UNIVERSITY OF CALIFORNIA

Los Angeles

Learning to Optimize with Guarantees

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Howard Wayne Heaton

2021

ABSTRACT OF THE DISSERTATION

Learning to Optimize with Guarantees

by

Howard Wayne Heaton

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2021

Professor Wotao Yin, Chair

Machine learning (ML) has evolved dramatically over recent decades, from relative infancy to a practical technology with widespread commercial use. This same time period increasingly saw applications modeled as large scale optimization problems, reviving interest in first-order optimization methods (due to their low per-iteration cost and memory requirements). ML enables computers to improve automatically through experience. At first glance, this may appear to signal a pending end to hand crafted optimization modeling.Yet, optimization can provide intuition for effective ML model design. Also, some ML models cannot be considered trustworthy without guarantees (*e.g.* satisfaction provided constraints) and/or explanations of their behaviors. Design intuition from optimization led to experiments unrolling optimization algorithms into ML model architectures via what is now known as the "learn to optimize" (L2O) paradigm. L2O models generalize hand-crafted optimization for use with big data and provide promising numerical results.

This work investigates L2O theory and implementations. Chiefly, we investigate L2O convergence guarantees, training of L2O models, how to interweave prior and data-driven knowledge, and how to certify L2O inferences comply with desired specifications. Thus, our core contribution is a fusion of optimization and machine learning, merging desirable properties from each field – model intuition, guarantees, practical performance, and the ability to leverage big data. A progression of novel frameworks and theory are developed herein for L2O models, extending prior work in multiple directions.

The dissertation of Howard Wayne Heaton is approved.

Lin Yang

Deanna Hunter

Stanley Osher

Wotao Yin, Committee Chair

University of California, Los Angeles

2021

To my big mama – for always inspiring and supporting my curiosity

To my princess – for your love and holding my hand at every step of this journey

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

This work is the accumulation of guided steps from several thoughtful and encouraging mentors. First, I owe my deepest thanks to Wotao Yin and Stan Osher for their willingness to advise me on my academic journey. Back as an unknown undergraduate, Stan answered my cold email for a meeting to learn about UCLA's program, which made clear UCLA was where I wanted to be. Wotao has generously shared from his vast knowledge, bestowing wisdom on everything from the fine details of proofs to seeing the forest for the trees. Both Stan and Wotao also taught me through my graduate coursework, which I am incredibly grateful for. I also owe thanks to Luminita Vese and Deanna Needell for their instruction, incredible kindness, and time shared to meet during my UCLA journey.

I am thankful for Reinhard Schulte introducing me to the world of research as an undergraduate. He mentored me through two summers of research at Loma Linda University and my senior project at Walla Walla University. His guidance molded the base of my research interests and abilities. During the second summer, I met Yair Censor, whom I cannot thank enough for patiently teaching me the ways of iterative projection methods and mentoring me through multiple projects. Yair was an incredible influence in developing my mathematical interests and finding my research passions. Around this time, I became acquainted with Kevin Vixie, whom I must thank for his assistance with getting me into graduate school, his phenomenal advice, and his continually shared wisdom through his writings.

Last, but not least, I wish to thank my family. They have been incredibly supportive of my esoteric interests in maths since childhood. I would not be here today were it not for my mom's inspiration, her constantly "tutoring" me from elementary school to university, and her encouragement to chase my passions. And endless thanks be to my wife for her love and patience, adventuring with me to Los Angeles, and supporting us via tireless work as an ICU nurse, particularly during the Covid-19 pandemic.

Howard Heaton

Los Angeles, CA

December 9, 2021

VITA

2011 – 2016 Bachelors of Science, Walla Walla University.
Computer Science, Mathematics, Physics

2016 – 2018 Masters of Arts, University of California Los Angeles.
Mathematics

2020 Data Science Intern, Retina AI

2021 Graduate Tech Research Intern, Walt Disney Animation Studios

PUBLICATIONS

- S. Wu Fung,* **H. Heaton**,* Q. Li, D. McKenzie, S. Osher, W. Yin. *JFB: Jacobian-free Backpropagation for Implicit Networks.* Proceedings of the AAAI Conference on Artificial Intelligence, 2022.
- J. Shen,* X. Chen,* **H. Heaton**,* T. Chen, J. Liu, W. Yin, Z. Wang. *Learning a Minimax Optimizer: A Pilot Study.* International Conference on Learning Representations, 2020.
- **H. Heaton**, X. Chen, Z. Wang, W. Yin. *Safeguarded Learned Convex Optimization.* arXiv preprint:2003.01880, 2020.
- T. Chen, X. Chen, W. Chen, **H. Heaton**, J. Liu, Z. Wang, W. Yin.† *Learning to optimize: A Primer and a Benchmark*. arXiv preprint:2103.12828, 2021.
- **H. Heaton**,* S. Wu Fung,* A.T. Lin,* S. Osher, W. Yin. *Wasserstein-based Projections with Applications to Inverse Problems*. arXiv preprint:2008.02200, 2020.

- **H. Heaton**, Y. Censor. *Asynchronous sequential inertial iterations for common fixed points problems with an application to linear systems*. Journal of Global Optimization, 2019.

- Y. Censor, **H. Heaton**, R. Schulte.† *Derivative-free superiorization with component-wise perturbations*. Numerical Algorithms, 2019.

- **H. Heaton**,⋆ D. McKenzie,⋆ Q. Li, S. Wu Fung, S. Osher, W. Yin. *Learn to Predict Equilibria via Fixed Point Networks*. arXiv preprint:2106.00906, 2021.

- **H. Heaton**, S. Wu Fung, A. Gibali, W. Yin. *Feasibility-Based Fixed Point Networks.* Fixed Point Theory and Algorithms for Sciences and Engineering, 2021.

- **H. Heaton**, S. Wu Fung, S. Osher, W. Yin. *Explainable AI via Learning to Optimize*. In Preparation, 2021.

⋆ Equal contribution

† Alphabetic ordering

Chapter 1: Introduction

[A] mathematical problem should be difficult in order to entice us, yet not completely inaccessible, lest it mock at our efforts. It should be to us a guide post on the mazy paths to hidden truths, and ultimately a reminder of our pleasure in the successful solution.

– David Hilbert[1]

[1] Taken from the "Mathematical Problems" lecture delivered before the International Congress of Mathematicians in 1900 [131].

An ever growing number of applications can be modeled as large (or even huge) scale optimization problems. During the early 2000s, this revived interest in first-order optimization methods that utilize low per-iteration cost and memory storage requirements. Optimization methods were originally hand-crafted by experts, based on theories and experience. As a paradigm shift from this conventional design, *learning to optimize* (L2O) uses machine learning to either improve an existing optimization method or generate a completely new one. This thesis focuses on continuous and model-based L2O schemes, culminating in novel model designs and theoretical guarantees for L2O model inferences.

Classic optimization methods are built upon components that are basic methods — such as gradient descent, conjugate gradient, Newton steps, and proximal point — in a theoretically justified manner.[2] Most conventional optimization methods can be written in a few lines, and their performance is theoretically guaranteed. To solve an optimization problem in practice, one selects a method supporting the type of problem at hand and applies the method, expecting a returned solution no worse than the method's guarantee.

[2] This introduction is based on the L2O survey [62]. We refer the reader to this survey for a wider coverage of L2O.

L2O is an alternative paradigm that develops an optimization method by training, *i.e.* learning from its performance on sample problems. A naïve L2O method may lack a solid theoretical basis, but it improves its performance during

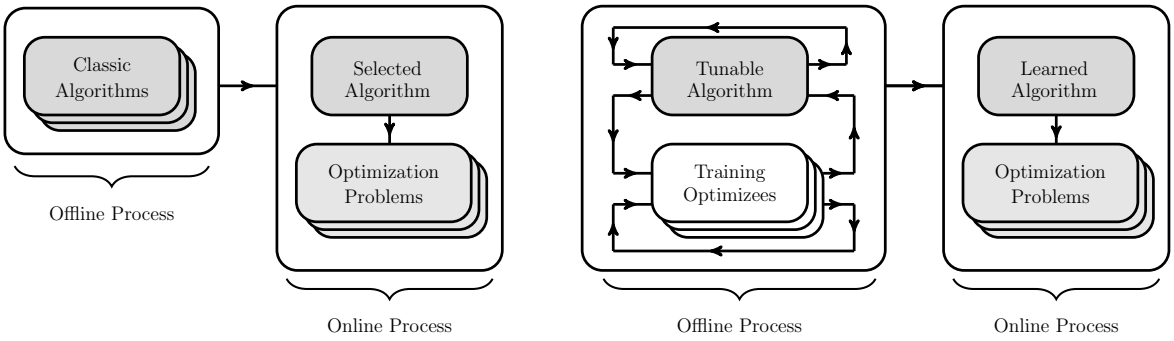


Figure 1.1: Classic optimizers are manually designed/selected, with few or no tuning parameters (left). Learned optimizers are trained offline in an L2O framework over a set of similar problems and designed to solve unseen problems from the same distribution (right).

the training process. Training often occurs offline and is time consuming. However, the online application of the method is (aimed to be) time saving. When it comes to problems where the desired solutions are difficult to obtain, such as nonconvex optimization and inverse problems, well-trained L2O methods can yield better quality inferences than classic methods.

Many applications involve the task of repeatedly solving a specific type of optimization problem over a fixed distribution of input data. Input data define optimizations that are similar, yet distinct. Conventional optimizers may be tuned for a particular distribution, but the underlying methods are designed for a theory-specified class of optimization problems. We often describe a conventional optimizer by the formulation (and its math properties), not the distribution of data.[3] In L2O, the training process shapes the optimizer according to both the formulation *and* the distribution of data. When the distribution is concentrated, the learned optimizer can "overfit" to the tasks and may discover "short cuts" that classic optimizers do not take.

[3] For example, we say an optimizer can minimize a *smooth-and-convex* objective function subject to *linear* constraints.

Two topics are core to the developments of this thesis. The first is operator-based optimization methods. The second is implicit ML models.[4] Before diving into L2O concepts, a brief overview is provided for each of topic. This chapter concludes by summarizing the contributions of works used to form this thesis.

[4] We henceforth drop "ML" and simply write models, particularly as this will align in Part II with the notion of optimization models.

Section 1.1: Optimization via Fixed Point Methods

This section overviews tools used by fixed point methods, which form an abstraction of many first-order optimization methods.[5] The set of fixed points of each operator[6] $T : \mathbb{H} \to \mathbb{H}$ is denoted by $\operatorname{fix}(T) \triangleq \{x : Tx = x\}$. For T with a nonempty fixed point set (i.e., $\operatorname{fix}(T) \neq \varnothing$), consider the fixed point problem

$$\text{Find } \overline{x} \text{ such that } \overline{x} \in \operatorname{fix}(T). \tag{1.1}$$

[5] Methods not covered by this section include conditional gradient (*a.k.a.* Frank-Wolfe) and conjugate gradient. L2O methods with these analytic counterparts are outside the scope of this thesis.

Many convex minimization problems, both constrained and unconstrained, may be equivalently rewritten as (1.1) for an appropriate operator T (*e.g.* see Table 1.1). We focus on fixed point formulations to provide a general approach for creating sequences that converge to solutions of (1.1) and, thus, of the corresponding optimization problem.

[6] Here $\mathbb{H}$ denotes a Hilbert space, typically the Euclidean space $\mathbb{R}^n$.

We begin with basic definitions and a classic convergence result. An operator $T : \mathbb{H} \to \mathbb{H}$ is *nonexpansive* if it is 1-Lipschitz, *i.e.*[7]

$$\|T(x) - T(y)\| \leq \|x - y\|, \quad \text{for all } x, y \in \mathbb{H}. \tag{1.2}$$

[7] Let $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ respectively be the Euclidean inner product and norm defined on $\mathbb{H}$.

An operator T is *averaged* if there is $\alpha \in (0,1)$ and nonexpansive $Q : \mathbb{H} \to \mathbb{H}$ such that $T = (1-\alpha)\mathrm{I} + \alpha Q$, with I the identity. If T is averaged with $\alpha = 1/2$, then T is *firmly nonexpansive.* An operator $A : \mathbb{H} \to \mathbb{H}$ is *monotone* provided[8]

$$\langle A(x) - A(y), x - y \rangle \geq 0, \quad \text{for all } x, y \in \mathbb{H}. \tag{1.3}$$

[8] Here A is single-valued; however, this notion naturally generalizes to set-valued operators. Commonplace examples of this include subgradients of convex functions. We refer readers to the textbook [213] for more on set-valued monotone operators.

The *resolvent* $J_{\alpha A}$ of A with parameter $\alpha > 0$ is defined by

$$J_{\alpha A} \triangleq (\mathrm{I} + \alpha A)^{-1}, \tag{1.4}$$

and the corresponding *reflected resolvent* $R_{\alpha A}$ is defined by

$$R_{\alpha A} \triangleq 2J_{\alpha A} - \mathrm{I}. \tag{1.5}$$

The operator $R_{\alpha A}$ is nonexpansive and $J_{\alpha A}$ is averaged. We refer readers to the texts [46, 27, 213] for further background on averaged operators.

A classic theorem states sequences generated by successively applying an averaged operator converge to a fixed point, *i.e.* a solution of (1.1). This method comes from [148, 172], yielding adoption of the name *Krasnosel'skiĭ-Mann* (KM) method. Their result is stated below and is known with various forms and proofs (*e.g.* see [25, Thm. 5.14], [38, Thm. 5.2], [45, Thm. 3.5.4], and [202, Thm. 2]).

Theorem 1.1.1 *If an averaged operator $T : \mathbb{H} \to \mathbb{H}$ has a nonempty fixed point set and a sequence $\{x^k\}$ with arbitrary initial iterate $x^1 \in \mathbb{H}$ satisfies the update relation*

$$x^{k+1} = T(x^k), \quad \text{for all } k \in \mathbb{N}, \tag{1.6}$$

then there is $x^\infty \in \operatorname{fix}(T)$ such that the sequence $\{x^k\}$ converges to x^∞, i.e. $x^k \to x^\infty$.

To make this theorem concrete, we provide explicit examples of well-known methods. To this end, we introduce a few more definitions. A function[9] $f : \mathbb{H} \to \overline{\mathbb{R}}$ is *proper* if its value is never $-\infty$ and is finite somewhere. A proper function f is *closed* if and only if f is lower semicontinuous. And, f is convex provided

$$f\big(\lambda x + (1-\lambda y)\big) \le \lambda f(x) + (1-\lambda) f(y), \quad \text{for all } x, y \in \operatorname{dom}(f) \text{ and } \lambda \in [0,1]. \tag{1.7}$$

[9] Throughout we use the notation $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{\infty\}$.

If f is closed, convex, and proper (CCP), then the proximal operator is well-defined[10] and averaged. For a scalar $\alpha > 0$, the proximal of a CCP function f is a resolvent that is the solution to a minimization problem, *i.e.*

$$\operatorname{prox}_{\alpha f}(x) \triangleq J_{\alpha \partial f}(x) = \operatorname*{arg\,min}_{z \in \mathbb{H}} f(z) + \frac{1}{2\alpha}\|z - x\|^2, \tag{1.8}$$

[10] The definition of the proximal yields a (posisbly empty) set of minimizers. The added CCP assumptions ensure the set consists of a unique element.

where $\partial f(x) \triangleq \{g : f(x) + \langle g, y - x\rangle \le f(y) \ \forall\ y \in \mathbb{H}\}$ is the subgradient of f. Proximal operators for several well-known functions can be expressed by explicit formulas (*e.g.* see page 177 in [31]). When the proximal admits a formula that is computationally cheap to evaluate, we (informally) say f is *proximable*. The fixed points of the proximal coincide with minimizers of f. Hence, by Theorem 1.1.1,

iteratively applying the proximal yields a sequence converging to a minimizer of f, known as the proximal point method. That is, if $x^1 \in \mathbb{H}$, f is CCP, $\alpha > 0$, and

$$x^{k+1} = \operatorname{prox}_{\alpha f}(x^k), \quad \text{for all } k \in \mathbb{N}, \tag{1.9}$$

then the sequence $\{x^k\}$ converges to a minimizer of f.

To illustrate use of operator splitting, we conclude with an example. Consider the minimization the sum of two CCP functions $f : \mathbb{H} \to \overline{\mathbb{R}}$ and $g : \mathbb{H} \to \overline{\mathbb{R}}$, *i.e.*

$$\min_{x \in \mathbb{R}^n} f(x) + g(x). \tag{1.10}$$

For $\alpha > 0$, by rearranging the optimality condition, each optimizer $\overline{x} \in \mathbb{H}$ satisfies[11]

$$\begin{aligned}
0 \in \partial f(\overline{x}) + \partial g(\overline{x}) &\iff 0 \in (\mathrm{I} + \alpha\partial f)(\overline{x}) - (\mathrm{I} - \alpha\partial g)(\overline{x}) && \text{(1.11a)}\\
&\iff 0 \in (\mathrm{I} + \alpha\partial f)(\overline{x}) - (2J_{\alpha\partial g} - \mathrm{I})(\mathrm{I} + \alpha\partial g)(\overline{x}) && \text{(1.11b)}\\
&\iff 0 \in (\mathrm{I} + \alpha\partial f)(\overline{x}) - R_{\alpha\partial g}(\overline{z}), \ \ \overline{z} \in (\mathrm{I} + \alpha\partial g)(\overline{x}) && \text{(1.11c)}\\
&\iff R_{\alpha\partial g}(\overline{z}) \in (\mathrm{I} + \alpha\partial f) J_{\alpha\partial g}(\overline{z}), \ \ \overline{x} = J_{\alpha\partial g}(\overline{z}) && \text{(1.11d)}\\
&\iff (J_{\alpha\partial f} \circ R_{\alpha\partial g})(\overline{z}) = J_{\alpha\partial g}(\overline{z}), \ \ \overline{x} = J_{\alpha\partial g}(\overline{z}) && \text{(1.11e)}\\
&\iff (R_{\alpha\partial f} \circ R_{\alpha\partial g})(\overline{z}) = \overline{z}, \ \ \overline{x} = J_{\alpha\partial g}(\overline{z}) && \text{(1.11f)}\\
&\iff \frac{1}{2}\left(\mathrm{I} + (R_{\alpha\partial f} \circ R_{\alpha\partial g})\right)(\overline{z}) = \overline{z}, \ \ \overline{x} = J_{\alpha\partial g}(\overline{z}). && \text{(1.11g)}
\end{aligned}$$

[11] If f has L-Lipschitz gradient, then the subgradient coincides with the gradient and the optimality condition in (1.11a) becomes $(\mathrm{I} - \alpha\nabla f)(x) \in (\mathrm{I} + \alpha\partial g)(x)$, and so $\overline{x} = J_{\alpha\partial g} \circ (\mathrm{I} - \alpha\nabla f)(\overline{x})$. Thus, $\overline{x}$ is a fixed point of the projected gradient operator. All sorts of variations of operator splitting derivations can be used to obtain different fixed point operators.

The composition $R_{\alpha\partial f} \circ R_{\alpha\partial g}$ of nonexpansive operators $R_{\alpha\partial f}$ and $R_{\alpha\partial g}$ is itself nonexpansive. Consequently, the fixed point operator on the left side of (1.11g) is averaged and can be used to find a fixed point $\overline{z}$. Having this fixed point, an optimizer $\overline{x}$ is directly obtained by applying the proximal $J_{\alpha\partial g}$. This splitting is known as Douglas Rachford splitting. Below Table 1.1 provides examples of several averaged operators arising from splitting schemes.

Problem	Method	Operator T	Assumption
$\min\ f(x)$	Gradient Descent	$\mathrm{Id} - \alpha\nabla f$	$\alpha < 2/L$
$\min\ f(x)$	Proximal Point	$\mathrm{prox}_{\alpha f}$	
$\min\ f(x)$ s.t. $x \in C$	Projected Gradient	$\mathrm{proj}_C \circ (\mathrm{Id} - \alpha\nabla f)$	$\alpha < 2/L$
$\min\ f(x) + g(x)$	Proximal Gradient	$\mathrm{prox}_{\alpha g} \circ (\mathrm{Id} - \alpha\nabla f)$	$\alpha < 2/L$
$\min f(x) + g(x)$	Douglas-Rachford	$\frac{1}{2}\left(\mathrm{Id} + R_{\alpha\partial f} \circ R_{\alpha\partial g}\right)$	
$\min_{(x,z)\in\Omega} f(x) + g(z)$	ADMM	$\frac{1}{2}\left(\mathrm{Id} + Q_1 \circ Q_2\right)$	$\Omega \triangleq \{(x,z) : Ax + Bz = b\}$ $Q_1 \triangleq R_{\alpha A\partial f^*(A^\top\cdot)}$ $Q_2 \triangleq R_{\alpha(B\partial g^*(B^\top\cdot)-b)}$
$\min f(x)$ s.t. $Ax = b$	Proximal Method of Multipliers	$(\mathrm{I} + \alpha\partial\mathcal{L})^{-1}$	
$\min f(x)$ s.t. $Ax = b$	Uzawa	$\mathrm{I} + \alpha\left(A\nabla f^*(-A^\top\cdot) - b\right)$	
$\min f(x) + g(Ax)$	PDHG	$(\mathrm{I} + M^{-1}\partial\mathcal{L})^{-1}$	$M = \begin{bmatrix} \alpha^{-1}\mathrm{I} & A^\top \\ -A & \beta^{-1}\mathrm{I} \end{bmatrix}$

Table 1.1: Averaged operators for well-known methods. We assume $\alpha > 0$ and L is the Lipschitz constant for shown gradients, and $\mathcal{L}$ is the Lagrangian associated with the presented problem. Here f^* denotes the convex conjugate of f. We refer the reader to [213] for a more comprehensive reference on convex conjugates and operator splitting.

Section 1.2: Implicit Models

A new direction emerged in deep learning, moving from explicit to implicit models[12] [239, 18, 19, 61, 101, 82, 135, 252, 151, 203, 165, 111]. Standard feedforward models explicitly prescribe a series of computations that map input data d to an inference y. On the other hand, each implicit model $\mathcal{N}_\Theta$ is defined via an equation. Commonly, this takes the form of a fixed point equation,[13] *i.e.*

$$\mathcal{N}_\Theta(d) \triangleq x_d, \quad \text{where} \quad x_d = T_\Theta(x_d; d), \tag{1.12}$$

where x_d is the unique fixed point of an operator $T_\Theta(\cdot; d)$ parameterized by weights Θ.[14] Throughout this work, the choice of T_Θ typically coincides with operators used to solve optimization problems (*e.g.* those in Table 1.1). However, their use can be more abstract (*e.g.* see the models in Chapter 3). Several core questions must be answered for this nascent class of models:

- Is the model $\mathcal{N}_\Theta$ well-defined for each input d?
- Are implicit models expressible, *i.e.* are they universal approximators?
- How are inferences $\mathcal{N}_\Theta(d)$ computed?
- How are the weights Θ tuned during training?
- What is the intuition behind various choices of T_Θ?
- What sorts of inference guarantees result from implicit models?
- How well do implicit models perform relative to their explicit counterparts?

Part II addresses variations of each inquiry. The short answers are yes, yes, fixed point iteration, Jacobian-Free Backpropagation, optimization-based modeling, constraint guarantees[15] and sharing similar features to training data,[16] and it depends on the setting (sometimes better, sometimes worse).

[12] Throughout this thesis, our use of the word *model* is synonymous with *neural network*.

[13] Later chapters will provide alternative formulations where $\mathcal{N}_\Theta$ is defined as a unique solution to a feasibility problem, a Nash equilibrium, or a minimizer to an optimization problem.

[14] That is, we assume $\text{fix}(T_\Theta(\cdot; d)) = \{x_d\}$.

[15] Constraints may include large systems of linear inequalities or other nontrivial set intersections.

[16] Data-driven regularization can be used to implicitly learn an "ideal" regularizer that distinguishes between inferences from inside or outside the training data.

Why Implicit Models?

Below we cover advantages of implicit models over feedforward models.

Implicit models for implicitly defined outputs. In some applications, model outputs are most aptly described implicitly as fixed points, not via an explicit function. As a toy example, consider predicting the variable $y \in \mathbb{R}$ given $d \in [-1/2, 1/2]$ when (d, y) is known to solve $y = d + y^5$. Using $y_1 = 0$ and the iteration

$$y_{k+1} = T(y_k; d) \triangleq d + y_k^5, \quad \text{for all } k \in \mathbb{N}, \tag{1.13}$$

one obtains convergence, *i.e.* $y_k \to y$. In this setting, y is exactly (and implicitly) characterized by the equation $y = T(y, d)$. On the other hand, an explicit solution to $y = d + y^5$ requires an infinite series representation,[17] unlike the simple formula $T(y, d) = d + y^5$. This illustrates it can be simpler and more appropriate to model a relationship implicitly. Later we illustrate this in the areas of convex feasibility, game theory and inverse problems. In these instances, inferences may naturally be characterized as fixed points of an operator parameterized by input data d.

[17] See Appendix F in [97] for further details.

"Infinite depth" with constant memory training. Solving for the fixed point of $T_\Theta(\cdot; d)$ is analogous to a forward pass through an "infinite depth" (in practice, very deep) weight-tied, input injected feedforward model. Yet, implicit models do not need to store intermediate quantities of the forward pass for backpropagation. Thus, implicit models can be trained using *constant memory costs* with respect to "depth," relieving a major bottleneck of training deep models.

No loss of expressiveness. Implicit models as defined in (1.12) are at least as expressive as feedforward models. More interestingly, the class of implicit models in which T_Θ is constrained to be an affine map and a single nonlinearity contains all feedforward models, and is, thus, at least as expressive [101], [18, Theorem 3]. Universal approximation properties of implicit models then follow directly from properties of conventional deep models (*e.g.* see [69, 169, 140]).

Section 1.3: Contributions

Large scale optimization-based problems can be tackled by using a distribution of training data to learn to optimize. For settings where exact optimization problem formulations are known, we provide two novel safeguard frameworks for steering L2O models toward convergence. The second class of problems considered models inferences as solutions to equations. For these, we present and theoretically justify an easy-to-implement scheme for backpropagating to train implicit models, which opens the door to practical implementation of several implicit model architectures. We provide examples of such architectures for convex feasibility problems, computing Nash equilibria (*e.g.* in games), and other L2O tasks. In each case, our numerical results are promising and implementations can have associated theoretical justifications. We conclude by showing these tools can be used to obtain novel and concrete notions of explainability and trustworthiness.

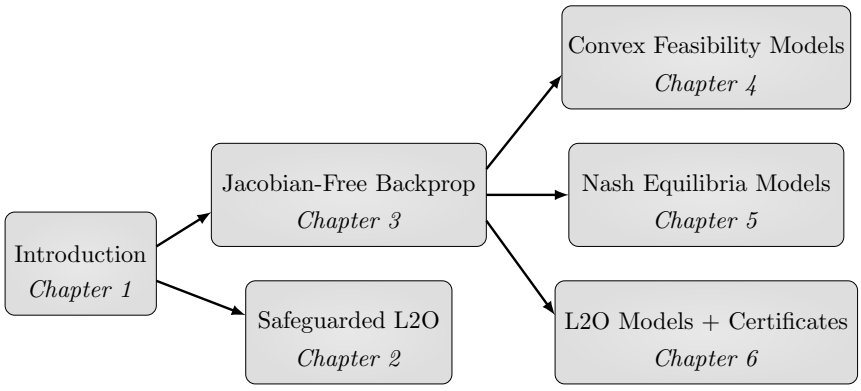


Figure 1.2: Graph for chapter dependencies. The introduction provides high-level L2O background used in all chapters. Chapter 3 establishes a backprop technique used in all subsequent parts of the thesis while the safeguarded L2O (Chapter 2) is self-contained and restricted to solving analytic optimization problems quickly. The three applications (Chapters 4, 5, and 6) can be considered independently of each other.

Part I: Deep Unrolling for Classic Optimization

Chapter 2: Safeguarded L2O

Being old and lame of my hands, and thereby uncapable of assisting my fellow citizens, when their houses are on fire; I must beg them to take in good part the following hints on the subject of fires. In the first place, as an ounce of prevention is worth a pound of cure, I would advise 'em to take care...

– Benjamin Franklin[1]

[1] Taken from Franklin's 1735 article "On Protection of Towns from Fire" in *The Pennsylvania Gazette* [92].

Solving scientific computing problems often requires application of efficient and scalable optimization algorithms. Data-driven algorithms can execute in much fewer iterations and with similar cost per iteration as state-of-the-art general purpose algorithms.[2] Inspired by one such algorithm, [112] proposed treating the entries in fixed matrices/vectors of the algorithm as learnable parameters that can vary by iteration. These entries were fine-tuned to obtain optimal performance on a data set for a fixed number of iterations. Empirically, this approach converged and showed roughly a 20-fold reduction in computational cost compared to the original algorithm. Several related works followed, also demonstrating numerical success.[3] These efforts opened the door to a new class of algorithms and analyses. Analytic optimization results often provide worst-case convergence rates, and limited theory exists pertaining to instances drawn from a common distribution (*e.g.* data supported on a low-dimensional manifold). That is, most L2O methods have little or no convergence guarantees, especially on data *distinct* from what is seen in training. Applying Franklin's wisdom, we wish to balance the desires to use data-driven algorithms and provide convergence guarantees. This chapter addresses this matter by answering the question:

[2] This chapter is primarily based on [127].

[3] We refer the reader to the L2O survey [62] for discussion of more related work.

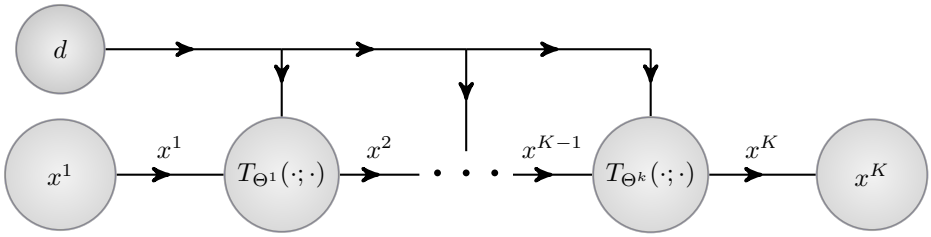


Figure 2.1: A common approach in various L2O schemes is to form feed forward networks by unrolling an iterative algorithm, truncated to K iterations, and tuning parameters at each layer/iteration k. This generalizes the update formula $x^{k+1} = T(x^k; d)$ to include a dependence on weights θ^k, denoted by a subscript. Image adapted from [62].

Can a safeguard be added to L2O algorithms to improve robustness and convergence guarantees without significantly hindering performance?

Here a safeguard is anything that identifies when a "bad" L2O update would occur and what to do in place of that "bad" update. We provide an affirmative answer to the question for convex problems with gradient and/or proximal oracles by providing such a safeguard and replacing "bad" L2O updates with updates from analytic methods. Our framework is called Safe-L2O. Since a trade-off is formed between per iteration costs and ensuring convergence, we clarify three properties of "practical" L2O safeguards.

1. Safeguards must only leverage known quantities related to convex problems (*e.g.* objective values, norms of gradients, distance between iterates).

2. Both L2O and Safe-L2O schemes should perform identically on "good" data, with comparable per-iteration costs.

3. Safeguards should apply only if "bad" L2O updates would otherwise occur.

The challenge is to create a simple safeguard that kicks in only when needed. Unlike classic optimization algorithms, exceptional L2O algorithms do *not* necessarily exhibit the behavior that each successive iterate is "better" than the current iterate (*i.e.* are not monotonically improving). Loosely speaking, this means there are cases where an L2O scheme that gets "worse" for a couple iterates yields a

better final output than an L2O scheme that is required to get "better" at each iterate. The intuition behind why this should be acceptable is that we are solely interested in the final output of the L2O algorithm, not the intermediate steps. From this insight, we deduce the safeguard should exhibit a form of trailing behavior, *i.e.* it should measure progress of previous iterates and only require that updates are "good" on average. If the safeguard follows too closely, then the Safe-L2O scheme's flexibility and performance are limited. If it follows from too far, then the Safe-L2O scheme may exhibit highly oscillatory behavior and converge too slowly. The appropriate amount for the safeguard to follow can be estimated by tuning L2O parameters for optimal performance on a training set *without* safeguarding and then using a validation set to test various safeguards with the L2O scheme. To avoid possible confusions, note *we are not trying to prove the convergence of any standalone L2O algorithm*. We instead 1) alarm on an L2O update when it may break convergence, 2) replace it with a fall-back update, and 3) show the resulting "hybrid optimization" converges.[4]

In addition to L2O updates, our method uses a safeguard condition with the update formula from a conventional algorithm. When the "good" condition holds, the L2O update is used; when it fails, the formula from the conventional algorithm is used. In the ideal case, L2O updates are applied often and the conventional algorithm formula provides a "fallback" for exceptional cases. This fallback is designed together with the safeguard condition to ensure convergence. This also implies, even when an L2O algorithm has a fixed number of iterations with tunable parameters, the algorithm may be extended to an arbitrary number of iterations by applying the fallback to compute latter updates (see Figure 2.2).

[4] Loosely speaking, our scheme updates via

$$x^{k+1} = \begin{cases} \text{L2O if "good"} \\ \text{Backup if "bad".} \end{cases}$$

Related L2O Methods. A seminal L2O work in the context of sparse coding was by [112]. Numerous follow-up papers also demonstrated empirical success at constructing rapid regressors approximating iterative sparse solvers, for compression, nonnegative matrix factorization, compressive sensing and other applications [221, 233, 234, 130, 247]. The majority of L2O works pertain to sparse coding and provide limited theoretical results. Some works have interpreted LISTA in various ways to provide proofs of different convergence properties [104, 179]. Others have investigated structures related to LISTA [242, 34, 35, 174], providing results varying by assumptions. [63] introduced necessary conditions for the LISTA weight structure to asymptotically achieve a linear convergence rate. This was followed by [164], which further simplified the weight conditions and provided a result stating that, with high probability, the convergence rate of LISTA is at most linear. The mentioned results are useful, yet can require intricate assumptions and proofs specific to sparse coding problems.

Our safeguarding scheme is related to existing Krasnosel'skiĭ-Mann (KM) methods. Indeed, [223] presents a KM safeguard method in a more hierarchical manner than ours; it differs from this chapter by solely refers to the current iterate residuals (plus a summable sequence). Additionally, [251] uses a similar safeguarding step for their Anderson accelerated KM method. To the best of our knowledge, neither method has been employed with L2O.

Contribution. We provide two simple Safe-L2O frameworks for creating data-driven algorithms with convergence guarantees. These framework can be used with all L2O algorithms that solve convex problems[5] for which proximal and/or gradient oracles are available. We incorporate several safeguarding procedures in a general setting and present a simple procedure for utilizing ML methods to instill knowledge from available data.

[5] They also apply to convex-convcave saddle point problems (*e.g.* matrix games).

Algorithm 1: L2O Model formed by deep unrolling without a safeguard.

$\mathcal{N}_\Theta(d)$:
  $x^1 \leftarrow \tilde{x}$ ◁ Initialize inference
  for $k = 1, 2, \ldots, K$
    $x^{k+1} \leftarrow T_{\Theta^k}(x^k; d)$ ◁ L2O Update
    $k \leftarrow k + 1$ ◁ Increment counter
  return x^k ◁ Output inference

Section 2.1: Safeguarded L2O via Fixed Point Residual

This section presents the Safe-L2O framework. We emphasize this safeguarding acts as a wrapper around a data-driven algorithm.[6] Each L2O operator T_Θ is parameterized by weights Θ. Each vector d of input data (*e.g.* the measurement vector in a least squares problem) defines an optimization problem. For each application of the algorithm, the fallback operator changes, depending upon the data d. To make explicit this dependence, we include an entry d via $T(\cdot; d)$. Often T_Θ can be viewed as forming one or multiple layers of a feed forward model. Thus, $\mathcal{N}_\Theta(d)$ in Algorithm 1 is precisely a feed forward model. In addition to an L2O operator T_Θ, our Safe-L2O method uses a fallback operator T and a scalar sequence $\{\mu_k\}$. Here $T(\cdot; d)$ defines an averaged operator from the update formula of a conventional optimization algorithm (see Table 1.1). Each μ_k defines a reference value to determine whether a tentative L2O update is "good." Each reference value μ_k in our safeguarding is relatable to a combination of $\|x^i - T(x^i)\|$ among previous iterates $i = 1, \ldots, k$. We illustrate L2O and fallback operators for data-driven step sizes in an example below.

[6] In this chapter, we assume each operator T is averaged.

Table 2.1: Choices to update μ_k that ensure Assumption 2.1.2 holds. Here $\alpha, \theta \in (0,1)$, C_k is the statement that $\|x^{k+1} - T(x^{k+1}; d)\| \leq \alpha\mu_k$, and we say a residual $\|x^n - T(x^n; d)\|$ is "good" if C_{n-1} holds.

Name	Update Formula
Geometric Sequence GS(θ)	$\mu_{k+1} = \begin{cases} \theta\mu_k & \text{if } C_k \text{ holds,} \\ \mu_k & \text{otherwise.} \end{cases}$ *Decrease μ_k by factor θ for "good" residuals.*
Recent Term RT	$\mu_{k+1} = \begin{cases} \|x^{k+1} - T(x^{k+1}; d)\| & \text{if } C_k \text{ holds,} \\ \mu_k & \text{otherwise.} \end{cases}$ *Take μ_k to be most recent "good" residual.*
Arithmetic Average AA	$m_{k+1} = \begin{cases} m_k + 1 & \text{if } C_k \text{ holds,} \\ m_k & \text{otherwise.} \end{cases}$ $\mu_{k+1} = \begin{cases} \dfrac{\|x^{k+1} - T(x^{k+1}; d)\| + m_k\mu_k}{m_{k+1}} & \text{if } C_k \text{ holds,} \\ \mu_k & \text{otherwise.} \end{cases}$ *Take μ_k to be average among "good" residuals.*
Exponential Moving Average EMA(θ)	$\mu_{k+1} = \begin{cases} \theta\|x^{k+1} - T(x^{k+1}; d)\| + (1-\theta)\mu_{k-1} & \text{if } C_k \text{ holds,} \\ \mu_k & \text{otherwise.} \end{cases}$ *Exponentially average μ_k with the latest "good" residuals.*
Recent Max RM(m)	$\Xi_k = \{\text{most recent } m \text{ indices } \ell : C_{\ell-1} \text{ holds}\}$ $\mu_{k+1} = \max_{\ell \in \Xi_k} \|x^\ell - T(x^\ell; d)\|$ *Take μ_k to be max of most recent "good" residuals.*

Remark 2.1.1 *Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be convex and differentiable with L-Lipschitz gradient with respect to the first argument. For $\Theta \in [0, \infty)$, define the L2O operator $T_\Theta : \mathbb{R}^n \to \mathbb{R}^n$ by*

$$T_\Theta(x; d) \triangleq x - \frac{2\Theta}{L} \cdot \nabla_x f(x; d). \tag{2.1}$$

Here T_Θ is a gradient descent operator with tunable step-size $2\Theta/L$, which is is averaged for $\Theta \in (0,1)$. The fallback operator T can be set to the gradient descent update with step size $1/L$. Although using $\Theta \gg 1$ may not be theoretically justifiable for L2O updates, this can be useful in accelerating the convergence of a method in some instances [105]. $\triangle$

Algorithm 2: Safe-L2O via fixed point residual decrease

1: $\mathcal{N}_\Theta(d, T, \alpha)$
2: $x^1 \leftarrow \tilde{x}$ ◁ Initialize inference
3: $\mu^1 \leftarrow \|\tilde{x} - T(\tilde{x}; d)\|$ ◁ Initialize safeguard
4: $k \leftarrow 1$ ◁ Initialize counter
5: **while** $\|x^{k+1} - x^k\| > \varepsilon$ ◁ Loop to fixed point
6: $y^{k+1} \leftarrow T_\Theta(x^k; d)$ ◁ L2O Prediction
7: **if** $\|T(y^k; d) - y^k\| \leq \alpha\mu_k$ ◁ Safeguard check
8: $x^{k+1} \leftarrow y^{k+1}$ ◁ L2O Update
9: **else**
10: $x^{k+1} \leftarrow T(x^k; d)$ ◁ Fallback KM Update
11: Update safeguard μ_{k+1} ◁ See Table 1
12: $k \leftarrow k + 1$ ◁ Increment counter
13: **return** x^k ◁ Output inference

Our first proposed Safe-L2O approach is Algorithm 2. For each iteration k, the weights Θ^k are used to define the L2O update $T_{\Theta^k}(\cdot; d)$. A fallback operator T is chosen in Line 3 (*e.g.* using Table 1.1). Each safeguard parameter μ_k is chosen in Line 10 (*e.g.* using Table 2.1). First the initial iterate x^1 is assigned in Line 2 to a fixed reference $\tilde{x}$ and similarly for the safeguard parameter μ^1 in Line 3. From Line 5 to Line 12, a repeated loop occurs[7] to compute each update x^{k+1}. In Line 6 the L2O operator is applied to the current iterate x^k get a tentative update y^{k+1}. This y^{k+1} is then determined to be "good" if the the inequality in Line 7 holds. In such a case, the L2O update is assigned to x^{k+1} in Line 8. Otherwise, the fallback operator T is used to obtain the update x^{k+1} in Line 10. Lastly, the safeguard parameter is updated in Line 11 (*e.g.* using Table 1.1).

