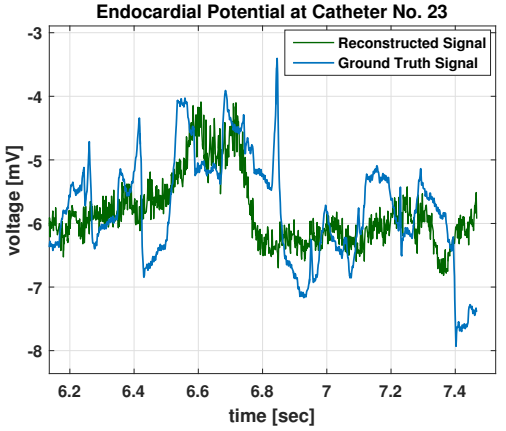
(a) Catheter No.8 Time Reconstruction



(b) Catheter No.8 Time Reconstruction Zoomed



(c) Catheter No.8 Time Reconstruction



(d) Catheter No.8 Time Reconstruction Zoomed

Figure 4.19: Time-domain reconstructions using a standard BEM method can produce significant error in the voltage signal (a-b). The presented polynomial reconstruction technique can both recover a significant portion of the missing large-signal information (c), but still exhibits some temporal inaccuracies in the small-signal reconstruction due to the inherent time-invariant nature of the model (d).

While our inverse model exhibited the high accuracy in voltage potentials in the time domain, there were some observed limitations in the spectral accuracy of the reconstruction.

Specifically low amplitude, high frequency transients were difficult to estimate with the model. This type of feature was observed in the original data, as can be seen in Fig. 4.19d along with our reconstruction result. This limitation is due to the small error contributed to overall objective function measured over the entire data set. The optimization will instead reconstruct a mean constant value during these events that still yields a low-average error for the entire duration of the signal. In this respect, proper fidelity of high frequency content of the endocardial signals continues to be a limitation faced by other inverse solutions [166].

The acceptance of a particular solution to inverse solutions in ECGi is still contested in the field [69]. The debate focuses on identifying and defining the optimal features in cardiac potential mappings that best serve clinical and diagnostic goals. In general spatio-temporal accuracy is almost universally regarded as critical for successful inversion of BPSMs, as invasive techniques used in ablation procedures mainly rely on surgeons manually probing the endocardium and/or epicardium in real-time to identify sources of AF. Although our approach optimized error in the temporal potential maps, other waveform constraints could easily be incorporated in our objective functions to help detect features of interest. This could be achieved, for example, by including a generalized Tikhonov regularization in addition to the Sobolev-norm regularization that was used, e.g. via a non-uniformly weighted-norm, or band-pass filter.

Finally we note that while the presented methods have demonstrated quantitative improvements in the time-domain-independent reconstruction of cardiac potential maps from BS potentials, the ultimate utility of this application depends on the clinical and diagnostic needs of electrophysiologists who would rely on such information to refine patient prognosis. In particular, the presented approach to non-invasive cardiac mapping provides a promising outlook for the use of non-invasive ECGi to monitor and study cardiac disease, because it offers a flexible learning-based framework that leverages past measurements to refine features of cardiac reconstructions that hold diagnostic value.

4.3 X-ray Computed Tomography

As deep neural networks (DNNs) have gained popularity, so has their use in medical imaging applications. In x-ray computed tomography (XRCT) and magnetic resonance imaging (MRI), for example, researchers have investigated the use of DNNs for performing both classification and reconstruction [107, 230]. In these and numerous other works, researchers have used standard DNN-prototype elements, such as convolutional, fully-connected, and pooling layers (originally developed in the signal processing and machine-learning community) to emulate various portions of the reconstruction process [230]. In Chen et al., for example, both a deep CNN and autoencoding neural network (a topology that feeds data through a bottleneck layer) were used to directly improve reconstruction performance in a patch-by-patch fashion from a limited number of projection angles [41, 42]. In Jin et al., a direct inversion formula extracted from a spectral iterative algorithm is combined with subsequent filtering by an autoencoding U-net architecture to improve artifacts introduced by initial higher-order method [104]. In Kang et al., a CNN is used to weight the wavelet coeffcients more optimally, resulting in noise suppression while maintaining understandability in the final layers [107]. In Zhu et al., this approach is taken a step further by using a convolutional neural network (a type of DNN topology) to emulate the fast-Fourier transform (FFT) or back-projection operation that is common to many k-space reconstruction algorithms based on Fourier theory [230].

Common to these algorithms is construction by first replacing (at a high level) components of the image-reconstruction algorithm with "equivalent" neural network models, which are either previously trained by an auxiliary objective (e.g. autoencoder architectures that seek to reconstruct an image its non-linear projections) or via mapping to a conventional classification task (e.g. location of a pathology). While classification results for these heuristically-designed black-box networks show promise as an emerging analysis and diagnostic tool, from the perspective of imaging they leave many unanswered technical questions (e.g. stability, resolution, accuracy) that brings understandable questions of

their reliability as a clinical tool, free from training-bias [8].

Further, while DNNs and similar non-linear regression techniques may be *capable* of harnessing powerful approximation properties that yield algorithms with superior resolution and accuracy, an approach to quantitatively measure or validate these properties on a trained network has thus far been elusive, beyond the standard technique of evaluating an asymptotically-intractable number of test cases. In fact, current research shows that conventional DNNs are easily fooled [154], suggesting that measuring classification accuracy on a withheld test-set is in many cases insufficient for demonstrating robustness, since even simple problems are severely under-sampled with respect to the dimensionality of their defining input and output maps. This is especially true in imaging applications such as computed tomography (CT), where the number of input/output pixels is commonly on order of $\geq 10^5$ (e.g. 30 projection angles with a camera of 64 x 64 pixels, or reconstruction volume of $\geq 64^3$ voxels).

This issue is not limited to the application of DNNs. Within the imaging literature itself, many models have been proposed and iterated on through the years, incorporating e.g. corrections for noise [126], motion of the patient [61], and sparsity constraints [219, 48, 223]. However, unlike typical DNN approaches, these models are usually designed to intrinsically satisfy a large (often infinite) number of mathematical constraints that correspond to our physical intuitions of the underlying physics problem, with the important caveat that inputs and parameters must belong to an assumed class of functions. Of course, when these assumptions are violated, as is common in real-world imaging scenarios, we observe degradation in the reconstruction that is *in addition to* (and compounds) other issues such as statistical emmisivity, noise, detector models, etc. that are typically handled separately.

As a practical note, this highlights the benefit of using a data-driven machine-learning approach, which may overcome these limitations when equipped with a sufficient amount of exemplary training data. But, as mentioned, the downside of using a black-box data-

driven approach such as DNNs is multi-faceted and compounding. Namely: (1) DNNs suffer from a lack of understandability that impedes the algorithm designer's ability to propose sensible modifications of the algorithm e.g. to incorporate priors and improve baseline accuracy[1], and (2) typical DNNs provide minimal theoretical guarantees or summaries of their assumptions, instead requiring a number of benchmark tests to assess an imaging algorithm's utility for a particular application. As such, there remains several complications with these methods such as the basin of convergence or quantification of achievable resolution. Moreover, this is a compelling reason why classical methods continue to be used despite their provably sub-optimal performance. In this vein, it seems useful to ask whether the understandability and predictability of classical approaches to imaging can be combined with modern data-driven machine learning techniques.

To this end, the purpose of this work is to outline a new scheme for designing deep learning algorithms for computed tomography and medical imaging applications, which leverages the abundance of literature that exists on the subject to develop more robust and accurate reconstruction algorithms without sacrificing interpretability. As an initial example, we pick a simple iterative algorithm for XRCT and SPECT, and show how these can be utilized to construct a deep imaging algorithm *initialized* at a well-understood baseline of performance. In particular, we show the performance is initially equivalent to the original algorithm, but with a natural representation that can be further expanded and trained as a neural network. The construction of the network should make it clear how this method can be applied to a number of different physical models of propagation or material interactions, such as attenuation, scattering and noise. We provide with some initial experiments to demonstrate the learning and predictive capability of such networks, and conclude by reminding readers that applying such techniques in a clinical setting will require a more careful treatment of algorithm optimization than is afforded by considering

[1]We conjecture that a large source of the confusion stems from the ad-hoc use of non-linearalities as information bottlenecks in the algorithm. This seems to prevent sensible weight initializations in multi-layer DNNs.

a single reconstruction metric on a small training set.

4.3.1 Iterative-Reconstruction Algorithms for CT

There are a number of different iterative-reconstruction formulations for computed tomography of x-ray data that are used in practice. These methods fall broadly into one of two categories: Fourier-based methods or algebraic methods. Spectral (Fourier) methods are based on explicit, deterministic inversion formulas for reconstruction a function from its line integrals (Radon transform), or small generalizations thereof. In contrast, algebraic techniques are based on minimizing or maximizing an objective, formulated by incorporating the expected stochastic variation in photon counts and other factors. Of course, the two categories are intimately related to the physics model of x-ray wave propagation in media, but recently there has been a push towards using model-based algebraic techniques for their ability to easily incorporate new models of emission and transmission as priors. We will focus on iterative techniques, since they are the most common algebraic techniques used in practice.

The basic idea of these iterative algebraic reconstruction is to (1) guess a distribution of parameters, (2) measure whether this guess makes sense with respect to the recorded data, and (3) use this measure to update the estimate of the parameters. To elucidate, in medical applications of transmission and emission tomography, the task is typically to estimate a 3D distribution of attenuation coefficients (an attenuation map) and concentration of radioactive molecules (activity-centers), respectively. To estimate these parameters, x-rays are collected at various projection angles by rotating a detector with respect to a patient or a sample. So, a fundamental question is how to use the measured x-ray projections to update an estimate of the distribution of parameters.

