UNIVERSITY OF CALIFORNIA,
MERCED

DISSERTATION

Large-Scale Optimization and Deep Learning Techniques for Data-Driven Signal Processing

by

Omar DeGuchy

A technical report submitted
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Applied Mathematics

2020

Committee Members:
Professor Roummel F. Marcia, Chair
Professor Harish S. Bhat
Professor Arnold D. Kim
Professor Suzanne S. Sindi
Professor Chrysoula Tsogka

The Dissertation of Omar DeGuchy is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Committee Member:

Professor Harish S. Bhat

Committee Member:

Professor Arnold D. Kim

Committee Member:

Professor Suzanne S. Sindi

Committee Member:

Professor Chrysoula Tsogka

Committee Chair / Research Advisor:

Professor Roummel F. Marcia

Date

Dedication

This dissertation is dedicated to my dearest parents, Isamaria, Louis, my brother Alex and to my lovely wife Ré.

Contents

Large-Scale Optimization and Deep Learning Techniques for Data-Driven Signal Processing

by

Omar DeGuchy

Doctorate in Applied Mathematics

Roummel F. Marcia, Chair

University of California, Merced

2020

Abstract

The collection of data has become an integral part of our everyday lives. The algorithms necessary to process this information become paramount to our ability to interpret this resource. This type of data is typically recorded in a variety of signals including images, sounds, time series, and bio-informatics. In this work, we develop a number of algorithms to recover these types of signals in a variety of modalities. This work is mainly presented in two parts.

Initially, we apply and develop large-scale optimization techniques used for signal processing. This includes the use of quasi-Newton methods to approximate second derivative information in a trust-region setting to solve regularized sparse signal recovery problems. We also formulate the compact representation of a large family of quasi-Newton methods known as the Broyden class. This extension of the classic quasi-Newton compact representation allows different updates to be used at every iteration. We also develop the algorithm to perform efficient solves with these representations. Within the realm of sparse signal recovery, but particular to photon-limited imaging applications, we also propose three novel algorithms for signal recovery in a low-light regime. First, we recover the support and lifetime decay of a flourophore from time dependent measurements. This type of modality is useful

in identifying different types of molecular structures in tissue samples. The second algorithm identifies and implements the Shannon entropy function as a regularization technique for the promotion of sparsity in reconstructed signals from noisy downsampled observations. Finally, we also present an algorithm which addresses the difficulty of choosing the optimal parameters when solving the sparse signal recovery problem. There are two parameters which effect the quality of the reconstruction, the norm being used, as well as the intensity of the penalization imposed by that norm. We present an algorithm which uses a parallel asynchronous search along with a metric in order to find the optimal pair.

The second portion of the dissertation draws on our experience with large-scale optimization and looks towards deep learning as an alternative to solving signal recovery problems. We first look to improve the standard gradient based techniques used during the training of these deep neural networks by presenting two novel optimization algorithms for deep learning. The first algorithm takes advantage of the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton algorithm in a trust-region setting in order to address the large scale minimization problem associated with deep learning. The second algorithm uses second derivative information in a trust-region setting where the Hessian is not explicitly stored. We then use a conjugate based method in order to solve the trust-region subproblem.

Finally, we apply deep learning techniques to a variety of applications in signal recovery. These applications include revisiting the photon-limited regime where we recover signals from noisy downsampled observations, image disambiguation which involves the recovery of two signals which have been superimposed, deep learning for synthetic aperture radar (SAR) where we recover information describing the imaging system as well as evaluate the impact of reconstruction on the ability to perform target detection, and signal variation detection in the human genome where we leverage the relationships between subjects to provide better detection.

List of Notations and Abbreviations

$\mathbf{f}^*$	Signal or image vector to be reconstructed
$\hat{\mathbf{f}}$	Estimate of the signal or image vector
$\mathbf{A}$	System matrix or observation matrix
$\mathbf{y}$	Measurement vector corrupted by Poisson noise
$\mathbf{b}$	Measurement vector corrupted by Gaussian noise
$\mathbb{1}$	Vector of ones
τ	Weight of the penalization term
p	p-norm parameter
pen	Non-convex penalty function
I	Identity matrix of appropriate size
B_k	Quasi-Newton Hessian approximation
H_k	Inverse quasi-Newton Hessian approximation
W	Weight matrix
g	Activation function
b	Bias
h	Hidden layer
d	Measured SAR data
ρ	Unknown reflectivity
θ	Trainable parameters
$\mathcal{L}(\theta)$	Loss or cost function
Adam	Adaptive moment estimation algorithm
backprop	Back-propagation algorithm
BCE	Binary cross entropy
CCD	Charged coupled devices
CG	Conjugate gradient