[7] In practice, this loops is often fixed to be some number K iterations rather than a satisfy the condition that successive iterates are "close."

To justify applying Algorithm 2, we use the following standard assumption.

Assumption 2.1.1 *The optimization problem has a solution and is equipped with an operator T such that i)* $\mathrm{fix}(T(\cdot; d))$ *equals the solution set and ii) $T(\cdot; d)$ is averaged.*

The next assumption ensures the used L2O updates approach the solution set, which is accomplished using the fixed point residual with the fallback operator.

Assumption 2.1.2 *The safeguard sequence $\{\mu_k\}$ is monotonically decreasing such that*

$$\|T(x^k; d) - x^k\| \leq \mu_k, \quad \textit{for all } k \in \mathbb{N}, \tag{2.2}$$

and there exists $\zeta \in (0, 1)$ such that

$$\mu_{k+1} \leq \zeta \mu_k, \quad \textit{whenever } x^{k+1} \textit{ is an L2O update.} \tag{2.3}$$

Our final assumption ties the L2O updates to the fallback operator. The assumption effectively states the residual of the L2O operator is bounded by a multiple of the residual of the fallback operator, *i.e.* T_Θ "looks similar" to T.[8]

[8] The reasonableness of this assumption comes from the fact T_Θ is heavily inspired by T and may even equal T for a specific choice of Θ.

Assumption 2.1.3 *There exists a bounded sequence $\{\tau_k\} \subset [0, \infty)$ such that*

$$\|T_\Theta(x^k; d) - x^k\| \leq \tau_k \|T(x^k; d) - x^k\|, \quad \textit{for all } k \in \mathbb{N}. \tag{2.4}$$

Our proposed methods (see Table 2.1) for choosing the sequence $\{\mu_k\}$ satisfy Assumption 2.1.2. These methods are *adaptive* in the sense that each update to μ_k depends upon the iterate x^k and (possibly) previous iterates. Each safeguard parameter μ_k also remains constant in k except for when the residual norm $\|x^{k+1} - T(x^{k+1}; d)\|$ decreases to less than a geometric factor of μ_k. This allows each μ_k to trail the value of the residual norm $\|x^k - T(x^k; d)\|$ and allows the residual norm to increase in k from time to time. As noted in the chapter introduction, this trailing behavior provides flexibility to the L2O updates. Our main convergence result is below and it is followed by a corollary justifying use of the schemes in Table 2.1 (both proven in the appendices).

Theorem 2.1.1 *If $\{x^k\}$ is a sequence generated by the repeated loop in the Safe-L2O (Algorithm 2) and Assumptions 2.1.1, 2.1.2, and 2.1.3 hold, then $\{x^k\}$ converges to a limit $x_d \in \text{fix}(T(\cdot; d))$, i.e. $x^k \to x_d$.*

Corollary 2.1.1 *If $\{x^k\}$ is a sequence generated by the Safe-L2O method (Algorithm 2) and Assumptions 2.1.1 and 2.1.3 hold, and $\{\mu_k\}$ is generated using a scheme outlined in Table 2.1, then Assumption 2.1.2 holds and, by Theorem 2.1.1, there is $x_d \in \mathrm{fix}(T(\cdot; d))$ such that $\{x^k\}$ converges to x_d, i.e. $x^k \to x_d$.*

We summarize the safeguard schemes as follows. The GS method decreases μ_k be a fixed geometric factor at each update. The AA method averages all past and current residuals where μ_k is/was modified. The EMA method performs similarly to AA, but using a moving average. The RM method sets μ_k to be the largest residual among sufficiently recent residuals. The RM and GS methods can make μ_k lag far behind the current residual. In our experience, the EMA is the most practical safeguard due to its adaptive and limited memory nature.

Section 2.2: Safeguarded L2O via Energy

This section introduces an alternative safeguard.[9] The results here assume T is firmly nonexpansive (*e.g.* a proximal update). Define the residual operator

$$F(x; d) \triangleq \frac{1}{2}\big(x - T(x; d)\big) \tag{2.5}$$

and, for each $k \in \mathbb{N}$, the energy $E_k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ by

$$E_k(x; d) \triangleq \|F(x; d)\|^2 - \frac{\lambda_k}{1 - \lambda_k} \left\langle F(x; d), x^1 - x \right\rangle, \tag{2.6}$$

where $\{\lambda_k\}$ is a sequence of step sizes given by $\lambda_k = 1/(k+1)$.

[9] We call this a safeguard via energy while, in reality, both safeguards may be viewed as utilizing energy dissipation. We use this naming solely to demarcate the approaches.

The second proposed Safe-L2O scheme is in Algorithm 3. A reference iterate $\tilde{x}$ can be understood to be the initialization x^1. Line 2 sets x^2 to be the average of the initialization and its image under T. Lines 4 to 10 yield a repeated loop. A tentative L2O update y^{k+1} is compute in Line 5. If this satisfies the check in Line 6, then it is "good" and used to update x^k in Line 7. Otherwise, the fallback method is a Halpern iteration [118], which is used to update in Line 9.

Algorithm 3: Safe-L2O via Energy Decrease

1: $\mathcal{N}_\Theta(d, T, C)$
2: $x^2 \leftarrow \frac{1}{2}(\tilde{x} + T(\tilde{x}; d))$ ◁ Initialize using reference $\tilde{x}$
3: $k \leftarrow 2$ ◁ Initialize counter
4: **while** $\|x^k - T(x^k; d)\| > \varepsilon$ ◁ Loop to fixed point
5: $y^{k+1} \leftarrow T_\Theta(x^k; d)$ ◁ L2O Prediction
6: **if** $E_{k+1}(y^{k+1}; d) \leq \frac{C}{k+1}$ ◁ Safeguard energy check
7: $x^{k+1} \leftarrow y^{k+1}$ ◁ L2O Update
8: **else**
9: $x^{k+1} \leftarrow \lambda_k \tilde{x} + (1 - \lambda_k) T(x^k; d)$ ◁ Fallback Update
10: $k \leftarrow k + 1$ ◁ Increment counter
11: **return** x^k ◁ Output inference

Our next result for safeguarding is stated below (proven in the appendices).

Theorem 2.2.1 *If the sequence $\{x^k\}$ is generated by the iteration in Algorithm 3 with firmly nonexpansive $T(\cdot; d)$ that has a fixed point and $\lambda_k = 1/(k+1)$, then*

$$\|x^k - T(x^k; d)\| \leq \frac{1}{2}\left(\frac{d_1}{k} + \sqrt{\frac{d_1^2}{k^2} + \frac{4C}{k}}\right), \quad \text{for all } k \geq 2, \tag{2.7}$$

where $d_1 \triangleq \min\{\|\tilde{x} - x\| : x \in \operatorname{fix}(T(\cdot; d))\}$ is the distance between the reference iterate $\tilde{x}$ and the set of fixed points and $C \geq 0$ is an arbitrary constant. In particular, this implies each limit point of $\{x^k\}$ is a fixed point.

The proof draws upon two ideas: the safeguarding technique already used and recent Halpern iteration analysis [76]. Note the inequality in Line 6 involves an energy different from Algorithm 2. Additionally, here the fallback update is anchored to the reference iterate $\tilde{x}$, providing stability of iterates (*i.e.* mitigating oscillatory behavior). This energy safeguard also provides a convergence rate,[10] $\mathcal{O}(1/k)$ if $C = 0$ and $\mathcal{O}(1/\sqrt{k})$ if $C > 0$. For numerical examples illustrating this safeguard with matrix games,[11] we refer readers to [218].

[10] Note $\mathcal{O}(1/k)$ is the *optimal* order of convergence rate for a resolvent oracle.

[11] Averaged and firmly nonexpansive operators can also be used for convex-concave minimax problems.

Section 2.3: Training and Averaged Operator Selection

Safe-L2O may be executed via inferences of a feed forward model. The input into the model is the data d, often in vector form. Each layer is designed so that its input is x^k, to which it applies either an L2O or fallback update (following the Safe-L2O method), and outputs x^{k+1} to the next layer. We encode all the model parameters with $\Theta \triangleq \{\Theta^k\}$. The set over which Θ is minimized may be chosen with great flexibility. If each Θ^k can be chosen independently, the model weights vary by layer. This is used in our numerical examples below. If instead each Θ^k is identical, *i.e.* the parameters across all layers are fixed, then we obtain a weight-tied recurrent neural network (RNN).[12]

[12] The implicit models covered in Part II all use an operator T_Θ that coincides with weight-tied models.

The "optimal" choice of weights Θ depends upon the application. Suppose each d is drawn from a common distribution $\mathcal{D}$. Then a choice of "optimal" parameters $\Theta^\star$ may be identified as those for which the expected value of $\phi_d(x^K)$ is minimized among $d \sim \mathcal{D}$, where $\phi_d : \mathbb{R}^n \to \mathbb{R}$ is an appropriate cost function (see Table 2.2). This is expressed mathematically by stating $\Theta^\star$ solves the problem[13]

[13] Different learning problems than (2.8) may be used (*e.g.* the min-max problem used by adversarial networks [108]).

$$\min_{\Theta} \mathbb{E}_{d\sim\mathcal{D}}[\phi_d(x^K(\Theta, d))], \tag{2.8}$$

where we emphasize the dependence of x^K on Θ and d by writing $x^K = x^K(\Theta, d)$. We approximately solve the problem (2.8) by sampling data $\{d^n\}_{n=1}^N$ from $\mathcal{D}$ and minimizing an empirical loss function. For our layer-dependent weights approach, we train by using successive "warm starts." First, the model is trained for approximately optimal performance with 1 iteration. These weights are then used to initialize the model for 2 iterations, which is then trained to approximate optimality. This process is repeated until the desired K iterations are reached for the model, with the majority of training time dedicated to training with K iterations.

L1	$\|x^K - x_d^\star\|_1$
L2	$\|x^K - x_d^\star\|_2^2$
Obj.	$f(x^K; d)$
Res.	$\|x^K - T(x^K; d)\|$

Table 2.2: Example choices for ϕ_d include ℓ_1 and ℓ_2 errors, objective values, and fixed point residuals.

Section 2.4: Numerical Examples

This section shows examples using Safe-L2O. We numerically investigate (i) the convergence rate of Safe-L2O relative to corresponding conventional algorithms, (ii) the efficacy of safeguarding procedures when inferences are performed on data for which L2O fails intermittently, and (iii) the convergence of Safe-L2O schemes even when the application of $\mathcal{N}_\Theta$ is not justified theoretically. We first use $\mathcal{N}_\Theta$ from ALISTA [164] on a synthetic LASSO problem. We use LISTA on a LASSO problem for image processing, differentiable linearized ADMM [241] on a sparse coding problem, and an L2O method for nonnegative least squares.[14]

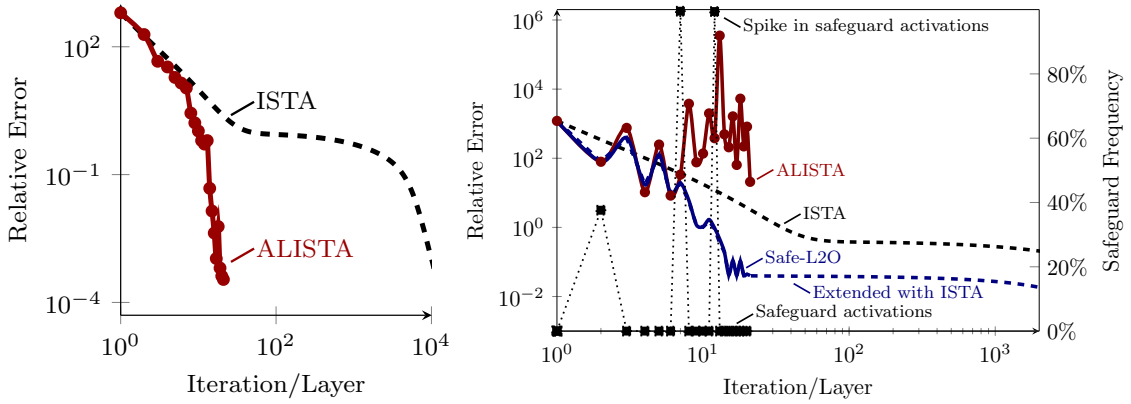
[14] Fine details of numerical implementations can be found in the appendices of [127].

In each example, $f_d^\star$ denotes the optimal value of $f_d(x)$ among all possible x. Performance is measured using a modified relative objective error:

$$\text{Relative Error} = \mathcal{R}_{f,\mathcal{D}}(x) := \frac{\mathbb{E}_{d\sim\mathcal{D}}[f_d(x) - f_d^\star]}{\mathbb{E}_{d\sim\mathcal{D}}[f_d^\star]}, \tag{2.9}$$

where the expectations are estimated numerically. We use (2.9) rather than the expectation of relative error to avoid high sensitivity to outliers.

Our numerical results are presented in several plots. When each iterate x^k is computed using data d drawn from the same distribution $\mathcal{D}_s$ that was used to train the L2O algorithm, we say the performance is on the "seen" distribution $\mathcal{D}_s$. These plots form the primary illustrations of the speedup of L2O algorithms. When each d is drawn from a distribution $\mathcal{D}_u$ that is *different* than $\mathcal{D}_s$, we refer to $\mathcal{D}_u$ as the *unseen* distribution. These plots show the ability of the safeguard to ensure convergence. A dotted plot with square markers is also added to show the frequency of safeguard activations among test samples, with the reference axis on the right hand side of the plots. We extend the Safe-L2O methods beyond their training iterations by applying the fallback operator T; we demarcate where this extension begins by changing the Safe-L2O plots from solid to dashed lines.



(a) Performance on *seen* distribution

(b) Performance on *unseen* distribution

Figure 2.2: Plot of error versus iteration for ALISTA example. Trained with $\phi_d = f_d$. Inferences used $\alpha = 0.99$ and EMA(0.25). In (b), how often the L2O update is "bad" and the safeguard activates for Safe-L2O is indicated in reference to the right vertical axis. This plot shows the safeguard is used only when $k = 2$, $k = 7$, and $k = 12$. Also, Safe-L2O converges in (b) while ALISTA displays divergent behavior.

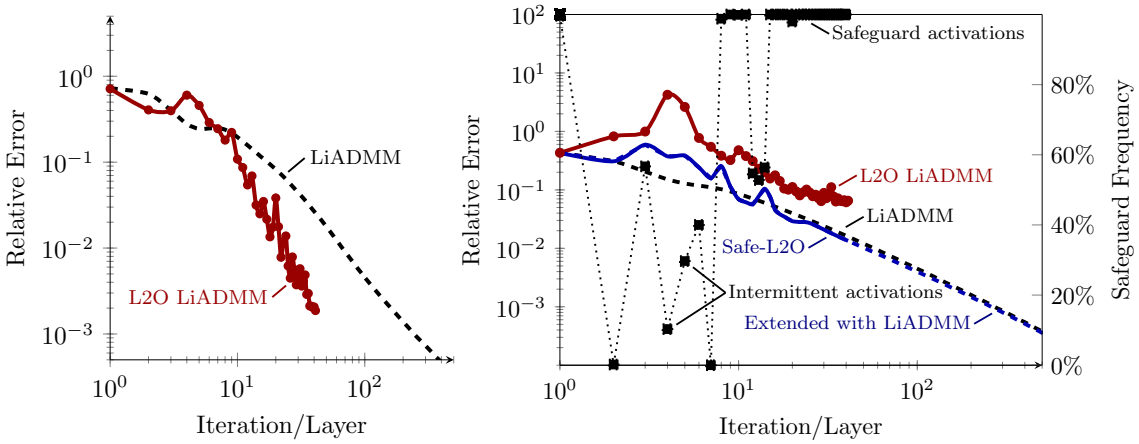
ALISTA for LASSO

Here we consider the LASSO problem for sparse coding. Let $x^\star \in \mathbb{R}^{500}$ be a sparse vector and $A \in \mathbb{R}^{250\times 500}$ be a dictionary. We assume access is given to noisy linear measurements $d \in \mathbb{R}^{250}$, where $\varepsilon \in \mathbb{R}^{250}$ is additive Gaussian white noise and $d = Ax^\star + \varepsilon$. Even for underdetermined systems, when $x^\star$ is sufficiently sparse and $\tau \in (0, \infty)$ is an appropriately chosen regularization parameter, $x^\star$ can often be recovered reasonably by solving the LASSO problem

$$\min_{x\in\mathbb{R}^n} f_d(x) := \frac{1}{2}\|Ax - d\|_2^2 + \tau\|x\|_1. \tag{2.10}$$

A classic method for solving (2.10) is the iterative shrinkage thresholding algorithm (ISTA) (*e.g.* see [73]).[15] [164] presents the L2O scheme ALISTA that we implement here. This L2O operator $\mathcal{N}_\Theta$ is parameterized by $\Theta = \{(\theta_k, \gamma_k)\}_{k=1}^K \in \mathbb{R}^{2K}$.

[15] This is a special case of the proximal-gradient in Table 1.1.



(a) Performance on *seen* distribution

(b) Performance on *unseen* distribution

Figure 2.3: Plot of error versus iteration for D-LADMM. Trained with $\phi_d = f_d$. Inferences used $\alpha = 0.99$ and EMA(0.75). In (b), how often the L2O update is "bad" and the safeguard activates for Safe-L2O is indicated in reference to the right vertical axis. This plot shows the safeguard is used about 10% and 30% of the time when $k = 4$ and $k = 5$, respectively.

Linearized ADMM

Let $A \in \mathbb{R}^{250\times 500}$ and $d \in \mathbb{R}^{250}$ be as in Subsection 2.4. Here we apply the L2O scheme differentiable linearized ADMM (D-LADMM) of [241] to the closely related sparse coding problem

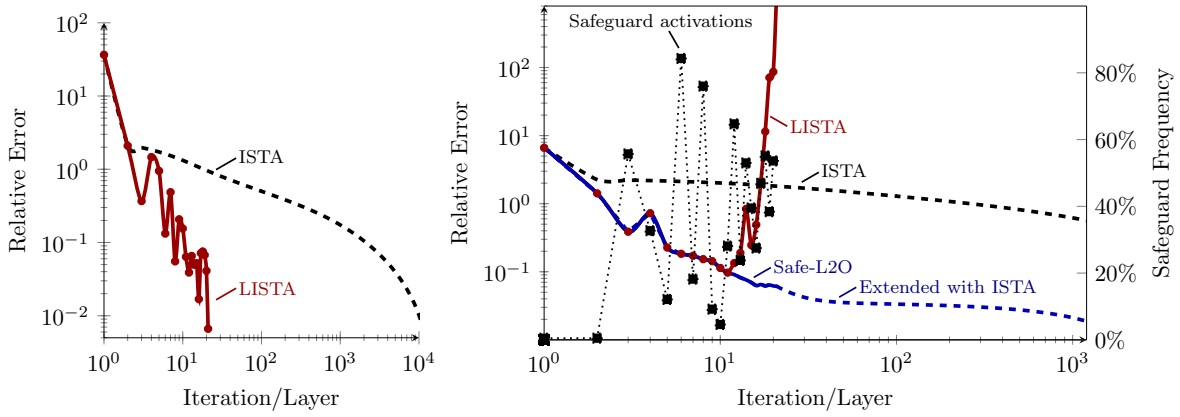
$$\min_{x\in\mathbb{R}^n} \|Ax - d\|_1 + \tau\|x\|_1. \tag{2.11}$$

The L2O operator $\mathcal{N}_\Theta$ and fallback linearized ADMM (LiADMM) operator T. Plots are provided in Figure 2.3.

LISTA for Natural Image Denoising

To evaluate our safeguarding mechanism in a more realistic setting, we apply safeguarded LISTA to a natural image denoising problem. In this subsection, we learn a LISTA-CP model [63] to perform natural image denoising. During training, L2O LISTA-CP model is trained to recover clean images from their Gaussian

noisy counterparts by solving (2.10). In (2.10), d is the noisy input to the model, and the clean image is recovered with $\hat{d} = Ax^\star$, where $x^\star$ is the optimal solution. The dictionary $A \in \mathbb{R}^{256\times512}$ is learned on the BSD500 dataset [173] by solving a dictionary learning problem [245]. During testing, however, the learned L2O LISTA-CP is applied to unseen pepper-and-salt noisy images. Comparison plots are provided in Figure 2.4 and implementation details are in the appendices.



(a) Performance on *seen* distribution (b) Performance on *unseen* distribution

Figure 2.4: Plot of error versus iteration for LISTA denoising. Trained with $\phi_d = f_d$. Inferences used $\alpha = 0.99$ and EMA(0.25). In (b), how often the L2O update is "bad" and the safeguard activates for Safe-L2O is indicated in reference to the right vertical axis. This plot shows the safeguard is used intermittently for $k > 2$.

Projected Gradient for Nonnegative Least Squares

Let $A \in \mathbb{R}^{500\times250}$ and $d \in \mathbb{R}^{500}$. Consider an overdetermined NNLS problem

$$\min_{x\in\mathbb{R}^{250}} f_d(x) := \frac{1}{2}\|Ax - d\|_2^2 \quad \text{s.t.} \quad x \geq 0. \tag{2.12}$$

Generalizing the projected-gradient method, we use

$$T_\Theta(x;\, d) := \max\left(x - \Theta(Ax - d), 0\right), \tag{2.13}$$

where $\Theta \in \mathbb{R}^{250\times500}$. The fallback method is projected gradient, *i.e.* $T(x;d) \triangleq T_{\alpha A^\top}(x;d)$ where $\alpha = 1/\|A^\top A\|_2$. Here $\Theta = \{\Theta^k\}_{k=1}^K$ consists of mnK trainable

parameters. A summary plot is given in Figure 2.5. Since this problem is unregularized, the L2O method learned very efficient updates, given A. This resulted in comparable performance on unseen data and the safeguard was never activated.

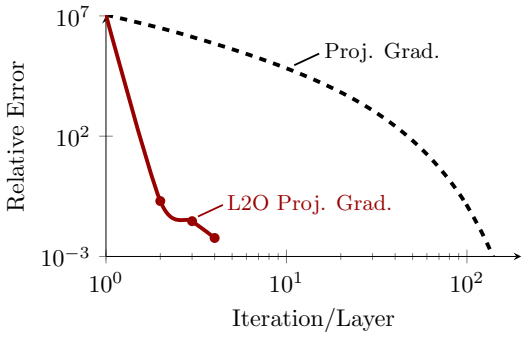


Figure 2.5: Performance on seen distribution $\mathcal{D}_s$ of L2O projected gradient scheme in (2.13).

Section 2.5: Conclusions

This chapter's safeguarding frameworks ensure convergence of L2O algorithms. Our Safe-L2O algorithm is also easy to implement, with our numerical experiments demonstrate rapid convergence by Safe-L2O methods and effective safeguarding when the L2O schemes appear to otherwise diverge. Roughly speaking, ALISTA, LISTA, and the L2O method for NNLS all reduced computational costs by at least one order of magnitude when applied to data from the same distribution as the training data. This is expected, given the results of previous works. More importantly, plots (b) of Figures 2.2 to 2.4 show the safeguard steers updates to convergence when they would otherwise diverge or converge slower than the conventional algorithm. Future work will provide a better data-driven fallback method and investigate stochastic extensions.

Part II: Implicit Models

Chapter 3: Jacobian-Free Backprop

I am enough of the artist to draw freely upon my imagination. Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world.

– Albert Einstein[1]

[1] Taken from Einstein's 1929 interview for *The Saturday Evening Post* [228].

[2] This chapter is based on the paper [97].

This chapter[2] delves into a relatively new direction of machine learning, the movement from explicit to implicit models [239, 18, 19, 61, 101, 82, 135, 252, 151, 203, 165, 111]. In the standard feedforward setting, a model prescribes a series of computations that map input data d to an inference y. Models can also explicitly leverage the assumption that high dimensional signals typically admit low dimensional representations in some latent space [226, 188, 197, 83, 225]. This may be done by designing the model to first map data to a latent space via a mapping Q_Θ and then apply a second mapping S_Θ to map the latent variable to the inference. Thus, a traditional feedforward $\mathcal{E}_\Theta$ may take the compositional form $\mathcal{E}_\Theta(d) = S_\Theta(Q_\Theta(d))$, illustrated by the red arrows in Figure 3.1.

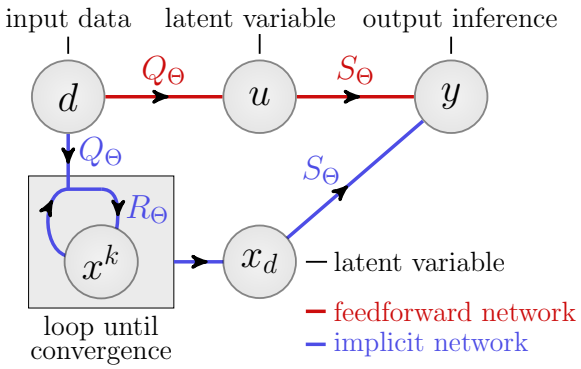


Figure 3.1: Feedforward models compute $S_\Theta \circ Q_\Theta$. Implicit models add a fixed point condition using R_Θ. If it is averaged, applying R_Θ to update a variable x^k gives convergence to a fixed point x_d of $R_\Theta(\cdot; Q_\Theta(d))$.

One can allow for computation in the latent space $\mathcal{U}$ by introducing a self-map $R_\Theta(\cdot; Q_\Theta(d))$ and the iteration

$$x^{k+1} = R_\Theta(x^k; Q_\Theta(d)), \quad \text{for all } k \in \mathbb{N}. \tag{3.1}$$

Iterating k times may be viewed as a weight-tied, input-injected model, where each feedforward step applies R_Θ [18]. As $k \to \infty$, *i.e.* the latent space portion becomes deeper, the limit of (3.1) yields a *fixed point equation*. As discussed in the introduction, implicit models capture this "infinite depth" behaviour by using $R_\Theta(\cdot\,; Q_\Theta(d))$ to define a fixed point condition:

$$\mathcal{N}_\Theta(d) \triangleq S_\Theta(x_d) \quad \text{where} \quad x_d = R_\Theta(x_d, Q_\Theta(d)), \tag{3.2}$$

as shown by blue in Figure 3.1. Special cases of the model in (3.2) recover architectures introduced in prior works [18, 19, 101, 239]. Continuing from the introduction, we note three immediate questions arising from the definition (3.2):

- Is the definition in (3.2) well-posed?
- How is $\mathcal{N}_\Theta(d)$ evaluated?
- How are the weights Θ of $\mathcal{N}_\Theta$ updated during training?

Since the first two points are well-established [239, 18], we briefly review these in Section 3.1 and focus on the third point. Using gradient-based methods for training requires computing $\mathrm{d}\mathcal{N}_\Theta/\mathrm{d}\Theta$ and, in particular, $\mathrm{d}x_d/\mathrm{d}\Theta$. Hitherto, previous works computed $\mathrm{d}x_d/\mathrm{d}\Theta$ by solving a Jacobian-based equation (see Section 3.2). Solving this linear system is computationally expensive and prone to instability [20], particularly when the dimension of the latent space is large and/or includes certain structures (*e.g.* batch normalization and/or dropout) [18, 19].

This chapter covers a novel and simple **Jacobian-Free Backpropagation** (JFB) technique for training implicit models that avoids *any* linear system solves. Instead, our scheme backpropagates by omitting the Jacobian term, resulting in

a form of preconditioned gradient descent. JFB yields much faster training of implicit models and allows for a wider array of architectures.

Section 3.1: Implicit Model Formulation

All terms presented in this section are provided in a general context, which is later made concrete for each application. We include a subscript Θ on various terms to emphasize the indicated mapping will ultimately be parameterized in terms of tunable weights[3] Θ. At the highest level, we are interested in constructing a model $\mathcal{N}_\Theta : \mathcal{D} \to \mathcal{Y}$ that maps from a data space[4] $\mathcal{D}$ to an inference space $\mathcal{Y}$. The implicit portion of the model uses a latent space $\mathcal{U}$, and data is mapped to this latent space by $Q_\Theta : \mathcal{D} \to \mathcal{U}$. We define the *model operator* $T_\Theta : \mathcal{U} \times \mathcal{D} \to \mathcal{U}$ by

$$T_\Theta(x; d) \triangleq R_\Theta(x, Q_\Theta(d)). \tag{3.3}$$

Provided input data d, our aim is to find the unique fixed point x_d of $T_\Theta(\cdot\,; d)$ and then map $u_d^\star$ to the inference space $\mathcal{Y}$ via a final mapping $S_\Theta : \mathcal{U} \to \mathcal{Y}$. This enables us to define an implicit model $\mathcal{N}_\Theta$ by

$$\mathcal{N}_\Theta(d) \triangleq S_\Theta(x_d) \quad \text{where} \quad x_d = T_\Theta(x_d; d). \tag{3.4}$$

[3] We use the same subscript for all terms, noting each operator typically depends on a portion of the weights.

[4] Each space is assumed to be a real-valued finite dimensional Hilbert space (*e.g.* $\mathbb{R}^n$) endowed with a product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$. It will be clear from context which space is being used.

Algorithm 4: Implicit Model with Fixed Point Iteration

```
N_Θ(d):                                  ◁ Input data is d
  x^1 ← x̃                                ◁ Assign latent term
  while ‖x^k − T_Θ(x^k; d)‖ > ε          ◁ Loop til converge
    x^{k+1} ← T_Θ(x^k; d)                ◁ Refine latent term
    k ← k + 1                            ◁ Increment counter
return S_Θ(x^k)                          ◁ Output estimate
```

Implementation considerations for T_Θ are discussed below. We also introduce assumptions on T_Θ that yield sufficient conditions to use the simple procedure in Algorithm 4 to approximate $\mathcal{N}_\Theta(d)$. In this algorithm,[5] the latent variable initialization $\tilde{x}$ can be any fixed quantity (*e.g.* the zero vector). The inequality in Step 3 gives a fixed point residual condition that measures convergence. Step 4 implements a fixed point update. The estimate of the inference $\mathcal{N}_\Theta(d)$ is computed by applying S_Θ to the latent variable x^k in Step 6. The blue path in Figure 3.1 visually summarizes Algorithm 4.

[5] Note the great similarity between Algorithms 1 and 4.

Convergence Finitely many loops in Steps 3 and 4 of Algorithm 4 is guaranteed by a classic functional analysis result [21]. This approach is used by several implicit models [101, 239, 135]. Below is a variation of Banach's classic result for application in our setting.

Assumption 3.1.1 *The mapping T_Θ is L-Lipschitz with respect to inputs (x, d), i.e.*

$$\|T_\Theta(x;\ d) - T_\Theta(v;\ w)\| \leq L\|(x,d) - (v,w)\|, \quad \textit{for all } (x,d),(v,w) \in \mathcal{U} \times \mathcal{D}. \tag{3.5}$$

Holding d fixed, the operator $T_\Theta(\cdot; d)$ is a contraction, i.e. there is $\gamma \in [0,1)$ such that

$$\|T_\Theta(x;d) - T_\Theta(v;d)\| \leq \gamma\|x - v\|, \quad \textit{for all } x, v \in \mathcal{U}. \tag{3.6}$$

Remark 3.1.1 *The L-Lipschitz condition on T_Θ is used since recent works show Lipschitz continuity with respect to inputs improves generalization [220, 110, 89] and adversarial robustness [66, 10].*

Theorem 3.1.1 (Banach) *For any $x^1 \in \mathcal{U}$, if Assumption 3.1.1 holds and if the sequence $\{x^k\}$ is generated via the update relation*[6]

$$x^{k+1} = T_\Theta(x^k;\ d), \quad \textit{for all } k \in \mathbb{N}, \tag{3.7}$$

then $\{x^k\}$ converges linearly to the unique fixed point x_d of $T_\Theta(\cdot; d)$.

[6] This result is closely related to Theorem 4.1.1, differing in its use of a stronger contraction assumption.

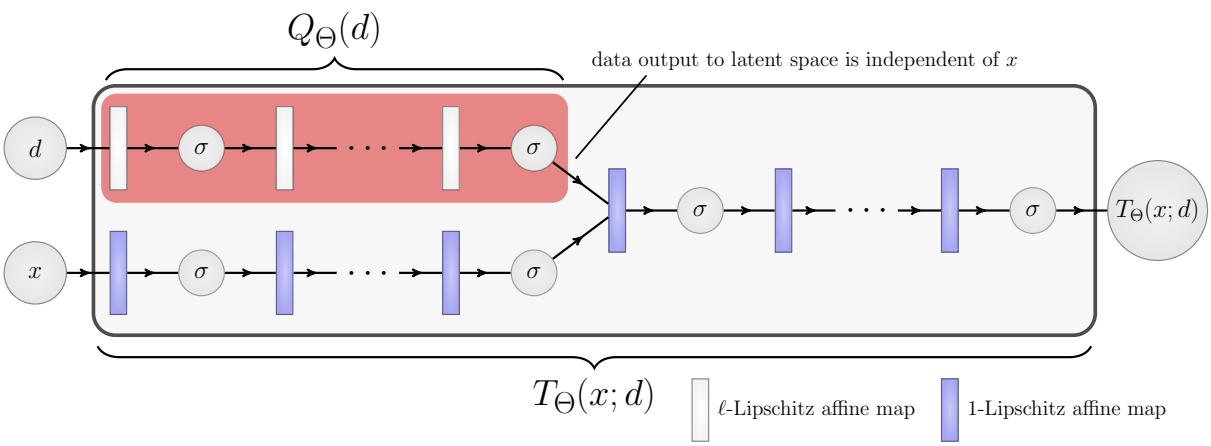


Figure 3.2: Diagram of a possible architecture for model operator T_Θ (in large rectangle). Data d and latent x variables are processed in two streams by nonlinearities (denoted by σ) and affine mappings (denoted by rectangles). These streams merge into a final stream that may also contain transformations. Light gray and blue affine maps are ℓ-Lipschitz and 1-Lipschitz, respectively. The mapping Q_Θ from data space to latent space is enclosed by the red rectangle.

Alternative Approaches In [18, 19] Broyden's method is used for finding x_d. Broyden's method is a quasi-Newton scheme and so at each iteration it updates a stored approximation to the Jacobian J_k and then solves a linear system in J_k. Since in this work our goal is to explore truly *Jacobian-free* approaches, we stick to the simpler fixed point iteration scheme when computing x_d (*i.e.* Algorithm 4). In the contemporaneous [103], it is reported that using fixed point iteration in conjunction with Anderson acceleration finds x_d faster than both vanilla fixed point iteration and Broyden's method. Combining JFB with Anderson accelerated fixed point iteration is a research direction we leave for future work.

Other Implicit Formulations A related implicit learning formulation is the well-known neural ODE model [61, 82, 210]. Neural ODEs leverage known connections between deep residual models and discretizations of differential equations [117, 237, 211, 60, 88, 168], and replace these discretizations by black-box ODE solvers in forward and backward passes. The implicit property of these models arise from their method for computing gradients. Rather than backprop-

agate through each layer, backpropagation is instead done by solving the adjoint equation [134] using a blackbox ODE solver as well. This is analogous to solving the Jacobian-based equation when performing backpropagation for implicit models (see (3.10)) and allows the user to alleviate the memory costs of backpropagation through deep neural models by solving the adjoint equation at additional computational costs. A drawback is the adjoint equation must be solved to high-accuracy; otherwise, a descent direction may not be guaranteed [102, 185, 186].

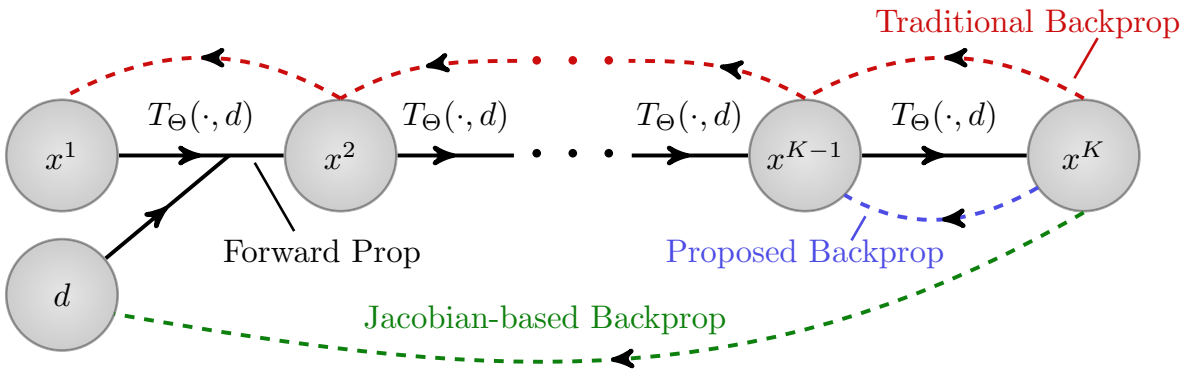


Figure 3.3: Diagram of backpropagation schemes for recurrent implicit depth models. Forward propagation is tracked via solid arrows point to the right (*n.b.* each forward step uses d). Backpropagation is shown via dashed arrows pointing to the left. Traditional backpropagation requires memory capacity proportional to depth (which is implausible for large K). Jacobian-based backpropagation solves an associated equation dependent upon the data d and operator T_Θ. JFB uses a single backward step, which avoids both large memory capacity requirements and solving a Jacobian-type equation.

Section 3.2: Backpropagation

We present a simple way to backpropagate with implicit models, called Jacobian-free backprop (JFB). Traditional backpropagation will *not* work effectively for implicit models since forward propagation during training could entail hundreds or thousands of iterations, requiring ever growing memory to store computational graphs. On the other hand, implicit models maintain fixed memory costs by backpropagating "through the fixed point" and solving a Jacobian-based equation (at potentially substantial added computational costs). The key step to circumvent this Jacobian-based equation with JFB is to tune weights by using a preconditioned gradient. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a smooth loss function, denoted by $\ell(x, y)$, and consider the training problem

$$\min_{\Theta} \mathbb{E}_{d\sim\mathcal{D}}\big[\,\ell\,(y_d, \mathcal{N}_\Theta(d))\,\big], \tag{3.8}$$

where we abusively write $\mathcal{D}$ to also mean a distribution. For clarity of presentation, in the remainder of this section we notationally suppress the dependencies on weights Θ by letting x_d denote the fixed point in (3.4). Unless noted otherwise, mapping arguments are implicit in this section; in each implicit case, this will correspond to entries in (3.4). We begin with standard assumptions enabling us to differentiate $\mathcal{N}_\Theta$.

Assumption 3.2.1 *The mappings S_Θ and T_Θ are continuously differentiable with respect to x and Θ.*

[7] This assumption is easy to ensure in practice. For notational brevity, we use the subscript Θ throughout.

Assumption 3.2.2 *The weights Θ may be written as a tuple $\Theta = (\theta_S, \theta_T)$ such that weight paramaterization of S_Θ and T_Θ depend only on θ_S and θ_T, respectively.*[7]

Let $\mathcal{J}_\Theta$ be defined as the identity operator, denoted by I, minus the Jacobian[8] of T_Θ at (x, d), *i.e.*

$$\mathcal{J}_\Theta(x; d) \triangleq \mathrm{I} - \frac{\mathrm{d}T_\Theta}{\mathrm{d}x}(x; d). \tag{3.9}$$

[8] Under Assumption 3.1.1, the Jacobian $\mathcal{J}_\Theta$ exists almost everywhere. However, presentation is cleaner by assuming smoothness.