The central technique in many model-based algorithms revolves around evaluating (2) by simulating projections of the current estimate. This can be done in a number of ways, e.g. incorporating a full-wave model of wave propagation, attenuation, noise,

multiple bounces, scattering, etc. In the simplest formulation, we could consider a simple ray-tracing model where the recorded count at detector i (positioned at angle θ) depends on the path of photons from each voxel, summarized as:

$$\mathrm{p}_i = \sum_{j \in I_i} c_{ij} \lambda_j \tag{4.23}$$

where c_{ij} represents the effective transmission coefficient from voxel j to detector i, I_i represents the set of voxels contributing to detector i, and λ_j represents the initial photon magnitude in voxel j in the straight-line direction to detector i. Notice that in this expression c_{ij} can summarize almost all information about the geometry and propagation of x-rays through media, including attenuation, recombination, and scattering.

Assuming a formula (e.g. Eq. 4.23) for simulating the projections at various θ, the first step is measuring how well this matches up with the recorded projection (photon count). The most basic comparisons are suprisingly simple, e.g. a measure of pixel-wise mean-square error:

$$r_{\mathrm{MSE}} = ||p_i - Y_i||_2^2 \tag{4.24}$$

or, the pixel-wise ratio:

$$r_{\mathrm{ratio}} = \frac{Y_i}{p_i} \tag{4.25}$$

where Y_i is the measured projection at detector i, and r_* represents a residual or type of error signal. In any algorithm, we would like r_{MSE} to tend towards zero, whereas we would like r_{ratio} to approach one.

Using this residual as feedback, the next step is to update the estimate of the unknown parameters. While there are many possible update rules, in general the update strategy that is chosen should be compatible with the choice of aforementioned feedback. For

example, the Newton-Raphson update in transmission tomography can be defined as:

$$\tau^{k+1} = \tau^k + \alpha \cdot \frac{F(\tau^k, Y)}{F'(\tau^k, Y)} \tag{4.26}$$

where $F(\tau_k, Y)$ denotes the computation of a residual such as r_{MSE} with respect to the current estimate τ^k, and $F'(\tau^k)$ represents its first-derivative with respect to the parameters τ^k. That is, we are seeking the root of the equation $F(\tau^k, Y) = 0$ for variable τ^k (with the complete set of measured projections Y is a constant).

The Newton-Raphson method is simple, but can be computationally expensive when the underlying physics model is complicated, since it is necessary to compute $F'(\cdot)$. Still, it is a strong method and is popular for reconstruction of attenuation coefficients in transmission-mode x-ray CT (XRCT).

We will now briefly work-through a simple classical derivation of the Maximum-Likelihood Expectation-Maximization (MLEM) algorithm for emission-mode CT [118], since this has been shown to have superior convergance properties compared to the Newton-Raphson method in many applications and for SPECT.

Maximum-Likelihood Expectation-Maximization (MLEM) Algorithm for Emission CT

Suppose the observed data is a only a sample y of a random-vector Y, that is described by a density function $g(Y, \Lambda)$, where Λ is the vector of parameters to be estimated (e.g. the intensity of source voxels in the sample). $g(Y, \Lambda)$ may be hard to know empirically, or model explicitly, because it is in general a function of the sample geometry, detector geometry, and the various source-elements we are looking for. This makes it hard to maximize $g(Y, \Lambda)$ w.r.t Λ. For this reason, many EM algorithms consider the "embedding" of Y into a larger sample-space, such that $Y = h(X)$, where X is supposed to have a density function $f(X, \Lambda)$ (of some assumed form) with respect to some measure $\mu(X)$. In

the discrete version of this problem, we can formulate the relation:

$$g(Y,\lambda) = \sum_{\mu} f(X,\lambda) \quad s.t. \quad \{X : h(X) = Y\} \tag{4.27}$$

Now the expectation-maximation (EM) iteration is simple. In the E-step of the n-th iteration, we form the conditional expectation:

$$E\big[ln\big(f(X,\lambda)\big)|Y,\Theta^n\big] \tag{4.28}$$

where ln denotes the natural logarithm, and Λ^n denotes the current vector of parameter estimates. Then, in the M-step of the n-th iteration, we fix Λ^n, and maximize the expectation with respect to Λ.

While such an algorithm generally has some nice convergence properties [178], there is uncertainty both in the specification of $h(\cdot)$ and the choice of $f(\cdot,\Lambda)$.In particular, there can be many ways of embedding Y into a larger sample-space, including physics-inspired techniques (e.g. particle considerations, ray-tracing, count-statistics). Thus, from a practical viewpoint, the art of using this simple EM algorithm lies in choosing an appropriate specification for X [118]. We will exploit this freedom later in our deep-learning formulation.

To use this EM formulation for emission computed tomography we must, essentially, define the expectation given in Eq. 4.28 explicitly in terms of Y, the samples we collect in the detector. This can be achieved using physical considerations; for example:

The mean number of detected photons originating from pixel j during the i-th projection could be computed as:

$$\Delta t_i \lambda_j b_{ij} = c_{ij}\lambda_j \tag{4.29}$$

where i is the projection subscript, j is the pixel subscript, Δt_i length of time over which the i-th projection is collected, b_{ij} is the probability that a photon leaving pixel j reaches

the i-th detector, and λ_j is the unknown source intensity we are looking for. Note that in the this formulation, b_{ij} and Δt_i summarize the physical features of the detector geometry and assumed attenuation constants; consequently, c_{ij} is assumed to be known and fixed with respect to an assumed (linear) physics model.

Further, using the ray-tracing model that is conventional applicable when considering the particulate-nature of radiation, we could define the total number of recorded photons for projection i to be:

$$Y_i = \sum_{j \in I_i} X_{ij} \tag{4.30}$$

where it is clear from context that Y_i is the vector representing counts at all the detector locations, I_i is the subset of source-locations contributing to the detector counts (it could be all of them), the mean of X_{ij} is $c_{ij}\lambda_j$, and that Y_i is Poisson-distributed if X_{ij} is assumed to be Poisson-distributed.

Using these assumptions, we can write the EM iterative-reconstruction algorithm explicitly as a method of moments estimate, as:

$$\lambda_j^{n+1} = \frac{\lambda_j^n}{\sum_{i \in J_j} c_{ij}} \cdot \sum_{i \in J_j} \frac{c_{ij} Y_i}{\sum_{k \in I_i} c_{ij} \lambda_k^n} \tag{4.31}$$

where J_j is the set of projects to which pixel j contributes, and λ_j^{n+1} is the new estimate of the source-intensity at pixel j at the after the n-th iteration. In this form, it is obvious that the update-rule is a multiplicative update, that incorporates the ratio of recorded data Y_i to a simulated-projection $\sum_{k \in I_i} c_{ij} \lambda_k^n$, that is re-weighted and applied to the current estimate of sources. This is a well-known derivation, for perhaps one of the simplest EM formulations for emission CT. We will now show even this simple algorithm is capable of deep inference.

4.3.2 Deep Iterative-Reconstruction Algorithms for CT

We begin by noticing that the both of the aforementioned iterative-reconstruction algorithms can be re-written as neural networks by simply writing out their data-flow graphs [169]. For example, we notice that the update rule (Eq. 4.31) can be thought of as a simple network that takes the initial estimate of the source-intensity λ^n as input, and produces the next estimate λ^{n+1} as output. In the simplest case the parameters, or weights of this network, are the assumed constants c_{ij} that incorporate the model and detector geometry and the assumed physical attributes, such as the attenuation or probability of detection. Although these are considered known constants in the vanilla-version of the EM algorithm, it is easy to see that they are based on simplistic assumptions of operative physics that are known to intimately affect the quality of the reconstruction (i.e. the art). In short, by considering the optimization of such parameters with respect to the input-output pairs we provide (e.g. phantom experiments, known cases where analytic solutions are possible, etc.), we can develop more-accurate imaging algorithms that are informed by real data and experiments, and flexible to modification in specific cases. This interpretation has a strong analogue to approaches that incorporate deformations of the original model [130, 6].

While Eq. 4.26 and Eq. 4.31 by themselves resemble shallow 1-layer networks with simple additive normalization[2], we further notice that by considering more than one iteration we can easily generate arbitrarily deep networks. That is, we can understand the output of the original Newton-Raphson or EM algorithm after r iterations to be the result of r nested function compositions, which resembles a DNN f with pooling [169]:

$$\lambda^r = f(\lambda^0, C) \tag{4.32}$$

$$= p_{r-1}\Big(p_{r-2}\big(p_{r-3}(...p_0(\lambda_j^0, C_0), C_{r-2})C_{r-1}\big), C_r\Big) \tag{4.33}$$

[2]e.g. to respect strict equality constraints that may arise, such as $\sum_i \sum_{j\in I_i} c_{ij}\lambda_j^n = \sum_i Y_i$

where λ^0 denotes the initial guess of the target distribution in vector form, λ^r denotes the reconstruction estimate after r iterations (loops), p_k represents the update operator at step k, and C^k denotes the system matrix at step k with entries $c^k_{i,j}$ representing the contribution from voxel j at detector pixel i (as in the standard notation). Notice here, that we have generalized to allow the system matrix to change at each iteration of the algorithm. This can take a physical interpretation as resolving or correcting for different physical effects as the iterative refinement progresses. Care must be taken here to either, not violate the convexity of the objective (e.g. in the EM case), or to ensure the basin of convergence is at least equivalent if not larger than the original algorithm.

In addition to growing deep, we can also grow wide, by considering algorithmic generalizations of the update rule used at each step. While the $c_{ij} \in C$ traditionally denote the various fixed parameters of the model, representing a summary of the various physics models that are assumed for the contribution of a voxel j to detector i, uncertainties in the parameters of these constituent models can be considered as free-variables ripe for optimization. As a simple example, emission-mode the photon contribution from voxel j can be modeled as:

$$\beta^{-1}(e^{-\beta t_i^1} - e^{-\beta t_i^2})\lambda_j b_{ij} = c_{ij}\lambda_j \tag{4.34}$$

When there is uncertainty in the parameters $\{\beta, t_i^1, t_i^2\}$, they can be tuned (either at run-time or *apriori*) to give the best reconstruction performance [3]. This was known even in the early literature, e.g. Lange and Carson point out that, for emission-mode SPECT, the MLEM algorithm can be modified to update the additional unknowns such as the "effect of randoms" in the detector count, as:

[3] In this formulation, with everything else held constant, this is a non-linear problem.

$$Y_i = \sum_{j \in I_i} X_{ij} + A_i \tag{4.35}$$

$$\text{so that,} \quad N_{ij} = \frac{c_{ij} \lambda_j^n Y_i}{\sum_{k \in I_i} c_{ik} \lambda_k^n + a_i^n} \tag{4.36}$$

where a_i^n is the estimate of a_i at the nth iteration [118]. However, as the number of unknowns increase, this strategy can be intractable for near-real-time algorithms of satisfactory performance, so we will focus our attention on the second, data-driven strategy of minimizing these stationary parameter uncertainties *apriori* with respect to a measure of quantitative performance on a representative set of training and validation data.

Just as the uncertainty in parameters of one model are amenable to optimization, when there is uncertainty in which physics models are at play, the contribution of each chosen model and its parameters can also be jointly optimized. For example, we could consider the effect anisotropic transmission, and the represent the attenuation map as a tensor. Initializing at the isotropic case, we could allow the parameter-update (learning) algorithm to determine how important each possibly-anisotropic transmission coefficient is to the final reconstruction. In this vein, we might want to enforce more structure to the distribution of c_{ij} (e.g. by asking that the effective transmission map is locally smooth or C is sparse) to incorporate what we know about the distributions these values are sampled from, while also providing flexibility for when our assumptions could use some tweaking.

Pushing the envelope further, we might allow c_{ij} to vary from iteration to iteration (either through global updates or data-dependent refinements), denoting them as c_{ij}^r. Even more radically, we could postulate multiple possible values for c_{ij}^r, evaluate them all, and pick the best estimate with respect to some local (e.g. time-of-flight) or global (e.g. total-variation, reconstruction error) measure of performance $\mathcal{M}$, as:

$$c_{ij}^r = \arg\max M[{}_w c_{ij}^r] \tag{4.37}$$

This formulation in particular may represent a family of auto-focussing algorithms, which iteratively selects which physical degradation or system models to correct for at various stages of the reconstruction.

Clearly, there are many such parametrizations that have highly-interpretable physical and algorithmic analogues. These algorithmic generalizations add many more parameters to the algorithm (similar to conventional DNN-based approaches), except that the overall structure and operation of the network of the original reconstuction algorithm is maintained. In fact, when we generalize so much, it is good to realize that the original Newton-Raphson or MLEM algorithm is easily recovered for specific choices of c_{ij}^r. Specifically, we suggest initializing any free-parameters with the effective constants used in the original version of the algorithm, so that a baseline of performance is achieved, prior to machine-learning-based optimization.

While the aforementioned generalizations provide robustness with respect to different models for c_{ij}, we see that we can also generalize in another way. Borrowing from the the consideration raised by Lange and Carson, for example, we can generalize Equation 4.36 further via a power-series approximation as:

$$N_{ij} = \frac{\sum_{p=0}^{\infty} \gamma_{ij}^p (b_{ij}\lambda_j Y_i)^p}{\sum_{k \in I_i} \sum_{p=0}^{\infty} \alpha_{ik}^p (b_{ik}\lambda_k)^p} \tag{4.38}$$

or suitable moments thereof, for a relevant choice of α_{ij}^p and γ_{ij}^p. Of course, a different basis expansion or parameterization may also be preferable to cover a different range of physical scenarios. We note that Equation 4.38 accounts for non-linearities that are both global and data-dependent (e.g. scattering only at particular voxels).

In short, by explicitly preserving the structure of the imaging algorithm within our deep architecture, and training within these constraints, we believe we can fill the gap between generic DNN approaches and classical reconstruction algorithms [174, 173, 172]. In particular, for this application we must show that such networks have

equivalent approximation power but offer greater control over unwanted oscillations (i.e. overfitting) [26].

Training

We note that all the formulas presented thusfar can be seen as rational polynomials, and therefore represent easily-differentiable functions amenable to optimization by gradient like methods. In particular, we can use higher-order stochastic gradient descent algorithms popular in the machine learning community, such as those that incorporate momentum terms into the update, as:

$$\omega_{n+1} = \sum_{0 \leq l < K} \beta_{n,l}\omega_{n-l} + \alpha_{n,l}(F'(\omega_{n-l}, Y))^{-1}F(\omega_{n-l}, Y) \tag{4.39}$$

where ω represents the vectorized collection of weights from the deep algorithm, Y represents the projection/sample data, F represents the deep imaging algorithm, and F' in this context refers to the gradient of F with respect to the parameters ω. In this work, we will stick to using the Adam optimizer [109].

4.3.3 Numerical Experiments and Discussion

In this section, we outline results from a series of small-phantom experiments on transmission-mode and emission-mode (SPECT) reconstruction tasks. In each case, we adopt a mild version of the generalization of a vanilla reconstruction algorithm, and show that with training the performance of the optimized algorithm has superior performance than the original formulation. To maintain fairness in our tests, we pick training and testing phantoms from disjoint sets, and randomize the attenuation and activity distributions.

Transmission-mode Reconstruction (XRCT)

We generalized a basic Newton-Raphson update, as follows:

$$\tau_{k+1} = \arg\min_{r} F\big(\tau_k - \alpha_k^r \frac{F(\tau_k, Y)}{F'(\tau_k, Y)}, Y\big) \tag{4.40}$$

$$\text{with} \quad F(\tau_k, Y) = \sum_{i}^{|Y|} ||p_{i,k} - Y_i||_2^2, \tag{4.41}$$

$$p_{i,k} = \text{ReLU}\big(\sum_{l}^{L} \big(\zeta_l exp(-l \cdot A\tau_k)\big)\big), \tag{4.42}$$

$$\text{and} \quad F'(\tau_k, Y) = \text{ReLU}\big(\hat{\nabla}_{\tau_k} F(\tau_k, Y) + \epsilon\big) \tag{4.43}$$

where τ_k represents the reconstruction estimate at iteration k, Y represents the recorded projection data, ReLU represents the rectified linear unit (alternatively implemented as the maximum of quantity and zero), $\hat{\nabla}_{\tau_k}$ represents the "normalized" derivative operator with respect to τ_k (its output is a unit vector indicating the gradient direction), and ϵ was chosen empirically as 1e-4 to prevent early termination due to zero-valued gradients. In this initial investigation, we fix the number of projection angles at 8, spaced equally from 0° to 180°.

We implemented this algorithm as a polynomial (rational function) neural network in Tensorflow, and optimized the aforementioned parameters with respect to the overall reconstruction loss, which we defined as:

$$\mathcal{L}_2 = \sum_{b} ||\hat{x}_b - x_b||_2^2 \tag{4.44}$$

where $\hat{x}_b$ represents the estimate of the reconstructed volume (output of the network), and x_b represents the true volume (phantom), of the b-th item in each training batch. To help control the behavior of the algorithm, additional constraints were placed on the projection-error in each iteration, as:

$$\mathcal{L} = \mathcal{L}_2 + \sum_b \sum_k \gamma_k \sum_{i=0}^{|Y|} ||{}_bp_{i,k} - {}_b\, Y_{i,k}||_2^2 \tag{4.45}$$

where ${}_bp_{i,k}$ represents the simulated projection at pixel i at each internal algorithm iteration k (e.g. Newton-iteration or MLEM-iteration) for the b-th item in the batch, ${}_bY_{i,k}$ represents the recorded value at the corresponding pixel, and γ_k represents a weight that is chosen according to the desired convergence criteria (e.g. $\gamma_{k+1} \geq \gamma_k$). Notice these additional terms represent an unsupervised objective, that only asks that the simulated projections get closer to the true projections. Therefore, independent optimization of these auxiliary terms could be used to train this algorithm on real-CT data, rather than phantoms (i.e. when the true geometry is not known).