Following [239, 18], we differentiate both sides of the fixed point relation in (3.4) to obtain, by the implicit function theorem,

$$\frac{\mathrm{d}x_d}{\mathrm{d}\Theta} = \frac{\partial T_\Theta}{\partial x}\frac{\mathrm{d}x_d}{\mathrm{d}\Theta} + \frac{\partial T_\Theta}{\partial \Theta} \quad \Longrightarrow \quad \frac{\mathrm{d}x_d}{\mathrm{d}\Theta} = \mathcal{J}_\Theta^{-1} \cdot \frac{\partial T_\Theta}{\partial \Theta}, \tag{3.10}$$

where $\mathcal{J}_\Theta^{-1}$ exists whenever $\mathcal{J}_\Theta$ exists (see Lemma 9.2.4). Using the chain rule gives the loss gradient

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\Theta}\left[\ell(y_d, \mathcal{N}_\Theta(d))\right] &= \frac{\mathrm{d}}{\mathrm{d}\Theta}\Big[\ell(y_d, S_\Theta(T_\Theta(x_d, d))\Big] \\ &= \frac{\partial \ell}{\partial y}\left[\frac{\mathrm{d}S_\Theta}{\mathrm{d}x}\mathcal{J}_\Theta^{-1}\frac{\partial T_\Theta}{\partial \Theta} + \frac{\partial S_\Theta}{\partial \Theta}\right]. \end{aligned} \tag{3.11}$$

The matrix $\mathcal{J}_\Theta$ satisfies the inequality (see Lemma 9.2.4)

$$\left\langle x, \mathcal{J}_\Theta^{-1} x\right\rangle \geq \frac{1-\gamma}{(1+\gamma)^2}\|x\|^2, \quad \text{for all } x \in \mathcal{U}. \tag{3.12}$$

Intuitively, this coercivity property makes $\mathcal{J}_\Theta^{-1}$ appear to act like a preconditioner and, thus, we may possibly remove[9] $\mathcal{J}_\Theta^{-1}$ from (3.11) to backpropagate using

$$\begin{aligned} p_\Theta &\triangleq -\frac{\mathrm{d}}{\mathrm{d}\Theta}\Big[\ell(y_d, S_\Theta(T_\Theta(x, d))\Big]_{x=x_d} \\ &= -\frac{\partial \ell}{\partial y}\left[\frac{\mathrm{d}S_\Theta}{\mathrm{d}x}\frac{\partial T_\Theta}{\partial \Theta} + \frac{\partial S_\Theta}{\partial \Theta}\right]. \end{aligned} \tag{3.13}$$

[9] Our knowledge for justifying this below is relatively limited; hence this is where imagination of what the situation "looks like" is more important, practically speaking.

The omission of $\mathcal{J}_\Theta^{-1}$ admits two straightforward interpretations. Note $\mathcal{N}_\Theta(d) = S_\Theta(T_\Theta(x_d; d))$, and so p_Θ is precisely the gradient of the expression $\ell(y_d, S_\Theta(T_\Theta(x_d; d)))$, treating x_d as a constant *independent* of Θ. The distinction is that using $S_\Theta(T_\Theta(x_d; d))$ assumes, perhaps by chance, the user chose the first iterate x^1 in their fixed point iteration (see Algorithm 4) to be precisely the fixed point x_d. This makes the iteration trivial, "converging" in one iteration. We can simulate this behavior by using the fixed point iteration to find x_d and only backpropagating through the final step of the fixed point iteration, as shown in Figure 3.3.

Since the weights Θ typically lie in a space of much higher dimension than the latent space $\mathcal{U}$, the Jacobians $\partial S_\Theta/\partial\Theta$ and $\partial T_\Theta/\partial\Theta$ effectively always have full column rank. We leverage this fact via the following assumption.

Assumption 3.2.3 *Under Assumption 3.2.2, given any weights $\Theta = (\theta_S, \theta_T)$ and data d, the matrix*

$$M \triangleq \begin{bmatrix} \dfrac{\partial S_\Theta}{\partial \theta_S} & 0 \\ 0 & \dfrac{\partial T_\Theta}{\partial \theta_T} \end{bmatrix} \tag{3.14}$$

has full column rank and is sufficiently well conditioned to satisfy the inequality[10]

$$\kappa(M^\top M) = \frac{\lambda_{\max}(M^\top M)}{\lambda_{\min}(M^\top M)} \leq \frac{1}{\gamma}. \tag{3.15}$$

[10] The term γ here refers to the contraction factor in (3.6).

Remark 3.2.1 *The conditioning portion of the above assumption is useful for bounding the worst-case behavior in our analysis. However, we found it unnecessary to enforce this in our experiments for effective training (e.g. see Figure 3.5), which we hypothesize is justified because worst case behavior rarely occurs in practice and we train using averages of p_Θ for samples drawn from large data sets.*

Assumption 3.2.3 gives rise to a second interpretation of JFB. Namely, the full column rank of M enables us to rewrite p_Θ as a preconditioned gradient, *i.e.*

$$p_\Theta = \underbrace{\left(M \begin{bmatrix} \mathrm{I} & 0 \\ 0 & \mathcal{J}_\Theta \end{bmatrix} M^+ \right)}_{\text{preconditioning term}} \frac{\mathrm{d}\ell}{\mathrm{d}\Theta}, \tag{3.16}$$

where M^+ is the Moore-Penrose pseudo inverse [178, 196]. These insights lead to this chapter's main result.

Theorem 3.2.1 *If Assumptions 3.1.1, 3.2.1, 3.2.2, and 3.2.3 hold for given weights Θ and data d, then*

$$p_\Theta \triangleq -\frac{\mathrm{d}}{\mathrm{d}\Theta}\Big[\ell(y_d, S_\Theta(T_\Theta(x, d))\Big]_{x=x_d} \tag{3.17}$$

is a descent direction for $\ell(y_d, \mathcal{N}_\Theta(d))$ with respect to Θ.

Theorem 3.2.1 shows we can avoid difficult computations associated with $\mathcal{J}_\Theta^{-1}$ in (3.11) (*i.e.* solving an associated linear system/adjoint equation) in implicit model literature [61, 82, 18, 239]. Thus, our scheme more naturally applies to general multilayered T_Θ and is substantially simpler to code. Our scheme is juxtaposed in Figure 3.3 with classic and Jacobian-based schemes.

Two additional considerations must be made when determining the efficacy of training a model using (3.17) rather than Jacobian-based gradients (3.11).

- Does use of p_Θ in (3.17) degrade training/testing performance relative to (3.11)?
- Is the term p_Θ in (3.17) resilient to errors in estimates of the fixed point x_d?

The first answer is our training scheme takes a different path to minimizers than using gradients with the implicit model. Thus, for nonconvex problems, one should not expect the results to be the same. In our experiments in Section 4.3, using (3.17) is competitive (3.11) for all tests (when applied to nearly identical models). The second inquiry is partly answered by the corollary below, which states JFB yields descent even for approximate fixed points.

Corollary 3.2.1 *Given weights Θ and data d, there exists $\varepsilon > 0$ such that if $x_d^\varepsilon \in \mathcal{U}$ satisfies $\|x_d^\varepsilon - x_d\| \leq \varepsilon$ and the assumptions of Theorem 3.2.1 hold, then*

$$p_\Theta^\varepsilon \triangleq -\frac{\mathrm{d}}{\mathrm{d}\Theta}\Big[\ell(y_d, S_\Theta(T_\Theta(x,d))\Big]_{x=x_d^\varepsilon} \tag{3.18}$$

is a descent direction of $\ell(y_d, \mathcal{N}_\Theta(d))$ with respect to Θ.

We are not aware of any other error tolerance results for implicit models.

Coding Backpropagation A key feature of JFB is its simplicity of implementation. In particular, the backpropagation of our scheme is similar to that of a standard backpropagation. We illustrate this in the sample of PyTorch [191] code

in Figure 3.4. Here `explicit_model` represents $S_\Theta(T_\Theta(x;d))$. The fixed point x_d = `x_fxd_pt` is computed by successively applying T_Θ (see Algorithm 4) within a `torch.no_grad()` block. With this fixed point, `explicit_model` evaluates and returns $S_\Theta(T_\Theta(x_d,d))$ to `y` in `train` mode (to create the computational graph). Thus, our scheme coincides with standard backpropagation through an explicit model with *one* latent space layer. On the other hand, standard implicit models backpropagate by solving a linear system to apply $\mathcal{J}_\Theta^{-1}$ as in (3.11). That approach requires manual updates of the parameters, more computational resources, and considerations (*e.g.* conditioning of $\mathcal{J}_\Theta^{-1}$) for each architecture used.

Figure 3.4: Sample PyTorch code for backpropagation

Implicit Forward + Proposed Backprop

```
x_fxd_pt = find_fixed_point(d)
y = explicit_model(x_fxd_pt, d)
loss = criterion(y, labels)
loss.backward()
optimizer.step()
```

Neumann Backpropagation The inverse of the Jacobian in (3.9) can be expanded using a Neumann series, *i.e.*

$$\mathcal{J}_\Theta^{-1} = \left(\mathrm{I} - \frac{\mathrm{d}T_\Theta}{\mathrm{d}u}\right)^{-1} = \sum_{k=0}^{\infty} \left(\frac{\mathrm{d}T_\Theta}{\mathrm{d}u}\right)^k. \tag{3.19}$$

Thus, JFB is a zeroth-order approximation to the Neumann series. In particular, JFB resembles the Neumann-RBP approach for recurrent models [157]. Neumann-RBP does not guarantee a descent direction or guidelines on how to truncate the Neumann series. This is generally difficult to achieve in theory and practice [6]. Our work differs from [157] in that we focus purely on implicit models, prove descent guarantees for JFB, and provide simple PyTorch implementations. Similar approaches exist in hyperparameter optimization, where truncated Neumann series are is used to approximate second-order updates during training [170, 167]. Similar zeroth-order truncations of the Neumann series have been employed,

albeit without proof, in meta-learning [90, 200] and in training transformers [99].

Section 3.3: Experiments

This section shows the effectiveness of JFB using PyTorch [191]. All models are ResNet-based such that Assumption 3.2.2 holds.[11] One can ensure Assumption 3.1.1 holds (*e.g.* via spectral normalization). Yet, in our experiments we found this unnecessary since tuning the weights automatically encouraged contractive behavior.[12] All experiments are run on a single NVIDIA TITAN X GPU with 12GB RAM. Further details are in Appendix E of [97].

[11] A weaker version of Assumption 3.2.1 also holds in practice, *i.e.* differentiability almost everywhere.

[12] We found (3.6) held for batches of data during training, even when using batch normalization.

MNIST

Method	Model size	Acc.
Explicit	54K	99.4%
Neural ODE†	84K	96.4%
Aug. Neural ODE†	84K	98.2%
MON ‡	84K	99.2%
JFB-trained Implicit ResNet (ours)	54K	**99.4%**

SVHN

Method	Model size	Acc.
Explicit	164K	93.7%
Neural ODE†	172K	81.0%
Aug. Neural ODE†	172K	83.5%
MON (Multi-tier lg)‡	170K	92.3%
JFB-trained Implicit ResNet (ours)	164K	**94.1%**

CIFAR-10

Method	Model size	Acc.
Explicit (ResNet-56)*	0.85M	93.0%
MON (Multi-tier lg)‡*	1.01M	89.7%
JFB-trained Implicit ResNet (ours)*	0.84M	**93.7%**
Multiscale DEQ*	10M	93.8%

Table 3.1: Test accuracy of JFB-trained Implicit ResNet compared to Neural ODEs, Augmented NODEs, and MONs; †as reported in [82]; ‡as reported in [239]; *with data augmentation

Classification

We train implicit models on three benchmark image classification datasets licensed under CC-BY-SA: SVHN [183], MNIST [152], and CIFAR-10 [150]. Ta-

	Dataset	Avg time per epoch (s)	# of $\mathcal{J}$ mat-vec products	Accuracy %
Jacobian based	MNIST	28.4	6.0×10^6	99.2
	SVHN	92.8	1.4×10^7	90.1
	CIFAR10	530.9	9.7×10^8	87.9
JFB	MNIST	17.6	0	99.4
	SVHN	36.9	0	94.1
	CIFAR10	146.6	0	93.67

Table 3.2: Comparison of Jacobian-based backpropagation (first three rows) and our proposed JFB approach. "Mat-vecs" denotes matrix-vector products.

ble 3.1 compares our results with state-of-the-art results for implicit models, including Neural ODEs [61], Augmented Neural ODEs [82], Multiscale DEQs [19], and MONs [239]. We also compare with corresponding explicit versions of our ResNet-based models as well as with state-of-the-art ResNet results [123] on the augmented CIFAR10 dataset. The explicit models are trained with the same setup as their implicit counterparts. Table 3.1 shows JFBs are an effective way to train implicit models, substantially outperform all the ODE-based models as well as MONs using similar or fewer parameters. Moreover, JFB is competitive with Multiscale DEQs [19] despite having less than a tenth as many parameters.

Comparison to Jacobian-based Backpropagation

Table 3.2 compares performance between using the standard Jacobian-based backpropagation and JFB. The experiments are performed on all the datasets described in Section 3.3. To apply the Jacobian-based backpropagation in (3.10), we use the conjugate gradient (CG) method on an associated set of normal equations similarly to [157]. To maintain similar costs, we set the maximum number of CG iterations to be the same as the maximum depth of the forward propagation. The remaining experimental settings are kept the same as those from our proposed approach (and are therefore not tuned to the best of our ability). Note the model architectures trained with JFB contain batch normalization in the latent space whereas those trained with Jacobian-based backpropagation do not. Removal

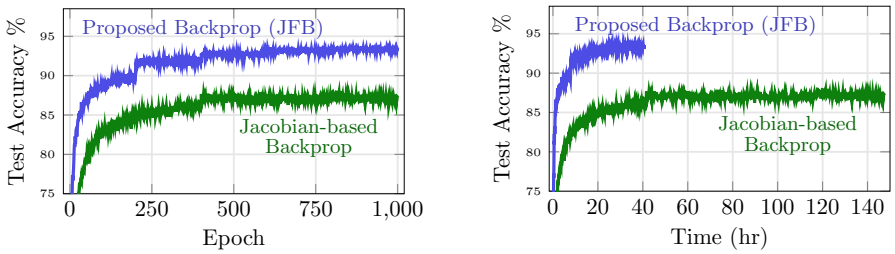


Figure 3.5: CIFAR10 results using comparable models and configurations, but with two backpropagation schemes: our proposed JFB method (blue) and standard Jacobian-based backpropagation in (3.11) (green), with fixed point tolerance $\epsilon = 10^{-4}$. JFB gives better test accuracy and is faster.

of batch normalization for the Jacobian-based method was necessary due to a lack of convergence when solving (3.10), thereby increasing training loss (see the appendices for further details). This phenomena is also observed in previous works [19, 18]. Thus, we find JFB to be (empirically) effective on a wider class of model architectures (*e.g.* including batch normalization). The main purpose of the Jacobian-based results in Figure 3.5 and Table 3.2 is to show speedups in training time while maintaining a competitive accuracy with previous state-of-the-art implicit models.

Higher Order Neumann Approximation

As explained above, JFB can be interpreted as an approximation to the Jacobian-based approach by using a truncated series expansion. In particular, JFB is the zeroth order (*i.e.* $k = 0$) truncation to the Neumann series expansion (3.19) of the Jacobian inverse $\mathcal{J}_\Theta^{-1}$. In Figure 3.6, we compare JFB with training that uses more Neumann series terms in the approximation of the the Jacobian inverse $\mathcal{J}_\Theta^{-1}$. Figure 3.6 shows JFB is competitive at reduced time cost. More significantly, JFB is also much easier to implement as shown in Figure 3.4. An additional experiment with SVHN data and discussion about code are provided in the appendices.

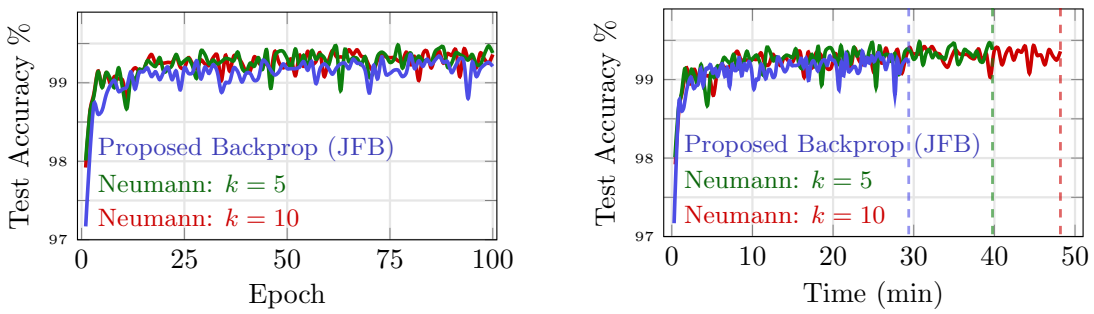


Figure 3.6: MNIST training using different truncations k of the Neumann series (3.19) to approximate the inverse Jacobian $\mathcal{J}_\Theta^{-1}$. Plots show faster training with fewer terms (fastest with JFB, *i.e.* $k = 0$) and competitive test accuracy.

Section 3.4: Conclusions

This chapter presents a new and simple Jacobian-free backpropagation (JFB) scheme. JFB enables training of implicit models with fixed memory costs (regardless of depth), is easy to code (see Figure 3.4), and yields efficient backpropagation (by removing computations to do linear solves at each step). Use of JFB is theoretically justified (even when fixed points are approximately computed). Our experiments show JFB yields competitive results for implicit models. Extensions in future work (and subsequent chapters) enable satisfaction of additional constraints for imaging and phase retrieval [143, 87, 128, 96, 139], geophysics [115, 94, 95], and games [230, 160, 155, 212]. Future work will also analyze JFB in stochastic settings.

Chapter 4: Convex Feasibility Problems

Each piece, or part, of the whole of nature is always merely an approximation to the complete truth, or the complete truth so far as we know it. In fact, everything we know is only some kind of approximation, because we know that we do not know all the laws as yet. Therefore, things must be learned only to be unlearned again or, more likely, to be corrected.

– Richard Feynman[1]

[1] Taken from his lecture "Atoms in Motion" [86].

Inverse problems consist of recovering a signal from a collection of noisy measurements.[2] These problems can often be cast as feasibility problems; however, additional regularization is typically necessary to ensure accurate and stable recovery with respect to data perturbations. Hand-chosen analytic regularization can yield theoretical guarantees, but such approaches can have limited practical effectiveness recovering signals due to their inability to leverage large amounts of available data. To this end, this chapter follows suit with Feynman's quote and suggests relearning/correcting convex feasibility problems to better align with available data. This is done by fusing data-driven regularization and convex feasibility in a theoretically sound manner using implicit feasibility-based models.[3] Each feasibility model defines a collection of nonexpansive operators, each of which is the composition of a projection-based operator and a data-driven regularization operator. Fixed point iteration is used to compute fixed points of these operators, and weights of the operators are tuned so fixed points closely represent available data. Numerical examples here show performance increases by feasibility models when compared to standard TV-based recovery methods for CT reconstruction and a comparable model based on algorithm unrolling.

[2] This chapter is based on [125].

[3] These models were originally called *Feasibility-based Fixed Point Networks* (F-FPNs).

Inverse problems arise in numerous applications such as medical imaging [12, 13, 120, 189], phase retrieval [29, 42, 96], geophysics [37, 94, 95, 114, 116, 139], and machine learning [70, 93, 117, 229, 240]. The goal of inverse problems is to recover a signal[4] $x_d^\star$ from a collection of indirect noisy measurements d. These quantities are typically related by a linear mapping A via

$$d = Ax_d^\star + \varepsilon, \tag{4.1}$$

where ε is measurement noise. Inverse problems are often ill-posed, making recovery of the signal $x_d^\star$ unstable for noise-affected data d. To overcome this, traditional approaches estimate the signal $x_d^\star$ by a solution x_d to the problem

$$\min_x \ \ell(Ax, d) + J(x), \tag{4.2}$$

where ℓ is a fidelity term that measures the discrepancy between the measurements and the application of the forward operator A to the signal estimate (*e.g.* least squares). The function[5] J serves as a regularizer, which ensures both that the solution to (4.2) is unique and that its computation is stable. In addition to ensuring well-posedness, regularizers are constructed in an effort to instill prior knowledge of the true signal, *e.g.* sparsity $J(x) = \|x\|_1$ [32, 41, 43, 78], Tikhonov $J(x) = \|x\|^2$ [40, 107], total variation (TV) $J(x) = \|\nabla x\|_1$ [57, 209], and, more recently, data-driven regularizers [3, 144, 171]. A further generalization of using data-driven regularization consists of Plug-and-Play (PnP) methods [58, 67, 128, 227], which replace the proximal operators in an optimization algorithm with previously trained denoisers.

[4] While we refer to signals, this phrase is generally meant to describe objects of interest that can be represented mathematically (*e.g.* images, parameters of a differential equation, and points in a Euclidean space).

[5] Throughout this chapter, J is used exclusively for a regularizer and does *not* correspond to a resolvent operator.

An underlying theme of regularization is that signals represented in high dimensional spaces often exhibit a common structure. Although hand picked regularizers may admit desirable theoretical properties leveraging *a priori* knowledge, they are typically unable to leverage available data. An ideal regularizer

will leverage available data to best capture the core properties that should be exhibited by output reconstruction estimates of true signals. Neural networks have demonstrated great success in this regard, achieving state of the art results [243, 137]. However, purely data-driven machine learning approaches do little to leverage the underlying physics of a problem, which can lead to poor compliance with data [176]. On the other hand, fast feasibility-seeking algorithms (*e.g.* see [56, 55, 109, 51, 49] and references therein) efficiently leverage known physics to solve inverse problems, being able to handle massive-scale sets of constraints [24, 56, 187, 194]. Thus, a relatively untackled question remains:

Is it possible to fuse feasibility-seeking algorithms with data-driven regularization in a manner that improves reconstructions and yields convergence?

This chapter answers the above inquiry affirmatively. The key idea is to use machine learning techniques to create mappings T_Θ, parameterized by weights Θ. For fixed measurement data d, $T_\Theta(\cdot\,; d)$ forms an operator possessing standard properties used in feasibility algorithms. Fixed point iteration finds fixed points of each $T_\Theta(\cdot\,; d)$, and weights Θ are tuned so these fixed points both resemble available signal data and are consistent with measurements (up to a noise level).

Contribution The contribution of this chapter is to connect powerful feasibility-seeking algorithms to data-driven regularization in a manner that maintains theoretical guarantees. This is done by presenting a feasibility model framework that solves a learned feasibility problem. Numerical examples are provided that demonstrate notable performance benefits to our proposed formulation when compared to TV-based methods and fixed-depth neural networks formed by algorithm unrolling.

Feasibility Problem Convex feasibility problems (CFPs) arise in many real-world applications, *e.g.* imaging, sensor networks, radiation therapy treatment planning (see [56, 30, 27] and the references therein). We formalize the CFP setting and relevant methods as follows. Let $\mathcal{U}$ and $\mathcal{D}$ be finite dimensional Hilbert spaces, referred to as the signal and data spaces, respectively. Given additional knowledge about a linear inverse problem, measurement data $d \in \mathcal{D}$ can be used to express a CFP solved by the true signal[6] $x_d^\star \in \mathcal{U}$ when measurements are noise-free. That is, data d can be used to define a collection $\{\mathcal{C}_{d,j}\}_{j=1}^m$ of closed convex subsets of $\mathcal{U}$ (*e.g.* hyperplanes) such that the true signal $x_d^\star$ is contained in their intersection, *i.e.* $x_d^\star$ solves the problem

[6] Note $x_d^\star$ is the signal sought whereas x_d is the solution to (CFP). Ideally, these are the same, but are often different in practice.

$$\text{Find } x_d \text{ such that } x_d \in \mathcal{C}_d \triangleq \bigcap_{j=1}^m \mathcal{C}_{d,j}. \tag{CFP}$$

A common approach to solving (CFP), *inter alia*, is to use projection algorithms [26], which utilize orthogonal projections onto the individual sets $\mathcal{C}_{d,j}$. For a closed, convex, and nonempty set $\mathcal{C} \subseteq \mathcal{U}$, the projection $P_{\mathcal{C}} : \mathcal{U} \to \mathcal{C}$ onto $\mathcal{C}$, is defined by

$$P_{\mathcal{C}}(x) \triangleq \arg\min_{v \in \mathcal{C}} \frac{1}{2}\|v - x\|^2. \tag{4.3}$$

Projection algorithms are iterative in nature and each update uses combinations of projections onto each set $\mathcal{C}_{d,j}$, relying on the principle that it is generally much easier to project onto the individual sets than onto their intersection. These methods date back to the 1930s [138, 65] and have been adapted to now handle huge-size problems of dimensions for which more sophisticated methods cease to be efficient or even applicable due to memory requirements [56]. Computational simplicity derives from the fact the building bricks of a projection algorithm are the projections onto individual sets. Memory efficiency occurs because the algo-

rithmic structure is either sequential or simultaneous (or hybrid) as in the block-iterative projection methods [5, 39] and string-averaging projection methods [56, 54, 52, 53]. These algorithms generate sequences that solve (CFP) asymptotically, and the update operations can be iteration dependent (*e.g.* cyclic projections). We let $\mathcal{A}_d^k$ be the update operator for the k-th step of a projection algorithm solving (CFP). Consequently, each projection algorithm generates a sequence $\{x^k\}$ via the fixed point iteration

$$x^{k+1} \triangleq \mathcal{A}_d^k(x^k), \quad \text{for all } k \in \mathbb{N}. \tag{FPI}$$

A common assumption for such methods is the intersection of all the algorithmic operators' fixed point sets contains or forms the desired set $\mathcal{C}_d$, *i.e.*

$$\mathcal{C}_d = \bigcap_{k=1}^{\infty} \operatorname{fix}(\mathcal{A}_d^k), \tag{4.4}$$

which automatically holds when $\{\mathcal{A}_d^k\}$ cycles over a collection of projections.

Data-Driven Feasibility Problem As noted previously, inverse problems are often ill-posed, making (CFP) insufficient to faithfully recover the signal $x_d^\star$. Additionally, when noise is present, it can often be the case that the intersection is empty (*i.e.* $\mathcal{C}_d = \varnothing$). This calls for a different model to recover $x_d^\star$. To date, projection methods have limited inclusion of regularization (*e.g.* superiorization [74, 50, 129, 122, 48], sparsified Kaczmarz [215, 166]). Beyond sparsity via ℓ_1 minimization, such approaches typically do not yield guarantees beyond feasibility (*e.g.* it may be desirable to minimize a regularizer over $\mathcal{C}_d$). We propose composing a projection algorithm and a data-driven regularization operator in a manner so each update is analogous to a proximal-gradient step. This is accomplished via a parameterized mapping $R_\Theta : \mathcal{U} \to \mathcal{U}$, with weights denoted by Θ. This mapping directly leverages available data to learn features shared among true signals of interest. We augment (CFP) by using operators $\{\mathcal{A}_d^k\}$ for solving (CFP)

and instead solve the learned common fixed points (L-CFP) problem

$$\text{Find } x_d \text{ such that } x_d \in \mathcal{C}_{\Theta,d} \triangleq \bigcap_{k=1}^{\infty} \operatorname{fix}(\mathcal{A}_d^k \circ R_\Theta). \tag{L-CFP}$$

Loosely speaking, when R_Θ is chosen well, the signal x_d closely approximates $x_d^\star$.

We briefly recap classic operator results to solve (L-CFP). Recall an operator $T{:}\mathcal{U} \to \mathcal{U}$ is *nonexpansive* if it is 1-Lipschitz, *i.e.*

$$\|T(u) - T(u)\| \le \|u - v\|, \quad \text{for all } u, v \in \mathcal{U}. \tag{4.5}$$

Also, T is *averaged* if there exists $\alpha \in (0,1)$ and a nonexpansive operator $Q : \mathcal{U} \to \mathcal{U}$ such that $T(x) = (1-\alpha)x + \alpha Q(x)$ for all $x \in \mathcal{U}$. For example, the projection $P_{\mathcal{S}}$ defined in (4.3) is averaged along with convex combinations of projections [46]. Our method utilizes the following standard assumptions, which are typically satisfied by projection methods (in the noise-free setting with R_Θ as the identity).

Assumption 4.1.1 *The intersection set $\mathcal{C}_{\Theta,d}$ defined in (L-CFP) is nonempty and $\{(\mathcal{A}_d^k \circ R_\Theta)\}$ forms a sequence of nonexpansive operators.*

Assumption 4.1.2 *For any sequence $\{x^k\} \subset \mathcal{U}$, the sequence of operators $\{(\mathcal{A}_d^k \circ R_\Theta)\}$ has the property*

$$\lim_{k\to\infty} \|(\mathcal{A}_d^k \circ R_\Theta)(x^k) - x^k\| = 0 \implies \liminf_{k\to\infty} \|P_{\mathcal{C}_{\Theta,d}}(x^k) - x^k\| = 0. \tag{4.6}$$

When a finite collection of update operations are used and applied (essentially) cyclically, the previous assumption automatically holds (*e.g.* setting $\mathcal{A}_d^k \triangleq P_{\mathcal{C}_{d,i_k}}$ and $i_k \triangleq k \bmod(m) + 1$). We use the learned fixed point iteration to solve (L-CFP)

$$x^{k+1} \triangleq (\mathcal{A}_d^k \circ R_\Theta)(x^k), \quad \text{for all } k \in \mathbb{N}. \tag{L-FPI}$$

Justification of the (L-FPI) iteration is provided by the following theorems, which are rewritten from their original form to match the present context.

Algorithm 5: Feasibility Model Computation

$\mathcal{N}_\Theta(d)$: ◁ Input data is d
  $x^1 \leftarrow \tilde{x}$ ◁ Initialize iterate to fixed reference
  $k \leftarrow 1$ ◁ Initialize iteration counter
  while $\|x^k - x^{k-1}\| \geq \delta$ **or** $k = 1$ ◁ Loop to convergence
    $x^{k+1} \leftarrow (\mathcal{A}_d^k \circ R_\Theta)(x^k; d)$ ◁ Apply regularization and feasibility
    $k \leftarrow k + 1$ ◁ Increment counter
  return x^k ◁ Output solution estimate

Theorem 4.1.1 (Krasnosel'skiĭ-Mann [148, 172]) *If $(\mathcal{A}_d \circ R_\Theta): \mathcal{U} \to \mathcal{U}$ is averaged and has a fixed point, then, for any $x^1 \in \mathcal{U}$, the sequence $\{x^k\}$ generated by (L-FPI), taking $\mathcal{A}_d^k \circ R_\Theta = \mathcal{A}_d \circ R_\Theta$, converges to a fixed point of $\mathcal{A}_d \circ R_\Theta$.*

Theorem 4.1.2 (Cegielski, Theorem 3.6.2, [46]) *If Assumptions 4.1.1 and 4.1.2 hold, and if $\{x^k\}$ is a sequence generated by the iteration (L-FPI) satisfying $\|x^{k+1} - x^k\| \to 0$, then $\{x^k\}$ converges to a limit $x^\infty \in \mathcal{C}_{\Theta,d}$.*

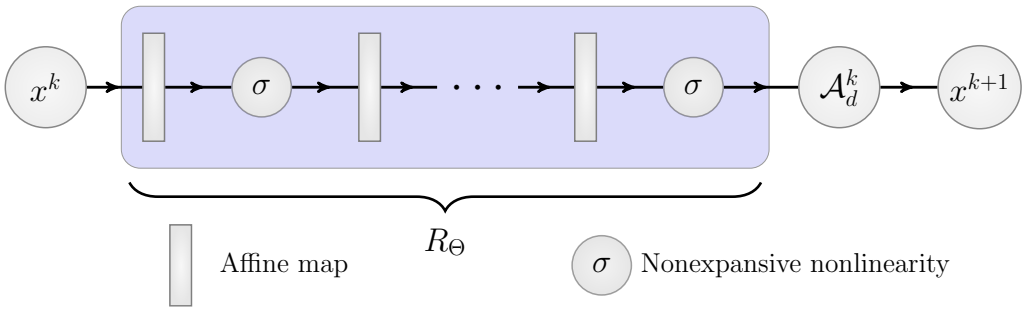


Figure 4.1: Diagram for update operations in the learned fixed point iteration (L-FPI) to solve (L-CFP). Here R_Θ is comprised of a finite sequence of applications of (possibly) distinct affine mappings (*e.g.* convolutions), and nonlinearities (*e.g.* projections on the nonnegative orthant, *i.e.* ReLUs). For each $k \in \mathbb{N}$, we let $\mathcal{A}_d^k$ be a projection-based algorithmic operator. The parameters Θ of R_Θ are tuned in an offline process by solving (3.8) to ensure signals are faithfully recovered.

Section 4.2: Feasibility Model

Herein we present the feasibility model. Although based on the setting of the previous chapter, here we replace the single operator of those models by a sequence of operators, each taking the form of a composition. Namely, we use updates in the iteration (L-FPI). The assumptions necessary for convergence can be approximately ensured (*e.g.* see the Appendix). This iteration yields the feasibility model $\mathcal{N}_\Theta$, defined by

$$\mathcal{N}_\Theta(d) \triangleq x_d, \quad \text{where} \quad x_d = \bigcap_{k=1}^{\infty} \operatorname{fix}(\mathcal{A}_d^k \circ R_\Theta), \tag{4.7}$$

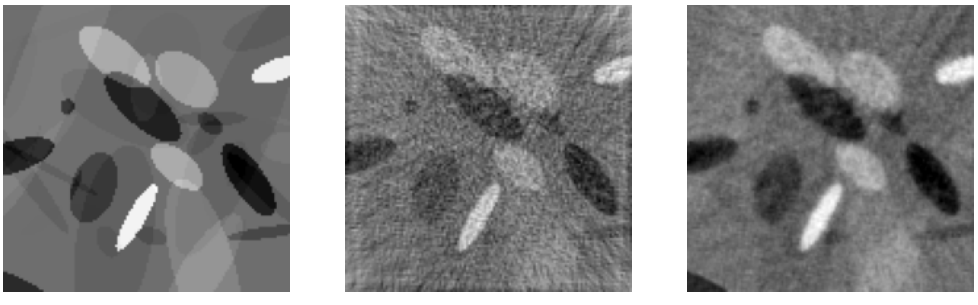
assuming the intersection is unique.[7] This is approximately implemented via Algorithm 5.

[7] Uniqueness is unlikely in practice; however, this assumption is justified since we use the same initial iterate u^1 for each initialization. This makes recovery of the same signal stable with respect to changes in Θ.

The weights Θ of the network $\mathcal{N}_\Theta$ are tuned by solving the training problem (3.8). In an ideal situation, the optimal weights $\Theta^\star$ solving (3.8) would yield feasible outputs (*i.e.* $\mathcal{N}_\Theta(d) \in \mathcal{C}_d$ for all data $d \in \mathcal{C}$) that also resemble the true signals $u_d^\star$. However, measurement noise in practice makes it unlikely that $\mathcal{N}_\Theta(d)$ is feasible, let alone that $\mathcal{C}_d$ is nonempty. In the noisy setting, this is no longer a concern since we augment (CFP) via (L-CFP) and are ultimately concerned with recovering a signal $x_d^\star$, not solving a feasibility problem. In summary, our model is based on the underlying physics of a problem (via the convex feasibility structure), but is also steered by available data via the training problem (3.8). Illustrations of the efficacy of this approach are provided in Section 4.3.

Section 4.3: Experiments

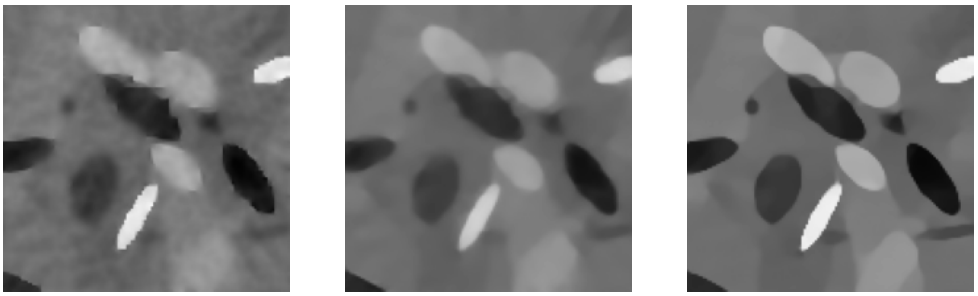
Experiments in this section show the relative reconstruction quality of feasibility models and comparable schemes – in particular, filtered backprojection (FBP) [81], total variation (TV) minimization (similarly to [184, 106]), total variation superiorization (based on [195, 133]), and an *unrolled* feasibility model.



Ground Truth
SSIM: 1.000
PSNR: ∞

FBP
SSIM: 0.273
PSNR: 18.224

TVS
SSIM: 0.582
PSNR: 25.88



TVM
SSIM: 0.786
PSNR: 27.80

Unrolling
SSIM: 0.811
PSNR: 26.01

Proposed
SSIM: 0.900
PSNR: 30.94

Figure 4.2: Ellipse reconstruction with test data for each method: filtered back projection (FBP), TV superiorization (TVS), TV minimization (TVM), unrolled network, and the proposed feasibility model.

Experimental Setup Comparisons are provided for two low-dose CT examples: a synthetic dataset, consisting of images of random ellipses, and the LoDoPab dataset [153], which consists of human phantoms. For both datasets, CT measurements are simulated with a parallel beam geometry with a sparse-angle setup of only 30 angles and 183 projection beams, resulting in 5,490 equations and 16,384 unknowns. Additionally, we add 1.5% Gaussian noise corresponding to each individual beam measurement. Moreover, the images have a resolution of 128×128 pixels. The quality of the image reconstructions are determined using the Peak Signal-To-Noise Ratio (PSNR) and structural similarity index measure (SSIM). We use the PyTorch deep learning framework [192] with the ADAM [141] optimizer. We use the Operator Discretization Library (ODL) python library [2]

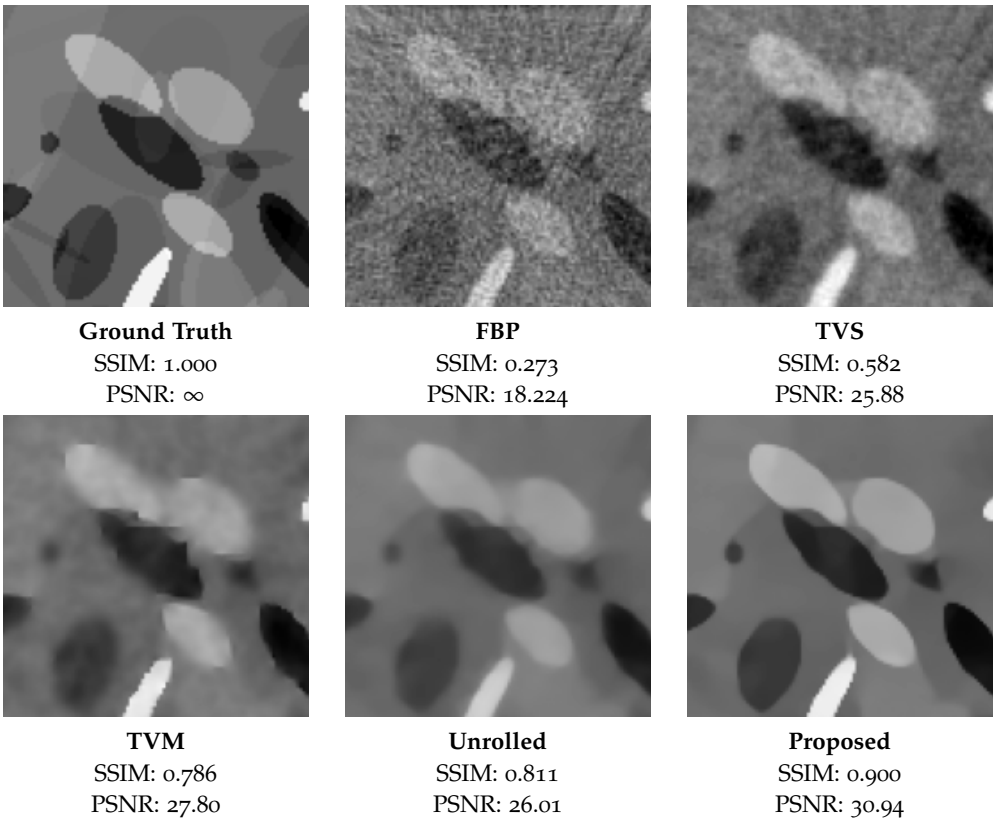
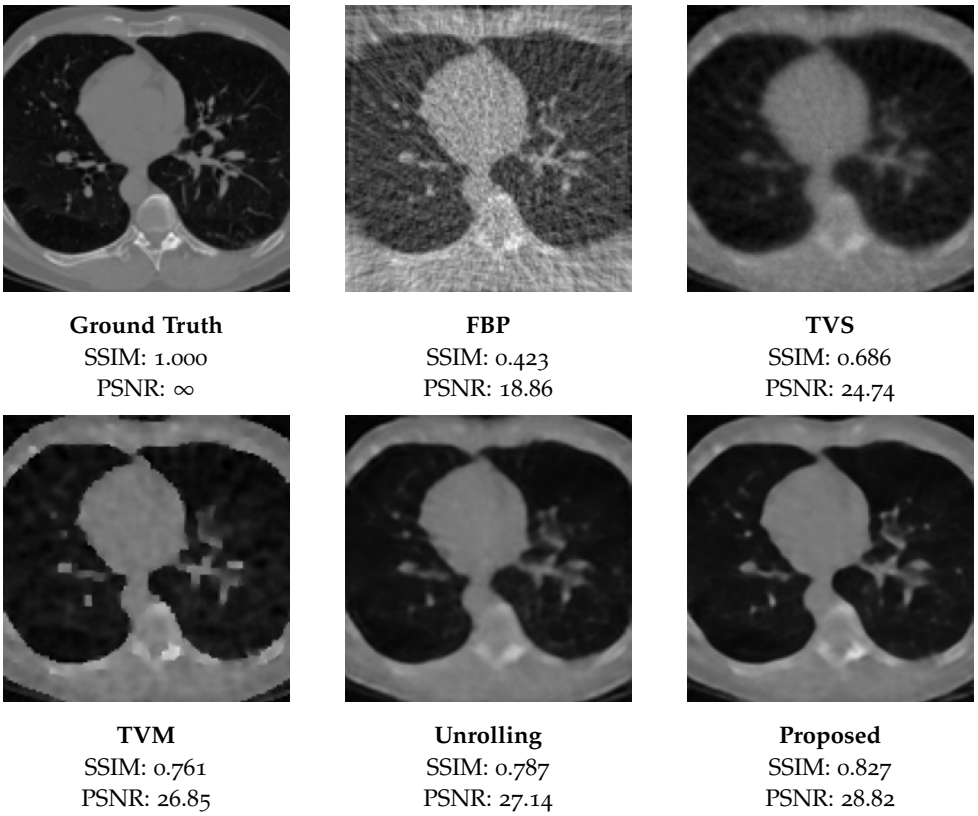


Figure 4.3: Zoomed-in ellipse reconstruction with test data of Figure 4.2 for each method: FBP, TVS, TVM, unrolling, and the proposed model.

to compute the filtered backprojection solutions. The CT experiments are run on a Google Colab notebook. For all methods, we use a single diagonally relaxed orthogonal projections (DROP) [55] operator for $\mathcal{A}_d$ (*i.e.* $\mathcal{A}_d^k = \mathcal{A}_d$ for all k), noting DROP is nonexpansive with respect to a norm dependent on A [124]. The loss function ℓ used for training is the mean squared error between reconstruction estimates and the corresponding true signals. We use a synthetic dataset consisting of random phantoms of combined ellipses as in [4]. The ellipse training and test sets contain 10,000 and 1,000 pairs, respectively. We also use phantoms derived from actual human chest CT scans via the benchmark Low-Dose Parallel Beam dataset (LoDoPaB) [153]. The LoDoPab training and test sets contain 20,000 and 2,000 pairs, respectively.