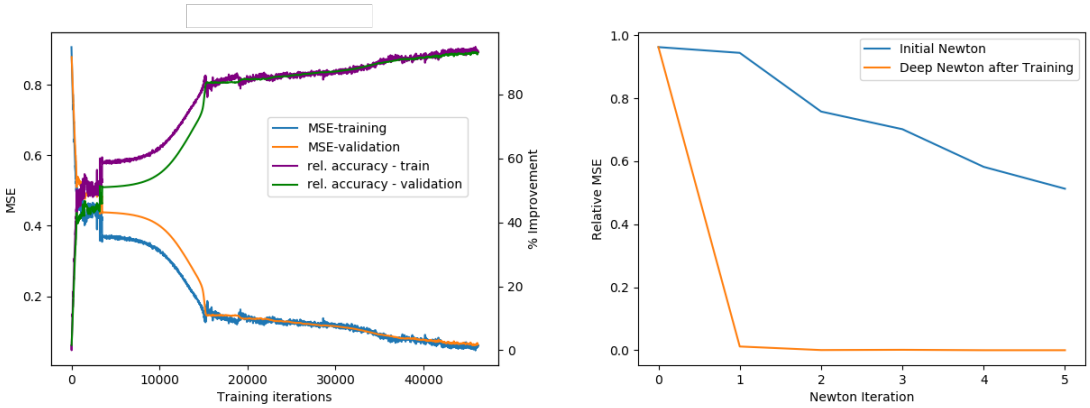


Figure 4.20: Performance of "Deep Newton-CT" for transmission tomography on 20x20x20 phantoms (a) averaged per-batch over the training period. (b) over internal algorithm iteration number.

The optimization was done using a version of stochchastic gradient descent, with the standard Adam optimizer (initial learning rate chosen empircally) with a batch size of 100, 20 x 20 x 20 phantoms for 50K training iterations. The training results, depicted in Figure 4.20, show a good correspondance between performance on the training and validation data. This is expected since the training, validation, and testing phantoms were all initialized randomly from a distribution. This highlights an important point that adjusting the distribution of training examples or objective constraints will be important

for using this data-driven method to overcome specific artifacts of interest, e.g. scattering, anisotropy, etc., in phenomenological- and detector-specific applications.

Emission-mode Reconstruction for SPECT

We generalized a basic MLEM update, as follows:

$$\lambda_j^{n+1} = \frac{\lambda_j^n}{\sum_{i\in J_j}\sum_p^P d_k(\rho_{i,j})^p} \cdot \sum_{i\in J_j} \frac{\sum_p^P b_k(\zeta_{i,j})^p}{\sum_{k\in I_i}\sum_p^P a_k(\gamma_{ij})^p} \tag{4.46}$$

$$\text{with} \quad \rho_{ij} = \text{ReLU}\big((_0W_{ij}\cdot c_{ij}) \cdot_1 W_{ij}\big), \tag{4.47}$$

$$\zeta_{ij} = \text{ReLU}\big((_0W_{ij}\cdot c_{ij}) \cdot_1 W_{ij}\cdot Y_i\big), \tag{4.48}$$

$$\text{and} \quad \gamma_{ij} = \text{ReLU}\big((_0W_{ij}\cdot c_{ij}) \cdot_2 W_{ij}\cdot \lambda_k\big) \tag{4.49}$$

where λ_j^n represents the estimated activity concentration of voxel j at iteration n, c_{ij} is the standard effective-transmission matrix (e.g. using simple, linear ray-like physics), and we have added the free parameters ρ, ζ, γ, a, b, c in a structured format. In this formulation, additional freedom can be gained by allowing these parameters to vary independently at each iteration of the algorithm. An algorithimic analogue of this approach is correcting for different artifacts at each iteration, as they become apparent to the algorithm or technician.

Clearly Eq. 4.46 is an embedding of the original rational (polynomial) function (Eq. 4.31) into a much larger function space of the same class. This network has the benefit of growing wide (by increasing P) and also deep ($n \to \infty$), while also maintaining the structure, intilialization, and understandability of the original MLEM algorithm. Of course, understanding the physical analogue and evaluating the reasonableness of the parameters that arise from such a model (deep or otherwise) remains an important task for any imaging scientist.

In this initial investigation, we fix the number of projection angles at 10, spaced equally from 0° to 360°. Parameters were initialized with all ones, except for $a_{k\neq 1}$, $b_{k\neq 1}$, and $c_{k\neq 1}$; this initialization allows the MLEM algorithm to start at the performance identical to

the original algorithm.

Similar to the Newton-Raphson case, we implemented the "Deep MLEM" as a polynomial neural network in Tensorflow, that was further trained using stochastic gradient descent (Adam optimizer) on randomly sampled activity and attenuation maps (30, 10 x 10 x 10 phantoms) that were generated as perturbations of a 3D Shepp-Logan-type phantom. The lower batch size was needed since the number of voxels that are assumed to contribute to a single detector pixel is much larger in the emission case than in the transmission case. The training results, depicted in Figure 4.21, again show a good correspondence between performance on the training and validation data. In particular, we recorded the convergence of the method at various levels of training, and show that this can systematically be enforced via training on the absolute error, and penalizing the r_{ratio} at successive iterations (layers) of the algorithm (network). Of course, for this size of phantom, the mean-absolute error (MAE) may not have much room for growth, as the r_{ratio} error is already quite small after a few iterations; this highlights a good reason why optimizing the overall reconstruction task can be preferable when ground-truth is available.

Discussion

The models presented in this section demonstrate a more-natural way to incorporate deep learning into physics-based imaging, without necessarily targeting a classification application. In particular, the presented framework is general enough to be applied to both Fourier and iterative image reconstruction methods that are popular in the high-energy medical imaging communities (e.g. XRCT, PET, SPECT, MRI, etc.). Of course, the convergence and resolution properties of any deep imaging algorithm developed using this methodology needs to be rigorously assessed; what we are providing here is a framework for approaching those tasks that leverages both our understanding of classical reconstruction algorithms, and new computational tools such as deep neural networks.

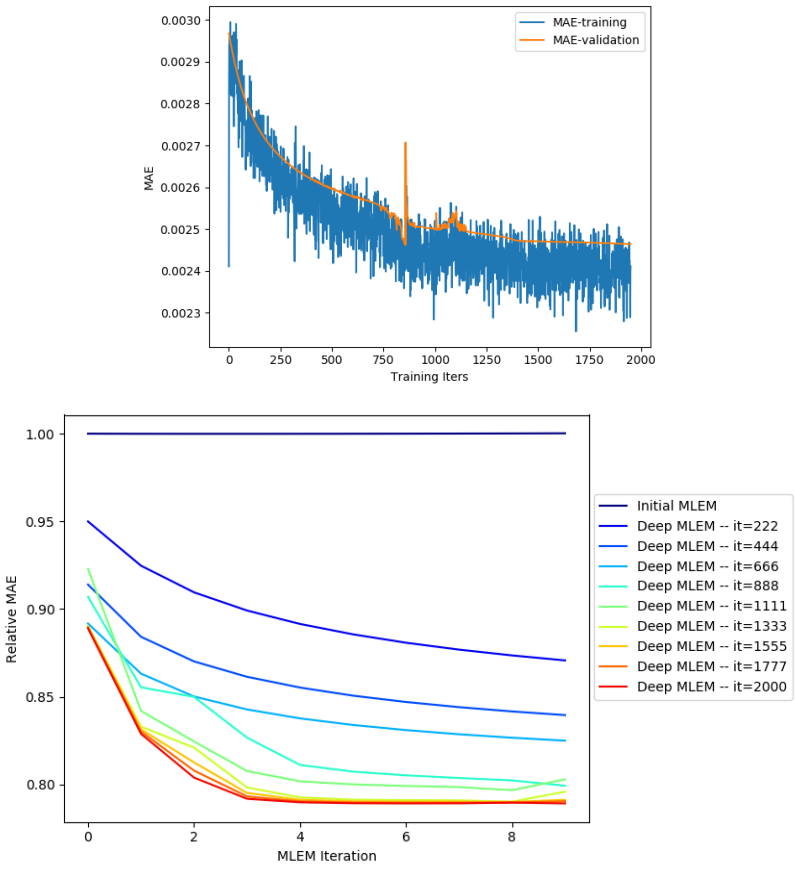


Figure 4.21: Performance of "Deep MLEM" for a SPECT task on 10x10x10 phantoms (a) averaged per-batch over the training period. (b) over internal algorithm iteration number after various levels of training (iteration number).

While the results presented here show systematic improvement in reconstruction performance, there are many difficulties in structuring the optimization to yield good and physically-meaningful results. One issue is potential bias introduced by the choice of training phantoms; the set of all possible input-output sequences is quite large, even when considering only medically-relevant distributions, and an appropriate subset must be chosen for practical optimization routines. Instead, we suggest initially using this methodology to investigate and explore whether addition of some parameters can aid in reconstruction or model the underlying physics. In this case, where the number of parameters and non-linearities are carefully added, incrementally, only a small number of phantoms should be required. We plan to explore this application in future work.

Another issue is that the initial reconstruction algorithm must perform reasonably well for the target application prior to the addition of additional free parameters. If this is not the case, it is possible that the actual improvements may be marginal. Adding more parameters is likely to increase the performance of even these initially suboptimal networks, but may make it difficult to traverse the function landscape to find a desirable local (or global) optima. Starting with a good algorithm not only provides guarantees of baseline performance, it defends against overfitting via addition of too many uninitialized (and uninterpretable) parameters.

Finally, we would like to point out that our methodology does not yet solve the problem of designing deep imaging algorithms for computed tomography, but rather introduces a platform on which machine learning algorithms should be tested and benchmarked (i.e. in comparison to existing methods). Further work, such as those investigating suitable optimization algorithms for constrained networks, or those that defend against training bias, should be investigated in future work.

Chapter 5

Conclusion

In this thesis, we used well-established techniques of polynomial approximation as a vehicle to investigate and understand the unreasonable effectiveness of deep learning in image reconstruction and image recognition problems. An outcome of this research is a set of conjectures relating the structure and performance of deep networks, and a set of design principles that realize this for practical problems.

As we saw in Chapter 2, deep networks can provably avoid the curse of dimensionality with respect to the degree of approximation when they utilize the compositional structure of the target function. Such networks also have provably small generalization error near training data. We conjecture that utilization of the compositional structure also yields benefits to the requisite data rate for a desired level of performance, which would otherwise grow exponentially in the dimension. In the case of $\max_d$, this was empirically shown to be the case up to $d = 32$ for point data sampled randomly in the unit hypercube $\mathbb{R}_{[-1,1]}$. The extension of this idea for high dimensional ($d \geq 1000$) image recognition problems is left as future work.

In particular, we note that to avoid the curse of dimensionality, the compositional representation must feature *low in-degree* nodes. For very high dimensional problems, this implies very deep networks. Numerical concerns arise in the training of these deep

networks. Various schemes can be used, but of immediate interest and concern is the numerical stability of backpropagation for gradient-based optimization of polynomial networks. This is left as future work.

On the other hand, there are many compositional structures (albeit of high in-degree) that have been discovered both by traditional algorithm designers (e.g. clustering, Newton's method, filtered back-projection) and by DNN practitioners (e.g. VGG-16, U-Net, ResNet) that indicate good performance on modern problems. Thus, an important line of investigation would be to ask how the approach of deep minimum Sobolev norm networks could be applied the training of these networks, and what guarantees on the generalization error can be derived. For example, conventional DNNs can be approximated by a deep polynomial network with a tractable expression for the Sobolev norm that is a simple (fixed) formula of the weights. It would be fruitful to ask if such networks are more robust, e.g. with respect to adversarial attacks.

More generally, there are many open questions. For example:

- Given samples of a high dimensional function, how do we uncover its compositional structure?
- Is there a penalty for using a graph structure that is larger than the compositional graph structure of the target function?
- Is there a paradigm for removing the acyclic requirement for the compositional structure of the target function, i.e. can we approximate functions with recurrent networks?
- Are there other classes of problems that can be encoded efficiently with deep networks, e.g. combinatorial problems or dynamic programming algorithms?

Answering these questions will help extend deep learning to address many open problems in imaging and image recognition, including accelerating the solution to several classes of inverse problems that are currently considered intractable [44].

Bibliography

[1] Ajith Abraham and Lakhmi Jain. "Evolutionary multiobjective optimization". In: *Evolutionary Multiobjective Optimization*. Springer, 2005, pp. 1–6.

[2] Howard M Adelman and Raphael T Haftka. "Sensitivity analysis of discrete structural systems". In: *AIAA journal* 24.5 (1986), pp. 823–832.

[3] Omar Y Al-Jarrah et al. "Efficient machine learning for big data: A review". In: *Big Data Research* 2.3 (2015), pp. 87–93.

[4] Eugene L Allgower and Kurt Georg. *Numerical continuation methods: an introduction*. Vol. 13. Springer Science & Business Media, 2012.

[5] Amir A Amini, Terry E Weymouth, and Ramesh C Jain. "Using dynamic programming for solving variational problems in vision". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 9 (1990), pp. 855–867.

[6] Habib Ammari et al. "Electrical impedance tomography by elastic deformation". In: *SIAM Journal on Applied Mathematics* 68.6 (2008), pp. 1557–1573.

[7] Peter Arbenz and Walter Gander. "Solving nonlinear eigenvalue problems by algorithmic differentiation". In: *Computing* 36.3 (1986), pp. 205–215.

[8] Ahmed Ashraf et al. "Learning to Unlearn: Building Immunity to Dataset Bias in Medical Imaging Studies". In: *arXiv preprint arXiv:1812.01716* (2018).

[9] Pierre Baldi and Kurt Hornik. "Neural networks and principal component analysis: Learning from examples without local minima". In: *Neural networks* 2.1 (1989), pp. 53–58.

[10] Andrew R Barron. "Approximation and estimation bounds for artificial neural networks". In: *Machine learning* 14.1 (1994), pp. 115–133.

[11] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

[12] LM Beda et al. "Programs for automatic differentiation for the machine BESM". In: *Inst. Precise Mechanics and Computation Techniques, Academy of Science, Moscow* (1959).

[13] R Berber and C Brosilow. "Insights into the Relationships Between Linear and Nonlinear Model Based Control and Issues for Further Research". In: *Nonlinear Model Based Process Control*. Springer, 1998, pp. 87–114.

[14] Dimitri P Bertsekas. "Nonlinear programming". In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334.

[15] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Vol. 5. Athena Scientific Belmont, MA, 1996.

[16] Dario Bini and Victor Y Pan. *Polynomial and matrix computations: fundamental algorithms*. Springer Science & Business Media, 2012.

[17] Dario Andrea Bini. "Numerical computation of polynomial zeros by means of Aberth's method". In: *Numerical algorithms* 13.2 (1996), pp. 179–200.

[18] Dario Andrea Bini and Giuseppe Fiorentino. "Design, analysis, and implementation of a multiprecision polynomial rootfinder". In: *Numerical Algorithms* 23.2-3 (2000), pp. 127–173.

[19] Chris M Bishop. "Neural networks and their applications". In: *Review of scientific instruments* 65.6 (1994), pp. 1803–1832.

[20] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[21] Mathieu Blondel et al. "Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms". In: *International Conference on Machine Learning*. 2016, pp. 850–858.

[22] Ranko Bojanic and Fuhua Cheng. "Rate of convergence of Bernstein polynomials for functions with derivatives of bounded variation". In: *Journal of Mathematical Analysis and Applications* 141.1 (1989), pp. 136–151.

[23] Andrew P Brown, Michael J Sheffler, and Katherine E Dunn. "Persistent electro-optical/infrared wide-area sensor exploitation". In: *Evolutionary and Bio-Inspired Computation: Theory and Applications VI*. Vol. 8402. International Society for Optics and Photonics. 2012, p. 840206.

[24] Hugh Calkins et al. "Treatment of atrial fibrillation with anti–arrhythmic drugs or radio frequency ablation: Two systematic literature reviews and meta–analyses". In: *Circulation: Arrhythmia and Electrophysiology* (2009), CIRCEP–108.

[25] Luca Caltagirone et al. "LIDAR–camera fusion for road detection using fully convolutional neural networks". In: *Robotics and Autonomous Systems* 111 (2019), pp. 125–131.

[26] Emmanuel J Candès. "Harmonic analysis of neural networks". In: *Applied and Computational Harmonic Analysis* 6.2 (1999), pp. 197–218.

[27] Emmanuel J Candès. "New ties between computational harmonic analysis and approximation theory". In: *Approximation Theory X* (2002), pp. 87–153.

[28] Emmanuel Jean Candes. "Ridgelets: theory and applications". PhD thesis. Stanford University Stanford, 1998.

[29] Matthew Carlberg et al. “Classifying urban landscape in aerial LiDAR using 3D shape analysis”. In: *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE. 2009, pp. 1701–1704.

[30] Ozgun Cciccek et al. “3D U-Net: learning dense volumetric segmentation from sparse annotation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 424–432.

[31] Judit Chamorro-Servent et al. “Improving the Spatial Solution of Electrocardiographic Imaging: A New Regularization Parameter Choice Technique for the Tikhonov Method”. In: *International Conference on Functional Imaging and Modeling of the Heart*. Springer. 2017, pp. 289–300.

[32] S Chandrasekaran, CH Gorman, and HN Mhaskar. “Minimum Sobolev norm interpolation of derivative data”. In: *arXiv preprint arXiv:1710.01419* (2017).

[33] S Chandrasekaran and HN Mhaskar. “A construction of linear bounded interpolatory operators on the torus”. In: *arXiv preprint arXiv:1011.5448* (2010).

[34] S Chandrasekaran and A Rajagopal. “Fast indefinite multi-point (IMP) clustering”. In: *Calcolo* 54.1 (2017), pp. 401–421.

[35] S Chandrasekaran et al. “Minimum Sobolev Norm schemes and applications in image processing”. In: *Wavelet applications in industrial processing VII*. Vol. 7535. International Society for Optics and Photonics. 2010, p. 753507.

[36] Shiv Chandrasekaran et al. “Higher order numerical discretization methods with sobolev norm minimization”. In: *Procedia Computer Science* 4 (2011), pp. 206–215.

[37] Shivkumar Chandrasekaran, CH Gorman, and Hrushikesh Narhar Mhaskar. “Minimum Sobolev norm interpolation of scattered derivative data”. In: *Journal of Computational Physics* 365 (2018), pp. 149–172.

[38] Shivkumar Chandrasekaran, KR Jayaraman, and Hrushikesh Narhar Mhaskar. "Minimum Sobolev norm interpolation with trigonometric polynomials on the torus". In: *Journal of Computational Physics* 249 (2013), pp. 96–112.

[39] Shivkumar Chandrasekaran and Hrushikesh Narhar Mhaskar. "A minimum Sobolev norm technique for the numerical discretization of PDEs". In: *Journal of Computational Physics* 299 (2015), pp. 649–666.

[40] Chaofan Chen et al. "This looks like that: deep learning for interpretable image recognition". In: *arXiv preprint arXiv:1806.10574* (2018).

[41] Hu Chen et al. "Low-dose CT via convolutional neural network". In: *Biomedical optics express* 8.2 (2017), pp. 679–694.

[42] Hu Chen et al. "Low-dose CT with a residual encoder-decoder convolutional neural network". In: *IEEE transactions on medical imaging* 36.12 (2017), pp. 2524–2535.

[43] Liang-Chieh Chen et al. "Semantic image segmentation with deep convolutional nets and fully connected crfs". In: *arXiv preprint arXiv:1412.7062* (2014).