Ground Truth
SSIM: 1.000
PSNR: ∞

FBP
SSIM: 0.423
PSNR: 18.86

TVS
SSIM: 0.686
PSNR: 24.74

TVM
SSIM: 0.761
PSNR: 26.85

Unrolling
SSIM: 0.787
PSNR: 27.14

Proposed
SSIM: 0.827
PSNR: 28.82

Figure 4.4: LoDoPab reconstruction with test data for each method: filtered back projection (FBP), TV superiorization (TVS), TV minimization (TVM), unrolled network, and the proposed feasibility model.

Experiment Methods

TV Superiorization Sequences generated by successively applying the operator $\mathcal{A}_d$ are known to converge even in the presence of summable perturbations, which can be intentionally added to lower a regularizer value (*e.g.* TV) without compromising convergence, thereby giving a "superior" feasible point. Compared to minimization methods, superiorization typically only guarantees feasibility, but is often able to do so at reduced computational cost. This scheme, denoted as TVS, generates updates

$$x^{k+1} = \mathcal{A}_d\left(x^k - \alpha\beta^k D_-^\top\left(\frac{D_+x^k}{\|D_+x^k\|_2+\varepsilon}\right)\right), \quad \text{for } k = 1,2,\ldots,20, \tag{4.8}$$

Method	Avg. PSNR (dB)	Avg. SSIM	# Parameters
Filtered Backprojection	17.79	0.211	1
TV Superiorization	27.35	0.721	2
TV Minimization	28.55	0.772	4
Unrolled Network	30.39	0.859	96,307
Proposed	**31.30**	**0.877**	96,307

Table 4.1: Average PSNR and SSIM on the 1,000 image ellipse testing dataset.

where D_- and D_+ are the forward and backward differencing operators, $\varepsilon > 0$ is added for stability, and 20 iterations are used as early stopping to avoid overfitting to noise. The differencing operations yield a derivative of isotropic TV (*e.g.* see [158]). The scalars $\alpha > 0$ and $\beta \in (0, 1)$ are chosen to minimize training mean squared error. See the superiorization bibliography [47] for further TVS materials.

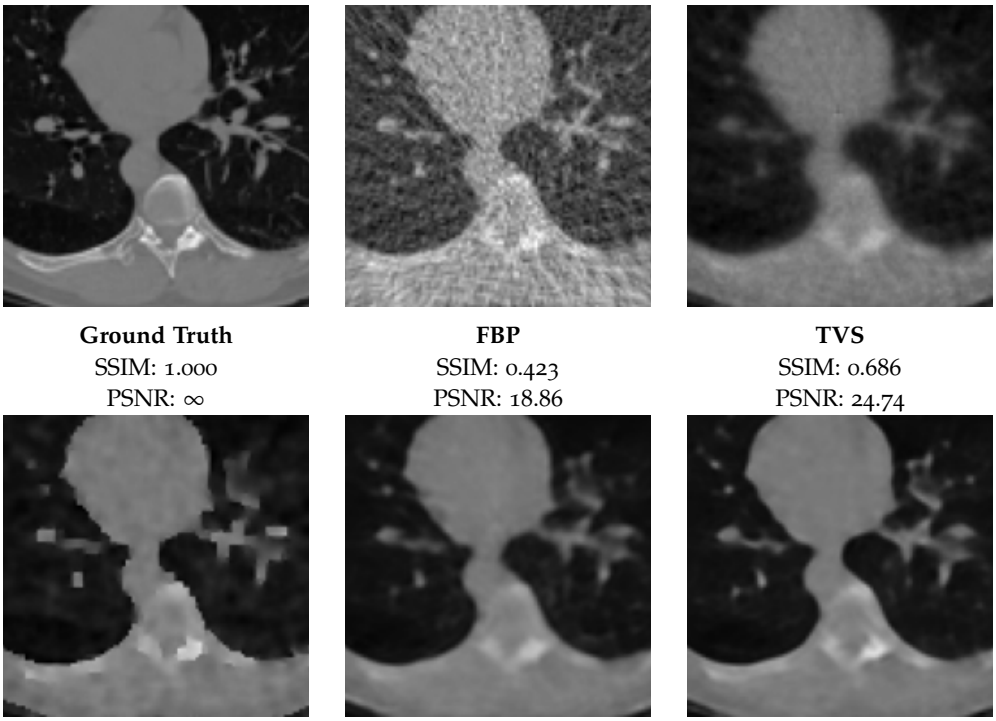
TV Minimization For a second analytic comparison method, we use anisotropic TV minimization (TVM). In this case, we solve the constrained problem

$$\min_{x \in [0,1]^n} \|D_+ x\|_1 \text{ such that } \|Ax - d\| \leq \varepsilon, \tag{TVM}$$

where $\varepsilon > 0$ is a hand-chosen scalar reflecting the level of measurement noise and the box constraints on u are included since all signals have pixel values in the interval $[0, 1]$. We use linearized ADMM [213] to solve (TVM) and refer to this model as TV minimization (TVM). Implementation details are in the appendices.

Method	Avg. PSNR (dB)	Avg. SSIM	# Parameters
Filtered Backprojection	19.27	0.354	1
TV Superiorization	26.65	0.697	2
TV Minimization	28.52	0.765	4
Deep Unrolling	29.30	0.800	96,307
Proposed	**30.46**	**0.832**	96,307

Table 4.2: Average PSNR/SSIM on the 2,000 image LoDoPab testing dataset.



Ground Truth
SSIM: 1.000
PSNR: ∞

FBP
SSIM: 0.423
PSNR: 18.86

TVS
SSIM: 0.686
PSNR: 24.74

TVM
SSIM: 0.761
PSNR: 26.85

Unrolling
SSIM: 0.787
PSNR: 27.14

Proposed
SSIM: 0.827
PSNR: 28.82

Figure 4.5: Zoomed-in LoDoPab reconstruction with test data of Figure 4.4 for each method: FBP, TVS, TVM, unrolling, and the proposed feasibility model.

Feasibility Model Structure The architecture of the operator R_Θ is modeled after the seminal work [123] on residual networks. The feasibility model and unrolled scheme both leverage the same structure R_Θ and DROP operator for $\mathcal{A}_d$. The operator R_Θ is the composition of four residual blocks. Each residual block takes the form of the identity mapping plus the composition of a leaky ReLU activation function and convolution (twice). The number of network weights in R_Θ for each setup was 96,307, a small number by machine learning standards. Training here used JFB [97].

Experiment Results Our results show the proposed feasibility models outperforms all classical methods as well as the unrolled data-driven method. We show the result on an individual reconstruction via wide and zoomed-in images from the ellipse and LoDoPab testing datasets in Figures 4.2 and 4.3 and Figures 4.4 and 4.5, respectively. The average SSIM and PSNR values on the entire

ellipse and LoDoPab datasets are shown in Tables 4.1 and 6.3. We emphasize the type of noise depends on each individual ray in a similar manner to [128], making the measurements more noisy than some related works. This noise and ill-posedness of our underdetermined setup are illustrated by the poor quality of analytic method reconstructions. (However, we note improvement by using TV over FBP and further improvement by TV minimization over TV superiorization.) Although nearly identical in structure to feasibility models, these results show the unrolled method to be inferior to feasibility models in these experiments. We hypothesize this is due to the large memory requirements of unrolling (unlike the proposed implicit models), which limits the number of unrolled steps ($\sim$ 20 steps versus 100+ steps of implicit feasibility models), and the proposed models are tuned to optimize a fixed point condition rather than a fixed number of updates.

Section 4.4: Conclusions

This chapter connects feasibility-seeking algorithms and data-driven algorithms. The presented implicit model leverages the elegance of fixed point methods while using state-of-the-art training methods for implicit deep learning. This results in a sequence of learned operators $\{\mathcal{A}_d^k \circ R_\Theta\}$ that can be repeatedly applied until convergence is obtained. This limit point is expected to be nearly compatible with provided constraints (up to the level of noise) and resemble the collection of true signals. The provided numerical examples show improved performance obtained by feasibility models over both classic methods and an unrolling-based network. Future work will extend feasibility models to a wider class of feasibility problems (*e.g.* split feasibility).

Chapter 5: Nash Equilibria

Faraday was asked: "What is the use of this discovery?"

He answered: "What is the use of a child - it grows to be a man."

– Alfred North Whitehead[1]

[1] Taken from *An Introduction to Mathematics* [238].

Introduction

Many recent works in deep learning have highlighted the power of using end-to-end learning in conjunction with known analytic models and constraints[2] [33, 16, 162, 161, 154, 146, 62]. This best-of-both worlds approach fuses the flexibility of learning-based approaches with the interpretability of models derived by domain experts. Moreover, hard-coding constraints into a data-driven algorithm can be used to guarantee safety and fairness. This chapter furthers this line of research by proposing a new framework for learning to predict the outcomes of contextual (*i.e.* parametrized) multi-player games from historical data while respecting constraints on players' actions.

[2] This chapter is adapted from [126].

Many social systems can profitably be analyzed as games, including competitive market economies [15], traffic routing [235], anti-poaching initiatives [246] and deployment of security resources [198]. Loosely speaking, any situation with multiple intelligent agents attempting to achieve conflicting aims may be viewed through a game-theoretic lens. Games with parameters depending on contextual information d that is beyond the control of the players are called *contextual games* [217]. For example, in traffic routing d may encode factors like weather, local sporting events or tolls influencing players' (*i.e.* drivers') commute times.

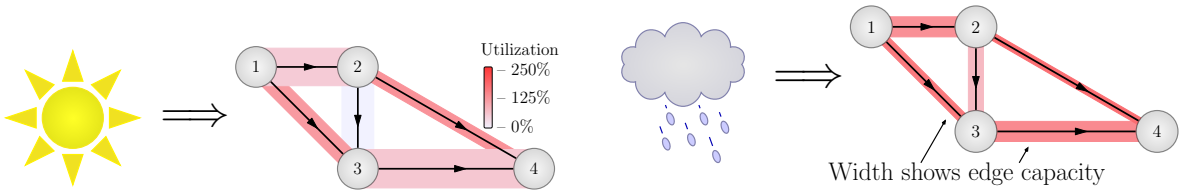


Figure 5.1: Nash models can predict traffic flow given contextual information (*e.g.* weather). For example, road capacities reduce on rainy days and light/dark red edges show light/heavy traffic.

Game-theoretic analyses frequently assume players' cost functions are known *a priori* and seek to predict how players will act, typically by computing a Nash equilibrium [182]. Informally, a Nash equilibrium is a choice of strategy for each player such that no player can improve their outcomes by unilaterally deviating from this strategy. We consider the problem of predicting equilibria, given only contextual information, without knowing the players' cost functions. Although the players' cost functions are unknown, historical data pairs $(d, x_d^\star)$ can be utilized, consisting of contexts d and the resulting equilibrium $x_d^\star$ [33, 250, 162, 161, 154]. This chapter proposes a new model framework based on Nash equilibria.[3] Here each model "learns to predict the appropriate game from context and then output game equilibria" by defining a tunable operator with fixed points that coincide with Nash equilibria. Forward propagation is formed by repeated application of the operator until a fixed point condition is satisfied. Thus, by construction, these models are implicit and the operator weights can be efficiently trained using JFB [97]. Importantly, the models presented here also avoid direct, costly projections onto sets of constraints for players' actions, which is the computational bottleneck of multiple prior works [162, 161, 154].

[3] The original work dubbed these models *Nash Fixed Point Networks* (N-FPNs).

Our framework applies equally to atomic games (*i.e.* the set of players is finite) and certain non-atomic games, particularly traffic routing [208]. As pointed out in [162, 154], learning to predict players' behaviours given the context is an important first step in manipulating the game towards a desirable outcome. For example, in traffic routing problems a central controller may seek to discourage

motorists from over-utilizing a road through a quiet neighborhood. Using our framework, and sufficient historical data, this hypothetical controller could predict the effect that exogenous variables (*e.g.* weather, a local sporting event) will have on traffic flow and then take corrective measures (*e.g.* increasing tolls) to decrease the predicted flow along this residential road.

Contributions This chapter provides a *scalable* data-driven framework for efficiently predicting equilibria in systems modeled as contextual games. Specifically, we contribute the following.

- Provide end-to-end trained model[4] that outputs Nash equilibria.
- Present scheme for decoupling constraints, enabling simple and computationally efficient forward and backward propagation.
- Demonstrate the efficacy of proposed models on traffic routing problems.
- Provide universal approximation result for game-based models.

[4] Plug-and-Play models are an example of something that is *not* trained end-to-end since training is separated from the algorithm used in deployment.

Attribute	Analytic	Feed Forward	Prior Implicit Models	Proposed Models
Output is Equilibria	✓		✓	✓
Data-Driven		✓	✓	✓
Constraint Decoupling	✓			✓
Simple Backprop	NA	✓		✓

Table 5.1: Comparison of different Nash equilibria prediction models. Analytic modeling algorithms yield game equilibria that are not data-driven. Traditional feed-forward models are data-driven and easy to train, but may not output a game equilibrium. Existing game-based implicit models are non-trivial to train (backpropagate) and require intricate forward propagation.

Section 5.1: Overview of Games

We begin with a brief review of game theory. After establishing notation, we provide a set of assumptions under which the mapping $d \mapsto x_d^\star$ is "well-behaved." We then recall the notion of a variational inequality and how Nash equilibria can be characterized using fixed point equations.

A K-player normal form contextual game is defined by action sets[5] $\mathcal{V}_k$ and cost functions $u_k : \mathbb{H} \times \mathcal{D} \to \mathbb{R}$ for $k \in [K]$, where $\mathcal{C} \triangleq \mathcal{V}_1 \times \ldots \times \mathcal{V}_K$ and $\mathcal{D}$ denotes the set of contexts (*i.e.* data space). The k-th player's actions x_k are constrained to the action set $\mathcal{V}_k$, yielding an action profile $x = (x_1, \ldots, x_K) \in \mathcal{C} \subseteq \mathbb{H}$. The actions of all players other than k are denoted by $x_{-k} = (x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_K)$. Assuming rationality, each player seeks to minimize their own cost function u_k by controlling only x_k while explicitly knowing u_k is impacted by the other players' actions x_{-k}. An action profile[6] x_d is a *Nash equilibrium* if

$$u_k(x_k, x_{d,-k}; d) \geq u_k(x_{d,k}, x_{d,-k}; d) \quad \text{for all } x_k \in \mathcal{V}_k \text{ and } k \in [K]. \tag{5.1}$$

In words, x_d is a Nash equilibrium if no player can lower their cost by unilaterally deviating from x_d. We make the following assumptions:

(A1) $\mathbb{H}$ is a finite dimensional Hilbert space.

(A2) $\mathcal{C} \subset \mathbb{H}$ is closed and convex.

(A3) The cost functions $u_k(\cdot; d)$ are continuously differentiable for all k and d.

(A4) $\nabla_k u_k(x; d)$ is Lipschitz continuous with respect to d for all k and all x.

(A5) Each cost function $u_k(x_k, x_{-k}; d)$ is α-strongly convex with respect to x_k.

(A6) $\mathcal{D}$ is compact.

Under these, the Nash equilibrium x_d is well-behaved with respect to d, as summarized by the following theorem.

Theorem 5.1.1 *If Assumptions (A1) to (A6) hold, then there is a unique Nash Equilibrium x_d for all $d \in \mathcal{D}$ and the map $d \mapsto x_d$ is Lipschitz continuous.*

[5] This is also known as the *decision set* and/or the *strategy set*.

[6] Here $x_d \in \mathcal{C}$ while $x_k \in \mathcal{V}_k$. The subscript letter will identify which space the point belongs to. In some instances, double subscripts are used for further clarity.

When each u_k is differentiable with respect to x, the *game gradient* is defined by

$$F(x;d) \triangleq \left[\nabla_{x_1} u_1(x;d)^\top, \ldots, \nabla_{x_K} u_K(x;d)^\top\right]^\top. \tag{5.2}$$

Remark 5.1.1 *Strong convexity is a strong assumption. It may be loosened at the possible cost of uniqueness. However, in practice uniqueness of model outputs can be ensured by fixing the initial iterate x^1.*

Variational Inequalities

Throughout, all sets $\mathcal{C}$ are assumed to be closed, convex and nonempty.

Definition 5.1.1 (Monotonicity) *For $\alpha > 0$, $d \in \mathcal{D}$, and a mapping $F : \mathbb{H} \times \mathcal{D} \to \mathbb{H}$, if*

$$\langle F(x;d) - F(y;d), x - y\rangle \geq \alpha \|F(x;d) - F(y;d)\|^2, \quad \textit{for all} \;\; x, y \in \mathbb{H}, \tag{5.3}$$

then $F(\cdot\,;d)$ is α-cocoercive[7]. If (5.3) holds taking $\alpha = 0$, then $F(\cdot\,;d)$ is monotone.

[7] This is also known as α-inverse strongly monotone

Definition 5.1.2 (VI) *Given $d \in \mathcal{D}$, find $x_d \in \mathcal{C}$ (called a solution of the VI) such that*

$$\langle F(x_d;d), x - x_d\rangle \geq 0, \quad \textit{for all} \;\; x \in \mathcal{C}. \tag{5.4}$$

The solution set to this variational inequality (VI) is denoted by $\mathrm{VI}(F(\cdot\,;d), \mathcal{C})$.

Nash equilibria may be characterized using VIs [84, Prop. 1.4.2]; namely,

$$x_d \text{ is a Nash Equilibrium} \iff x_d \in \mathrm{VI}(F(\cdot\,;d), \mathcal{C}). \tag{5.5}$$

In summary, x_d is a Nash equilibrium if no unilateral change lowers any individual cost and a VI solution if no feasible update reduces the sum of individual costs. By (5.5), these views are equivalent.

Section 5.2: Nash Equilibria Model

We propose networks defined by Nash equilibria. That is, by the equivalence between Nash equilibria and certain variational inequalities (see (5.5)), we use models defined by the optimality condition[8]

[8] Under the assumptions above, the VI solution is unique, making $\mathcal{N}_\Theta$ well-defined.

$$\mathcal{N}_\Theta(d) \triangleq \mathrm{VI}(F_\Theta(\cdot; d), \mathcal{C}), \tag{5.6}$$

where $\mathcal{C}$ is a product of action sets $\mathcal{V}_k$ and $F_\Theta(\cdot;\cdot)$ is a feed forward model with weights Θ. We first establish the proposed model $\mathcal{N}_\Theta$ can have sufficient capacity to accurately approximate the correspondence $d \mapsto x_d^\star$ for the games of interest.

Theorem 5.2.1 (Universal Approximation) *Suppose Assumptions (A1)–(A6) hold. Then for any $\varepsilon > 0$ there exists an $F_\Theta(\cdot;\cdot)$ such that $\max_{d\in\mathcal{D}} \|x_d^\star - \mathcal{N}_\Theta(d)\|_2 \le \varepsilon$.*

Two core questions naturally arise in the implementation of $\mathcal{N}_\Theta$ in (5.6):

1. For a given d, how are inferences of $\mathcal{N}_\Theta(d)$ computed?
2. Given training data $\{d, x_d^\star\}$, how does one select weights Θ?

The second item is addressed by assuming each $x_d^\star$ takes the form of a Nash equilibrium and then minimizing a training loss function by using JFB,[9] the scheme detailed in Chapter 3. As is well-known [84], for all $\alpha > 0$,

[9] In particular, see Section 3.2.

$$x_d \in \mathrm{VI}(F_\Theta(\cdot\,; d), \mathcal{C}) \iff 0 \in F_\Theta(x_d; d) + \partial\delta_{\mathcal{C}}(x_d) \tag{5.7a}$$

$$\iff x_d = P_{\mathcal{C}}(x_d - \alpha F_\Theta(x_d; d)). \tag{5.7b}$$

Thus, when the operator $P_{\mathcal{C}} \circ (\mathrm{I} - \alpha F_\Theta)$ on the right hand side of (5.7) is tractable and well-behaved, inferences of $\mathcal{N}_\Theta(d)$ can be computed via a fixed point iteration.[10] Unfortunately computing $P_{\mathcal{C}}$ and $\mathrm{d}P_{\mathcal{C}}/\mathrm{d}z$ (required for backprop) can be prohibitively expensive (*e.g.* when each $\mathcal{V}_k$ is the intersection of sets). To overcome these difficulties, we *formulate* (5.6) *as a fixed point problem*. Our formulation, while superficially more complicated, avoids expensive projections *and* is

[10] Although not explicit, the projected gradient-type iteration requires F_Θ to be cocoercive and an appropriate step-size α to converge.

easy to backpropagate through. The key ingredient is a novel use of Davis-Yin three operator splitting [75], which admits simple and explicit formulae for each computation. To the best of our knowledge, this splitting has not already been utilized in the VI literature.

Theorem 5.2.2 *Suppose $\mathcal{C} = \mathcal{C}^1 \cap \mathcal{C}^2$ for convex $\mathcal{C}^1$ and $\mathcal{C}^2$. If both $\mathcal{C}^i$ are polyhedral or have relative interiors with a point in common and the VI admits a unique solution, then defining*

$$T_\Theta(x;d) \triangleq x - P_{\mathcal{C}^1}(x) + P_{\mathcal{C}^2}\left(2P_{\mathcal{C}^1}(x) - x - \alpha F_\Theta(P_{\mathcal{C}^1}(x);d)\right) \tag{5.8}$$

yields the equivalence[11]

$$\mathcal{N}_\Theta(d) = \mathrm{VI}\big(F_\Theta(\cdot;d), \mathcal{C}\big) = P_{\mathcal{C}^1}(z_d) \iff z_d = T_\Theta(z_d;d). \tag{5.9}$$

[11] Observe the fixed point z_d is *not* the VI solution, but its projection onto $\mathcal{C}^1$ is a solution. This distinction can be subtle, but significant in practice.

Corollary 5.2.1 *In the setting of Theorem 5.2.2, if $F_\Theta(\cdot;d)$ is α-cocoercive and z^1 is given, then the iteration $z^{k+1} = T_\Theta(z^k;d)$ yields convergence $z^k \to z_d \in \mathrm{fix}(T_\Theta(\cdot;d))$.*

Algorithm 6: Computation of Nash Equilibria (Special Case)

```
1: N_Θ(d) :                                  ◁ Input data is d
2:   x^1 ← x̃, n ← 2                          ◁ Initializations
3:   x^2 ← P_C(x^1 − α F_Θ(x^1; d))          ◁ Apply T_Θ update
4:   while ‖x^n − x^{n−1}‖ > ε               ◁ Loop to fixed point
5:     x^{n+1} ← P_C(x^n − α F_Θ(x^n; d))    ◁ Apply T update
6:     n ← n + 1                             ◁ Increment counter
7:   return x^n                              ◁ Output inference
```

The formulation (5.9) is computationally cheaper than (5.7) when $P_{\mathcal{C}_1}$ and $P_{\mathcal{C}_2}$ are computationally cheaper than $P_{\mathcal{C}}$. This is the case in many applications of interest. Practically, we use Krasnosel'skiĭ-Mann iteration to (approximately) find a fixed point of $T_\Theta(;d)$. We present a concrete instantiation of Nash model in Algorithm 7, where we simplify the update procedure by introducing auxiliary sequences $\{x^k\}$ and $\{y^k\}$. This framework may be extended to constraint sets of the form $\mathcal{C} = \mathcal{C}_1 \cap \ldots \cap \mathcal{C}_k$ and[12] $\mathcal{C} = \mathcal{C}_1 + \ldots + \mathcal{C}_k$ (discussed below). Although

[12] Here, "+" denotes the Minkowski sum of sets

Algorithm 7: Decoupled Computation of Nash Equilibria (Abstract Form)

$\mathcal{N}_\Theta(d)$: ◁ Input data is d
  $z^1 \leftarrow \tilde{z},\ z^0 \leftarrow \tilde{z},\ n \leftarrow 1$ ◁ Initialize iterates and counter
  while $\|z^n - z^{n-1}\| > \varepsilon$ **or** $n = 1$ ◁ Loop to convergence
    $x^{n+1} \leftarrow P_{\mathcal{C}^1}(z^n)$ ◁ Project onto constraint set
    $y^{n+1} \leftarrow P_{\mathcal{C}^2}(2x^{n+1} - z^n - \alpha F_\Theta(x^{n+1}; d))$ ◁ Project reflected gradient
    $z^{n+1} \leftarrow z^n - x^{n+1} + y^{n+1}$ ◁ Combine auxiliary sequences
    $n \leftarrow n + 1$ ◁ Increment counter
  return $P_{\mathcal{C}^1}(z^n)$ ◁ Output inference

we find Algorithm 7 to be most practical, other operator-based methods (*e.g.* ADMM and PDHG) can be used via equivalences similar to (5.9).

Constraint Decoupling

The provided formulation assumes $\mathcal{C}$ is expressed as the intersection of two sets $\mathcal{C}^1$ and $\mathcal{C}^2$. When an explicit and relatively simple formula exists for $P_\mathcal{C}$ (*e.g.* $\mathcal{C}$ is the probability simplex [80, 232]), one can set $\mathcal{C}^1 = \mathbb{H}$ and $\mathcal{C}^2 = \mathcal{C}$ so that the iteration in Algorithm 7 performs updates via the projected gradient-type scheme in Algorithm 6. However, it is often the case that $P_\mathcal{C}$ does not admit a closed form while $P_{\mathcal{C}^1}$ and $P_{\mathcal{C}^2}$ admit explicit and computationally cheap expressions (*e.g.* $\mathcal{C}^1 = \mathbb{R}^n_{\geq 0}$ and $\mathcal{C}^2$ an affine hyperplane). Loosely speaking, using T_Θ as in (10.112) enables replacement of a potentially "difficult" projection onto $\mathcal{C}$ with "easy" projections[13] onto individual constraints $\mathcal{C}^i$. More generally, in some real-world applications (*e.g.* traffic routing), the set $\mathcal{C}$ is a Minkowski sum of intersections of simple sets, *i.e.* $\mathcal{C} = \mathcal{C}_1 + \cdots + \mathcal{C}_K$ where $\mathcal{C}_k = \mathcal{C}_k^1 \cap \mathcal{C}_k^2$. By using a product space, the constraints may be further decoupled, thereby avoiding an iterative subroutine to compute $P_\mathcal{C}$ for each update. This is shown[14] by Algorithms 8 and 9. In particular, our decoupled scheme updates only require *a single application of each* $P_{\mathcal{C}_k^i}$. See the appendices for lemmas proving the equivalences.

[13] This does imply the iterate x^n estimating $\mathcal{N}_\Theta(d)$ may not, at any iteration, be feasible. That is, one can have $x^n \in \mathcal{C}^1$ and $x^n \notin \mathcal{C}$ for all $n \in \mathbb{N}$ while $x^n \to \mathcal{N}_\Theta(d)$.

[14] Each overlined element $\overline{x}$ is in the K-product space $\underbrace{\mathcal{U} \times \cdots \times \mathcal{U}}_{K \text{ terms}}$.

Crucially, decoupling projections onto $\mathcal{C}$ into projections onto each $\mathcal{C}_k^i$ that

Algorithm 8: Nash Equilibria (Minkowski Sum $\mathcal{C} = \mathcal{C}_1 + \cdots + \mathcal{C}_K$)

$\mathcal{N}_\Theta(d)$: ◁ Input data is d
  $n \leftarrow 1$ ◁ Initialize counter
  for $k = 1, 2, \ldots, K$
    $\overline{z}_k^1 \leftarrow \hat{z}$ ◁ Initialize iterates to $\hat{z} \in \mathbb{H}$
  while $\sum_{k=1}^K \|\overline{z}_k^n - \overline{z}_k^{n-1}\| > \varepsilon$ **or** $n = 1$ ◁ Loop until convergence
    for $k = 1, 2, \ldots, K$ ◁ Loop over constraints $\mathcal{C}_k^1$
      $\overline{x}_k^{n+1} \leftarrow P_{\mathcal{C}_k^1}(\overline{z}_k^n)$ ◁ Project onto constraint set
    $v^{n+1} \leftarrow \sum_{k=1}^K \overline{x}_k^{n+1}$ ◁ Combine projections
    for $k = 1, 2, \ldots, K$ ◁ Loop over constraints $\mathcal{C}_k^2$
      $\overline{y}_k^{n+1} \leftarrow P_{\mathcal{C}_k^2}(2\overline{x}_k^{n+1} - \overline{z}_k^n - \alpha F_\Theta(v^{n+1}; d))$ ◁ Project reflected gradients
      $\overline{z}_k^{n+1} \leftarrow \overline{z}_k^n - \overline{x}_k^{n+1} + \overline{y}_k^{n+1}$ ◁ Apply block-wise updates
    $n \leftarrow n + 1$ ◁ Increment counter
  return v^n ◁ Output inference

possess analytic formulas enables efficient forward propagation (*i.e.* evaluation of $\mathcal{N}_\Theta$) and backward propagation (to tune weights Θ) through the projections. These projection formulas can be directly coded into the feed forward operation for $\mathcal{N}_\Theta$, enabling built-in autodifferentation tools to perform backpropagation.

Limitations Our approach tunes an operator so that its fixed points match given contextual Nash equilibria, but says little about the players' cost functions. Thus, T_Θ cannot be used to design interventions to increase social welfare (*i.e.* the negative of the sum of all players costs) [201, 145, 154]. However, T_Θ *can* be used to design interventions to discourage agents from playing a given action.

Related Works There are two distinct learning problems for games. The first considers repeated rounds of the same game and operates from the player's perspective. Players have imperfect knowledge of the game, and the goal is to learn the optimal strategy (*i.e.* the Nash equilibrium or, more generally, a coarse correlated equilibrium), given only the cost incurred in each round. This problem is not studied here, and we refer readers to [119, 121, 222, 217] for details.

Algorithm 9: Nash Equilibria (Intersection Constraints $\mathcal{C} = \mathcal{C}_1 \cap \cdots \mathcal{C}_K$)

$\mathcal{N}_\Theta(d)$: ◁ Input data is d
  $n \leftarrow 1$ ◁ Initialize counter
  for $k = 1, 2, \ldots, K$
    $\overline{z}_k^1 \leftarrow \hat{z}$ ◁ Initialize iterates
  while $\sum_{k=1}^K \|\overline{z}_k^n - \overline{z}_k^{n-1}\| > \varepsilon$ **or** $n = 1$ ◁ Loop until convergence
    for $k = 1, 2, \ldots, K$
      $\overline{x}_k^{n+1} \leftarrow P_{\mathcal{C}_k}(\overline{z}_k^n)$ ◁ Project onto each $\mathcal{C}_k$
    $v^{n+1} \leftarrow \frac{1}{K} \sum_{k=1}^K \left(2\overline{x}_k^{n+1} - \overline{z}_k^n - \alpha F_\Theta(\overline{x}_k^{n+1}; d)\right)$ ◁ Average reflected
    for $k = 1, 2, \ldots, K$ "gradient" updates
      $\overline{z}_k^{n+1} \leftarrow \overline{z}_k^n - \overline{x}_k^{n+1} + v^{n+1}$ ◁ Combine sequences
    $n \leftarrow n + 1$ ◁ Increment counter
  return $\overline{x}_1^n$ ◁ Output inference

The second problem supposes historical context-action data pairs $(d, x_d^\star)$ are available to an external observer. The observer wishes to learn something about the players' cost functions, assuming each $x_d^\star$ is (approximately) a Nash equilibrium. The works [236, 201, 145, 33, 249, 248, 250, 7] and many others approach this as an inverse problem, positing a parametrized form of each players cost function and then tuning these weights to minimize empirical risk. Several recent works [162, 161, 154] abandon learning cost functions directly in favor of learning an operator approximating the game gradient within an *implicit deep learning* framework, which is in line with our approach. A differentiable game solver for two player games is proposed in [162]. Backpropagation is done by solving a $p \times p$ linear system[15] for each $(d, x_d^\star)$. As pointed out in [161], this is prohibitively expensive for high dimensional games. In [161], this approach was modified, leading to a fast backpropagation algorithm, but only for two-player games admitting a compact extensive form representation. Moreover, both [162] and [161] only consider the case where $\mathcal{C}$ is a product of simplices, and these works avoid projections by only considering particular types of games[16]. The

[15] In this section, p denotes the dimension of the action space $\mathbb{H}$.

[16] Precisely, they add an entropic regularizer, which guarantees x_d is in the interior of $\mathcal{C}$.

	R	P	S
R	0	$-\langle w^1, d\rangle$	$\langle w^2, d\rangle$
P	$\langle w^1, d\rangle$	0	$-\langle w^3, d\rangle$
S	$-\langle w^2, d\rangle$	$\langle w^3, d\rangle$	0

$$\longrightarrow \quad B(d) \triangleq \begin{bmatrix} 0 & -\langle w^1, d\rangle & \langle w^2, d\rangle \\ \langle w^1, d\rangle & 0 & -\langle w^3, d\rangle \\ -\langle w^2, d\rangle & \langle w^3, d\rangle & 0 \end{bmatrix}$$

Table 5.2: Payoff matrix $B(d)$ for contextual Rock-Paper-Scissors (RPS).

approach of [154] is most similar to ours, but crucially they do not exploit constraint decoupling or Jacobian-free backpropagation. Instead, they use an iterative $\mathcal{O}(p^3)$ algorithm [8] to compute $P_{\mathcal{C}}$ and $\mathrm{d}P_{\mathcal{C}}/\mathrm{d}z$ and solve a Jacobian-based equation in every backward pass.

We also highlight recent work applying deep learning to traffic flow prediction [71, 156, 113, 216]. Unlike us, these works consider *non-equilibrium* traffic flows and use fine-grained spatiotemporal data (*e.g.* traffic densities on every road segment at five minute intervals [71]) to predict the traffic flow in the near future. This is in contrast with our approach of using coarse-grained data (*e.g.* weather) to predict the equilibrium traffic flow. A common method for solving the VI arising in traffic routing (see Section 5.3) is the Frank-Wolfe algorithm [91]. More sophisticated approaches are given in [22, 77] and [23]. Although the equivalence (5.7) is well-known in this community (see, *e.g.* [193, 23]), we are not aware of any prior works utilizing operator splitting to decouple the constraints.

Section 5.3: Experiments

We show the efficacy of Nash models on two games types: rock-paper-scissors and traffic routing. JFB is used throughout for model training.

Rock Paper Scissors

We perform a rock-paper-scissors experiment similar to [162]. Each player's actions are restricted to the unit simplex $\Delta^3 \triangleq \{x \in \mathbb{R}^3_{\geq 0} : \|x\|_1 = 1\} \subset \mathbb{R}^3$ so that $\mathcal{C} = \Delta^3 \times \Delta^3$ and actions x_i are interpreted as probability distributions over three choices: "rock", "paper" and "scissors." Equilibria $x_d^\star$ are drawn from

VI($F(\cdot\,;d),\mathcal{C}$), using the game gradient F in (5.2) with cost functions given by

$$u_1(x;d) \triangleq \langle x_1, B(d)x_2\rangle \quad \text{and} \quad u_2(x;d) \triangleq -\langle x_1, B(d)x_2\rangle\,, \tag{5.10}$$

where the payoff matrix $B(d) \in \mathbb{R}^{3\times 3}$ (see Table 5.2) defines the players' cost functions using $w^i \in \mathbb{R}^3_{\geq 0}$. Contextual data d are drawn from a distribution $\mathcal{D}$ that is uniform over $[0,1]^3$. An Nash model is trained to predict $x_d^\star$ from d using training data context-action pairs $\{(d^i, x_{d^i}^\star)\}_{i=1}^{1000}$, without using knowledge of F. The tunable operator F_Θ in (10.112) consists of a residual update with two fully connected layers and a leaky ReLU activation. Forward propagation uses Algorithm 6. For illustration, we simulate play between two players. The first player acts optimally using knowledge of $B(d)$ and the second player's strategy. Three options are used for the second player: another optimal player, a player that only has access to d trained to move according to the Nash-based model, and uniform choices. With two optimal players, a Nash equilibria is obtained where the expected cost after each game is zero. If the Nash-based model is well-trained, then the second case yields the same result. In the final case, the optimal player has an advantage, yielding first player costs less than zero (*i.e.* the first player usually wins). Here

$$\text{(Exp. Abs. Nash Player } k\text{-Game Ave. Cost)} \equiv y^k \triangleq \mathbb{E}_{d\sim\mathcal{D}}\left[\left|\frac{1}{k}\sum_{\ell=1}^{k} u_1\big(s^\ell;d\big)\right|\right], \tag{5.11}$$

where s^k is a tuple of two one-hot vectors (*e.g.* $s_1^k \sim x_d^\star$ and $s_2^k \sim \mathcal{N}_\Theta(d)$). If $\mathcal{N}_\Theta(d) = x_d^\star$, then the expected cost u_1 is zero and $y^k \to 0$ (*n.b.* simulated games have nonzero variance due to one-hot sampling s_i^k whereas $x_d^\star$ is continuous). This behavior is illustrated in Figure 5.2.

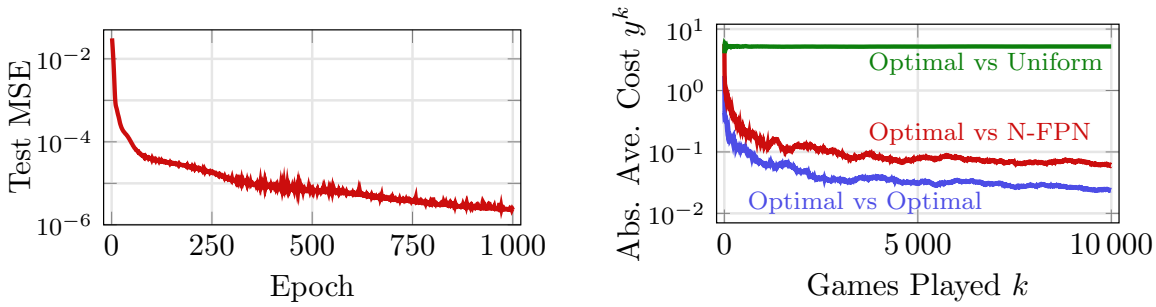


Figure 5.2: Rock-paper-scissors example. Left plot shows test loss during training. Right plot shows the cost expression y^k over the course of k games in three settings. The first player always acts optimally, knowing both the true cost $u_1(\cdot; d)$ and the second player's strategy. The second player either also acts optimally, chooses uniformly randomly, or uses trained model predictions only knowing d. Both players acting optimally yields a Nash equilibrium, making $y^k \to 0$. When the second player is uniform, the first player typically wins. This plot shows the modeled Nash equilibria player chooses nearly optimally.