[44] Victor C Chen. *Inverse Synthetic Aperture Radar Imaging; Principles*. Institution of Engineering and Technology, 2014.

[45] Xi Chen et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *Advances in neural information processing systems*. 2016, pp. 2172–2180.

[46] Fuhua Cheng. "On the rate of convergence of Bernstein polynomials of functions of bounded variation". In: *Journal of approximation theory* 39.3 (1983), pp. 259–274.

[47] Youngmin Cho and Lawrence K Saul. "Kernel methods for deep learning". In: *Advances in neural information processing systems*. 2009, pp. 342–350.

[48] Jiyoung Choi et al. “Sparsity driven metal part reconstruction for artifact removal in dental CT”. In: *Journal of X-ray Science and Technology* 19.4 (2011), pp. 457–475.

[49] Lark Kwon Choi, Jaehee You, and Alan Conrad Bovik. “Referenceless prediction of perceptual fog density and perceptual image defogging”. In: *IEEE Transactions on Image Processing* 24.11 (2015), pp. 3888–3901.

[50] City of Vancouver. *2013 Dublin LiDAR Dataset*. 2013. URL: `https://data.vancouver.ca/datacatalogue/LiDAR2013.htm`.

[51] Taco Cohen and Max Welling. “Group equivariant convolutional networks”. In: *International conference on machine learning*. 2016, pp. 2990–2999.

[52] Annie AM Cuyt and Louis B Rall. “Computational implementation of the multivariate Halley method for solving nonlinear systems of equations”. In: *ACM Transactions on Mathematical Software (TOMS)* 11.1 (1985), pp. 20–36.

[53] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[54] George Cybenko. “Mathematical problems in neural computing”. In: *Signal Processing Scattering and Operator Theory and Numerical Processing* 3 (1989), pp. 47–64.

[55] Angela Dai et al. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5828–5839.

[56] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *international Conference on computer vision & Pattern Recognition (CVPR'05)*. Vol. 1. IEEE Computer Society. 2005, pp. 886–893.

[57] Zolzaya Dashdorj et al. “Classification of news by topic using location data”. In: *Joint International Semantic Technology Conference*. Springer. 2016, pp. 305–314.

[58] Ilke Demir et al. “Deepglobe 2018: A challenge to parse the earth through satellite images”. In: *ArXiv e-prints* (2018).

[59] Arnaud Dessein et al. “Seamless texture stitching on a 3D mesh by Poisson blending in patches”. In: *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 2031–2035.

[60] Ronald A DeVore, Ralph Howard, and Charles Micchelli. “Optimal nonlinear approximation”. In: *Manuscripta mathematica* 63.4 (1989), pp. 469–478.

[61] Joyoni Dey and Michael A King. “Theoretical and numerical study of MLEM and OSEM reconstruction algorithms for motion correction in emission tomography”. In: *IEEE transactions on nuclear science* 56.5 (2009), pp. 2739–2749.

[62] Umesh R Dhond and Jake K Aggarwal. “Structure from stereo-a review”. In: *IEEE transactions on systems, man, and cybernetics* 19.6 (1989), pp. 1489–1510.

[63] Min Ding, Kristian Lyngbaek, and Avideh Zakhor. “Automatic registration of aerial imagery with untextured 3d lidar models”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[64] Gintare Karolina Dziugaite and Daniel M Roy. “Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data”. In: *arXiv preprint arXiv:1703.11008* (2017).

[65] Martin Engelcke et al. “Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks”. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 1355–1361.

[66] L.C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2010. ISBN: 9780821849743. URL: `https://books.google.com/books?id=Xnu0o\_EJrCQC`.

[67] Georg Faber. “Über die interpolatorische Darstellung stetiger Funktionen”. In: *Jber. Deutsch. Math. Verein* 23 (1914), pp. 192–210.

[68] Andriy Fedorov et al. “3D Slicer as an image computing platform for the Quantitative Imaging Network”. In: *Magnetic resonance imaging* 30.9 (2012), pp. 1323–1341.

[69] Carlos Figuera et al. “Regularization techniques for ECG imaging during atrial fibrillation: A computational study”. In: *Frontiers in physiology* 7 (2016), p. 466.

[70] R Fletcher and CM Reeves. “A mechanization of algebraic differentiation and the automatic generation of formulae for molecular integrals of Gaussian orbitals”. In: *The Computer Journal* 6.3 (1963), pp. 287–292.

[71] Cidalia Costa Fonte et al. “An automated methodology for converting OSM data into a land use/cover map”. In: *Proceedings of the 6 th International Conference on Cartography & GIS*. 2016, pp. 462–473.

[72] Edward W Forgy. “Cluster analysis of multivariate data: efficiency versus interpretability of classifications”. In: *biometrics* 21 (1965), pp. 768–769.

[73] Roy Frostig, Matthew James Johnson, and Chris Leary. *Compiling machine learning programs via high-level tracing*. 2018.

[74] Mariano Gasca and Thomas Sauer. “Polynomial interpolation in several variables”. In: *Advances in Computational Mathematics* 12.4 (2000), p. 377.

[75] Andreas Geiger et al. “Vision meets robotics: The KITTI dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[76] Raja N Ghanem et al. “Noninvasive electrocardiographic imaging (ECGI): comparison to intraoperative mapping in patients”. In: *Heart Rhythm* 2.4 (2005), pp. 339–354.

[77] MICHAEL GOLOMB and HANS F WEINESERGER. *Optimal approximation and error bounds, in On Numerical Approximation, proceedings of a symposium held at Madison, Wisconsin, April, 1958 (RE Langer, Ed.)* 1959.

[78] GH Golub and CF Van Loan. "Matrix Computations 4th Edition The Johns Hopkins University Press". In: *Baltimore, MD* (2013).

[79] Susana Gomez, Angel Perez, and Rosa M Alvarez. "Multiscale optimization for aquifer parameter identification with noisy data". In: *WIT Transactions on Ecology and the Environment* 24 (1970).

[80] Kuang Gong et al. "EMnet: an unrolled deep neural network for PET image reconstruction". In: *Medical Imaging 2019: Physics of Medical Imaging*. Vol. 10948. International Society for Optics and Photonics. 2019, p. 1094853.

[81] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[82] Christopher H. Gorman. "Applications of the Minimum Sobolev Norm and Associated Fast Algorithms". PhD thesis. University of California, Santa Barbara, 2019.

[83] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural turing machines". In: *arXiv preprint arXiv:1410.5401* (2014).

[84] Andreas Griewank et al. "On automatic differentiation". In: *Mathematical Programming: recent developments and applications* 6.6 (1989), pp. 83–107.

[85] L Grippo, F Lampariello, and S Lucidi. "A truncated Newton method with nonmonotone line search for unconstrained optimization". In: *Journal of Optimization Theory and Applications* 60.3 (1989), pp. 401–419.

[86] Ingo Guhring, Gitta Kutyniok, and Philipp Petersen. "Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms". In: *arXiv preprint arXiv:1902.07896* (2019).

[87] Maria S Guillem et al. "Noninvasive localization of maximal frequency sites of atrial fibrillation by body surface potential mapping". In: *Circulation. Arrhythmia and electrophysiology* 6.2 (2013), p. 294.

[88] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. "Perceptual organization and recognition of indoor scenes from RGB-D images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 564–571.

[89] Saurabh Gupta et al. "Learning rich features from RGB-D images for object detection and segmentation". In: *European Conference on Computer Vision*. Springer. 2014, pp. 345–360.

[90] William H Guss. "Deep function machines: Generalized neural networks for topological layer expression". In: *arXiv preprint arXiv:1612.04799* (2016).

[91] Michel Haïssaguerre et al. "Mapping and ablation of idiopathic ventricular fibrillation". In: *Circulation* 106.8 (2002), pp. 962–967.

[92] David M Harrild and Craig S Henriquez. "A computer model of normal conduction in the human atria". In: *Circulation research* 87.7 (2000), e25–e36.

[93] Simon Haykin. "A comprehensive foundation". In: *Neural networks* 2.2004 (2004), p. 41.

[94] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[95] Raymond M Hicks and Preston A Henne. "Wing design by numerical optimization". In: *Journal of Aircraft* 15.7 (1978), pp. 407–412.

[96] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

[97] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[98] Albert S Huang et al. "Visual odometry and mapping for autonomous flight using an RGB-D camera". In: *Robotics Research*. Springer, 2017, pp. 235–252.

[99] Russell T Hyde et al. *Aircraft-based topographical data collection and processing system*. US Patent 5,557,397. 1996.

[100] Aapo Hyvärinen and Erkki Oja. "Independent component analysis: algorithms and applications". In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[101] Michael Innes et al. "Fashionable Modelling with Flux". In: *arXiv preprint arXiv:1811.01457* (2018).

[102] M Isenburg. "LAStools-efficient tools for LiDAR processing". In: *Available at: http: http://www. cs. unc. edu/˜ isenburg/lastools/[Accessed October 9, 2012]* (2012).

[103] John David Jackson. *Classical electrodynamics*. 1999.

[104] Kyong Hwan Jin et al. "Deep convolutional neural network for inverse problems in imaging". In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522.

[105] T Kaczorek. "Generalizations of the Cayley-Hamilton theorem with applications". In: *Archives of Electrical Engineering* 56.1 (2007), pp. 3–41.

[106] T Kaczorek and A Stajniak. "An extension of the Cayley-Hamilton theorem for singular 2-D linear systems with non-square matrices". In: *Bulletin of the Polish Academy of Sciences. Technical Sciences* 43.1 (1995), pp. 39–48.