Contextual Traffic Routing

Setup Consider a road network represented by a directed graph with vertices V and arcs E. Let $N \in \mathbb{R}^{|V|\times|E|}$ denote the vertex-arc incidence matrix. An origin-destination pair (OD-pair) is a triple (v_1, v_2, q) with $v_i \in V$ and $q \in \mathbb{R}_{>0}$, encoding the constraint of routing q units of traffic from v_1 to v_2. Each OD-pair is encoded by a vector $b \in \mathbb{R}^{|V|}$ with $b_{v_1} = -q$, $b_{v_2} = q$ and all other entries zero. A valid *traffic flow* $x \in \mathbb{R}^{|E|}$ for an OD-pair has nonnegative entries satisfying the flow equation $Nx = b$. The e-th entry x_e represents the traffic density along the e-th arc. The flow equation ensures the number of cars entering an intersection equals the number leaving, except a net movement of q units of traffic from v_1 to v_2. For K OD-pairs, a valid traffic flow x is the sum of traffic flows for each OD-pair, which is in the Minkowski sum

$$\mathcal{C} = \mathcal{C}_1 + \cdots + \mathcal{C}_K, \quad \text{with} \quad \mathcal{C}_k = \underbrace{\{x : Nx = b_k\}}_{\mathcal{C}_k^1} \bigcap \underbrace{\{x : x \geq 0\}}_{\mathcal{C}_k^2} \quad \text{for all } k \in [K]. \tag{5.12}$$

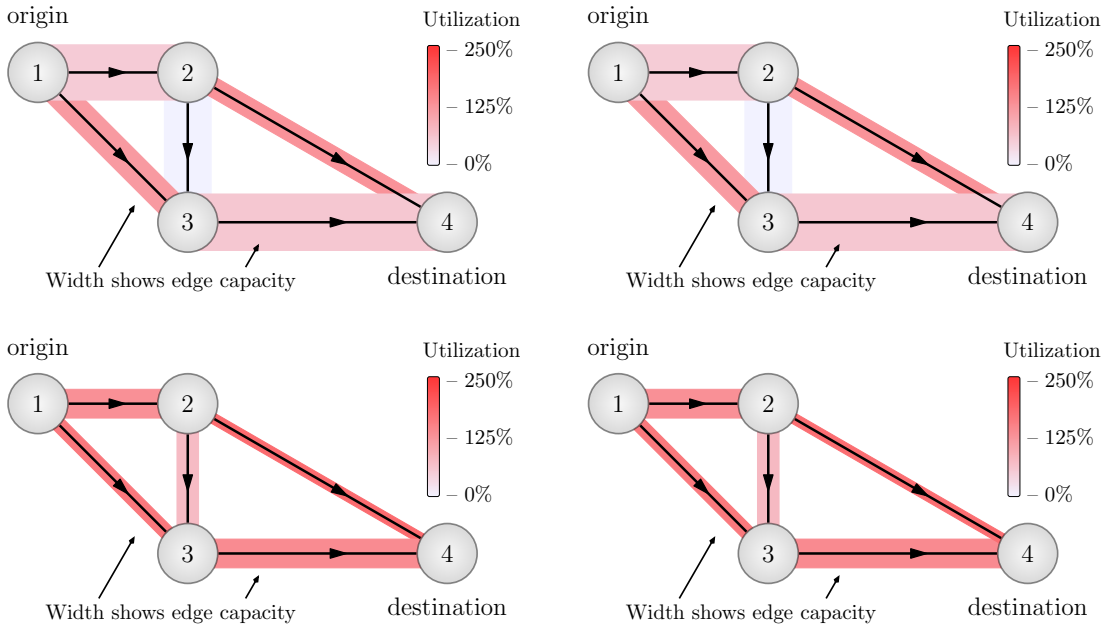


Figure 5.3: Top left: True traffic flow for "sunny" context. Top right: Predicted traffic by $\mathcal{N}_\Theta$ for "sunny" context. Bottom left: True traffic flow for "rainy" context. Bottom right: Predicted traffic by $\mathcal{N}_\Theta$ for "rainy" context

A *contextual travel time function*[17] $t_e(x_e; d)$ is associated with each arc, where d encodes contextual data. Here the equilibrium of interest is, roughly speaking, a flow configuration $x_d^\star$ where the travel time between each OD-pair is as short as possible when taking into account congestion effects [44]. This is known as a *Wardrop equilibrium* (also called the *user equilibrium*) [235], a special case of Nash equilibria where $F = [t_1(x_1; d)^\top \cdots t_{|E|}(x_{|E|}; d)^\top]^\top$. In certain cases, a Wardrop equilibrium is the limit of a sequence of Nash equilibria as the number of drivers goes to infinity [190].

[17] This time is monotonically increasing as a function of traffic density x_e (for any fixed d).

TRAFIX Scores Accuracy of traffic routing predictions are measured by a *TRAFIX score*. This score forms an intuitive alternative to mean squared error. An error tolerance $\varepsilon > 0$ is chosen (*n.b.* $\varepsilon = 5 \times 10^{-3}$ in our experiments). For an estimate x of $x^\star$, the TRAFIX score with parameter ε is the percentage of edges for which x has relative error (with tolerance[18] $\tau > 0$) less than ε, *i.e.*

[18] The parameter τ is added to handle the case when the e-th component of $x^\star$ is zero, *i.e.* $x_e^\star = 0$.

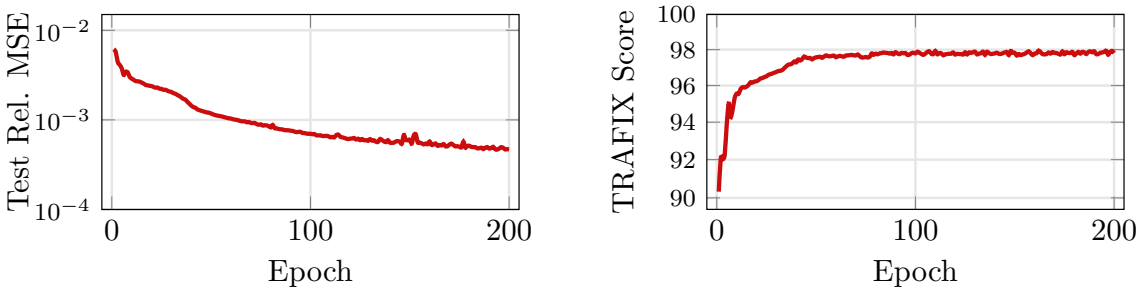


Figure 5.4: Plots for Nash model performance on Eastern Massachusetts testing data. The left plot shows convergence of expected relative mean squared error on testing data after each training epoch and the right shows the expected TRAFIX score on testing data after each training epoch.

$$\text{(relative error of edge } e) \triangleq \frac{|x_e - x_e^\star|}{|x_e^\star| + \tau}, \tag{5.13a}$$

$$\text{TRAFIX}(x, x^\star; \varepsilon, \tau) \triangleq \frac{\text{(\# edges with relative error} < \varepsilon)}{\text{(\# edges)}} \times 100\%. \tag{5.13b}$$

Our plots and tables show expected TRAFIX scores over testing data.

Datasets and Training We construct datasets for a (collection of) large scale datasets constructed from the traffic networks of real-world cities curated by the Transportation Networks for Research Project [224]. We construct this data by fixing a choice of $t_e(x;d)$ for each arc e, randomly generating a large set of contexts $d \in [0,1]^{10}$ and then, for each d, finding a solution x_d in $\text{VI}(F(\cdot\,;d), \mathcal{C})$. For illustrative purposes, we also consider a toy example, illustrated in Figure 5.3. We train a Nash model using Algorithm 8 for forward propagation to predict $x_d^\star$ from d for each data set.

Results Table 5.3 shows a description of the traffic networks datasets, including the numbers of edges, nodes, and OD-pairs. To illustrate the effectiveness of Nash models for each setting, this table also shows the number of tunable parameters in column four, relative mean squared error (MSE) in column five and the TRAFIX score in column six for the testing dataset. The convergence during training of the relative MSE and TRAFIX score on the Eastern-Massachusetts testing dataset is shown in Figure 5.4. Additional plots can be found in the appendices.

dataset	edges/nodes	OD-pairs	# params	rel. MSE	TRAFIX score
Sioux Falls	76/24	528	46K	1.9×10^{-3}	94.42%
Eastern Massachusetts	258/74	1113	99K	4.7×10^{-4}	97.94%
Berlin-Friedrichshain	523/224	506	179K	5.3×10^{-4}	97.42%
Berlin-Tiergarten	766/361	644	253K	7.6×10^{-4}	95.95%
Anaheim	914/416	1406	307K	2.4×10^{-3}	95.28%

Table 5.3: Expected values of Nash model predictions on traffic routing test data. First and second columns show the number of edges, nodes, and origin-destination pairs for corresponding dataset. Second column shows number of tunable parameters.

Section 5.4: Conclusions

The fusion of big data and optimization algorithms offers potential for predicting equilibria in systems with many interacting agents. The models proposed in this chapter form a *scalable* data-driven framework for efficiently predicting equilibria for such systems that can be modeled as contextual games. The model architecture yields equilibria outputs that satisfy constraints while also being trained end-to-end. Moreover, the provided constraint decoupling schemes enable simple forward and backward propagation using explicit formulae for each projection. The efficacy of the models is illustrated on large-scale traffic routing problems using a contextual traffic routing benchmark dataset and TRAFIX scoring system.[19] Future work will investigate applications on larger datasets, convergence acceleration, and weakening assumptions for convergence.

[19] Returning to the quote by Faraday, this chapter develops a novel tool, but we cannot yet know how useful it is until new datasets are made available to truly find its computational limitations.

Chapter 6: Explainable L2O Models

[W]hat is important is the gradual development of a theory, based on a careful analysis of the ordinary everyday interpretation of economic facts. This preliminary stage is necessarily heuristic, i.e. the phase of transition from unmathematical plausibility consideration to the formal procedure of mathematics. The theory finally obtained must be mathematically rigorous and conceptually general. Its first applications are necessarily to elementary problems where the result has never been in doubt and no theory is actually required.

– John Von Neumann[1]

[1] Taken from the concluding remarks in Section 1 of the seminal textbook [231] on game theory and economic behavior.

A current paradigm shift in machine learning is to construct explainable and transparent models, often called explainable AI (XAI). This is a crucial task for sensitive applications like medical imaging and finance (*e.g.* see recent expositions on the role of explainability [14, 1, 79, 214]). However, commonplace models (*e.g.* fully connected models) do not often offer this interpretability. Some works have provided explanations using sensitivity analysis [214] or layer-wise propagation [17], but these do not necessarily shed light on how to correct "bad" behaviors or concretely quantify whether an inference is trustworthy. This chapter shows how L2O can be used to directly embed explainability into the structure of models to obtain trustworthiness and interpretability.

As before, we focus on ML applications where domain experts can create approximate models by hand.[2] Here each inference $\mathcal{N}_\Theta(d)$ of a model $\mathcal{N}_\Theta$ with input d solves an optimization problem. That is, we use models defined by

[2] Unlike previous chapters with games and feasibility problems, here we solely consider minimization problems.

$$\mathcal{N}_\Theta(d) \triangleq \underset{x \in \mathcal{C}_\Theta(d)}{\arg\min}\, f_\Theta(x; d), \tag{6.1}$$

where f_Θ is a function and $\mathcal{C}_\Theta(d) \subseteq \mathbb{R}^n$ is a constraint set (*e.g.* encoding prior information like physical quantities), and each (possibly) includes dependencies on weights Θ.[3]

[3] Note the model $\mathcal{N}_\Theta$ is *implicit* since its output is defined by an optimality condition rather than an explicit computation. This notation intentionally coincides with the notation in Section 1.2.

Switching gears, a standard practice in software engineering is to code post-conditions after function calls return. Post-conditions are criteria used to validate what the user expects from the code and ensure code is not executed under the wrong assumptions [9]. We propose use of these for ML model inferences (see Figures 6.1 and 6.5). These conditions enable use of certificates with labels – pass, warning or fail – to describe each model inference. We define[4] an inference to be *trustworthy provided it satisfies all post-conditions*.

[4] Certainly, many alternatives exist for trustworthiness; however, we believe our choice is apt for the situations considered in that it is unambiguous.

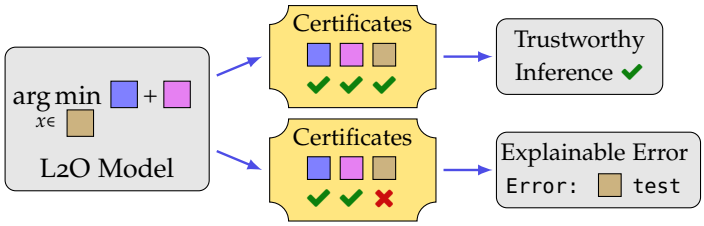


Figure 6.1: Each L2O model is composed of parts (shown as colored blocks) that are analytic or learned. L2O inferences solve an optimization problem for given model inputs. Certificates label if each inference is consistent with training data. If so, it is trustworthy; otherwise, the faulty model part errs.

These two discussed ideas, optimization and certificates, form a concrete notion of XAI. Prior and data-driven knowledge can be encoded via optimization, and this encoding can be verified via certificates (see Figure 6.1). To illustrate, consider inquiring why a model generated a "bad" inference (*e.g.* an inference disagrees with observed measurements). The first diagnostic step is to check certificates. If no fails occurred, the model was not designed to handle the instance encountered. In this case, the model in (6.1) can be redesigned to encode prior knowledge of the situation. Alternatively, each failed certificate shows a type of error and often corresponds to portions of the model (see Figures 6.1 and 6.2). The L2O model allows debugging of algorithmic implementations and assumptions to correct errors. In a sense, this setup enables one to manually backpropagate errors to fix models (similar to training).

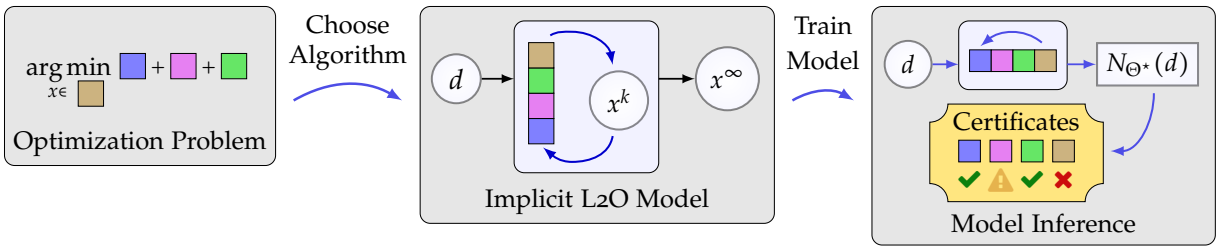


Figure 6.2: Diagram of L2O architectures and trustworthiness certificates. Colored blocks denote prior knowledge and data-driven terms. Middle shows an iterative algorithm formed from the blocks (*e.g.* corresponding proximal operators) to solve the optimization problem. Right shows a model inference $N_{\Theta^\star}(d)$ and associated certificates identifying if properties of inferences are consistent with training data. For troubleshooting, each label is associated with properties of specific blocks (indicated by labels next to blocks). Labels take value pass ✔, warning ⚠, or fail ✖. Each certificate identifies if inference features for model parts are trustworthy.

Contributions This chapter brings new explainability and guarantees to machine learning applications using prior knowledge. We propose novel implicit L2O models with intuitive design, memory efficient training, inferences that satisfy optimality/constraint conditions, and certificates that either indicate trustworthiness or flag inconsistent inference features.

Related Works

Closely related to our work is deep unrolling, a subset of L2O wherein models consist of a fixed number of iterations of a data-driven optimization algorithm. Deep unrolling has garnered great success and provides intuitive model design. We refer readers to recent surveys [62, 219, 177] for further L2O background. Downsides of unrolling primarily include growing memory requirements with unrolling depth and a lack of guarantees.[5]

Implicit models circumvent these two shortcomings by defining models using an equation (*e.g.* as in (6.1)) rather than prescribe a fixed number of computations as in deep unrolling. This enables inferences to be computed by iterating until convergence, thereby enabling theoretical guarantees. Memory-efficient

[5] Although it is difficult to verify in general, comparing the Safe L2O experiments to those with implicit models as in this chapter, it seems implicit models are more stable (*i.e.* "easier") to train.

training techniques were also developed for this class of models, which have been applied successfully in games [126], music source separation [147], language modeling [18], segmentation [19], and inverse problems [125, 103]. The recent work [103] most closely aligns with our L2O methodology.

Related XAI works use labels/cards similar to our proposed certificates. Model Cards [175] document intended and appropriate uses of models; these are short documents that companion trained models and provide benchmarked evaluation in a variety of conditions (*e.g.* across different cultural, demographic, or phenotypic group). Care labels [180, 181] are similar, testing properties like expressivity, runtime, and memory usage. FactSheets [11] are another option, modeled after supplier declarations of conformity and aim to identify models' intended use, performance, safety, and security. These works push forward the role of explainability and complement this work. Our proposed scheme differs both in quantities measured and methodology. Rather than provide statistics at the distribution level, our certificates are focused on providing trustworthiness feedback at the level individual inferences. Unlike these works, our certificates can check with any given data (*e.g.* inference from a different model) is consistent with the true data, even if it was *not* produced using the trained model.[6]

[6] This is illustrated in our CT experiments below where certificates are created not only for the proposed models' inferences, but also for other approaches.

L2O	Implicit	Flags	Model Property
✓			Intuitive Design
	✓		Memory Efficient
✓	✓		Satisfy Constraints + (above)
		✓	Trustworthy Inferences
✓	✓	✓	Explainable Errors + (above)

Table 6.1: Summary of design features and corresponding model properties. Design features yield additive properties, as indicated by "+ (above)." Proposed implicit L2O models with certificates have intuitive design, memory efficient training, inferences that satisfy optimality/constraint conditions, certificates of trustworthiness, and explainable errors.

Model Design L2O model design is naturally decomposed into two steps: optimization formulation and algorithm choice. The first step identifies a tentative objective to encode prior knowledge via regularization (*e.g.* sparsity) or constraints (*e.g.* unit simplex for classification). We may add entirely data-driven terms too. Informally, this step identifies a special case of (6.1) of the form[7]

$$\mathcal{N}_\Theta(d) \triangleq \arg\min_x (\text{prior knowledge}) + (\text{data-driven terms}), \tag{6.2}$$

[7] Constraints are encoded in the objective using indicator functions, equaling ∞ when constraint is satisfied and 0 otherwise.

The second design step is to choose an algorithm for solving the chosen optimization problem (*e.g.* proximal-gradient or ADMM). We use iterative algorithms, and the update formula for each iteration is given by a *model operator* $T_\Theta(x; d)$. Updates are typically composed in terms of gradient and proximal operations. Some parameters (*e.g.* step sizes) may be included in the weights Θ to be tuned during training. Given data d, computation of the inference $\mathcal{N}_\Theta(d)$ is completed by generating a sequence $\{x_d^k\}$ via the relation

$$x_d^{k+1} = T_\Theta(x_d^k; d), \quad \text{for all } k \in \mathbb{N}. \tag{6.3}$$

By design, $\{x_d^k\}$ converges to a solution of (6.1), and we set[8]

$$\mathcal{N}_\Theta(d) = \lim_{k\to\infty} x_d^k. \tag{6.4}$$

[8] If the optimization problem has multiple solutions, uniqueness of $\mathcal{N}_\Theta(d)$ is obtained in practice by fixing the initialization x_d^1.

In our context, each model inference $\mathcal{N}_\Theta(d)$ is defined to be an optimizer as in (6.1). Hence *properties of inferences can be interpreted via the optimization model* (6.1). The iterative algorithm is applied successively until stopping criteria are met (*i.e.* in practice we choose an iterate K, possibly dependent on d, so that $\mathcal{N}_\Theta(d) \approx x_d^K$). Since $\{x_d^k\}$ converges, we may adjust stopping criteria to approximate the limit to arbitrary precision, which implies we may provide guarantees on model inferences (*e.g.* satisfying a linear system to a desired precision [103, 125, 126]).

Example of Model Design. To make the model design procedure concrete, we illustrate this process on a classic problem: sparse recovery from linear measurements. Here the task is to estimate a signal $x_d^\star$ via access to linear measurements d satisfying $d = Ax^\star$ for a known matrix A.

Step 1. Since true signals are known to be sparse, we include ℓ_1 regularization. To comply with measurements, we add a fidelity term. Lastly, to capture hidden features of the data distribution, we also add a data-driven regularization. Putting these together gives the problem

$$\min_{x\in\mathbb{R}^n} \underbrace{\tau\|x\|_1}_{\text{sparsity}} + \underbrace{\frac{1}{2}\|Ax-d\|_2^2}_{\text{fidelity}} + \underbrace{\frac{1}{2}\|W_1x\|^2 + \langle x, W_2 d\rangle}_{\text{data-driven regularizer}}, \tag{6.5}$$

where $\tau > 0$ and W_1 and W_2 are two tunable matrices. This model encodes a balance of three terms: sparsity, fidelity, data-driven regularization.

Step 2. The proximal-gradient scheme generates a sequence $\{z^k\}$ converging to a limit which solves (6.5). Upon simplifying and combining terms, the proximal-gradient method can be expressed via the iteration[9]

$$z^{k+1} = \eta_{\tau\lambda}\big(z^k - \lambda W(Az^k - d)\big), \quad \text{for all } k \in \mathbb{N}, \tag{6.6}$$

[9] The shrink η_θ is given by $\eta_\theta(x) \triangleq \operatorname{sign}(x)\max(|x|-\theta, 0)$.

where $\lambda > 0$ is a step-size and W is a matrix defined in terms of W_1, W_2, and $A^\top$. From the update on the right hand side of (6.6), we see the step size λ can be "absorbed" into the tunable matrix W and the shrink function parameter can set to $\theta > 0$. That is, this example model has weights $\Theta = (W, \theta, \tau)$ with model operator

$$T_\Theta(x; d) \triangleq \eta_\theta\big(x - W(Ax - d)\big), \tag{6.7}$$

which resembles the updates of previous L2O works.[163, 112, 64] Inferences can be computed by generating a sequence $\{x_d^k\}$ via the iteration

$$x_d^{k+1} = T_\Theta(x_d^k; d), \quad \text{for all } k \in \mathbb{N}. \tag{6.8}$$

The model inference is the limit x_d^∞ of this sequence $\{x_d^k\}$.

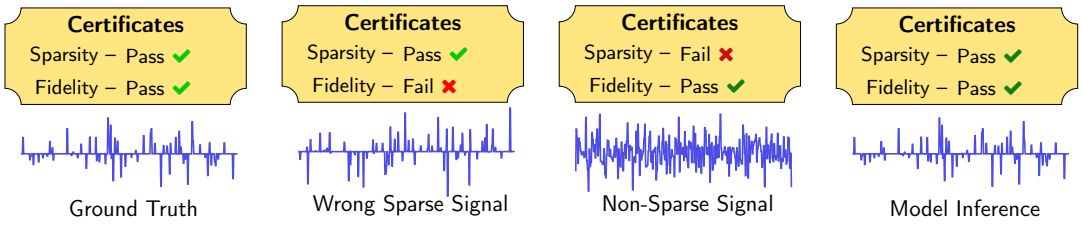


Figure 6.3: Example inferences for test data d. The sparsified version Kx of each inference x is shown (*c.f.* Figure 6.6) along with certificates. Ground truth was taken from test dataset of analysis dictionary experiment. The second from left is sparse and inconsistent with measurement data. The second from right complies with measurements but is not sparse. The rightmost is generated using our proposed model (ADM), which approximates the ground truth well and is trustworthy.

Convergence Evaluation of the model $\mathcal{N}_\Theta(d)$ is well-defined and tractable under a simple assumption. As stated in the introduction, by the classic result of Banach [21], it suffices to ensure T_Θ be γ-Lipschitz with respect to x for a $\gamma \in [0,1)$, *i.e.*

$$\|T_\Theta(x;d) - T_\Theta(v;d)\| \leq \gamma\|x - v\|, \quad \text{for all } x, v, d. \tag{6.9}$$

When this property holds, the sequence $\{x^k\}$ in (6.3) converges linearly. This may appear to be a strong assumption; however, common operations in convex optimization algorithm (*e.g.* proximals) are τ-Lipschitz for some $\tau \in [0,1]$. For entirely data-driven portions of T_Θ, we note several activation functions are 1-Lipschitz [68, 98] (*e.g.* ReLU and softmax).[10,11]

[10] Packages like PyTorch [192] include functions to force affine mappings to be γ-Lipschitz (*e.g.* spectral normalization).

[11] Even without forcing T_Θ to be a contraction, it has been observed for trained models that $\{x^k\}$ often converges in practice [125, 18, 103].

Section 6.2: Trustworthiness Certificates

Explainable models justify whether each inference is trustworthy. We consider providing justification in the form of certificates that verify various properties of the inference are consistent with model inferences on training data and/or prior knowledge. Each certificate is a tuple of the form (name, label) with a property name and a corresponding label which has one of three values: pass, warning, or fail. This section gives a framework for using these certificates.[12]

[12] For certain checks, this will be obvious, which is in line with Von Neumann's quote. As we will see, a perhaps unexpected novelty is the ability to now quantify reliability using regularization.

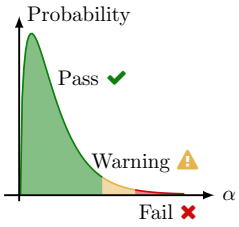


Figure 6.4: Example probability curve for property values α. Lower values of α are desired. Each value has a label: pass, warning, or fail. These are shown by green, yellow, and red.

Certificate Design Each certificate label is generated by two steps. The first is to apply a function that maps inferences (or intermediate states) to a *nonnegative scalar value* α quantifying a property of interest. The second step is to map this scalar to a label. Labels are generated via the flow:

$$\text{Inference} \to \text{Property Value} \to \text{Certificate Label.} \tag{6.10}$$

The property value function is assumed to be given and can take various forms. In the model design example above, a sparsity property can be quantified by counting the number of nonzero entries in a signal, and a fidelity property can use the relative error $\|Ax - d\|/\|d\|$ (see Figure 6.3). Given property values, we next assign labels according to the probabilities of each property value.

Typical certificate labels should follow a trend where inferences often obtain a pass label to indicate trustworthiness while warnings occur occasionally and failures are obtained in extreme situations. Let $\mathcal{A}$ be the sample space of model inference property values α generated from training data. Since small values of α are desirable, labels are assigned according to the probability of observing a value less than or equal to α. That is, we evaluate the cumulative density function (CDF) defined for probability measure $\mathbb{P}_{\mathcal{A}}$ by

$$\text{CDF}(\alpha) = \int_0^{\alpha} \mathbb{P}_{\mathcal{A}}(\overline{\alpha})\, \mathrm{d}\overline{\alpha}, \tag{6.11}$$

Labels are chosen according to the task at hand. Let p_{p}, p_{w}, and $p_{\text{f}} = 1 - p_{\text{p}} - p_{\text{w}}$ be the desired probabilities for the pass, warning, and fail labels, respectively. Assignments are made for α via

$$\text{Label}(\alpha) = \begin{cases} \text{pass} & \text{if CDF}(\alpha) < p_{\text{p}} \\ \text{warning} & \text{if CDF}(\alpha) \in [p_{\text{p}}, 1 - p_{\text{f}}) \\ \text{fail} & \text{otherwise.} \end{cases} \tag{6.12}$$

The remaining task is to estimate the CDF. This is done using a map ϕ_Λ parameterized by weights Λ. The probability measure $\mathbb{P}_\mathcal{A}$ can be estimated from α samples drawn from $\mathcal{A}$, and the derivative ϕ'_Λ forms a probability measure that should approximate $\mathbb{P}_\mathcal{A}$. Consequently, the "ideal" weights $\Lambda^\star$ minimize a divergence D (*e.g.* Wasserstein or Kullback–Leibler) between the probability measure ϕ'_Λ and the true probability $\mathbb{P}_\mathcal{A}$, *i.e.*

$$\Lambda^\star \in \arg\min_\Lambda D\left(\phi'_\Lambda \mid \mathbb{P}_\mathcal{A}\right). \tag{6.13}$$

The training problem (6.13) can be solved using a variant of stochastic gradient descent (SGD) [36, 205] or ADAM [142].

Certificate Implementation As noted in the introduction, trustworthiness certificates are proof that an inference satisfies post-conditions (*i.e.* passes various tests). Thus, they are to be used in code in the same manner as standard software engineering practice. Consider the snippet of code in Figure 6.5. As usual, an inference is generated by calling the model. However, alongside the inference `x`, certificates `certs` are returned that label whether the inference `x` passes tests that identify consistency with training data and prior knowledge.

Inference + Certificates → Trustworthy Inference

```
def TrustworthyInference(d):
    x, certs = model(d)
    if 'warning' in certs:
        warnings.warn('Warning Msg')
    if 'fail' in certs:
        raise Exception('Error Msg')
    return x
```

Figure 6.5: Python snippet code to use certificates as post-conditions. Actual implementation should use specific warning/exception messages for flagged entries in `certs`.

Concept	Quantity	Formula
Sparsity	Nonzeros	$\|x\|_0$
Measurements	Relative Error	$\|Ax - d\|/\|d\|$
Constraints	Distance to Set $\mathcal{C}$	$d_{\mathcal{C}}(x)$
Smooth Images	Total Variation	$\|\nabla x\|_1$
Classifier Confidence	Probability short of one-hot label	$1 - \max_i x_i$
Convergence	Iterate Residual	$\|x^k - x^{k-1}\|$
Regularization	Prox Residual	$\|x - \text{prox}_{f_\Omega}(x)\|$

Table 6.2: Examples of certificate property value choices. Each certificate is tied to a high-level concept, and then quantified in a formula. For classifier confidence, we assume x is in the unit simplex. The proximal is a data-driven update for f_Ω with weights Ω.

Property Value Functions Several quantities may be used to generate certificates. To be most effective, these are chosen to coincide with the optimization problem used to design the L2O model, *i.e.* to quantify structure of prior and data-driven knowledge. This enables each certificate to clearly validate a portion of the model (see Figure 6.2). Since various concepts are useful for different types of modeling, we provide a brief (and non-comprehensive) list of ideas and possible corresponding property values in Table 6.2.

One property concept deserves particular attention: data-driven regularization. This form of regularization is important for discriminating between inference features that are qualitatively intuitive but difficult to quantify by hand. Rather than approximate a function, implicit L2O models directly approximate gradients/proximals. These provide a way to measure regularization indirectly via gradient norms/residual norms of proximals. Moreover, these norms (*e.g.* see last row of Table 6.2) are easy to compute and equal zero only at local minima of regularizers, thereby providing a satisfactory substitute for the regularizer function values.[13]

[13] The learned proximal/gradient might not coincide with *any* function. However, this is of no concern since the operators still satisfy convergence properties and the model intuition still holds.

Section 6.3: Experiments

Each numerical experiment shows an application of novel implicit L2O models, which were designed directly from prior knowledge. Associated certificates of trustworthiness are used to emphasize the explainability of each model and illustrate use-cases of certificates. Experiments were coded using Python with the PyTorch library [192], the Adam optimizer[142], and, for ease of re-use, were run via Google Colab. We emphasize these experiments are for illustration of intuitive and novel model design and trustworthiness and are not benchmarked against state-of-the-art models.

Implicit Model Training

Standard backpropagation cannot be used for implicit models as it requires memory capacities beyond existing computing devices. Indeed, storing gradient data for each iteration in the forward propagation (see (6.3)) scales the memory during training linearly with respect to the number of iterations. Since the limit x^∞ solves a fixed point equation, implicit models can be trained by differentiating implicitly through the fixed point to obtain a gradient. This implicit differentiation requires further computations and coding. Instead of using gradients, we utilize Jacobian-Free Backpropagation (JFB) [97] to train models. JFB further simplifies training by only backpropagating through the final iteration, which was proven to yield preconditioned gradients. JFB trains using fixed memory (with respect to the K steps used to estimate $\mathcal{N}_\Theta(d)$) and avoids numerical issues arising from computing exact gradients [20], making JFB and its variations [100, 132] apt for training implicit L2O models.

Implicit Dictionary Learning

Setup In practice, high dimensional signals often approximately admit low dimensional representations (*e.g.* see the dictionary learning and compressed sensing surveys [253]). For illustration, we thus consider a linear inverse problem where true data admit sparse representations. Here each signal $x_d^\star \in \mathbb{R}^{250}$ admits a representation $s_d^\star \in \mathbb{R}^{50}$ via a transformation M (*i.e.* $x_d^\star = Ms_d^\star$). A matrix $A \in \mathbb{R}^{100\times 250}$ is applied to each signal $x_d^\star$ to provide linear measurements $d = Ax_d^\star$. Our task is to recover $x_d^\star$ given knowledge of A and d *without* the matrix M. Since the linear system is quite under-determined, schemes solely minimizing measurement error fail to recover true signals (*e.g.* least squares solutions).

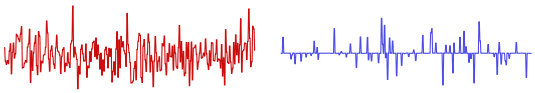


Figure 6.6: Training ADM yields sparse representation of inferences. Diagram shows true data x (sample from test dataset) on left and its sparsified representation Kx on right.

Model Design All convex regularization approaches are known lead to biased estimators whose expectation does not equal the true signal [85]. However, the seminal work [43] of Candes and Tao shows ℓ_1 minimization (rather than additive regularization) enables exact recovery under suitable assumptions. Thus, we minimize a sparsified signal subject to linear constraints via the implicit dictionary model (IDM)

$$\mathcal{N}_\Theta(d) \triangleq \underset{x\in\mathbb{R}^{250}}{\arg\min} \|Kx\|_1 \;\text{ s.t. }\; Ax = d. \tag{6.14}$$

The square matrix K is used to leverage the fact x has a low-dimensional representation by transforming x into a sparse vector. Linearized ADMM [213] (L-ADMM) is used to create a sequence $\{x_d^k\}$ as in (6.3). The model $\mathcal{N}_\Theta$ has weights $\Theta = K$. If it exists, the matrix K^{-1} is known as a dictionary and $K\mathcal{N}_\Theta(d)$ is the corresponding sparse code; hence the name IDM for (6.14). To this end, we em-

phasize K is learned during training and is *different* from M, but these matrices are related since we aim for the product $Kx_d^\star = KMs_d^\star$ to be sparse. Note we use L-ADMM to *provably* solve (6.14), and $\mathcal{N}_\Theta$ is easy to train.

Discussion IDM combines intuition from dictionary learning with a reconstruction algorithm. Two properties are used to identify trustworthy inferences: sparsity and measurement compliance (*i.e.* fidelity). Sparsity and fidelity are quantified using the ℓ_1 norm of the sparsified inference (*i.e.* $K\mathcal{N}_\Theta(d)$) and relative measurement error. Figure 6.6 shows the training the model yields a sparsifying transformation K. Figure 6.3 shows the proposed certificates identify "bad" inferences that might, at first glance, appear to be "good" due to their compatibility with constraints. Lastly, observe the utility of learning K, rather than approximating M, is K makes it is easy to check if an inference admits a sparse representation. Using M to check for sparsity is nontrivial.

CT Image Reconstruction

Setup Comparisons are provided for low-dose CT examples derived from the Low-Dose Parallel Beam dataset (LoDoPab) dataset [153], which consists of phantoms derived from actual human chest CT scans. Here CT measurements are simulated with a parallel beam geometry with a sparse-angle setup of only 30 angles and 183 projection beams, resulting in 5,490 equations and 16,384 unknowns. We add 1.5% Gaussian noise to *each individual beam measurement*. Images have resolution 128×128. To make errors easier to contrast between methods, the linear systems here are under-determined and have more noise than those in some similar works. Image quality is determined using the Peak Signal-To-Noise Ratio (PSNR) and structural similarity index measure (SSIM). Mean squared error was used for the training loss. Training/test datasets consist of 20,000/2,000 samples.

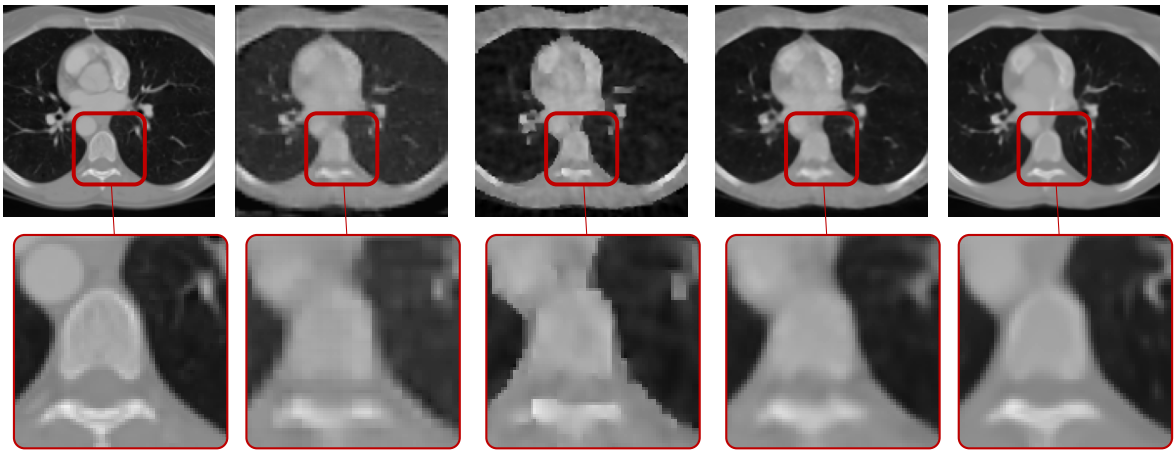


Figure 6.7: Reconstructions on test data computed via U-Net [137], TV minimization, F-FPNs [125], and Implicit L2O (left to right). Bottom row shows expansion of region indicated by red box. Pixel values outside $[0,1]$ are flagged. Fidelity is flagged when images do not comply with measurements, and regularization is flagged when texture features of images are sufficiently inconsistent with true data (*e. g.* grainy images). Labels are provided beneath each image (*n. b.* fail is assigned to images that are worse than 99% of L2O inferences on training data). All comparison methods fail while the Implicit L2O image passes all tests.

Method	Avg. PSNR	Avg. SSIM	# Params
U-Net	27.32 dB	0.761	533,593
TV Min	28.52 dB	0.765	4
F-FPN†	30.46 dB	0.832	96,307
Implicit L2O	31.09 dB	0.851	59,697

Table 6.3: Average PSNR and SSIM on the 2,000 image LoDoPab testing dataset. † Reported from original work [125]. U-Net was trained with filtered backprojection as in prior work [137].

Model Design The model for the CT experiment extends the IDM. In practice, it has been helpful to utilize a sparsifying transform [136, 244]. We accomplish this via a linear operator K, which is applied and then this product is fed into a data-driven regularizer f_Ω with parameters Ω. We additionally ensure compliance with measurements from the radon transform matrix A, up to a tolerance δ. In our setting, all pixel values are also known to be in the interval $[0, 1]$. Combining these pieces yields the implicit L2O model

$$\mathcal{N}_\Theta(d) \triangleq \underset{x\in[0,1]^n}{\arg\min}\, f_\Omega(Kx) \;\; \text{s.t.} \;\; \|Ax - d\| \leq \delta \tag{6.15}$$

Here $\mathcal{N}_\Theta$ has weights $\Theta = (\Omega, K, \delta, \alpha, \beta, \lambda)$ with α, β and λ step-sizes in L-ADMM.

Discussion The L2O model is designed with three features: compliance with measurements (*i.e.* fidelity), valid pixel values, and data-driven regularization. This knowledge can identify trustworthy inferences. The property values used are, respectively, relative measurement error, the indicator function for pixel constraints, and the relative residual norm for the proximal operator prox_{f_Ω}. Comparisons with U-Net [137], F-FPNs[14] [125], and total variation (TV) Minimization are given in Figure 6.7 and Table 6.3. In Figure 6.7, the only image to pass all provided tests is the proposed implicit L2O model. This is intuitive as this model outperforms the others (see Table 6.3) and was specifically designed to embed all of our knowledge,[15] unlike the others. Observe data-driven regularization enabled certificates to detect and flag "bad" TV Minimization features (*e.g.* visible staircasing effects [204, 59]), which shows novelty of certificates as these features

[14] This is precisely the model from Chapter 4.