[107] Eunhee Kang, Junhong Min, and Jong Chul Ye. "A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction". In: *Medical physics* 44.10 (2017).

[108] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[109] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[110] Peter M Kistler et al. "Validation of Three-Dimensional Cardiac Image Integration: Use of Integrated CT Image into Electroanatomic Mapping System to Perform Catheter Ablation of Atrial Fibrillation". In: *Journal of cardiovascular electrophysiology* 17.4 (2006), pp. 341–348.

[111] Aleksey Kondratyev, Hans J Stetter, and Franz Winkler. "Numerical computation of Gröbner bases". In: *Proceedings of CASC2004 (Computer Algebra in Scientific Computing)* (2004), pp. 295–306.

[112] Paul F Kough. "The indefinite quadratic programming problem". In: *Operations Research* 27.3 (1979), pp. 516–533.

[113] C Kredler and W Kowarschick. "A Convenient Way of Computing ML-Estimates: Use of Automatic Differentiation". In: *COMPSTAT*. Springer. 1986, pp. 108–113.

[114] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[115] Michael Kusenbach, Michael Himmelsbach, and Hans-Joachim Wuensche. "A new geometric 3D LiDAR feature for model creation and classification of moving objects". In: *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE. 2016, pp. 272–278.

[116] Vivek Kwatra et al. "Texture optimization for example-based synthesis". In: *ACM Transactions on Graphics (ToG)*. Vol. 24. ACM. 2005, pp. 795–802.

[117] Debra F Laefer et al. "2015 aerial laser and photogrammetry survey of dublin city collection record". In: *New York University, Tech. Rep* (2017). DOI: http://dx.doi.org/10.17609/N8MQ0N.

[118] Kenneth Lange, Richard Carson, et al. "EM reconstruction algorithms for emission and transmission tomography". In: *J Comput Assist Tomogr* 8.2 (1984), pp. 306–16.

[119] Bjornar Larsen and Chinatsu Aone. "Fast and effective text mining using linear-time document clustering". In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. Citeseer. 1999, pp. 16–22.

[120] Charles L Lawson. "C^1 SURFACE INTERPOLATION FOR SCATTERED DATA ON A SPHERE". In: *The Rocky Mountain journal of mathematics* (1984), pp. 177–202.

[121] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[122] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010), p. 18.

[123] Yann LeCun et al. "A theoretical framework for back-propagation". In: *Proceedings of the 1988 connectionist models summer school*. Vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann. 1988, pp. 21–28.

[124] Yann LeCun et al. "Generalization and network design strategies". In: *Connectionism in perspective*. Vol. 19. Citeseer, 1989.

[125] Tsu-Tian Lee and Jin-Tsong Jeng. "The Chebyshev-polynomials-based unified model neural networks for function approximation". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.6 (1998), pp. 925–935.

[126] Shuai Leng et al. "Noise reduction in spectral CT: Reducing dose and breaking the trade-off between image noise and energy bin selection". In: *Medical physics* 38.9 (2011), pp. 4946–4957.

[127] Moshe Leshno et al. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural networks* 6.6 (1993), pp. 861–867.

[128] Holger Leuck and H-H Nagel. "Automatic differentiation facilitates OF-integration into steering-angle-based road vehicle tracking". In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 2. IEEE. 1999, pp. 360–365.

[129] Anton Leykin, Jan Verschelde, and Ailing Zhao. "Newton's method with deflation for isolated singularities of polynomial systems". In: *Theoretical Computer Science* 359.1-3 (2006), pp. 111–122.

[130] T Li et al. "Motion correction for improved target localization with on-board cone-beam computed tomography". In: *Physics in Medicine & Biology* 51.2 (2005), p. 253.

[131] Hongcheng Liu and Yinyu Ye. "High-Dimensional Learning under Approximate Sparsity: A Unifying Framework for Nonsmooth Learning and Regularized Neural Networks". In: *arXiv preprint arXiv:1903.00616* (2019).

[132] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. "On the computational efficiency of training neural networks". In: *Advances in neural information processing systems*. 2014, pp. 855–863.

[133] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[134] Alejandro Lopez-Rincon, Mostafa Bendahmane, and BedrEddine Ainseba. "On 3D numerical inverse problems for the bidomain model in electrocardiology". In: *Computers & Mathematics with Applications* 69.4 (2015), pp. 255–274.

[135] Hans-Gerd Maas. "Fast determination of parametric house models from dense airborne laserscanner data". In: *International Workshop on Mobile Mapping Technology*. Vol. 32. 2W1. 1999, pp. 1–6.

[136] Hans-Gerd Maas and George Vosselman. "Two algorithms for extracting building models from raw laser altimetry data". In: *ISPRS Journal of photogrammetry and remote sensing* 54.2-3 (1999), pp. 153–163.

[137] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. "Autograd: Effortless gradients in numpy". In: *ICML 2015 AutoML Workshop*. 2015.

[138] Shahin Mahdizadehaghdam et al. "Deep dictionary learning: A parametric network approach". In: *arXiv preprint arXiv:1803.04022* (2018).

[139] Andelo Martinovic et al. "3d all the way: Semantic segmentation of urban scenes from start to end in 3d". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4456–4465.

[140] David Mascharka et al. "Transparency by design: Closing the gap between performance and interpretability in visual reasoning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4942–4950.

[141] Moritz Menze and Andreas Geiger. "Object scene flow for autonomous vehicles". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3061–3070.

[142] Gregory P Meyer et al. "Sensor Fusion for Joint 3D Object Detection and Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[143] Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. "Learning functions: when is deep better than shallow". In: *arXiv preprint arXiv:1603.00988* (2016).

[144] Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. "When and why are deep networks better than shallow ones?" In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[145] Hrushikesh Mhaskar and Tomaso Poggio. “An analysis of training and generalization errors in shallow and deep networks”. In: *arXiv preprint arXiv:1802.06266* (2018).

[146] Hrushikesh N Mhaskar. “Neural networks for optimal approximation of smooth and analytic functions”. In: *Neural computation* 8.1 (1996), pp. 164–177.

[147] Hrushikesh N Mhaskar and Charles A Micchelli. “Approximation by superposition of sigmoidal and radial basis functions”. In: *Advances in Applied mathematics* 13.3 (1992), pp. 350–373.

[148] Hrushikesh N Mhaskar, Paul Nevai, and Eugene Shvarts. “Applications of classical approximation theory to periodic basis function networks and computational harmonic analysis”. In: *Bulletin of Mathematical Sciences* 3.3 (2013), pp. 485–549.

[149] Hrushikesh N Mhaskar and Tomaso Poggio. “Deep vs. shallow networks: An approximation theory perspective”. In: *Analysis and Applications* 14.06 (2016), pp. 829–848.

[150] Alexis Mignon and Frédéric Jurie. “Pcca: A new approach for distance learning from sparse pairwise constraints”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2666–2672.

[151] Mike Mintz et al. “Distant supervision for relation extraction without labeled data”. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics. 2009, pp. 1003–1011.

[152] Grégoire Montavon, Geneviève Orr, and Klaus-Robert Müller. *Neural networks: tricks of the trade*. Vol. 7700. springer, 2012.

[153] Isidor P Natanson and Alexis N Obolensky. *Constructive Function Theory: Vol. I: Uniform Approximation*. Frederick Ungar Publishing Company, 1964.

[154] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 427–436.

[155] Joachim Niemeyer, Franz Rottensteiner, and Uwe Soergel. "Contextual classification of lidar data and building object detection in urban areas". In: *ISPRS journal of photogrammetry and remote sensing* 87 (2014), pp. 152–165.

[156] David Novotny et al. "Semi-convolutional operators for instance segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 86–102.

[157] Office of the Chief Technology Officer (OCTO), District of Columbia GIS program. *LiDAR - DC Point Cloud - 2018*. 2018. URL: https://opendata.dc.gov/datasets/e412b5f1f3bc4a89a5bebc496c1a5279.

[158] Sung-Kwun Oh, Witold Pedrycz, and Byoung-Jun Park. "Polynomial neural networks architecture: analysis and design". In: *Computers & Electrical Engineering* 29.6 (2003), pp. 703–725.

[159] Ohio Geographicall Referenced Information Program. *OSIP LiDAR*. 2017. URL: https://ogrip.oit.ohio.gov/ProjectsInitiatives/StatewideImagery.aspx.

[160] Frank Pannizzo et al. "Automatic methods for detection of tachyarrhythmias by antitachycardia devices". In: *Journal of the American College of Cardiology* 11.2 (1988), pp. 308–316.

[161] Jorge Pedrón-Torrecilla et al. "Noninvasive Estimation of Epicardial Dominant High-Frequency Regions During Atrial Fibrillation". In: *Journal of cardiovascular electrophysiology* 27.4 (2016), pp. 435–442.

[162] Nicholas Pippenger and Michael J Fischer. “Relations among complexity measures”. In: *Journal of the ACM (JACM)* 26.2 (1979), pp. 361–381.

[163] André Platzer. “Differential-algebraic dynamic logic for differential-algebraic programs”. In: *Journal of Logic and Computation* 20.1 (2008), pp. 309–352.

[164] Tomaso Poggio, Fabio Anselmi, and Lorenzo Rosasco. *I-theory on depth vs width: hierarchical function composition*. Tech. rep. Center for Brains, Minds and Machines (CBMM), 2015.

[165] Tomaso Poggio et al. “Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review”. In: *International Journal of Automation and Computing* 14.5 (2017), pp. 503–519.