[15] The F-FPN model utilized measurement constraints, but not direct constraints on pixel values.

are intuitive yet previously were difficult to quantify. Our model used 11% and 62% as many weights as U-Net and F-FFPN, indicating greater efficiency of the implicit L2O framework.

Section 6.4: Conclusions

Explainable ML models can be concretely developed by fusing certificates with the L2O methodology. The implicit L2O methodology enables prior and data-driven knowledge to be directly embedded into models, thereby providing clear and intuitive design. This approach is theoretically sound and compatible with state-of-the-art ML tools. The L2O model also enables construction of our certificate framework with easy-to-read labels, certifying if each inference is trustworthy. These certificates are derived from the L2O model and are trained to be consistent with true data. In particular, using data-driven regularization, our certificates provide a principled scheme for the detection and quantification of "bad" features in inferences. Thanks to this optimization-based model design, failed certificates can be further debugged and corrected by investigating the architecture. This reveals the interwoven nature of pairing implicit L2O with certificates. Our experiments illustrate these ideas in different settings, presenting novel model designs and interpretable results. Future work will study extensions to physics-based applications where PDE-based physics can be integrated into the model [199, 212, 159].

Chapter 7: Conclusions

Large scale scientific problems are commonplace in the current era of big data. Even when solutions are in high dimensional spaces, practicality demands these problems be quickly and accurately solved. Two fields – optimization and machine learning – provide complementing properties in several applications. This is particularly the case when a problem structure is repeatedly used, each time with new but similar data. Optimization provides desirable theory (*e.g.* interpretation and guarantees) while machine learning enables improvement through experience. Within this L2O paradigm, we first consider settings where exact optimization problem formulations are known (Chapter 2). Here the aim is to use learning to obtain rapid convergence. Two novel safeguard frameworks are provided that provably give convergence of deep unrolling methods to optimizers. Numerical experiments show this can enable speedup by an order of magnitude (or more) when compared to analytic methods and converges even when the nonsafeguarded L2O method diverges.[1]

[1] Even if the nonsafeguarded L2O method "blew up," it was found that by simply added a safeguard it could still not only converge, but also do so much more quickly than an analytic method.

The second class of problems considered requires computing inferences for which prior knowledge suggests the inferences can be well-approximated by a parameterized model. Here we represent inferences by implicit models, which are defined in terms of an optimality condition.[2] For such training, we present novel theory for a simple backprop technique: JFB (Chapter 3). This technique is easy-to-implement (see Figure 3.4) and improved performance over comparable methods for image classification. Moreover, JFB was numerically observed to be efficacious even with architectures not covered by theory (*e.g.* batch normalization and dropout) that prior implicit model training schemes cannot handle.

[2] These typically are explicitly formulated in terms of fixed point equations for an associated model operator.

JFB and related techniques open the door to several new optimization-based models. In particular, we show JFB enables one to beneficially augment convex feasibility problems in a data-driven and theoretically sound manner (Chapter 4). JFB also allows the computation of Nash equilibria in contextual games with the additional use of three-operator splitting techniques (Chapter 5).[3] In addition, we show implicit models for Nash equilibria can universally approximate true data that are constrained to action sets. Lastly, we cover implicit L2O models. These inherit theory directly from the classic result of KM and averagedness of operator splitting schemes. Moreover, we show implicit L2O schemes admit interpretability and the ability to identify certificates of trustworthiness (Chapter 6). These certificates can be beneficial in critical situations and act as a standard check to ensure model inferences were computed appropriately. In particular, these certificates are the first XAI tool, to the best of our knowledge, to concretely quantify regularization-type qualities of inferences.[4] Future work will investigate extensions to applications with partial differential equation, extensions of implicit L2O theory, loosening assumptions needed to justify JFB, and further notions of explainability.

[3] A novel application of Davis Yin splitting was introduced to making forward and backward propagation simple and computationally practical.

[4] That is, regularization certificates are able to concretely determine whether seemingly qualitative features are "good" or "bad", *e.g.* the stair-casing phenomena in CT images created from TV minimization.

Appendices

Safe L2O Proofs

This section contains proofs of a sequence of lemmas followed by the two main safeguarding theorems and corollary.

Lemma 8.1.1 *If $\{x^k\}$ is a sequence generated by Safe-L2O and Assumptions 2.1.1, 2.1.2, and 2.1.3 hold, then there exists $\tau \geq 1$ such that*

$$\|x^{k+1} - x^k\| \leq \tau \|T(x^k; d) - x^k\|, \quad \textit{for all } k \in \mathbb{N}. \tag{8.1}$$

Proof: Fix any $x^\star \in \operatorname{fix}(T(\cdot; d))$. Set $\mathcal{I}_1 \subseteq \mathbb{N}$ to be the set of all indices such that the update relation $x^{k+1} = T_{\Theta^k}(x^k; d)$ holds, and set $\mathcal{I}_2 \triangleq \mathbb{N} - \mathcal{I}_1$ so that $\mathbb{N} = \mathcal{I}_1 \cup \mathcal{I}_2$. Also define

$$\tau \triangleq \max\left(1, \sup_{k \in \mathbb{N}} \tau_k\right). \tag{8.2}$$

If $k \in \mathcal{I}_1$, then Assumption 2.1.3 implies

$$\|x^{k+1} - x^k\| = \|T_{\Theta_k}(x^k; d) - x^k\| \leq \tau_k \|T(x^k; d) - x^k\| \leq \tau \|T(x^k; d) - x^k\|. \tag{8.3}$$

Additionally, if $k \in \mathcal{I}_2$, then

$$\|x^{k+1} - x^k\| = \|T(x^k; d) - x^k\|. \tag{8.4}$$

In either case, we see (8.1) holds, taking τ as in (8.2). ∎

Lemma 8.1.2 *If $\{x^k\}$ is a sequence generated by Safe-L2O and Assumptions 2.1.1, 2.1.2, and 2.1.3 hold, then* $\lim_{k\to\infty} \|x^{k+1} - x^k\| = 0$.

Proof: We consider two cases. If $|\mathcal{I}_1| < \infty$, then there is $N \in \mathbb{N}$ such that $n \geq N$ implies the $x^{k+1} = T(x^k; d)$, which is a KM iteration that converges[5] by Theorem 1.1.1. Thus, in this case, there exists $x_d \in \text{fix}(T(\cdot; d))$ such that, together with the triangle inequality, we deduce

[5] Recall all operators T in Chapter 2 are assumed to be averaged.

$$0 \leq \lim_{k\to\infty} \|x^{k+1} - x^k\| \leq \lim_{k\to\infty} \|x^{k+1} - x_d\| + \|x_d - x^k\| = 0. \tag{8.5}$$

Assume $|\mathcal{I}_1| = \infty$. By Assumption 2.1.2, there is an increasing sequence $\{n_k\} \subseteq \mathbb{N}$ and $\zeta \in (0, 1)$ such that

$$\mu_{n_{k+1}} \leq \zeta \mu_{n_k}, \quad \text{for all } k \in \mathbb{N}. \tag{8.6}$$

By induction and the fact $\{\mu_k\}$ is monotonically decreasing,

$$\mu_{n_k} \leq \zeta^k \mu_{n_1} = \zeta^k \mu_1, \quad \text{for all } k \in \mathbb{N}. \tag{8.7}$$

Since $\zeta \in (0, 1)$ and $\mu_1 \in [0, \infty)$, it follows that $\mu_{n_k} \to 0$. With the monotonicity and nonnegativity of $\{\mu_k\}$, we further have $\mu_k \to 0$. Next applying Lemma 8.1.1 reveals there exists $\tau \geq 1$ such that

$$0 \leq \|x^{k+1} - x^k\| \leq \tau \|T(x^k; d) - x^k\| \leq \tau \mu_k, \quad \text{for all } k \in \mathbb{N}. \tag{8.8}$$

By the squeeze lemma, we conclude $\|x^{k+1} - x^k\| \to 0$, completing the proof. ∎

Lemma 8.1.3 *If $\{x^k\}$ is a sequence generated by Safe-L2O and Assumptions 2.1.1, 2.1.2, and 2.1.3 hold, then $\{x^k\}$ is bounded and there is a summable sequence $\{\delta_k\} \subset [0,\infty)$ such that*

$$\|x^{k+1} - x^\star\| \leq \|x^k - x^\star\| + \delta_k, \quad \textit{for all } k \in \mathbb{N} \textit{ and } x^\star \in \operatorname{fix}(T(\cdot; d)). \tag{8.9}$$

Proof: Let $d \in \mathcal{D}$ be given and fix any $x^\star \in \operatorname{fix}(T(\cdot;d))$. Set $\mathcal{I}_1 \subseteq \mathbb{N}$ to be the set of all indices such that the update relation $x^{k+1} = T_{\Theta^k}(x^k; d)$ holds, and set $\mathcal{I}_2 \triangleq \mathbb{N} - \mathcal{I}_1$ so that $\mathbb{N} = \mathcal{I}_1 \cup \mathcal{I}_2$. Next observe

$$\mu_{k+1} \leq \zeta \mu_k, \quad \text{for all } k \in \mathcal{I}_1. \tag{8.10}$$

This implies

$$\begin{aligned}
\sum_{k\in\mathcal{I}_1} \|x^{k+1} - x^k\| &= \sum_{k\in\mathcal{I}_1} \|T_\Theta(x^k) - x^k\| && \text{(8.11a)}\\
&\leq \sum_{k\in\mathcal{I}_1} \tau_k \|T(x^k) - x^k\| && \text{(8.11b)}\\
&\leq \left(\sup_{k\in\mathbb{N}} \tau_k\right) \sum_{k\in\mathcal{I}_1} \mu_k && \text{(8.11c)}\\
&\leq \left(\sup_{k\in\mathbb{N}} \tau_k\right) \sum_{k\in\mathcal{I}_1} \mu_1 \zeta^k && \text{(8.11d)}\\
&\leq \underbrace{\left(\sup_{k\in\mathbb{N}} \tau_k\right) \cdot \frac{\mu_1}{1-\zeta}}_{\triangleq B}, && \text{(8.11e)}
\end{aligned}$$

where the underbraced quantity, defined to be B, is finite by Assumptions 2.1.2 and 2.1.3. A classic result (*e.g.* see Cor. 2.15 in [25]) states, for all $\theta \in \mathbb{R}$,

$$\|\theta x + (1-\theta)y\|^2 = \theta\|x\|^2 + (1-\theta)\|y\|^2 - \theta(1-\theta)\|x-y\|^2, \quad \text{for all } x, y \in \mathbb{R}^n. \tag{8.12}$$

Additionally, because $T(\cdot;d)$ is averaged, there exists $\alpha \in (0,1)$ such that $T(\cdot;d) =$

$(1-\alpha)\mathrm{I} + \alpha Q$ for a 1-Lipschitz operator Q. These facts imply, for all $k \in \mathcal{I}_2$,

$$\|x^{k+1} - x^\star\|^2 = \|T(x^k; d) - x^\star\|^2 \tag{8.13a}$$

$$= \|(1-\alpha)(x^k - x^\star) + \alpha(Q(x^k) - x^\star)\|^2 \tag{8.13b}$$

$$= (1-\alpha)\|x^k - x^\star\|^2 + \alpha\|Q(x^k) - x^\star\|^2 - \alpha(1-\alpha)\|Q(x^k) - x^k\|^2 \tag{8.13c}$$

$$\leq \|x^k - x^\star\|^2 - \alpha(1-\alpha)\|Q(x^k) - x^k\|^2 \tag{8.13d}$$

$$\leq \|x^k - x^\star\|^2. \tag{8.13e}$$

Thus, by the above inequality and the triangle inequality,

$$\|x^{k+1} - x^\star\| \leq \begin{cases} \|x^k - x^\star\| + \|x^{k+1} - x^k\| & \text{if } k \in \mathcal{I}_1, \\ \|x^k - x^\star\| & \text{if } k \in \mathcal{I}_2. \end{cases} \tag{8.14}$$

Because the terms $\|x^{k+1} - x^k\|$ for $k \in \mathcal{I}_1$ are summable by (8.11) and $\|x^{k+1} - x^\star\| \leq \|x^k - x^\star\|$ for $k \in \mathcal{I}_2$, the sequence $\{\delta_k\}$ defined by $\delta_k = \|x^{k+1} - x^k\|$ for $k \in \mathcal{I}_1$ and o otherwise is summable, which establishes (8.9) in the second claim of the lemma. Moreover, applying this inequality inductively reveals

$$\|x^k\| \leq \|x^\star\| + \|x^{k+1} - x^\star\| \tag{8.15a}$$

$$\leq \|x^\star\| + \|x^1 - x^\star\| + \sum_{\ell \in \mathcal{I}_1} \delta_k \tag{8.15b}$$

$$\leq \|x^\star\| + \|x^1 - x^\star\| + B, \qquad \text{for all } k \in \mathbb{N}, \tag{8.15c}$$

which proves boundedness of $\{x^k\}$. ∎

Theorem 2.1.1. If $\{x^k\}$ is a sequence generated by the repeated loop in the Safe-L2O (Algorithm 2) and Assumptions 2.1.1, 2.1.2, and 2.1.3 hold, then $\{x^k\}$ converges to a limit $x_d \in \text{fix}(T(\cdot; d))$, *i.e.* $x^k \to x_d$.

Proof: We first show $\{x^k\}$ contains a limit point in the fixed point set of $T(\cdot; d)$. This is then used to show the entire sequence $\{x^k\}$ converges to this limit point. By Lemma 8.1.3, the sequence $\{x^k\}$ is bounded, and so there exists a convergent subsequence $\{x^{n_k}\} \subseteq \{x^k\}$ with limit x^∞. By Lemma 8.1.1, there is $\tau \geq 1$ such that

$$0 \leq \|T(x^k; d) - x^k\| \leq \tau \|x^{k+1} - x^k\|, \quad \text{for all } k \in \mathbb{N}. \tag{8.16}$$

Applying the squeeze lemma with the result of Lemma 8.1.2 to (8.16) yields

$$\lim_{k\to\infty} \|T(x^{n_k}; d) - x^{n_k}\| = 0. \tag{8.17}$$

With the 2-Lipschitz continuity of $T - \text{I}$ and continuity of norms, (8.17) implies

$$\|T(x^\infty; d) - x^\infty\| = 0 \quad \Longrightarrow \quad x^\infty \in \text{fix}(T(\cdot; d)). \tag{8.18}$$

All that remains is to show the entire sequence $\{x^k\}$ converges to x^∞. To this end, let $\varepsilon > 0$ be given. It suffices to show there exists $N \in \mathbb{N}$ such that

$$\|x^k - x^\infty\| \leq \varepsilon, \quad \text{for all } k \geq N. \tag{8.19}$$

By Lemma 8.1.3, there is a summable sequence $\{\delta_k\} \subset [0, \infty)$ such that

$$\|x^{k+1} - x^\star\| \leq \|x^k - x^\star\| + \delta_k, \quad \text{for all } k \in \mathbb{N} \text{ and } x^\star \in \text{fix}(T(\cdot; d)). \tag{8.20}$$

Since $\{\delta_k\}$ is summable, there exists $N_1 \in \mathbb{N}$ such that

$$\sum_{k=N_1}^{\infty} \delta_k \leq \frac{\varepsilon}{2}. \tag{8.21}$$

Since $x^{n_k} \to x^\infty$, there exists $N_2 \geq N_1$ such that

$$\|x^{N_2} - x^\infty\| \leq \frac{\varepsilon}{2}. \tag{8.22}$$

Combining (8.21) and (8.22), we see

$$\|x^k - x^\infty\| \leq \|x^{N_2} - x^\infty\| + \sum_{\ell=N_2}^{k} \delta_\ell \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon, \quad \text{for all } k \geq N_2. \tag{8.23}$$

This verifies (8.19), taking $N = N_2$, and the proof is complete. ∎

Corollary 2.1.1. If $\{x^k\}$ is a sequence generated by the Safe-L2O method (Algorithm 2) and Assumptions 2.1.1 and 2.1.3 hold, and $\{\mu_k\}$ is generated using a scheme outlined in Table 2.1, then Assumption 2.1.2 holds and, by Theorem 2.1.1, there is $x_d \in \text{fix}(T(\cdot; d))$ such that $\{x^k\}$ converges to x_d, *i.e.* $x^k \to x_d$.

Proof: The proof is parsed into four parts, one for each particular choice of the sequence $\{\mu_k\}$ in Table 2.1, where we note "Recent Term" is a special case of "Recent Max" obtained by taking $m = 1$. Each proof part is completely independent of the others and is separated by italic text. However, to avoid excessive writing, in each section let $\Gamma \subseteq \mathbb{N}$ be the set of all indices for which the inequality in the conditional definitions of $\{\mu_k\}$ hold, the sequence $\{t_k\}$ be an ascending enumeration of Γ, and m_k be the number of times the inequality in the conditional definition of $\{\mu_k\}$ has been satisfied by iteration k. In each case, note $|\Gamma| = \infty$.

Geometric Sequence. Define the sequence $\{\mu_k\}$ using, for each $k \in \mathbb{N}$, the Geometric Sequence update formula in Table 2.1. This implies

$$\mu_k = \alpha^{m_k} \mu_1. \tag{8.24}$$

Since Γ is infinite, $\lim_{k\to\infty} m_k = \infty$, and it follows that

$$\lim_{k\to\infty} \mu_k = \lim_{k\to\infty} (1-\delta)^{m_k} \mu_1 = 0 \cdot \mu_1 = 0, \tag{8.25}$$

i.e. Assumption 2.1.2 holds.

Arithmetic Average. Define the sequence $\{\mu_k\}$ using the AA update formula in Table 2.1. Then observe that, at each index t_k,

$$0 \le \mu_{t_k+1} \le \frac{\alpha\mu_{t_k} + m_{t_k}\mu_{t_k}}{m_{t_k}+1} = \left(1 - \frac{1-\alpha}{m_{t_k}+1}\right)\mu_{t_k} \le \mu_{t_k}, \quad \text{for all } k \in \mathbb{N}. \tag{8.26}$$

Since $\mu_{k+1} = \mu_k$ whenever $k \notin \Gamma$, (8.26) shows $\{\mu_k\}$ is monotonically decreasing. Consequently, using induction reveals

$$0 \le \mu_{t_k} - \frac{1-\alpha}{m_{t_k}+1}\mu_{t_k} \le \mu_1 - \sum_{\ell=1}^{k} \frac{(1-\alpha)\mu_{t_\ell}}{m_{t_\ell}+1} = \mu_1 - \sum_{\ell=1}^{k} \frac{(1-\alpha)\mu_{t_\ell}}{\ell+1} \quad \text{for all } k \in \mathbb{N}, \tag{8.27}$$

where we note $m_{t_\ell} = \ell$ in the sum since m_ℓ increments once each time a modification occurs in the sequence $\{\mu_k\}$. By way of contradiction, suppose there exists $\tau \in (0, \infty)$ such that

$$\liminf_{k\to\infty} \mu_k \ge \tau > 0. \tag{8.28}$$

With the monotonicity of $\{\mu_k\}$, (8.27) implies

$$\sum_{\ell=1}^{k} \frac{(1-\alpha)\tau}{\ell+1} \le \sum_{\ell=1}^{k} \frac{(1-\alpha)\mu_{t_\ell}}{\ell+1} \le \mu_1, \quad \text{for all } k \in \mathbb{N}. \tag{8.29}$$

However, the sum on the left hand side becomes a divergent harmonic series as $k \to \infty$, contradicting the finite upper bound on the right hand side. This contradiction proves assumption (8.28) is false, from which it follows that

$$\liminf_{k\to\infty} \mu_k = 0. \tag{8.30}$$

By the monotone convergence theorem and nonnegativity of each μ_k, we deduce $\mu_k \to 0$, *i.e.* Assumption 2.1.2 holds.

Exponential Moving Average. Given $\theta \in (0,1)$, define the sequence $\{\mu_k\}$ using the EMA(θ) formula in Table 2.1. For each k when μ_{t_k} changes value, observe

$$\begin{aligned} \mu_{t_k+1} &= \theta\|x^{t_k+1} - T(x^{t_k+1})\| + (1-\theta)\mu_{t_k} \\ &\le \theta\alpha\mu_{t_k} + (1-\theta)\mu_{t_k} \\ &= \theta(1-\alpha)\mu_{t_k}. \end{aligned} \tag{8.31}$$

This implies the sequence $\{\mu_k\}$ is nonincreasing and, when a decrease does occur, it is by a geometric factor of the current iterate. By induction, it follows

$$\mu_k \leq [\theta(1-\alpha)]^{m_k}\mu_1, \quad \text{for all } k \in \mathbb{N}. \tag{8.32}$$

Since Γ is infinite, $\lim_{k\to\infty} m_k = \infty$. With the fact $\theta(1-\alpha) \in (0,1)$, we see

$$0 \leq \lim_{k\to\infty} \mu_k \leq \lim_{k\to\infty} [\theta(1-\alpha)]^{m_k}\mu_1 = 0 \cdot \mu_1 = 0, \tag{8.33}$$

from which Assumption 2.1.2 holds by the squeeze theorem.

Recent Max. Let $m \in \mathbb{N}$. Set Ξ_k to be the set of the most recent m indices in Γ, counting backwards from k, where $\{\mu_k\}$ is defined by the update formula in Table 2.1. When there are less than m indices in $\Gamma \cap \{1, 2, \ldots, k\}$, we let Ξ_k be all of the indices in the intersection. The sequence $\{\mu_k\}$ is monotonically decreasing since, for each k in Γ, the new term $\|x^k - T(x^k; d)\|$ is introduced so that $\|x^k - T(x^k; d)\| \in \Xi_{k+1}$, and this new term is no larger than the largest term in Ξ_k. All that remains is to show this sequence converges to zero. By way of contradiction, suppose there exists $\tau \in (0, \infty)$ such that

$$\liminf_{k\to\infty} \mu_k = \tau > 0. \tag{8.34}$$

Then choose

$$\varepsilon = \frac{(1-\alpha)\tau}{2\alpha}, \tag{8.35}$$

which implies $\alpha(\tau + \varepsilon) < \tau$. By (8.34) and the fact Γ is infinite, there exists $\tilde{N} \in \mathbb{N}$ with $\tilde{N} > m$ such that

$$\|\mu_{t_{\tilde{N}}} - \tau| < \varepsilon \quad \Longrightarrow \quad \mu_{t_{\tilde{N}}} < \tau + \varepsilon. \tag{8.36}$$

Note each new element to Ξ_k is no larger than $\alpha\mu_{t_{\tilde{N}}}$. And, for any k after m such replacements occur,

$$\mu_k = \max_{\ell \in \Xi_k} \|x^\ell - T(x^\ell; d)\| \leq \alpha\mu_{t_{\tilde{N}}} \leq \alpha(\tau + \varepsilon) < \tau, \tag{8.37}$$

a contradiction to (8.34). This shows our assumption (8.34) must be false, and so

$$\liminf_{k\to\infty} \mu_k = 0. \tag{8.38}$$

By the monotone convergence theorem, we conclude Assumption 2.1.2 holds. ■

We conclude this section with a proof of the second safeguarding theorem.

Theorem 2.2.1. If the sequence $\{x^k\}$ is generated by the iteration in Algorithm 3 with firmly nonexpansive $T(\cdot;d)$ and $\lambda_k = 1/(k+1)$, then

$$\|x^k - T(x^k;d)\| \leq \frac{1}{2}\left(\frac{d_1}{k} + \sqrt{\frac{d_1^2}{k^2} + \frac{4C}{k}}\right), \quad \text{for all } k \geq 2, \tag{8.39}$$

where $d_1 \triangleq \min\{\|\tilde{x} - x\| : x \in \operatorname{fix}(T(\cdot;d))\}$ is the distance between the reference iterate $\tilde{x}$ and the set of fixed points and $C \geq 0$ is an arbitrary constant. In particular, this implies each limit point of $\{x^k\}$ is a fixed point.

Proof: We proceed in the following manner, with *much* credit due to the analysis in [76]. First we verify an inequality with the energy sequence $\{E_k(x^k)\}$ (Step 1). This is used to obtain the convergence rate (Step 2). Resulting implications about limit points are established last (Step 3). Below we assume $x^1 = \tilde{x}$.

Step 1. We claim

$$E_k(x^k) \leq \frac{C}{k}, \quad \text{for all } k \geq 2. \tag{8.40}$$

We proceed by induction. First note $T(\cdot;d)$ is firmly nonexpansive, and so $2F(\cdot;d) = \mathrm{I} - T(\cdot;d)$ is also firmly nonexpansive [28], which implies

$$\|2F(x;d) - 2F(v;d)\|^2 \leq \langle 2F(x;d) - 2F(v;d), x - v\rangle, \quad \text{for all } x, v. \tag{8.41}$$

Using (8.41) with $x = x^2$ and $v = x^1$ together with our choice of step sizes $\{\lambda_k\}$,

we find

$$E_2(x^2) = \|F(x^2)\|^2 - \frac{\lambda_1}{1-\lambda_1}\langle F(x^2), x^1 - x^2\rangle \tag{8.42a}$$

$$= \|F(x^2)\|^2 - \langle F(x^2), x^1 - x^2\rangle \tag{8.42b}$$

$$= \langle F(x^2), F(x^2) - F(x^1)\rangle \tag{8.42c}$$

$$= \|F(x^2) - F(x^1)\|^2 + \langle F(x^1), F(x^2) - F(x^1)\rangle \tag{8.42d}$$

$$\leq \frac{1}{2}\|2F(x^2) - 2F(x^1)\|^2 + \langle F(x^1), F(x^2) - F(x^1)\rangle \tag{8.42e}$$

$$\leq \langle F(x^2) - F(x^1), x^2 - x^1\rangle + \langle F(x^1), F(x^2) - F(x^1)\rangle \tag{8.42f}$$

$$= -\langle F(x^2) - F(x^1), F(x^1)\rangle + \langle F(x^1), F(x^2) - F(x^1)\rangle \tag{8.42g}$$

$$= 0. \tag{8.42h}$$

Thus, $E_2(x^2) \leq 0 \leq C/2$, and the base case holds. Inductively, suppose (8.40) holds taking $k = n$ for some $n \geq 2$. If $x^{n+1} = y^{n+1}$, then (8.40) holds, taking $k = n+1$, by the conditional statement in Line 6 of Algorithm 3. Alternatively, suppose $x^{n+1} \neq y^{n+1}$. Applying (8.41) with $x = x^{n+1}$ and $v = x^n$ yields

$$\|F(x^{n+1}) - F(x^n)\|^2 \leq 2\|F(x^{n+1}) - F(x^n)\|^2 \leq \langle F(x^{n+1}) - F(x^n), x^{n+1} - x^n\rangle. \tag{8.43}$$

Upon expansion of the left hand side, we discover

$$\|F(x^{n+1})\|^2 \leq \langle F(x^{n+1}), x^{n+1} - x^n + 2F(x^n)\rangle - \langle F(x^n), x^{n+1} - x^n + F(x^n)\rangle. \tag{8.44}$$

Algebraic manipulations of the update formula for x^{n+1} yield the relations

$$x^{n+1} - x^n + 2F(x^n) = \frac{\lambda_n}{1-\lambda_n}(x^1 - x^{n+1}), \tag{8.45a}$$

$$x^{n+1} - x^n + F(x^n) = \lambda_n(x^1 - x^n) - (1 - 2\lambda_n)F(x^n), \tag{8.45b}$$

Substituting (8.45) in (8.44) gives

$$\|F(x^{n+1})\|^2 \leq \frac{\lambda_n}{1-\lambda_n}\langle F(x^{n+1}), x^1 - \mathbf{x}^{n+1}\rangle \tag{8.46a}$$

$$-\lambda_n\langle F(x^n), x^1 - x^n\rangle + (1 - 2\lambda_n)\|F(x^n)\|^2. \tag{8.46b}$$

and we collect terms with $F(x^{n+1})$ on the left hand side to obtain

$$\|F(x^{n+1})\|^2 - \frac{\lambda_n}{1-\lambda_n}\langle F(x^{n+1}), x^1 - x^{n+1}\rangle \le (1-2\lambda_n)\|F(x^n)\|^2 \tag{8.47a}$$

$$- \lambda_n \langle F(x^n), x^1 - x^n\rangle. \tag{8.47b}$$

Furthermore, by our choice of step size sequence $\{\lambda_n\}$,

$$1 - 2\lambda_n = \frac{n-1}{n+1} \tag{8.48}$$

and, for $n \ge 2$,

$$\lambda_n = \frac{n-1}{n+1}\cdot\frac{1}{n-1} = \frac{n-1}{n+1}\cdot\frac{\lambda_{n-1}}{1-\lambda_{n-1}}. \tag{8.49}$$

Combining (8.47), (8.48), and (8.49) with the definition of E_n in (2.6) yields

$$E_{n+1}(x^{n+1}) \le \frac{n-1}{n+1}\cdot E_n(x^n). \tag{8.50}$$

Applying the inductive hypothesis, we deduce

$$E_{n+1}(x^{n+1}) \le \frac{n-1}{n+1}\cdot\frac{C}{n} = \frac{n-1}{n}\cdot\frac{C}{n+1} \le \frac{C}{n+1}, \tag{8.51}$$

and this inequality closes the induction. Thus, (8.40) holds by the principle of mathematical induction.

Step 2. Let $x^\star$ be the projection of x^1 onto $\mathrm{fix}(T(\cdot;d))$ so that

$$\|x^1 - x^\star\| = \arg\min\{\|x^1 - x\| : x \in \mathrm{fix}(T(\cdot;d))\} = d_1. \tag{8.52}$$

Note this projection is well defined since the set of fixed points is closed and convex. By (8.40), for $k \ge 2$,

$$\|F(x^k)\|^2 \le \frac{\lambda_k}{1-\lambda_k}\langle F(x^k), x^1 - x^k\rangle + \frac{C}{k} \tag{8.53}$$

$$= \underbrace{\frac{\lambda_k}{1-\lambda_k}}_{=1/k}\left(\langle F(x^k), x^1 - x^\star\rangle + \underbrace{\langle F(x^k) - F(x^\star), x^\star - x^k\rangle}_{=0}\right) + \frac{C}{k} \tag{8.54}$$

$$= \frac{1}{k}\langle F(x^k), x^1 - x^\star\rangle + \frac{C}{k} \tag{8.55}$$

$$\le \frac{1}{k}\|F(x^k)\|\|x^1 - x^\star\| + \frac{C}{k}. \tag{8.56}$$

where the third line holds since $F(x^\star) = 0$ and F is monotone. Using the quadratic formula with the fact that $\|F(x^k)\|^2 \geq 0$, we obtain (8), as desired.

Step 3. Let x^∞ be a limit point of $\{x^k\}$. This implies there exists a subsequence $\{x^{n_k}\}$ that converges to x^∞. Since $T(\cdot; d)$ is 1-Lipschitz and norms are continuous, it follows that

$$0 \leq \|x^\infty - T(x^\infty; d)\| = \lim_{k\to\infty} \|x^{n_k} - T(x^{n_k}; d)\| \leq \lim_{k\to\infty} \frac{1}{2}\left(\frac{d_1}{n_k} + \sqrt{\frac{d_1^2}{n_k^2} + \frac{4C}{n_k}}\right) = 0. \tag{8.57}$$

By the squeeze lemma, we deduce $x^\infty \in \text{fix}(T(\cdot; d))$, *i.e.* $x^\infty \in \text{fix}(T(\cdot; d))$. Because x^∞ was an arbitrarily chosen limit point, each limit point of $\{x^k\}$ is a fixed point of $T(\cdot; d)$. ∎

JFB Proofs

This section provides proofs for results in Chapter 3. For the reader's convenience, we restate all results before proving them.

Lemma 9.2.4 *If Assumption 3.1.1 and 3.2.1 hold, then $\mathcal{J}_\Theta$ in (3.9) exists and*

$$\langle x, \mathcal{J}_\Theta x \rangle \geq (1-\gamma)\|x\|^2, \quad \textit{for all } x \in \mathcal{U}. \tag{9.58}$$

Additionally, $\mathcal{J}_\Theta$ is invertible, and its inverse $\mathcal{J}_\Theta^{-1}$ satisfies the coercivity inequality

$$\left\langle x, \mathcal{J}_\Theta^{-1} x \right\rangle \geq \frac{1-\gamma}{(1+\gamma)^2}\|x\|^2, \quad \textit{for all } x \in \mathcal{U}. \tag{9.59}$$

Proof: We proceed in the following manner. First we establish the coercivity inequality (9.58) (Step 1). This is used to show $\mathcal{J}_\Theta$ is invertible (Step 2). The previous two results are then combined to establish the inequality (9.59) (Step 3). All unproven results that are quoted below about operators are standard and may be found standard functional analysis texts (*e.g.* [149]).

Step 1. To obtain our coercivity inequality, we identify a bound on the operator norm for $\partial T_\Theta / \partial x$. Fix any unit vector $v \in \mathcal{U}$. By the definition of differentiation,

$$\frac{\mathrm{d}T_\Theta}{\mathrm{d}x} v = \lim_{\varepsilon \to 0^+} \frac{T_\Theta(x_d + \varepsilon v; d) - T_\Theta(x_d; d)}{\|(x_d + \varepsilon v) - x_d\|} = \lim_{\varepsilon \to 0^+} \frac{T_\Theta(x_d + \varepsilon v; d) - T_\Theta(x_d; d)}{\varepsilon}. \tag{9.60}$$

Thus,

$$\left\| \frac{\mathrm{d}T_\Theta}{\mathrm{d}x} v \right\| = \left\| \lim_{\varepsilon \to 0^+} \frac{T_\Theta(x_d + \varepsilon v; d) - T_\Theta(x_d; d)}{\varepsilon} \right\| = \lim_{\varepsilon \to 0^+} \frac{\|T_\Theta(x_d + \varepsilon v; d) - T_\Theta(x_d; d)\|}{\varepsilon}, \tag{9.61}$$

where the first equality follows from (9.60) and the second holds by the continuity of norms. Combining (9.62) with the Lipschitz assumption (3.6) gives the

upper bound

$$\left\| \frac{\mathrm{d}T_\Theta}{\mathrm{d}x} v \right\| \leq \lim_{\varepsilon \to 0^+} \frac{\gamma \| (x_d + \varepsilon v) - x_d \|}{\varepsilon} = \gamma. \tag{9.62}$$

Because the upper bound relation in (9.62) holds for an arbitrary unit vector $v \in \mathcal{U}$, we deduce

$$\left\| \frac{\mathrm{d}T_\Theta}{\mathrm{d}x} \right\| \triangleq \sup \left\{ \left\| \frac{\mathrm{d}T_\Theta}{\mathrm{d}x} v \right\| : \ \|v\| = 1 \right\} \leq \gamma. \tag{9.63}$$

That is, the operator norm is bounded by γ. Together the Cauchy-Schwarz inequality and (9.63) imply

$$\left\langle x, \frac{\mathrm{d}T_\Theta}{\mathrm{d}x} x \right\rangle \leq \left\| \frac{\mathrm{d}T_\Theta}{\mathrm{d}x} \right\| \|x\|^2 \leq \gamma \|x\|^2, \quad \text{for all } x \in \mathcal{U}. \tag{9.64}$$

Thus, the bilinear form $\langle \cdot , \mathcal{J}_\Theta \cdot \rangle$ is $(1-\gamma)$ coercive, *i.e.*

$$\langle x, \mathcal{J}_\Theta x \rangle = \|x\|^2 - \left\langle x, \frac{\mathrm{d}T_\Theta}{\mathrm{d}x} x \right\rangle \geq (1-\gamma)\|x\|^2, \quad \text{for all } x \in \mathcal{U}. \tag{9.65}$$

Step 2. Consider any kernel element $w \in \ker(\mathcal{J}_\Theta)$. Then (9.65) implies

$$(1-\gamma)\|w\|^2 \leq \langle w, \mathcal{J}_\Theta w \rangle = \langle w, 0 \rangle = 0 \quad \Longrightarrow \quad (1-\gamma)\|w\|^2 \leq 0 \quad \Longrightarrow \quad w = 0. \tag{9.66}$$

Consequently, the kernel of $\mathcal{J}_\Theta$ is trivial, *i.e.*

$$\ker(\mathcal{J}_\Theta) \triangleq \{x : \mathcal{J}_\Theta x = 0\} = \{0\}, \tag{9.67}$$

and wherefore the linear operator $\mathcal{J}_\Theta$ is invertible.

Step 3. By (9.62) and an elementary result in functional analysis,

$$\|\mathcal{J}_\Theta^\top \mathcal{J}_\Theta\| = \|\mathcal{J}_\Theta\|^2 \leq \left(\|\mathrm{I}\| + \left\| \frac{\mathrm{d}T_\Theta}{\mathrm{d}u} \right\| \right)^2 \leq (1+\gamma)^2. \tag{9.68}$$

Hence

$$\|x\|^2 = \langle x, x \rangle = \left\langle \mathcal{J}_\Theta^{-1} x, (\mathcal{J}_\Theta^\top \mathcal{J}_\Theta) \mathcal{J}_\Theta^{-1} x \right\rangle \leq (1+\gamma)^2 \left\| \mathcal{J}_\Theta^{-1} x \right\|^2, \quad \text{for all } x \in \mathcal{U}. \tag{9.69}$$

Combining (9.65) and (9.69) reveals, for all $x \in \mathcal{U}$,

$$\frac{1-\gamma}{(1+\gamma)^2} \langle x, x \rangle \leq (1-\gamma) \|\mathcal{J}_\Theta^{-1} x\|^2 \leq \left\langle \mathcal{J}_\Theta^{-1} x, \mathcal{J}_\Theta (\mathcal{J}_\Theta^{-1} x) \right\rangle = \left\langle \mathcal{J}_\Theta^{-1} x, x \right\rangle. \tag{9.70}$$

This establishes (9.59), and we are done. ∎

Lemma 9.2.5 *If $A \in \mathbb{R}^{t\times t}$ is symmetric with positive eigenvalues,*

$$\overline{\lambda} \triangleq \frac{\lambda_{\max}(A) + \lambda_{\min}(A)}{2} \quad \textit{and} \quad S \triangleq \overline{\lambda}\mathrm{I} - A, \tag{9.71}$$

then

$$\|S\| = \frac{\lambda_{\max}(A) - \lambda_{\min}(A)}{2}. \tag{9.72}$$

Proof: Since A is symmetric, the spectral theorem asserts it possesses a set of eigenvectors that form an orthogonal basis for $\mathbb{R}^t$. This same basis forms the set of eigenvectors for $\overline{\lambda}\mathrm{I} - A$, with eigenvalues of A denoted by $\{\lambda_i\}_{i=1}^t$. So, there exists orthogonal $P \in \mathbb{R}^{t\times t}$ and diagonal Λ with entries given by each of the eigenvalues λ_i such that

$$S = \overline{\lambda}\mathrm{I} - P^\top \Lambda P = P^\top \left(\overline{\lambda}\mathrm{I} - \Lambda\right) P. \tag{9.73}$$

Substituting this equivalence into the definition of the operator norm yields

$$\|S\| \triangleq \sup\{\|S\xi\| : \ \|\xi\| = 1\} = \sup\left\{\|P^\top(\overline{\lambda}I - \Lambda)P\xi\| : \ \|\xi\| = 1\right\}. \tag{9.74}$$

Leveraging the fact P is orthogonal enables the supremum to be restated via

$$\|S\| = \sup\left\{\|(\overline{\lambda}I - \Lambda)P\xi\| : \ \|\xi\| = 1\right\} = \sup\left\{\|(\overline{\lambda}I - \Lambda)\zeta\| : \ \|\zeta\| = 1\right\}. \tag{9.75}$$

Because $\overline{\lambda}\mathrm{I} - \Lambda$ is diagonal, (9.75) implies

$$\|S\| = \max_{i\in[t]} |\overline{\lambda} - \lambda_i| = \frac{\lambda_{\max}(A) - \lambda_{\min}(A)}{2}, \tag{9.76}$$

and the proof is complete. ∎

Theorem 3.2.1. *If Assumptions 3.1.1, 3.2.1, 3.2.2, and 3.2.3 hold for given weights Θ and data d, then*

$$p_\Theta \triangleq -\frac{\mathrm{d}}{\mathrm{d}\Theta}\Big[\ell(y_d, S_\Theta(T_\Theta(u,d))\Big]_{u=u_d^\star} \tag{9.77}$$

forms a descent direction for $\ell(y_d, \mathcal{N}_\Theta(d))$ with respect to Θ.

Proof: To complete the proof, it suffices to show

$$\left\langle \frac{\mathrm{d}\ell}{\mathrm{d}\Theta}, p_\Theta \right\rangle < 0, \quad \text{for all } \frac{\mathrm{d}\ell}{\mathrm{d}\Theta} \neq 0. \tag{9.78}$$

Let any weights Θ and data d be given, and assume the gradient $\mathrm{d}\ell/\mathrm{d}\Theta$ is nonzero. We proceed in the following manner. First we show p_Θ is equivalent to a preconditioned gradient (Step 1). We then show $M^\top \mathrm{d}\ell/\mathrm{d}\Theta$ is nonzero, with M as in (3.14) of Assumption 3.2.3 (Step 2). These two results are then combined to verify the descent inequality (9.78) for the provided Θ and d (Step 3).

[6] We assumed each space is a real-valued finite dimensional Hilbert space, making it equivalent to some Euclidean space. So, it suffices to show everything in Euclidean spaces.

Step 1. Denote the dimension of each component of the gradient $\mathrm{d}\ell/\mathrm{d}\Theta$ using[6]

$$\frac{\partial T_\Theta}{\partial \Theta} \in \mathbb{R}^{p\times n}, \quad \mathcal{J}_\Theta^{-1} \in \mathbb{R}^{n\times n}, \quad \frac{\partial S_\Theta}{\partial \Theta} \in \mathbb{R}^{p\times c}, \quad \frac{\mathrm{d}S_\Theta}{\mathrm{d}u} \in \mathbb{R}^{n\times c}, \quad \frac{\partial \ell}{\partial y} \in \mathbb{R}^{c\times 1}. \tag{9.79}$$

Combining each of these terms yields the gradient expression[7]

$$\frac{\mathrm{d}\ell}{\mathrm{d}\Theta} = \left[\frac{\partial T_\Theta}{\partial \Theta}\mathcal{J}_\Theta^{-1}\frac{\mathrm{d}S_\Theta}{\mathrm{d}u} + \frac{\mathrm{d}S_\Theta}{\mathrm{d}\Theta}\right]\frac{\partial \ell}{\partial y}. \tag{9.80}$$

[7] In the main text, the ordering was used to make clear application of the chain rule, but here we reorder terms to get consistent dimensions in each matrix operation.

By Assumption 3.2.2, S_Θ and T_Θ depend on separate components of $\Theta = (\theta_S, \theta_T)$. Thus,

$$\frac{\mathrm{d}\ell}{\mathrm{d}\Theta} = \begin{bmatrix} \dfrac{\partial S_\Theta}{\partial \theta_S} \\ \dfrac{\partial T_\Theta}{\partial \theta_T}\mathcal{J}_\Theta^{-1}\dfrac{\mathrm{d}S_\Theta}{\mathrm{d}u} \end{bmatrix}\frac{\partial \ell}{\partial y} = \underbrace{\begin{bmatrix} \dfrac{\partial S_\Theta}{\partial \theta_S} & 0 \\ 0 & \dfrac{\partial T_\Theta}{\partial \theta_T} \end{bmatrix}}_{M}\underbrace{\begin{bmatrix} \mathrm{I} & 0 \\ 0 & \mathcal{J}_\Theta^{-1} \end{bmatrix}}_{\tilde{\mathcal{J}}_\Theta^{-1}}\underbrace{\begin{bmatrix} \mathrm{I} \\ \dfrac{\mathrm{d}S_\Theta}{\mathrm{d}u} \end{bmatrix}\frac{\partial \ell}{\partial y}}_{v}, \tag{9.81}$$

where we define[8] $M \in \mathbb{R}^{p\times(n+c)}$, $\tilde{\mathcal{J}}_\Theta^{-1} \in \mathbb{R}^{(n+c)\times(n+c)}$, and $v \in \mathbb{R}^{(n+c)\times 1}$ to be the

[8] Note this choice of M coincides with the matrix M in Assumption 3.2.3.

underbraced quantities. This enables the gradient to be concisely expressed via the relation

$$\frac{\mathrm{d}\ell}{\mathrm{d}\Theta} = M\tilde{\mathcal{J}}_{\Theta}^{-1}v, \tag{9.82}$$

and our proposed gradient alternative in (9.77) is given by

$$p_{\Theta} = -Mv. \tag{9.83}$$

Because M has full column rank (by Assumption 3.2.3), $M^{+}M = \mathrm{I}$, enabling us to rewrite p_{Θ} via

$$p_{\Theta} = -M\tilde{\mathcal{J}}_{\Theta}M^{+}M\mathcal{J}_{\Theta}^{-1}v = -(M\tilde{\mathcal{J}}_{\Theta}M^{+})\frac{\mathrm{d}\ell}{\mathrm{d}\Theta}. \tag{9.84}$$

[9] The preconditioner is not necessarily symmetric.

Hence p_{Θ} is a preconditioned gradient.[9]

Step 2. Set

$$w \triangleq M^{\top}\frac{\mathrm{d}\ell}{\mathrm{d}\Theta} = M^{\top}M\tilde{\mathcal{J}}_{\Theta}^{-1}v. \tag{9.85}$$

The fact that M has full column rank implies it has a trivial kernel. In particular,

$$0 \neq \frac{\mathrm{d}\ell}{\mathrm{d}\Theta} = M\tilde{\mathcal{J}}_{\Theta}^{-1}v \quad \Longrightarrow \quad 0 \neq \tilde{\mathcal{J}}_{\Theta}^{-1}v. \tag{9.86}$$

Again leveraging the full column rank of M, we know $M^{\top}M$ is invertible and, thus, has trivial kernel as well. This fact together with (9.86) reveals

$$0 \neq (M^{\top}M)\tilde{\mathcal{J}}_{\Theta}^{-1}v = w. \tag{9.87}$$

Step 3. Inserting the definition of w and p_{Θ} formulation of (9.84) into the scalar product in (9.78) yields

$$\left\langle \frac{\mathrm{d}\ell}{\mathrm{d}\Theta}, p_{\Theta} \right\rangle = -\left\langle M^{\top}M\tilde{\mathcal{J}}_{\Theta}^{-1}v, \tilde{\mathcal{J}}_{\Theta}M^{+}M\tilde{\mathcal{J}}_{\Theta}^{-1}v \right\rangle = -\left\langle w, \tilde{\mathcal{J}}_{\theta}(M^{\top}M)^{-1}w \right\rangle, \tag{9.88}$$

noting $M^{+} = (M^{\top}M)^{-1}M^{\top}$. Let λ_{+} and λ_{-} be the maximum and minimum eigenvalues of $(M^{\top}M)^{-1}$, respectively. Note $(M^{\top}M)$ is positive definite since the full column rank of M implies

$$\left\langle \xi, M^{\top}M\xi \right\rangle = \|M\xi\|^{2} > 0, \quad \text{for all nonzero } \xi \in \mathbb{R}^{n+c}. \tag{9.89}$$

Thus, $(M^\top M)^{-1}$ is positive definite, making $\lambda_+, \lambda_- > 0$. Let $\overline{\lambda}$ be the average

$$\overline{\lambda} \triangleq \frac{\lambda_+ + \lambda_-}{2}. \tag{9.90}$$

Plugging in this choice of $\overline{\lambda}$ to (9.88) by adding and subtracting $\overline{\lambda}\mathrm{I}$ gives

$$-\left\langle w, \tilde{\mathcal{J}}_\theta (M^\top M)^{-1} w \right\rangle \leq -\overline{\lambda}(1-\gamma)\|w\|^2 + \left\langle w, \tilde{\mathcal{J}}_\Theta (\overline{\lambda}\mathrm{I} - (M^\top M)^{-1}) w \right\rangle, \tag{9.91}$$

noting $\tilde{\mathcal{J}}_\Theta$ is $1-\gamma$ coercive because it is the block diagonal composition of $\mathcal{J}_\Theta$, which is $1-\gamma$ coercive by (9.58) in Lemma 9.2.4, and the identity matrix, which is 1-coercive. Application of the Cauchy Schwarz inequality to the right hand side of (9.91) reveals

$$-\left\langle w, \tilde{\mathcal{J}}_\theta (M^\top M)^{-1} w \right\rangle \leq -\overline{\lambda}(1-\gamma)\|w\|^2 + \|\tilde{\mathcal{J}}_\Theta\| \|\overline{\lambda}\mathrm{I} - (M^\top M)^{-1})\| \|w\|^2. \tag{9.92}$$

By Lemma 9.2.5,

$$\|\overline{\lambda}\mathrm{I} - (M^\top M)^{-1}\| = \frac{\lambda_+ - \lambda_-}{2}. \tag{9.93}$$

Similar block diagonal argument as used above to verify $\tilde{\mathcal{J}}_\Theta$ is coercive can also be applied to bound the operator norm of $\tilde{\mathcal{J}}_\Theta$. Indeed, (9.63) implies

$$\|\mathcal{J}_\Theta\| \leq 1+\gamma \quad \Longrightarrow \quad \|\tilde{\mathcal{J}}_\Theta\| \leq 1+\gamma. \tag{9.94}$$

Hence (9.88), (9.92), (9.93), and (9.94) together yield

$$\left\langle \frac{\mathrm{d}\ell}{\mathrm{d}\Theta}, p_\Theta \right\rangle \leq -\frac{1}{2}\big((1-\gamma)(\lambda_+ + \lambda_-) - (1+\gamma)(\lambda_+ - \lambda_-)\big)\|w\|^2 \tag{9.95a}$$

$$= -2(\lambda_- - \gamma\lambda_+)\|w\|^2. \tag{9.95b}$$

The right hand expression in (9.95) is negative since (9.87) shows $w \neq 0$ and the conditioning inequality (3.15) in Assumption 3.2.3 implies $(\lambda_- - \gamma\lambda_+)$ is positive. This verifies (9.78), completing the proof. ∎

Corollary 3.2.1. *Given weights Θ and data d, there exists $\varepsilon > 0$ such that if $x_d^\varepsilon \in \mathcal{U}$ satisfies $\|x_d^\varepsilon - x_d\| \leq \varepsilon$ and the assumptions of Theorem 3.2.1 hold, then*

$$p_\Theta^\varepsilon \triangleq -\frac{\mathrm{d}}{\mathrm{d}\Theta}\Big[\ell(y_d, S_\Theta(T_\Theta(x,d))\Big]_{x=x_d^\varepsilon} \tag{9.96}$$

is a descent direction of $\ell(y_d, \mathcal{N}_\Theta(d))$ with respect to Θ.

Proof: For notational convenience, for all $\tilde{x} \in \mathcal{U}$, define

$$p_\Theta(\tilde{x}) \triangleq -\frac{\mathrm{d}}{\mathrm{d}\Theta}\Big[\ell(y_d, S_\Theta(T_\Theta(x;d))\Big]_{x=\tilde{x}} \tag{9.97}$$

noting $p_\Theta^\varepsilon = p_\Theta(x_d^\varepsilon)$. Also define the quantity

$$\nabla \triangleq \frac{\mathrm{d}}{\mathrm{d}\Theta}\left[\ell(y_d, \mathcal{N}_\Theta(d))\right]. \tag{9.98}$$

Assuming $\nabla \neq 0$, it suffices to show

$$\left\langle p_\Theta^\varepsilon, \nabla \right\rangle < 0. \tag{9.99}$$

By the smoothness of ℓ, S_Θ, and T_Θ (Assumption 3.2.1), there is $\delta > 0$ such that

$$\|x - x_d\| \leq \delta \quad \Longrightarrow \quad \|p_\Theta(x) - p_\Theta(x_d)\| \leq \frac{(\lambda_- - \gamma\lambda_+)\|M^\top \nabla\|^2}{\|\nabla\|}, \tag{9.100}$$

where λ_+ and λ_- are the maximum and minimum eigenvalues of $(M^\top M)^{-1}$, respectively. Also note $M^\top \nabla \neq 0$ since $M^\top$ has full column rank.[10] Substituting the inequality (9.95) in the proof of Theorem 3.2.1 into (9.99) reveals

[10] See w in Step 2 of the proof of Theorem 3.2.1.

$$\langle p_\Theta(x), \nabla\rangle = \langle p_\Theta(x_d), \nabla\rangle + \langle p_\Theta(x) - p_\Theta(x_d), \nabla\rangle \tag{9.101a}$$

$$\leq -2(\lambda_- - \gamma\lambda_+)\|M^\top \nabla\|^2 + \langle p_\Theta(x) - p_\Theta(x_d), \nabla\rangle\,. \tag{9.101b}$$

The Cauchy Schwarz inequality and (9.100) enable us to obtain the upper bound

$$|\langle p_\Theta(x) - p_\Theta(x_d), \nabla\rangle| \leq (\lambda_- - \gamma\lambda_+)\|M^\top \nabla\|^2, \quad \text{for all } x \in B(x_d, \delta), \tag{9.102}$$

where $B(x_d, \delta)$ is the δ-ball centered at x_d. Combining (9.101) and (9.102) yields

$$\langle p_\Theta(x), \nabla\rangle \leq -(\lambda_- - \gamma\lambda_+)\|M^\top \nabla\|^2, \quad \text{for all } x \in B(x_d, \delta). \tag{9.103}$$

In particular, this shows (9.99) holds when we set $\varepsilon = \delta$. ∎

Nash Equilibria Proofs

Here we provide proofs for all theorems stated in the main text. For the readers convenience we reproduce each statement before proving it

Theorem 5.1.1. *If Assumptions (A1) to (A6) hold, then there is a unique Nash Equilibrium x_d for all $d \in \mathcal{D}$ and the map $d \mapsto x_d$ is Lipschitz continuous.*

Proof: (A3) implies the game gradient $F(;d)$ is strictly (in fact, strongly) monotone for all d, *i.e.*

$$\langle F(x;d) - F(y;d), x - y\rangle > 0 \quad \forall x, y \tag{10.104}$$

In game theory this is sometimes referred to as *diagonal strict convexity*. By [207, Theorem 2] the Nash equilibrium $x_d^\star$ is unique. See also [84, Theorem 2.2.3]. Next observe (A4) implies F is Lipschitz continuous with respect to d while (A5) guarantees F is α strongly monotone. [72, Theorem 2.1] then shows that around any fixed $\bar{d} \in \mathcal{D}$ the map $d \mapsto x_d^\star$ is locally Lipschitz, *i.e.* there exists a constant $L_{\bar{d}}$ and an open neighborhood $N_{\bar{d}} \subset \mathcal{D}$ of $\bar{d}$ upon which $d \mapsto x_d^\star$ is $L_{\bar{d}}$-Lipschitz continuous. As $\bar{D}$ is compact a standard covering argument converts this local Lipschitz property to a global Lipschitz property. ∎

Theorem 5.2.1. If Assumptions (A1) to (A6) hold, then for any $\varepsilon > 0$ there exists an $F_\Theta(\cdot;\cdot)$ such that

$$\max_{d\in\mathcal{D}} \|x_d^\star - \mathcal{N}_\Theta(d)\|_2 \leq \varepsilon. \tag{10.105}$$

Proof: Let $\varepsilon > 0$ and $\zeta \in \mathcal{D}$ be given. Denote[11] the map $d \mapsto x_d^\star$ by $\mathcal{L}$, *i.e.* $\mathcal{L}(d) \triangleq x_d^\star$. By Theorem 5.1.1, $\mathcal{L}$ is well-defined and Lipschitz continuous. Combined with the compactness of $\mathcal{D}$ via (A6), this implies, by standard universal approximation properties of neural networks [140], there exists a continuous model $G_\Theta : \mathcal{D} \to \mathcal{X}$

[11] In this proof, the notation $\mathcal{L}$ is *not* to be confused with a Lagrangian.

such that

$$\max_{d\in\mathcal{D}} \|\mathcal{L}(d) - G_\Theta(d)\|_2 \le \frac{\varepsilon}{2}. \tag{10.106}$$

Next fix $\alpha > 0$ and define the operator $F_\Theta\colon \mathcal{X}\times\mathcal{D}\to\mathcal{X}$ by

$$F_\Theta(x;d) \triangleq \frac{x - G_\Theta(d)}{\alpha}. \tag{10.107}$$

Note F_Θ is continuous by the continuity of G_Θ, and so the VI and fixed point equivalence (5.7) implies, letting x_ζ be the fixed point of the projected gradient operator,

$$\mathcal{N}_\Theta(\zeta) = P_\mathcal{C}(x_\zeta - \alpha F_\Theta(x_\zeta;\zeta)) = P_\mathcal{C}(G_\Theta(\zeta)). \tag{10.108}$$

By definition of the projection $P_\mathcal{C}$,

$$\|P_\mathcal{C}(G_\Theta(\zeta)) - G_\Theta(\zeta)\|_2 = \min_{x\in\mathcal{C}} \|x - G_\Theta(\zeta)\|_2, \tag{10.109}$$

which implies, since $x_d^\star = \mathcal{L}(\zeta) \in \mathcal{C}$,

$$\|P_\mathcal{C}(G_\Theta(\zeta)) - G_\Theta(\zeta)\|_2 \le \|\mathcal{L}(\zeta) - G_\Theta(\zeta)\|_2. \tag{10.110}$$

Together with the triangle inequality, (10.106) and (10.110) yield

$$\begin{aligned}
\|x_\zeta^\star - \mathcal{N}_\Theta(\zeta)\|_2 &= \|\mathcal{L}(\zeta) - P_\mathcal{C}(G_\Theta(\zeta))\|_2 && \text{(10.111a)}\\
&\le \|\mathcal{L}(\zeta) - G_\Theta(\zeta)\|_2 + \|G_\Theta(\zeta) - P_\mathcal{C}(G_\Theta(\zeta))\|_2 && \text{(10.111b)}\\
&\le 2\|\mathcal{L}(\zeta) - G_\Theta(\zeta)\|_2 && \text{(10.111c)}\\
&= \varepsilon. && \text{(10.111d)}
\end{aligned}$$

Since (10.111) holds for arbitrarily chosen $\zeta \in \mathcal{D}$, we deduce (10.105) holds for the provided ε. As $\varepsilon > 0$ was also arbitrarily chosen, the result follows. ∎

Below we provide a lemma justifying the decoupling of constraints in the action set $\mathcal{C}$. Here we make use of polyhedral sets[12]; however, this result also holds in a more general setting utilizing relative interiors of $\mathcal{C}^1$ and $\mathcal{C}^2$.

[12] A set is polyhedral if it is of the form $\{x : \langle x, a^i \rangle \leq b_i, \text{ for } i \in [p]\}$, for $p \in \mathbb{N}$.

Theorem 5.2.2. *Suppose $\mathcal{C} = \mathcal{C}^1 \cap \mathcal{C}^2$ for convex $\mathcal{C}^1$ and $\mathcal{C}^2$. If both $\mathcal{C}^i$ are polyhedral or have relative interiors with a point in common and the VI admits a unique solution, then defining*

$$T_\Theta(x;d) \triangleq x - P_{\mathcal{C}^1}(x) + P_{\mathcal{C}^2}\left(2P_{\mathcal{C}^1}(x) - x - \alpha F_\Theta(P_{\mathcal{C}^1}(x);d)\right) \tag{10.112}$$

yields the equivalence[13]

$$\mathcal{N}_\Theta(d) = \mathrm{VI}\big(F_\Theta(\cdot;d), \mathcal{C}\big) = P_{\mathcal{C}^1}(z_d) \iff z_d = T_\Theta(z_d;d). \tag{10.113}$$

[13] Observe the fixed point z_d is *not* the VI solution, but its projection onto $\mathcal{C}^1$ is a solution. This distinction can be subtle, but significant, in practice.

Proof: We begin with the well-known equivalence relation [84]

$$x_d^\circ \in \mathrm{VI}(F(\cdot;d), \mathcal{C}) \iff 0 \in F(x_d^\circ;d) + \partial\delta_{\mathcal{C}}(x_d^\circ). \tag{10.114}$$

Using our assumption on $\mathcal{C}^1$ and $\mathcal{C}^2$, we may apply [206, Theorem 23.8.1] to assert

$$\partial\delta_{\mathcal{C}} = \partial\delta_{\mathcal{C}^1} + \partial\delta_{\mathcal{C}^2}. \tag{10.115}$$

Next consider three maximal[14] monotone operators A, B and C, with C single-valued. For each $\alpha > 0$, let $J_{\alpha A}$ and $R_{\alpha A}$ be the resolvent of αA and reflected resolvent of αA, respectively, *i.e.*

$$J_{\alpha A} \triangleq (\mathrm{I} + \alpha A)^{-1} \quad \text{and} \quad R_{\alpha A} \triangleq 2J_{\alpha A} - \mathrm{I}. \tag{10.116}$$

[14] A monotone operator M is *maximal* if there is no other monotone operator S such that $\mathrm{Gra}(M) \subset \mathrm{Gra}(S)$ properly [213]. This is a technical assumption that holds for all cases of our interest.

In particular, note the resolvent of $\partial\delta_{\mathcal{C}^i}$ is precisely the projection operator $P_{\mathcal{C}^i}$ [27, Example 23.4]. Using three operator splitting (*e.g.* see [75, Lemma 2.2] and [213]), we obtain the equivalence

$$0 \in (A + B + C)(x) \tag{10.117a}$$

$$\iff x = J_{\alpha B}(z), \quad \text{where } z = z - J_{\alpha B}(z) + J_{\alpha A}(R_{\alpha B} - \alpha C J_{\alpha B})(z). \tag{10.117b}$$

Setting $A = \partial\delta_{\mathcal{C}^2}$, $B = \partial\delta_{\mathcal{C}^1}$, and $C = F$, (10.117) reduces to

$$0 \in F(x_d; d) + \partial\delta_{\mathcal{C}^1}(x_d) + \partial\delta_{\mathcal{C}^2}(x_d) \iff x_d = P_{\mathcal{C}_1}(z_d), \text{ where } z_d = T(z_d; d). \tag{10.118}$$

Combining (10.114), (10.115), and (10.118) yields (10.113), as desired. ∎

Corollary 5.2.1. *In the setting of Theorem 5.2.2, if $F_\Theta(\cdot; d)$ is α-cocoercive and z^1 is given, then the iteration $z^{k+1} = T_\Theta(z^k; d)$ yields convergence $z^k \to z_d \in \text{fix}(T_\Theta(\cdot; d))$.*

Proof: Because all sets considered in this work are closed and convex, the projection operators $P_{\mathcal{C}^1}$ and $P_{\mathcal{C}^2}$ are averaged [46, Theorem 2.2.21]. Combined with the fact that F is α-cocoercive, the operator T is averaged [75, Proposition 2.1]. By Theorem 1.1.1, given any z^1, if a sequence $\{z^k\}$ is generated using updates of the form $z^{k+1} = T(z^k; d)$ for an averaged operator $T(\cdot; d)$, then $\{z^k\}$ converges to a fixed point $z_d \in \text{fix}(T(\cdot; d))$. ∎

Constraint Decoupling

Minkowski Sum

This subsection provides a decoupling scheme for constraints structured as a Minkowski sum,[15] *i.e.*

$$\mathcal{C} \triangleq \mathcal{C}_1 + \cdots + \mathcal{C}_K, \tag{10.119}$$

[15] This arises in the modeling Wardrop equilibria in traffic routing problems.

where $\mathcal{C}_k \subset \mathcal{X}$ and $\mathcal{C}_k = \mathcal{C}_k^1 \cap \mathcal{C}_k^2$ for all $k \in [K]$. The core idea is to avoid attempting to directly project onto $\mathcal{C}$ and instead perform simple projections onto each set $\mathcal{C}_k^i$, assuming the projection onto $\mathcal{C}_k^i$ admits an explicit formula. First, define the product space

$$\overline{\mathcal{X}} \triangleq \underbrace{\mathcal{X} \times \mathcal{X} \times \ldots \times \mathcal{X}}_{K \text{ times}}. \tag{10.120}$$

For notational clarity, we denote elements of $\overline{\mathcal{X}}$ by overlines so each element $\overline{x} \in \overline{\mathcal{X}}$ is of the form $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_K)$ with $\overline{x}_k \in \mathcal{X}$ for all $k \in [K]$. Because $\mathcal{X}$ is a Hilbert space, $\overline{\mathcal{X}}$ is naturally endowed with a scalar product $\langle \cdot, \cdot \rangle_{\overline{\mathcal{X}}}$ defined by

$$\langle \overline{x}, \overline{y} \rangle_{\overline{\mathcal{X}}} \triangleq \sum_{k=1}^{K} \langle \overline{x}_k, \overline{y}_k \rangle. \tag{10.121}$$

Between $\mathcal{X}$ and the product space $\overline{\mathcal{X}}$ we define two natural maps $Q^- : \overline{\mathcal{X}} \to \mathcal{X}$ and $Q^+ : \mathcal{X} \to \overline{\mathcal{X}}$ by

$$Q^-(\overline{x}) \triangleq \sum_{k=1}^{K} \overline{x}_k \quad \text{and} \quad Q^+(x) \triangleq \underbrace{(x, x, \ldots, x)}_{K \text{ copies}}. \tag{10.122}$$

In words, $Q^-(\overline{x})$ maps down to $\mathcal{X}$ by adding together the blocks of $\overline{x}$ and $Q^+(x)$ maps up to $\overline{\mathcal{X}}$ by making K copies of x, thus motivating the use of "+" and "−" signs. Define the Cartesian product

$$\mathcal{A} \triangleq \mathcal{C}_1 \times \ldots \times \mathcal{C}_K \subseteq \overline{\mathcal{X}}, \tag{10.123}$$

and note $Q^-(\mathcal{A}) = \mathcal{C}$. To further decouple each set $\mathcal{C}_k$, also define the Cartesian products

$$\mathcal{A}^i \triangleq \mathcal{C}_1^i \times \ldots \times \mathcal{C}_K^i \quad \text{for all } i \in [2]. \tag{10.124}$$

so $\mathcal{A} = \mathcal{A}^1 \cap \mathcal{A}^2$. The projection onto $\mathcal{A}^i$ can be computed component-wise; namely,

$$P_{\mathcal{A}^i}(\overline{x}) = \left(P_{\mathcal{C}_1^i}(\overline{x}_1), \ldots, P_{\mathcal{C}_K^i}(\overline{x}_K) \right) \quad \text{for all } i \in [2]. \tag{10.125}$$

We now rephrase Algorithm 7, applied to a VI in the product space $\mathrm{VI}(Q^+ \circ F \circ Q^-, \mathcal{A})$, into Algorithm 8 using $\mathcal{A}^i$ in lieu of $\mathcal{C}^i$. The use of Algorithm 8 is justified by the following two lemmas. The first shows the product space operator is monotone whenever F is. The second shows the solution sets to the two VIs coincide, after applying Q^- to map down from $\overline{\mathcal{X}}$ to $\mathcal{X}$.

Lemma 10.3.6 *If $F : \mathcal{X} \to \mathcal{X}$ is α-cocoercive, then $Q^+ \circ F \circ Q^-$ on $\overline{\mathcal{X}}$ is (α/K)-cocoercive.*

Proof: Fix any $\overline{x}, \overline{y} \in \overline{\mathcal{X}}$ and set $R_{\overline{x}} \triangleq (F \circ Q^-)(\overline{x})$ and $R_{\overline{y}} \triangleq (F \circ Q^-)(\overline{y})$. Then

$$\left\langle Q^+(R_{\overline{x}}) - Q^+(R_{\overline{y}}), \overline{x} - \overline{y} \right\rangle_{\overline{\mathcal{X}}} = \sum_{k=1}^{K} \left\langle R_{\overline{x}} - R_{\overline{y}}, \overline{x}_k - \overline{y}_k \right\rangle \tag{10.126a}$$

$$= \left\langle R_{\overline{x}} - R_{\overline{y}}, Q^-(\overline{x}) - Q^-(\overline{y}) \right\rangle. \tag{10.126b}$$

Substituting in the definition of $R_{\overline{x}}$ and $R_{\overline{y}}$ reveals

$$\left\langle Q^+(R_{\overline{x}}) - Q^+(R_{\overline{y}}), \overline{x} - \overline{y} \right\rangle_{\overline{\mathcal{X}}} = \left\langle F(Q^-(\overline{x})) - F(Q^-(\overline{y})), Q^-(\overline{x}) - Q^-(\overline{y}) \right\rangle \tag{10.127a}$$

$$\geq \alpha \| F(Q^-(\overline{x})) - F(Q^-(\overline{y})) \|^2 \tag{10.127b}$$

$$= \frac{\alpha}{K} \| Q^+ \circ F \circ Q^-(\overline{x}) - Q^+ \circ F \circ Q^-(\overline{y}) \|_{\overline{\mathcal{X}}}^2, \tag{10.127c}$$

where the final equality follows from the definition of the norm on $\overline{\mathcal{X}}$. Because (10.127) holds for arbitrary $\overline{x}, \overline{y} \in \overline{\mathcal{X}}$, the result follows. ∎

Lemma 10.3.7 *For* $F\colon \mathcal{X} \to \mathcal{X}$, $\overline{x}^\circ \in \mathrm{VI}\left(Q^+ \circ F \circ Q^-, \mathcal{A}\right)$ *if and only if* $Q^-(\overline{x}^\circ) \in \mathrm{VI}\left(F, \mathcal{C}\right)$.

Proof: Fix $\overline{y} \in \mathcal{A}$ and $\overline{x}^\circ \in \mathrm{VI}\left(Q^+ \circ F \circ Q^-, \mathcal{A}\right)$. Similarly to the proof of Lemma 10.3.6, observe

$$\left\langle (Q^+ \circ F \circ Q^-)(\overline{x}^\circ), \overline{y} - \overline{x}^\circ \right\rangle_{\overline{\mathcal{X}}} = \sum_{k=1}^{K} \left\langle (F \circ Q^-)(\overline{x}^\circ), \overline{y}_k - \overline{x}_k^\circ \right\rangle \tag{10.128a}$$

$$= \left\langle F(Q^-(\overline{x}^\circ)), Q^-(\overline{y}) - Q^-(\overline{x}^\circ) \right\rangle. \tag{10.128b}$$

Since $Q^-(\mathcal{A}) = \mathcal{C}$, it follows that $x^\circ \triangleq Q^-(\overline{x}^\circ) \in \mathcal{C}$ and $w \triangleq Q^-(\overline{y}) \in \mathcal{C}$. Consequently,

$$0 \leq \left\langle (Q^+ \circ F \circ Q^-)(\overline{x}^\circ), \overline{y} - \overline{x}^\circ \right\rangle_{\overline{\mathcal{X}}} = \left\langle F(x^\circ), w - x^\circ \right\rangle. \tag{10.129}$$

As $\overline{y}$ was arbitrarily chosen, (10.129) holds for all $w \in \mathcal{C}$ and $Q^-(\overline{x}^\circ) \in \mathrm{VI}\left(F, \mathcal{C}\right)$.

Conversely, fix $\overline{y} \in \mathcal{A}$ and $\overline{x}^\circ \in \overline{\mathcal{X}}$ such that $Q^-(\overline{x}^\circ) \in \mathrm{VI}(F, \mathcal{C})$. Then $Q^-(\overline{y}) \in \mathcal{C}$ and

$$0 \leq \left\langle F(Q^-(\overline{x}^\circ)), Q^-(\overline{y}) - Q^-(\overline{x}^\circ) \right\rangle \tag{10.130a}$$

$$= \sum_{k=1}^{K} \left\langle F(Q^-(\overline{x}^\circ)), \overline{y}_k - \overline{x}_k^\circ \right\rangle \tag{10.130b}$$

$$= \left\langle (Q^+ \circ F \circ Q^-)(\overline{x}^\circ)), \overline{y} - \overline{x}^\circ \right\rangle_{\overline{\mathcal{X}}}. \tag{10.130c}$$

Together the inequality (10.130) and the fact $\overline{y} \in \mathcal{A}$ was arbitrarily chosen imply $\overline{x}^\circ \in \mathrm{VI}(Q^+ \circ F \circ Q^-, \mathcal{A})$. This completes the proof. ∎

Intersections of Constraints

For completeness, we also consider constraints $\mathcal{C}$ that may be expressed as the intersection of several sets, *i.e.* $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2 \cdots \cap \mathcal{C}_K$. Let $\overline{\mathcal{X}}$, $\langle \cdot, \cdot \rangle_{\overline{\mathcal{X}}}$, Q^+ and Q^- be as in Appendix 7. Next define[16]

[16] Note $\mathcal{A}$ in Appendix 7 is the same as $\mathcal{B}^1$ in (10.131), *i.e.* $\mathcal{B}^1 = \mathcal{A}$.

$$\mathcal{B}^1 \triangleq \mathcal{C}_1 \times \cdots \times \mathcal{C}_K \quad \text{and} \quad \mathcal{B}^2 \triangleq Q^+(\mathcal{X}) = \{\overline{x} \in \overline{\mathcal{X}} : \; x_1 = \cdots = x_K\}, \tag{10.131}$$

and $\mathcal{B} \triangleq \mathcal{B}^1 \cap \mathcal{B}^2$. Note $Q^-(\mathcal{B}) = \mathcal{C}$. The logic is now the same as before; rephrase Algorithm 1 using $\mathcal{B}^i$ in place of $\mathcal{C}^i$. The projection $P_{\mathcal{B}^1}$ can be computed component-wise via

$$P_{\mathcal{B}^1}(\overline{x}) = \left(P_{\mathcal{C}_1}(\overline{x}_1), \ldots, P_{\mathcal{C}_K}(\overline{x}_K)\right), \tag{10.132}$$

and $P_{\mathcal{B}^2}(\overline{x})$ has a simple closed form given in the following lemma.

Lemma 10.3.8 *With notation as above,* $P_{\mathcal{B}^2}(\overline{x}) = Q^+\left(\frac{1}{K}\sum_{k=1}^K \overline{x}_k\right)$.

Proof: By the definition of a projection and the norm on $\overline{\mathcal{X}}$,

$$P_{\mathcal{B}^2}(\overline{x}) \triangleq \underset{\overline{z} \in \mathcal{B}^2}{\arg\min} \, \|\overline{z} - \overline{x}\|_{\overline{\mathcal{X}}}^2 \tag{10.133a}$$

$$= \underset{\overline{z} \in \mathcal{B}^2}{\arg\min} \sum_{k=1}^K \|\overline{z}_k - \overline{x}_k\|^2 \tag{10.133b}$$

$$= Q^+(z^\#), \quad \text{where } z^\# = \underset{z \in \mathcal{X}}{\arg\min} \sum_{k=1}^K \|z - \overline{x}_k\|^2, \tag{10.133c}$$

so $z^\#$ satisfies the following optimality condition

$$0 = \frac{\mathrm{d}}{\mathrm{d}z}\left[\sum_{k=1}^K \|z - \overline{x}_k\|^2\right]_{z=z^\#} = \sum_{k=1}^K 2(z^\# - \overline{x}_k) = 2K\left(z^\# - \frac{1}{K}\sum_{k=1}^K \overline{x}_k\right). \tag{10.134}$$

This implies

$$z^\# = \frac{1}{K}\sum_{k=1}^K \overline{x}_k. \tag{10.135}$$

Together (10.133) and (10.135) yield the result, completing the proof. ∎

For each operator $F: \mathcal{X} \to \mathcal{X}$, we define a corresponding product space operator $\overline{F}: \overline{\mathcal{X}} \to \overline{\mathcal{X}}$ via

$$\overline{F}(\overline{x}) \triangleq (F(\overline{x}_1), \ldots, F(\overline{x}_K). \tag{10.136}$$

This definition enables us to show a direct equivalence between a VI in the original space $\mathcal{X}$ and the product space $\overline{\mathcal{X}}$. That is, we complete the analysis in the following lemmas by showing the solution set of an appropriate VI in the product space coincides with that of the original VI.

Lemma 10.3.9 *If $F: \mathcal{X} \to \mathcal{X}$ is α-cocoercive, then the operator $\overline{F}: \overline{\mathcal{X}} \to \overline{\mathcal{X}}$ is α-cocoercive.*

Proof: Fix any $\overline{x}, \overline{y} \in \overline{\mathcal{X}}$. Then observe

$$\left\langle \overline{F}(\overline{x}) - \overline{F}(\overline{y}), \overline{x} - \overline{y} \right\rangle_{\overline{\mathcal{X}}} = \sum_{k=1}^{K} \left\langle F(\overline{x}_k) - F(\overline{y}_k), \overline{x}_k - \overline{y}_k \right\rangle \tag{10.137a}$$

$$\geq \alpha \sum_{k=1}^{K} \|F(\overline{x}_k) - F(\overline{y}_k)\|^2 \tag{10.137b}$$

$$= \alpha \|\overline{F}(\overline{x}) - \overline{F}(\overline{y})\|_{\overline{\mathcal{X}}}^2. \tag{10.137c}$$

Because (10.137) holds for arbitrarily chosen $\overline{x}, \overline{y} \in \overline{\mathcal{X}}$, we conclude $\overline{F}$ is α-cocoercive. ∎

Lemma 10.3.10 *For α-cocoercive $F: \mathcal{X} \to \mathcal{X}$, $x^\circ \in \mathrm{VI}(F, \mathcal{C})$ if and only if $Q^+(x^\circ) \in \mathrm{VI}\left(\overline{F}, \mathcal{B}\right)$.*

Proof: Fix any $x^\circ \in \mathrm{VI}(F, \mathcal{C})$ and set $\overline{x}^\circ = Q^+(x^\circ)$. An elementary proof shows $Q^+ : \mathcal{C} \to \mathcal{B}$ is a bijection. Together with the fact x° is a VI solution, this implies

$$0 \leq K \left\langle F(x^\circ), y - x^\circ \right\rangle, \quad \text{for all } y \in \mathcal{C} \tag{10.138a}$$

$$\iff 0 \leq \sum_{k=1}^{K} \left\langle F(x^\circ), \overline{y}_k - x^\circ \right\rangle, \quad \text{for all } \overline{y} \in \mathcal{B} \tag{10.138b}$$

$$\iff 0 \leq \sum_{k=1}^{K} \left\langle F(\overline{x}_k^\circ), \overline{y}_k - \overline{x}_k^\circ \right\rangle, \quad \text{for all } \overline{y} \in \mathcal{B} \tag{10.138c}$$

$$\iff 0 \leq \left\langle \overline{F}(\overline{x}^\circ), \overline{y} - \overline{x}^\circ \right\rangle_{\overline{\mathcal{X}}}, \quad \text{for all } \overline{y} \in \mathcal{B}. \tag{10.138d}$$

By the transitive property, the first and final expressions in (10.138) are equivalent, and we are done. ∎

Projections onto Intersections of Hyperplanes

Consider the set $\mathcal{C} \triangleq \{x : Nx = b\} \subseteq \mathcal{X}$, and note $\mathcal{C}$ is closed and convex so the projection operator onto $\mathcal{C}$ is well-defined and given by

$$P_{\mathcal{C}}(z) = \operatorname*{arg\,min}_{x \in \mathcal{C}} \frac{1}{2}\|x - z\|^2 = \operatorname*{arg\,min}_{x \in \mathcal{X}} \frac{1}{2}\|x - z\|^2 \;\text{ s.t. }\; Nx = b. \tag{10.139}$$

For completeness we express (and prove) a projection formula for $\mathcal{C}$ via the following lemma.

Lemma 10.3.11 *For nonempty $\mathcal{C} \triangleq \{x : Nx = b\}$, the projection $P_{\mathcal{C}}$ is given by*

$$P_{\mathcal{C}}(z) = z - N^{\dagger}(Nz - b), \tag{10.140}$$

where $N^{\dagger} \triangleq U\Sigma^{-1}V^{\top}$ and $U\Sigma V$ is the compact singular value decomposition of N such that U and V have orthonormal columns and Σ is invertible.