[166] Andrew J Pullan et al. “The inverse problem of electrocardiography”. In: *Comprehensive Electrocardiology*. Springer, 2010, pp. 299–344.

[167] Charles R Qi et al. “Frustum pointnets for 3d object detection from rgb-d data”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 918–927.

[168] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108.

[169] Abhejit Rajagopal, Shivkumar Chandrasekaran, and Hrushikesh N Mhaskar. “Deep algorithms: designs for networks”. In: *arXiv preprint arXiv:1806.02003* (2018).

[170] Abhejit Rajagopal et al. “A machine learning pipeline for automated registration and classification of 3D lidar data”. In: *Geospatial Informatics, Fusion, and Motion Video Analytics VII*. Vol. 10199. International Society for Optics and Photonics. 2017, p. 101990D.

[171] Abhejit Rajagopal et al. “Deep-OSM-3D: Scalable Recognition in Wide-Area LiDAR & RGBD Imagery”. In: *in submission to Winter Conerence on Applications of Computer Vision (WACV)*. 2020.

[172] Abhejit Rajagopal et al. “Nonlinear electrocardiographic imaging using polynomial approximation networks”. In: *APL Bioengineering* 2.4 (2018), p. 046101.

[173] Abhejit Rajagopal et al. “Towards non-invasive electrocardiographic imaging using regularized neural networks”. In: *Medical Imaging 2018: Physics of Medical Imaging*. Vol. 10573. International Society for Optics and Photonics. 2018, 105732O.

[174] Abhejit Rajagopal et al. “TRACKING INFORMATION IN SAR IMAGE FORMATION AND CLASSIFICATION ALGORITHMS”. In: International Foundation for Telemetering. 2017.

[175] Louis B Rall. “Applications of software for automatic differentiation in numerical computation”. In: *Fundamentals of Numerical Computation (Computer-Oriented Numerical Analysis)*. Springer, 1980, pp. 141–156.

[176] Prajit Ramachandran, Barret Zoph, and Quoc V Le. “Searching for activation functions”. In: *arXiv preprint arXiv:1710.05941* (2017).

[177] Charulatha Ramanathan et al. “Noninvasive electrocardiographic imaging for cardiac electrophysiology and arrhythmia”. In: *Nature medicine* 10.4 (2004), p. 422.

[178] Javier Ramírez et al. “Effective emission tomography image reconstruction algorithms for SPECT data”. In: *International Conference on Computational Science*. Springer. 2008, pp. 741–748.

[179] Sebastian Riedel, Matko Bosnjak, and Tim Rocktäschel. “Programming with a differentiable forth interpreter”. In: *CoRR, abs/1605.06640* (2016).

[180] Miguel Rodrigo et al. “Solving inaccuracies in anatomical models for electrocardiographic inverse problem resolution by maximizing reconstruction quality”. In: *IEEE transactions on medical imaging* 37.3 (2018), pp. 733–740.

[181] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[182] Riccardo Rovatti. "Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in Sobolev norms". In: *IEEE Transactions on Fuzzy Systems* 6.2 (1998), pp. 235–249.

[183] Debaditya Roy, K Sri Rama Murty, and C Krishna Mohan. "Feature selection using deep neural networks". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2015, pp. 1–6.

[184] Yoram Rudy and John E Burnes. "Noninvasive electrocardiographic imaging". In: *Annals of Noninvasive electrocardiology* 4.3 (1999), pp. 340–359.

[185] DE Rumelhart, GE Hinton, and RJ Williams. "Learning internal representations by error propagation". In: *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*. MIT Press. 1986, pp. 318–362.

[186] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic routing between capsules". In: *Advances in neural information processing systems*. 2017, pp. 3856–3866.

[187] Grant Schindler and Frank Dellaert. "Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. IEEE. 2004, pp. I–I.

[188] Johannes L Schonberger et al. "Semantic Visual Localization". In: *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)* (2018).

[189] Muhammad Shahzad et al. "Buildings Detection in VHR SAR Images Using Fully Convolution Neural Networks". In: *arXiv preprint arXiv:1808.06155* (2018).

[190] Shashi Shekhar et al. “Spatial contextual classification and prediction models for mining geospatial data”. In: *IEEE Transactions on Multimedia* 4.2 (2002), pp. 174–188.

[191] Patrice Simard, Yann LeCun, and John S Denker. “Efficient pattern recognition using a new transformation distance”. In: *Advances in neural information processing systems*. 1993, pp. 50–58.

[192] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[193] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.

[194] Michael Steinbach, George Karypis, Vipin Kumar, et al. “A comparison of document clustering techniques”. In: *KDD workshop on text mining*. Vol. 400. 1. Boston. 2000, pp. 525–526.

[195] S Adam Strickberger et al. “Mapping and ablation of ventricular tachycardia guided by virtual electrograms using a noncontact, computerized mapping system”. In: *Journal of the American College of Cardiology* 35.2 (2000), pp. 414–421.

[196] Ilya Sutskever and Geoffrey E Hinton. “Deep, narrow sigmoid belief networks are universal approximators”. In: *Neural computation* 20.11 (2008), pp. 2629–2636.

[197] Shin Suzuki. “Constructive function-approximation by three-layer artificial neural networks”. In: *Neural Networks* 11.6 (1998), pp. 1049–1058.

[198] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[199] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[200] Graham W Taylor et al. “Convolutional learning of spatio-temporal features”. In: *European conference on computer vision*. Springer. 2010, pp. 140–153.

[201] Matus Telgarsky. “Benefits of depth in neural networks”. In: *Conference on Learning Theory*. 2016, pp. 1517–1539.

[202] Matus Telgarsky. “Representation benefits of deep feedforward networks”. In: *arXiv preprint arXiv:1509.08101* (2015).

[203] Demetri Terzopoulos. “Regularization of inverse visual problems involving discontinuities”. In: *IEEE Transactions on pattern analysis and Machine Intelligence* 4 (1986), pp. 413–424.

[204] Aleksandr Filippovich Timan. *Theory of Approximation of Functions of a Real Variable*. Vol. 34. Courier Corporation, 1994.

[205] Carlo Traverso and Alberto Zanoni. “Numerical stability and stabilization of Groebner basis computation”. In: *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*. ACM. 2002, pp. 262–269.

[206] Lloyd N Trefethen. *Approximation theory and approximation practice*. Vol. 128. Siam, 2013.

[207] Lloyd N Trefethen. “Inverse Yogiisms”. In: *Notices of the American Mathematical Society* 63.11 (2016).

[208] Martin Uecker et al. “Image reconstruction by regularized nonlinear inversionjoint estimation of coil sensitivities and image content”. In: *Magnetic Resonance in Medicine* 60.3 (2008), pp. 674–682.

[209] Michael Unser. “A representer theorem for deep neural networks”. In: *arXiv preprint arXiv:1802.09210* (2018).

[210] Alessandro Vicini and Domenico Quagliarella. “Airfoil and wing design through hybrid optimization strategies”. In: *AIAA journal* 37.5 (1999), pp. 634–641.

[211] Fei Wang et al. “Demystifying differentiable programming: Shift/reset the penultimate backpropagator”. In: *arXiv preprint arXiv:1803.10228* (2018).

[212] Martin Weinmann et al. “Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 105 (2015), pp. 286–304.

[213] Erik Wissner et al. “Noninvasive epicardial and endocardial mapping of premature ventricular contractions”. In: *Europace* 19.5 (2016), pp. 843–849.

[214] Jonathan Wray and Gary GR Green. “Neural networks, approximation theory, and finite precision computation”. In: *Neural networks* 8.1 (1995), pp. 31–37.

[215] Bichen Wu et al. “Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1887–1893.

[216] Zhirong Wu et al. “3d shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[217] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. “Wider or deeper: Revisiting the resnet model for visual recognition”. In: *Pattern Recognition* 90 (2019), pp. 119–133.

[218] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[219] Qiaofeng Xu et al. “Sparsity-regularized image reconstruction of decomposed K-edge data in spectral CT”. In: *Physics in Medicine & Biology* 59.10 (2014), N65.

[220] Yong Xu et al. “Review of video and image defogging algorithms and related studies on image restoration and enhancement”. In: *Ieee Access* 4 (2016), pp. 165–188.

[221] Chyuan-Huei Thomas Yang, Shang-Hong Lai, and Long-Wen Chang. "Hybrid image matching combining Hausdorff distance with normalized gradient matching". In: *Pattern Recognition* 40.4 (2007), pp. 1173–1181.

[222] Dmitry Yarotsky. "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94 (2017), pp. 103–114.

[223] Hengyong Yu and Ge Wang. "SART-type image reconstruction from a limited number of projections with the sparsity constraint". In: *Journal of Biomedical Imaging* 2010 (2010), p. 3.

[224] Deniz Yuret. "Knet: beginning deep learning with 100 lines of julia". In: *Machine Learning Systems Workshop at NIPS*. Vol. 2016. 2016, p. 5.

[225] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *British Machine Vision Conference 2016*. British Machine Vision Association. 2016.

[226] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 5067–5073.

[227] Chiyuan Zhang et al. "Understanding deep learning requires rethinking generalization". In: *arXiv preprint arXiv:1611.03530* (2016).

[228] Dengsheng Zhang and Guojun Lu. "Review of shape representation and description techniques". In: *Pattern recognition* 37.1 (2004), pp. 1–19.

[229] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499.

[230] Bo Zhu et al. "Image reconstruction by domain-transform manifold learning". In: *Nature* 555.7697 (2018), p. 487.