Proof: Referring to (10.139), we see the associated Lagrangian is given by

$$\mathcal{L}(x, \lambda) \triangleq \frac{1}{2}\|x - z\|^2 + \langle \lambda, Nx - b \rangle . \tag{10.141}$$

The optimizer $x^{\#} \triangleq P_{\mathcal{C}}(z)$ satisfies the optimality condition $0 = \nabla \mathcal{L}(x^{\#}, \lambda^{\#})$ for some $\lambda^{\#}$, which can be expanded as

$$0 = \nabla_x \Big[\mathcal{L}(x, \lambda)\Big]_{(x,\lambda)=(x^{\#},\lambda^{\#})} = x^{\#} - z + N^{\top}\lambda^{\#}, \tag{10.142a}$$

$$0 = \nabla_{\lambda} \Big[\mathcal{L}(x, \lambda)\Big]_{(x,\lambda)=(x^{\#},\lambda^{\#})} = Nx^{\#} - b. \tag{10.142b}$$

We claim it suffices to choose

$$\lambda^{\#} = (U\Sigma^{-2}U^{\top})(Nz - b). \tag{10.143}$$

By (10.142a), this choice yields

$$x^{\#} = z - N^{\top}\lambda^{\#} \tag{10.144a}$$

$$= z - N^{\top}(U\Sigma^{-2}U^{\top})(Nz - b) \tag{10.144b}$$

$$= z - (V\Sigma U^{\top})(U\Sigma^{-2}U^{\top})(Nz - b) \tag{10.144c}$$

$$= z - (V\Sigma^{-1}U^{\top})(Nz - b) \tag{10.144d}$$

$$= z - N^{\dagger}(Nz - b). \tag{10.144e}$$

To prove this formula for $x^{\#}$ gives the projection, it suffices to show the remaining condition $Nx^{\#} = b$ is satisfied. Decomposing N into its singular value decomposition, observe

$$Nx^{\#} = N(z - (V\Sigma^{-1}U^{\top})(Nz - b)) \tag{10.145a}$$

$$= Nz - (U\Sigma V^{\top})(V\Sigma^{-1}U^{\top})(Nz - b) \tag{10.145b}$$

$$= Nz - (U\Sigma V^{\top})(V\Sigma^{-1}U^{\top})(U\Sigma V^{\top}z - b) \tag{10.145c}$$

$$= Nz - U\Sigma V^{\top}z + UU^{\top}b \tag{10.145d}$$

$$= UU^{\top}b. \tag{10.145e}$$

The range of N is contained in the subspace spanned by the orthonormal columns of U, *i.e.* $\operatorname{range}(N) \subseteq \operatorname{span}(u^1, \ldots, u^r)$, where u^i is the i-th column of U and r is the rank of N. Because $\mathcal{C}$ is nonempty, $b \in \operatorname{range}(N)$ and it follows that there exists scalars $\alpha_1, \ldots, \alpha_r$ such that

$$b = \sum_{i=1}^{r} \alpha_i u^i. \tag{10.146}$$

Through direct substitution, we deduce

$$UU^{\top}b = UU^{\top}\sum_{i=1}^{r}\alpha_i u^i = U\left(\sum_{i,j=1}^{r}\alpha_i\left\langle u^j, u^i\right\rangle\right) = U\sum_{i=1}^{r}\alpha_i e^i = \sum_{i=1}^{r}\alpha_i u^i = b. \tag{10.147}$$

Thus, (10.145) and (10.147) together show the final optimality condition is satisfied, proving the claim. ∎

References

[1] Amina Adadi and Mohammed Berrada. "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)". In: *IEEE access* 6 (2018), pp. 52138–52160.

[2] Jonas Adler, Holger Kohr, and Ozan Öktem. *Operator Discretization Library (ODL)*. Jan. 2017.

[3] Jonas Adler and Ozan Öktem. "Learned primal-dual reconstruction". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1322–1332.

[4] Jonas Adler and Ozan Öktem. "Solving ill-posed inverse problems using iterative deep neural networks". In: *Inverse Problems* 33.12 (2017), p. 124007.

[5] Ron Aharoni and Yair Censor. "Block-iterative projection methods for parallel computation of solutions to convex feasibility problems". In: *Linear Algebra and Its Applications* 120 (1989), pp. 165–175.

[6] Christopher Aicher, Nicholas J Foti, and Emily B Fox. "Adaptively truncating backpropagation through time to control gradient bias". In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 799–808.

[7] Stephanie Allen, John P Dickerson, and Steven A Gabriel. "Using Inverse Optimization to Learn Cost Functions in Generalized Nash Games". In: *arXiv preprint arXiv:2102.12415* (2021).

[8] Brandon Amos and J Zico Kolter. "Optnet: Differentiable optimization as a layer in neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 136–145.

[9] Mariano Anaya. *Clean Code in Python: Refactor your legacy code base*. Packt Publishing Ltd, 2018.

[10] Cem Anil, James Lucas, and Roger Grosse. "Sorting out Lipschitz function approximation". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 291–301.

[11] Matthew Arnold et al. "FactSheets: Increasing trust in AI services through supplier's declarations of conformity". In: *IBM Journal of Research and Development* 63.4/5 (2019), pp. 6–1.

[12] Simon R Arridge. "Optical tomography in medical imaging". In: *Inverse problems* 15.2 (1999), R41.

[13] Simon R Arridge and John C Schotland. "Optical tomography: forward and inverse problems". In: *Inverse problems* 25.12 (2009), p. 123010.

[14] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115.

[15] Kenneth J Arrow and Gerard Debreu. "Existence of an equilibrium for a competitive economy". In: *Econometrica: Journal of the Econometric Society* (1954), pp. 265–290.

[16] Filipe de Avila Belbute-Peres et al. "End-to-end differentiable physics for learning and control". In: *Advances in neural information processing systems* 31 (2018), pp. 7178–7189.

[17] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[18] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. “Deep equilibrium models”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 690–701.

[19] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. “Multiscale Deep Equilibrium Models”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[20] Shaojie Bai, Vladlen Koltun, and Zico Kolter. “Stabilizing Equilibrium Models by Jacobian Regularization”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 554–565.

[21] Stefan Banach. “Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales”. In: *Fund. math* 3.1 (1922), pp. 133–181.

[22] Hillel Bar-Gera. “Origin-based algorithm for the traffic assignment problem”. In: *Transportation Science* 36.4 (2002), pp. 398–417.

[23] Hillel Bar-Gera. “Traffic assignment by paired alternative segments”. In: *Transportation Research Part B: Methodological* 44.8-9 (2010), pp. 1022–1046.

[24] H. H. Bauschke and V. R. Koch. “Projection methods: Swiss army knives for solving feasibility and best approximation problems with halfspaces”. In: *Contemporary Mathematics* 636 (2015), pp. 1–40.

[25] Heinz Bauschke and Patrick Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaace*. 2nd. Springer, 2017.

[26] Heinz H Bauschke and Jonathan M Borwein. "On projection algorithms for solving convex feasibility problems". In: *SIAM review* 38.3 (1996), pp. 367–426.

[27] Heinz H Bauschke, Patrick L Combettes, et al. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. 2nd. Springer, 2017.

[28] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*. Vol. 408. Springer, 2011.

[29] Heinz H Bauschke, Patrick L Combettes, and D Russell Luke. "Phase retrieval, error reduction algorithm, and Fienup variants: a view from convex optimization". In: *JOSA A* 19.7 (2002), pp. 1334–1345.

[30] Heinz H Bauschke and Valentin R Koch. "Projection methods: Swiss army knives for solving feasibility and best approximation problems with halfspaces". In: *Contemporary Mathematics* 636 (2015), pp. 1–40.

[31] Amir Beck. *First-order methods in optimization*. Vol. 25. SIAM, 2017.

[32] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[33] Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch Paschalidis. "Data-driven estimation in equilibrium using inverse optimization". In: *Mathematical Programming* 153.2 (2015), pp. 595–633.

[34] Thomas Blumensath and Mike E. Davies. "Iterative hard thresholding for compressed sensing". In: *Applied and Computational Harmonic Analysis* 27.3 (2009), pp. 265–274. ISSN: 1063-5203.

[35] M. Borgerding, P. Schniter, and S. Rangan. "AMP-Inspired Deep Networks for Sparse Linear Inverse Problems". In: *IEEE Transactions on Signal Processing* 65.16 (2017), pp. 4293–4308.

[36] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[37] Tan Bui-Thanh et al. "A computational framework for infinite-dimensional Bayesian inverse problems Part I: The linearized case, with application to global seismic inversion". In: *SIAM Journal on Scientific Computing* 35.6 (2013), A2494–A2523.

[38] Charles L Byrne. *Applied Iterative Methods*. en. A K Peters, Ltd., 2008.

[39] Charles L Byrne. "Block-iterative methods for image reconstruction from projections". In: *IEEE Transactions on Image Processing* 5.5 (1996), pp. 792–794.

[40] Daniela Calvetti and Lothar Reichel. "Tikhonov regularization of large linear problems". In: *BIT Numerical Mathematics* 43.2 (2003), pp. 263–283.

[41] Emmanuel J Candes and Justin Romberg. "Quantitative robust uncertainty principles and optimally sparse decompositions". In: *Foundations of Computational Mathematics* 6.2 (2006), pp. 227–254.

[42] Emmanuel J Candes et al. "Phase retrieval via matrix completion". In: *SIAM review* 57.2 (2015), pp. 225–251.

[43] Emmanuel J Candès, Justin Romberg, and Terence Tao. "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information". In: *IEEE Transactions on information theory* 52.2 (2006), pp. 489–509.

[44] Guillaume Carlier and Filippo Santambrogio. "A continuous theory of traffic congestion and Wardrop equilibria". In: *Journal of Mathematical Sciences* 181.6 (2012), pp. 792–804.

[45] Andrzej Cegielski. *Iterative Methods for Fixed Point Problems in Hilbert Spaces*. Lecture Notes in Mathematics 2057. Springer, 2012.

[46] Andrzej Cegielski. *Iterative methods for fixed point problems in Hilbert spaces*. Vol. 2057. Springer, 2012.

[47] Yair Censor. "Superiorization and perturbation resilience of algorithms: a continuously updated bibliography". In: *arXiv preprint arXiv:1506.04219* (2021).

[48] Yair Censor. "Weak and Strong Superiorization: Between Feasibility-Seeking and Minimization". In: *Analele Universitatii "Ovidius" Constanta - Seria Matematica* 23.3 (2017), pp. 41–54. DOI: doi:10.1515/auom-2015-0046. URL: https://doi.org/10.1515/auom-2015-0046.

[49] Yair Censor and Andrzej Cegielski. "Projection Methods: An Annotated Bibliography of Books and Reviews". In: *Optimization* 64.11 (2015), pp. 2343–2358.

[50] Yair Censor, Ran Davidi, and Gabor T Herman. "Perturbation resilience and superiorization of iterative algorithms". In: *Inverse Problems* 26.6 (2010), p. 065008.

[51] Yair Censor and Alexander Segal. "Iterative projection methods in biomedical inverse problems". In: *Mathematical methods in biomedical imaging and intensity-modulated radiation therapy (IMRT)* 10 (2008), pp. 65–96.

[52] Yair Censor and Alexander Segal. “On the string averaging method for sparse common fixed-point problems”. In: *International Transactions in Operational Research* 16.4 (2009), pp. 481–494.

[53] Yair Censor and Eli Tom. “Convergence of string-averaging projection schemes for inconsistent convex feasibility problems”. In: *Optimization Methods and Software* 18.5 (2003), pp. 543–554.

[54] Yair Censor and Alexander J Zaslavski. “Convergence and perturbation resilience of dynamic string-averaging projection methods”. In: *Computational Optimization and Applications* 54.1 (2013), pp. 65–76.

[55] Yair Censor et al. “On diagonally relaxed orthogonal projection methods”. In: *SIAM Journal on Scientific Computing* 30.1 (2008), pp. 473–504.

[56] Yair Censor et al. “On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints”. In: *Computational Optimization and Applications* 51.3 (2012), pp. 1065–1088.

[57] Raymond H Chan et al. “A two-stage method for spectral–spatial classification of hyperspectral images”. In: *Journal of Mathematical Imaging and Vision* (2020), pp. 1–18.

[58] Stanley H Chan, Xiran Wang, and Omar A Elgendy. “Plug-and-play ADMM for image restoration: Fixed-point convergence and applications”. In: *IEEE Transactions on Computational Imaging* 3.1 (2016), pp. 84–98.

[59] Tony Chan, Antonio Marquina, and Pep Mulet. “High-order total variation-based image restoration”. In: *SIAM Journal on Scientific Computing* 22.2 (2000), pp. 503–516.

[60] Bo Chang et al. “Reversible architectures for arbitrarily deep residual neural networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[61] Ricky TQ Chen et al. “Neural ordinary differential equations”. In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.

[62] Tianlong Chen et al. “Learning to optimize: A primer and a benchmark”. In: *arXiv preprint arXiv:2103.12828* (2021).

[63] Xiaohan Chen et al. “Theoretical Linear Convergence of Unfolded ISTA and Its Practical Weights and Thresholds”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 9061–9071.

[64] Xiaohan Chen et al. “Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds”. In: *arXiv preprint arXiv:1808.10038* (2018).

[65] Gianfranco Cimmino. “Cacolo approssimato per le soluzioni dei systemi di equazioni lineari”. In: *La Ricerca Scientifica (Roma)* 1 (1938), pp. 326–333.

[66] Moustapha Cisse et al. “Parseval networks: Improving robustness to adversarial examples”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 854–863.

[67] Regev Cohen, Michael Elad, and Peyman Milanfar. “Regularization by denoising via fixed-point projection (red-pro)”. In: *arXiv preprint arXiv:2008.00226* (2020).

[68] Patrick L Combettes and Jean-Christophe Pesquet. “Lipschitz certificates for layered network structures driven by averaged activation operators”. In: *SIAM Journal on Mathematics of Data Science* 2.2 (2020), pp. 529–557.

[69] Balázs Csanád Csáji et al. “Approximation with artificial neural networks”. In: *Faculty of Sciences, Eötvös LoràND University, Hungary* 24.48 (2001), p. 7.

[70] Felipe Cucker and Steve Smale. “Best choices for regularization parameters in learning theory: on the bias-variance problem”. In: *Foundations of computational Mathematics* 2.4 (2002), pp. 413–428.

[71] Zhiyong Cui et al. “Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.11 (2019), pp. 4883–4894.

[72] Stella Dafermos. “Sensitivity analysis in variational inequalities”. In: *Mathematics of Operations Research* 13.3 (1988), pp. 421–434.

[73] I. Daubechies, M. Defrise, and C. De Mol. “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint”. en. In: *Communications on Pure and Applied Mathematics* 57.11 (2004), pp. 1413–1457.

[74] Ran Davidi, Gabor T Herman, and Yair Censor. “Perturbation-resilient block-iterative projection methods with application to image reconstruction from projections”. In: *International Transactions in Operational Research* 16.4 (2009), pp. 505–524.

[75] Damek Davis and Wotao Yin. “A three-operator splitting scheme and its optimization applications”. In: *Set-valued and variational analysis* 25.4 (2017), pp. 829–858.

[76] Jelena Diakonikolas. “Halpern iteration for near-optimal and parameter-free monotone inclusion and strong solutions to variational inequalities”. In: *Conference on Learning Theory*. PMLR. 2020, pp. 1428–1451.

[77] Robert B Dial. “A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration”. In: *Transportation Research Part B: Methodological* 40.10 (2006), pp. 917–936.

[78] David L Donoho. “Compressed sensing”. In: *IEEE Transactions on information theory* 52.4 (2006), pp. 1289–1306.

[79] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. “Explainable artificial intelligence: A survey”. In: *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE. 2018, pp. 0210–0215.

[80] John Duchi et al. “Efficient projections onto the l 1-ball for learning in high dimensions”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 272–279.

[81] Dan E. Dudgeon and Russell M. Mersereau. “Multidimensional Digital Signal Processing”. In: *Prentice Hall Professional Technical Reference* (1990).

[82] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. “Augmented Neural ODEs”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf.

[83] Michael Elad, Mario AT Figueiredo, and Yi Ma. “On the role of sparse and redundant representations in image processing”. In: *Proceedings of the IEEE* 98.6 (2010), pp. 972–982.

[84] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[85] Jianqing Fan and Runze Li. "Variable selection via nonconcave penalized likelihood and its oracle properties". In: *Journal of the American statistical Association* 96.456 (2001), pp. 1348–1360.

[86] Richard Feynman. *Atoms in Motion*. Remarks by Richard Feynman during his lecture at California Institute of Technology [Accessed: 2021 12 01]. Sept. 1961. URL: https://www.feynmanlectures.caltech.edu/I_01.html.

[87] James R Fienup. "Phase retrieval algorithms: A comparison". In: *Applied optics* 21.15 (1982), pp. 2758–2769.

[88] Chris Finlay et al. "How to train your neural ODE". In: *arXiv preprint arXiv:2002.02798* (2020).

[89] Chris Finlay et al. "Lipschitz regularized deep neural networks generalize and are adversarially robust". In: *arXiv preprint arXiv:1808.09540* (2018).

[90] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proc. International Conference on Machine Learning (ICML)*. 2017, pp. 1126–1135.

[91] Marguerite Frank and Philip Wolfe. "An algorithm for quadratic programming". In: *Naval research logistics quarterly* 3.1-2 (1956), pp. 95–110.

[92] Benjamin Franklin. *On Protection of Towns from Fire, 4 February 1735*. 1734. URL: https://founders.archives.gov/documents/Franklin/01-02-02-0002.

[93] Samy Wu Fung. "Large-Scale Parameter Estimation in Geophysics and Machine Learning". PhD thesis. Emory University, 2019.

[94] Samy Wu Fung and Lars Ruthotto. "A multiscale method for model order reduction in PDE parameter estimation". In: *Journal of Computational and Applied Mathematics* 350 (2019), pp. 19–34.

[95] Samy Wu Fung and Lars Ruthotto. "An uncertainty-weighted asynchronous ADMM method for parallel PDE parameter estimation". In: *SIAM Journal on Scientific Computing* 41.5 (2019), S129–S148.

[96] Samy Wu Fung and Zichao Wendy. "Multigrid optimization for large-scale ptychographic phase retrieval". In: *SIAM Journal on Imaging Sciences* 13.1 (2020), pp. 214–233.

[97] Samy Wu Fung et al. "JFB: Jacobian-free Backpropagation for Implicit Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2022).

[98] Bolin Gao and Lacra Pavel. "On the properties of the softmax function with application in game theory and reinforcement learning". In: *arXiv preprint arXiv:1704.00805* (2017).

[99] Zhengyang Geng et al. "Is Attention Better Than Matrix Decomposition?" In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=1FvkSpWosOl.

[100] Zhengyang Geng et al. "On Training Implicit Models". In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021.

[101] Laurent El Ghaoui et al. "Implicit Deep Learning". In: *arXiv preprint arXiv:1908.06315* (2019).

[102] Amir Gholami, Kurt Keutzer, and George Biros. "ANODE: Unconditionally accurate memory-efficient gradients for neural ODEs". In: *arXiv preprint arXiv:1902.10298* (2019).

[103] Davis Gilton, Gregory Ongie, and Rebecca Willett. "Deep Equilibrium Architectures for Inverse Problems in Imaging". In: *arXiv preprint arXiv:2102.07944* (2021).

[104] R. Giryes et al. "Tradeoffs Between Convergence Speed and Reconstruction Accuracy in Inverse Problems". In: *IEEE Transactions on Signal Processing* 66.7 (2018), pp. 1676–1690.

[105] Pontus Giselsson, Mattias Falt, and Stephen Boyd. "Line search for averaged operator iteration". en. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. Las Vegas, NV, USA: IEEE, Dec. 2016, pp. 1015–1022.

[106] Tom Goldstein and Stanley Osher. "The Split Bregman Method for L1-Regularized Problems". In: *SIAM Journal on Imaging Sciences* 2.2 (2009), pp. 323–343.

[107] Gene H Golub, Per Christian Hansen, and Dianne P O'Leary. "Tikhonov regularization and total least squares". In: *SIAM journal on matrix analysis and applications* 21.1 (1999), pp. 185–194.

[108] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[109] Dan Gordon and Rachel Gordon. "Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems". In: *SIAM Journal on Scientific Computing* 27.3 (2005), pp. 1092–1117.

[110] Henry Gouk et al. "Regularisation of neural networks by enforcing Lipschitz continuity". In: *Machine Learning* 110.2 (Feb. 2021), pp. 393–

416. ISSN: 1573-0565. DOI: 10.1007/s10994-020-05929-w. URL: https://doi.org/10.1007/s10994-020-05929-w.

[111] Stephen Gould, Richard Hartley, and Dylan Campbell. "Deep declarative networks: A new hope". In: *arXiv preprint arXiv:1909.04866* (2019).

[112] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th international conference on international conference on machine learning*. 2010, pp. 399–406.

[113] Shengnan Guo et al. "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 922–929.

[114] E Haber et al. "Fast simulation of 3D electromagnetic problems using potentials". In: *Journal of Computational Physics* 163.1 (2000), pp. 150–171.

[115] Eldad Haber. *Computational methods in geophysical electromagnetics*. SIAM, 2014.

[116] Eldad Haber, Uri M Ascher, and Douglas W Oldenburg. "Inversion of 3D electromagnetic data in frequency and time domain using an inexact all-at-once approach". In: *Geophysics* 69.5 (2004), pp. 1216–1228.

[117] Eldad Haber and Lars Ruthotto. "Stable architectures for deep neural networks". In: *Inverse Problems* 34.1 (2017), p. 014004.

[118] Benjamin Halpern. "Fixed points of nonexpanding maps". In: *Bulletin of the American Mathematical Society* 73.6 (1967), pp. 957–961.

[119] James Hannan. "APPROXIMATION TO BAYES RISK IN REPEATED PLAY". In: *Contributions to the Theory of Games* 21.39 (1957), p. 97.

[120] Per Christian Hansen, James G Nagy, and Dianne P O'leary. *Deblurring images: matrices, spectra, and filtering*. SIAM, 2006.

[121] Sergiu Hart and Andreu Mas-Colell. "A simple adaptive procedure leading to correlated equilibrium". In: *Econometrica* 68.5 (2000), pp. 1127–1150.

[122] Hongjin He and Hong-Kun Xu. "Perturbation resilience and superiorization methodology of averaged mappings". In: *Inverse Problems* 33.4 (2017), p. 044007.

[123] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[124] Howard Heaton and Yair Censor. "Asynchronous sequential inertial iterations for common fixed points problems with an application to linear systems". In: *Journal of Global Optimization* 74.1 (May 2019), pp. 95–119. ISSN: 1573-2916.

[125] Howard Heaton et al. "Feasibility-based fixed point networks". In: *Fixed Point Theory and Algorithms for Sciences and Engineering* (2021).

[126] Howard Heaton et al. "Learn to Predict Equilibria via Fixed Point Networks". In: *arXiv preprint arXiv:2106.00906* (2021).

[127] Howard Heaton et al. "Safeguarded Learned Convex Optimization". In: *arXiv preprint arXiv:2003.01880* (2020).

[128] Howard Heaton et al. "Wasserstein-based Projection with Applications to Inverse Problems". In: *arXiv preprint arXiv:2008.02200* (2020).

[129] Gabor T Herman et al. "Superiorization: An optimization heuristic for medical physics". In: *Medical physics* 39.9 (2012), pp. 5532–5546.

[130] John R. Hershey, Jonathan Le Roux, and Felix Weninger. "Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures". In: *arXiv:1409.2574* (2014).

[131] David Hilbert. "Mathematical problems". In: *Bulletin of the American Mathematical Society* 8.10 (1902), pp. 437–479.

[132] Zhichun Huang, Shaojie Bai, and J Zico Kolter. "Implicit^2: Implicit Layers for Implicit Representations". In: *Advances in Neural Information Processing Systems* 34 (2021).

[133] Tony Humphries, J Winn, and Adel Faridani. "Superiorized algorithm for reconstruction of CT images from sparse-view and limited-angle polyenergetic data". In: *Physics in Medicine & Biology* 62.16 (2017), p. 6762.

[134] Antony Jameson. "Aerodynamic design via control theory". In: *Journal of scientific computing* 3.3 (1988), pp. 233–260.

[135] Younghan Jeon, Minsik Lee, and Jin Young Choi. "Differentiable Forward and Backward Fixed-Point Iteration Layers". In: *IEEE Access* (2021).

[136] Changhui Jiang et al. "Super-resolution ct image reconstruction based on dictionary learning and sparse representation". In: *Scientific reports* 8.1 (2018), pp. 1–10.

[137] 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.

[138] S Kaczmarz. "Angenaherte Auflosung von Systemen linearer Gleichungen". In: *Bulletin International de L'Académie Polonaise des Sciences et des Lettres A* (1937).

[139] Kelvin Kan, Samy Wu Fung, and Lars Ruthotto. "PNKH-B: A Projected Newton–Krylov Method for Large-Scale Bound-Constrained Optimization". In: *SIAM Journal on Scientific Computing* 0 (2021), S704–S726.

[140] Patrick Kidger and Terry Lyons. "Universal approximation with deep narrow networks". In: *Conference on Learning Theory*. PMLR. 2020, pp. 2306–2327.

[141] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR (Poster)*. 2015.

[142] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[143] Mikhail Viktorovich Klibanov. "Determination of a compactly supported function from the argument of its Fourier transform". In: *Doklady Akademii Nauk*. Vol. 289. 3. Russian Academy of Sciences. 1986, pp. 539–540.

[144] Erich Kobler et al. "Variational networks: connecting variational methods and deep learning". In: *German conference on pattern recognition*. Springer. 2017, pp. 281–293.

[145] Ioannis C Konstantakopoulos et al. "Social game for building energy efficiency: Utility learning, simulation, and analysis". In: *arXiv preprint arXiv:1407.0727* (2014).

[146] James Kotary et al. "End-to-End Constrained Optimization Learning: A Survey". In: *arXiv preprint arXiv:2103.16378* (2021).

[147] Yuichiro Koyama et al. “Music Source Separation with Deep Equilibrium Models”. In: *arXiv preprint arXiv:2110.06494* (2021).

[148] M.A. Krasnosel’skiĭ. “Two remarks about the method of successive approximations”. In: *Uspekhi Mat. Nauk* 10 (1955), pp. 123–127.

[149] Erwin Kreyszig. *Introductory Functional Analysis with Applications*. Vol. 1. Wiley New York, 1978.

[150] Alex Krizhevsky and G. Hinton. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, 2009.

[151] Nathan Lawrence et al. “Almost Surely Stable Deep Dynamics”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 18942–18953. URL: https://proceedings.neurips.cc/paper/2020/file/daecf755df5b1d637033bb29b319c39a-Paper.pdf.

[152] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[153] Johannes Leuschner et al. “The LoDoPaB-CT dataset: A benchmark dataset for low-dose CT reconstruction methods”. In: *arXiv preprint arXiv:1910.01113* (2019).

[154] Jiayang Li et al. “End-to-End Learning and Intervention in Games”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[155] Shuang Li et al. “Cubic regularization for differentiable games”. In: *NeurIPS Workshop 2019*.

[156] Yaguang Li et al. “Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting”. In: *International Conference on Learning Representations*. 2018.

[157] Renjie Liao et al. “Reviving and improving recurrent back-propagation”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3082–3091.

[158] Johan Lie and Jan M Nordbotten. “Inverse scale spaces for nonlinear regularization”. In: *Journal of Mathematical Imaging and Vision* 27.1 (2007), pp. 41–50.

[159] Alex Tong Lin et al. “Alternating the population and control neural networks to solve high-dimensional stochastic mean-field games”. In: *Proceedings of the National Academy of Sciences* 118.31 (2021).

[160] Alex Tong Lin et al. “APAC-Net: Alternating the population and agent control via two neural networks to solve high-dimensional stochastic mean field games”. In: *arXiv preprint arXiv:2002.10113* (2020).

[161] Chun Kai Ling, Fei Fang, and J Zico Kolter. “Large scale learning of agent rationality in two-player zero-sum games”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6104–6111.

[162] Chun Kai Ling, Fei Fang, and J Zico Kolter. “What game are we playing? end-to-end learning in normal and extensive form games”. In: *arXiv preprint arXiv:1805.02777* (2018).

[163] Jialin Liu and Xiaohan Chen. “ALISTA: Analytic weights are as good as learned weights in LISTA”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[164] Jialin Liu et al. “ALISTA: Analytic Weights Are As Good As Learned Weights in LISTA”. In: *International Conference on Learning Representations*. 2019.

[165] Andreas Look et al. “Differentiable Implicit Layers”. In: *arXiv preprint arXiv:2010.07078* (2020).

[166] Dirk A Lorenz et al. “A sparse Kaczmarz solver and a linearized Bregman method for online compressed sensing”. In: *2014 IEEE international conference on image processing (ICIP)*. IEEE. 2014, pp. 1347–1351.

[167] Jonathan Lorraine, Paul Vicol, and David Duvenaud. “Optimizing millions of hyperparameters by implicit differentiation”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 1540–1552.

[168] Yiping Lu et al. “Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3276–3285.

[169] Zhou Lu et al. “The expressive power of neural networks: A view from the width”. In: *arXiv preprint arXiv:1709.02540* (2017).

[170] Jelena Luketina et al. “Scalable gradient-based tuning of continuous regularization hyperparameters”. In: *International conference on machine learning*. PMLR. 2016, pp. 2952–2960.

[171] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. “Adversarial regularizers in inverse problems”. In: *Advances in Neural Information Processing Systems* (2018), pp. 8507–8516.

[172] Robert Mann. “Mean Value Methods in Iteration”. In: 4.3 (1953), pp. 506–510.

[173] D. Martin et al. “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics”. In: *Proc. 8th Int’l Conf. Computer Vision*. Vol. 2. July 2001, pp. 416–423.

[174] Chris Metzler, Ali Mousavi, and Richard Baraniuk. “Learned D-AMP: Principled Neural Network based Compressive Image Recovery”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 1772–1783.

[175] Margaret Mitchell et al. “Model cards for model reporting”. In: *Proceedings of the conference on fairness, accountability, and transparency*. 2019, pp. 220–229.

[176] Michael Moeller, Thomas Mollenhoff, and Daniel Cremers. “Controlling neural networks via energy dissipation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3256–3265.

[177] Vishal Monga, Yuelong Li, and Yonina C Eldar. “Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing”. In: *IEEE Signal Processing Magazine* 38.2 (2021), pp. 18–44.

[178] Eliakim H Moore. “On the reciprocal of the general algebraic matrix”. In: *Bulletin of the American Mathematical Society* 26 (1920), pp. 394–395.

[179] Thomas Moreau and Joan Bruna. “Understanding Trainable Sparse Coding with Matrix Factorization”. In: 2017.

[180] Katharina Morik et al. “The Care Label Concept: A Certification Suite for Trustworthy and Resource-Aware Machine Learning”. In: *arXiv preprint arXiv:2106.00512* (2021).

[181] Katharina Morik et al. “Yes We Care!–Certification for Machine Learning Methods through the Care Label Framework”. In: *arXiv preprint arXiv:2105.10197* (2021).

[182] John F Nash et al. “Equilibrium points in n-person games”. In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[183] Yuval Netzer et al. “Reading digits in natural images with unsupervised feature learning”. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 2011.

[184] Daniel O’Connor and Lieven Vandenberghe. “Primal-dual decomposition by operator splitting and applications to image deblurring”. In: *SIAM Journal on Imaging Sciences* 7.3 (2014), pp. 1724–1754.

[185] Derek Onken and Lars Ruthotto. “Discretize-Optimize vs. Optimize-Discretize for Time-Series Regression and Continuous Normalizing Flows”. In: *arXiv preprint arXiv:2005.13420* (2020).

[186] Derek Onken et al. “OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.10 (2021), pp. 9223–9232. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17113.

[187] C. E. Ordoñez et al. “A real-time image reconstruction system for particle treatment planning using proton computed tomography (pct)”. In: *Physics Procedia* 90 (2017), pp. 193–199.

[188] Stanley Osher, Zuoqiang Shi, and Wei Zhu. “Low dimensional manifold model for image processing”. In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1669–1690.

[189] Stanley Osher et al. "An iterative regularization method for total variation-based image restoration". In: *Multiscale Modeling & Simulation* 4.2 (2005), pp. 460–489.

[190] Dario Paccagnan et al. "Nash and Wardrop equilibria in aggregative games with coupling constraints". In: *IEEE Transactions on Automatic Control* 64.4 (2018), pp. 1373–1388.

[191] Adam Paszke et al. "Automatic differentiation in PyTorch". In: (2017).

[192] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems*. 2019, pp. 8026–8037.

[193] Michael Patriksson and R Tyrrell Rockafellar. "Sensitivity analysis of variational inequalities over aggregated polyhedra, with application to traffic equilibria". In: *Transportation Science* 37.1 (2003), pp. 56–68.

[194] S. Penfold et al. "Block-iterative and string-averaging projection algorithms in proton computed tomography image reconstruction". In: *in: Censor Y., Jiang M., Wang G. (eds), Biomedical Mathematics: Promising Directions in Imaging, Therapy Planning and Inverse Problems, Medical Physics Publishing* Madison (2010), pp. 347–368.

[195] S.N. Penfold et al. "Total variation superiorization schemes in proton computed tomography image reconstruction". In: *Medical physics* 37.11 (2010), pp. 5887–5895.

[196] Roger Penrose. "A generalized inverse for matrices". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 51. 3. Cambridge University Press. 1955, pp. 406–413.

[197] Gabriel Peyré. "Manifold models for signals and images". In: *Computer vision and image understanding* 113.2 (2009), pp. 249–260.

[198] James Pita et al. "Deployed ARMOR protection: The application of a game theoretic model for security at the Los Angeles International Airport". In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*. 2008, pp. 125–132.

[199] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[200] Aravind Rajeswaran et al. "Meta-Learning with Implicit Gradients". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/072b030ba126b2f4b2374f342be9ed44-Paper.pdf.

[201] Lillian J Ratliff et al. "Social game for building energy efficiency: Incentive design". In: *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2014, pp. 1011–1018.

[202] Simeon Reich. "Weak convergence theorems for nonexpansive mappings in Banach spaces". In: *Journal of Mathematical Analysis and Applications* 67.2 (1979), pp. 274–276.

[203] Max Revay and Ian Manchester. "Contracting implicit recurrent neural networks: Stable models with improved trainability". In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 393–403.

[204] Wolfgang Ring. "Structural properties of solutions to total variation regularization problems". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 34.4 (2000), pp. 799–810.

[205] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[206] R Tyrrell Rockafellar. *Convex analysis*. Vol. 36. Princeton university press, 1970.

[207] J Ben Rosen. "Existence and uniqueness of equilibrium points for concave n-person games". In: *Econometrica: Journal of the Econometric Society* (1965), pp. 520–534.

[208] Tim Roughgarden. "Routing games". In: *Algorithmic game theory* 18 (2007), pp. 459–484.

[209] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[210] Lars Ruthotto and Eldad Haber. "An Introduction to Deep Generative Modeling". In: *arXiv preprint arXiv:2103.05180* (2021).

[211] Lars Ruthotto and Eldad Haber. "Deep neural networks motivated by partial differential equations". In: *Journal of Mathematical Imaging and Vision* (2019), pp. 1–13.

[212] Lars Ruthotto et al. "A machine learning framework for solving high-dimensional mean field game and mean field control problems". In: *Proceedings of the National Academy of Sciences* 117.17 (2020), pp. 9183–9193.

[213] Ernest Ryu and Wotao Yin. *Large-Scale Convex Optimization: Algorithm Designs via Monotone Operators*. Cambridge, England: Cambridge University Press, 2022.

[214] Wojciech Samek and Klaus-Robert Müller. "Towards explainable artificial intelligence". In: *Explainable AI: interpreting, explaining and visualizing deep learning*. Springer, 2019, pp. 5–22.

[215] Frank Schöpfer and Dirk A Lorenz. "Linear convergence of the randomized sparse Kaczmarz method". In: *Mathematical Programming* 173.1 (2019), pp. 509–536.

[216] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/3a0844cee4fcf57de0c71e9ad3035478-Paper.pdf.

[217] Pier Giuseppe Sessa et al. "Contextual Games: Multi-Agent Learning with Side Information". In: *Advances in Neural Information Processing Systems* 33 (2020).

[218] Jiayi Shen et al. "Learning A Minimax Optimizer: A Pilot Study". In: *International Conference on Learning Representations (ICLR)*. 2021.

[219] Nir Shlezinger et al. "Model-Based Deep Learning". In: *arXiv preprint arXiv:2012.08405* (2020).

[220] Jure Sokolić et al. "Robust large margin deep neural networks". In: *IEEE Transactions on Signal Processing* 65.16 (2017), pp. 4265–4280.

[221] P. Sprechmann, A. M. Bronstein, and G. Sapiro. "Learning Efficient Sparse and Low Rank Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (Sept. 2015), pp. 1821–1833.

[222] Gilles Stoltz and Gábor Lugosi. "Learning correlated equilibria in games with compact sets of strategies". In: *Games and Economic Behavior* 59.1 (2007), pp. 187–208.

[223] Andreas Themelis and Panagiotis Patrinos. "SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators". In: (2019).

[224] *Transportation Networks for Research Core Team. Transportation Networks for Research.* `https://github.com/bstabler/TransportationNetworks`. Accessed: 2021-05-24.

[225] Madeleine Udell and Alex Townsend. "Why are big data matrices approximately low rank?" In: *SIAM Journal on Mathematics of Data Science* 1.1 (2019), pp. 144–160.

[226] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[227] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. "Plug-and-Play priors for model based reconstruction". In: *2013 IEEE Global Conference on Signal and Information Processing*. 2013, pp. 945–948.

[228] Geroge Viereck. "What Life Means to Einstein". In: *The Saturday Evening Post* (1929).

[229] Ernesto De Vito et al. "Learning from examples as an inverse problem". In: *Journal of Machine Learning Research* 6.May (2005), pp. 883–904.

[230] John Von Neumann. “On the theory of games of strategy”. In: *Contributions to the Theory of Games* 4 (1959), pp. 13–42.

[231] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1953.

[232] Weiran Wang and Miguel A Carreira-Perpinán. “Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application”. In: *arXiv preprint arXiv:1309.1541* (2013).

[233] Zhangyang Wang, Qing Ling, and Thomas S. Huang. “Learning Deep ℓ_0 Encoders”. en. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[234] Zhangyang Wang et al. “D3: Deep Dual-Domain Based Fast Restoration of JPEG-Compressed Images”. In: 2016, pp. 2764–2772.

[235] John Glen Wardrop. “Some theoretical aspects of road traffic research.” In: *Proceedings of the institution of civil engineers* 1.3 (1952), pp. 325–362.

[236] Kevin Waugh, Brian D Ziebart, and J Andrew Bagnell. “Computational rationalization: the inverse equilibrium problem”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, pp. 1169–1176.

[237] E Weinan. “A proposal on machine learning via dynamical systems”. In: *Communications in Mathematics and Statistics* 5.1 (2017), pp. 1–11.

[238] Alfred North Whitehead. *An introduction to mathematics*. Henry Holt and Company, 1911.

[239] Ezra Winston and J. Zico Kolter. “Monotone operator equilibrium networks”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 10718–

10728. URL: https://proceedings.neurips.cc/paper/2020/file/798d1c2813cbdf8bcdb388db0e32d496-Paper.pdf.

[240] Samy Wu Fung et al. "ADMM-SOFTMAX: AN ADMM APPROACH FOR MULTINOMIAL LOGISTIC REGRESSION". In: *Electronic Transactions on Numerical Analysis* 52 (2020), pp. 214–229.

[241] Xingyu Xie et al. "Differentiable Linearized ADMM". In: *arXiv:1905.06179* (2019).

[242] Bo Xin et al. "Maximal Sparsity with Deep Networks?" In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 4340–4348.

[243] Li Xu et al. "Deep convolutional neural network for image deconvolution". In: *Advances in neural information processing systems* 27 (2014), pp. 1790–1798.

[244] Qiong Xu et al. "Low-dose X-ray CT reconstruction via dictionary learning". In: *IEEE transactions on medical imaging* 31.9 (2012), pp. 1682–1697.

[245] Yangyang Xu and Wotao Yin. "A fast patch-dictionary method for whole image recovery". In: *arXiv preprint arXiv:1408.3740* (2014).

[246] Rong Yang et al. "Adaptive resource allocation for wildlife protection against illegal poachers." In: *AAMAS*. 2014, pp. 453–460.

[247] Yan Yang et al. "Deep ADMM-Net for Compressive Sensing MRI". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 10–18.

[248] Jing Zhang and Ioannis Ch Paschalidis. "Data-driven estimation of travel latency cost functions via inverse optimization in multi-class

transportation networks". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 6295–6300.

[249] Jing Zhang et al. "The price of anarchy in transportation networks by estimating user cost functions from actual traffic data". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 789–794.

[250] Jing Zhang et al. "The price of anarchy in transportation networks: Data-driven evaluation and reduction strategies". In: *Proceedings of the IEEE* 106.4 (2018), pp. 538–553.

[251] Junzi Zhang, Brendan O'Donoghue, and Stephen Boyd. "Globally convergent type-I Anderson acceleration for non-smooth fixed-point iterations". In: *arXiv:1808.03971* (2018).

[252] Qianggong Zhang et al. "Implicitly defined layers in neural networks". In: *arXiv preprint arXiv:2003.01822* (2020).

[253] Zheng Zhang et al. "A survey of sparse representation: algorithms and applications". In: *IEEE access* 3 (2015), pp. 490–530.