CNN	Convolutional Neural Network
CPU	Central processing unit
DNN	Deep neural network
EM	Expectation-maximization algorithm
FLIM	Fluorescence lifetime imaging
FPA	Focal plane array
GPSR	Gradient projection for sparse reconstruction
GPU	Graphics processing unit
GSS	Generating set search
HOPSPACK	Hybrid optimization parallel search package
L-BFGS	Limited memory Broyden-Fletcher-Goldfarb-Shanno update
MLP	Multilayer perceptron
MRI	Magnetic resonance imaging
MSE	Mean-squared error
PAIN	Poisson autoencoder inverting network
PET	Positron emission tomography
PICS	Poisson inverting convolutions
QR	Quick Response
ReLU	Rectified linear unit
RGB	Red, green, blue
RMSE	Root mean square error
RMSProp	Root mean square propagation
SAR	Synthetic aperture radar
SDA	Stacked denoising autoencoders
SGD	Stochastic gradient descent
SNR	Signal to noise ratio

SR1	Symmetric rank one update
SV	Structural variant
YALL1	Your algorithm for L1

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my adviser Professor Roummel F. Marcia for his mentorship throughout my graduate studies. He has been my greatest champion, encouraging my research efforts and providing me with opportunities to disseminate my work. His influence on my professional development as a researcher is immeasurable and will carry on through the rest of my career. There are no words to express the positive impact he has had on my life. Perhaps the only way to thank him is to follow his example and try to pass on the knowledge he has passed on to me. Thank you for taking a chance on me.

A very special thank you goes to my committee members, Professor Harish S. Bhat in Applied Mathematics for introducing me to deep learning and changing the course of my research, Professor Arnold D. Kim for his guidance in our research on light propagation and for becoming a friend with whom I can discuss a variety of topics, including food. I would also like to thank Professor Suzanne S. Sindi for being an outstanding mentor in research and in graduate life, particularly in the first year when I needed it most. Finally, I would like to thank Professor Chrysoula Tsogka for sharing her knowledge in synthetic aperture radar and for being yet another great mentor in my life.

Besides my Ph.D. committee, I would like to take this opportunity to thank all of my research collaborators. From the department of electrical engineering and computer science at UC Merced I would like to thank Dr. Jacob Rafatti, Prof. Mukesh Singhal, Aditya Runganath, and Michael Fernando Mendez Jimenez for their collaborative efforts. I would also like to thank Prof. Jennifer Erway from Wake Forest University and her contributions to my research. I give thanks to Dr. Reheman Baikejiang and his expertise on positron emission tomography. A very special thank you to my fellow interns at the Air Force Research Labs (AFRL) John Kaminski and Joseph Sebastian and mentors Benjamin Lewis, Fred Garber and Ed Zelnio. I would also like to thank my mentors at the Lawrence Livermore National Laboratory, Dr. Ignacio Aravena Solis and Dr. Cosmin Petra. Dr. Petra, thank you for giving me the opportunity to work on your team, I cannot thank you enough for the opportunities afforded to me from that experience and the impact it will have on my career. I would also like to thank Prof. Michael Stobb for countless discussions about our research and much appreciated suggestions. Finally, to my collaborators Fabian Santiago of Sindi Lab at UC Merced and Prof. Mario Banuelos at Fresno State, *el regalo màs grande de la vida es la amistad, y yo lo he recibido*, thank you brothers.

I would like to thank my academic family, Dr. Lasith Adhikari and Dr. Johannes Brust, thank you for laying out the blueprint. My labmates, Jackie Alvarez, Alex Ho, Ashley De Luna, and Andrew Lazar for being great collaborators and being willing to give me their best suggestions. I would also like to thank the members of my graduate cohort Matea Santiago, Alex Quijano and Dr. Jessica Taylor for their support. Much appreciation is given to the UC Merced SIAM student chapter and our fearless leader Prof. Noemi Petra, for her willingness to fight for our voice to be heard. Lastly, I just wanted to thank all of the staff, faculty and graduate students of the Applied Mathematics

department at UC Merced for making it feel like home.

A special gratitude goes out to NSF grants CMMI-1333326 and IIS-1741490, AFOSR grant FA9550-17-1-0238, and applied math department summer fellowships for helping to provide the funding for my research.

I would also like to thank Prof. Hubertus von Bremen, Prof. Arlo Caine, Prof. Randall Swift, Prof. John Rock and the rest of the Mathematics Department at Cal Poly Pomona who provided me with support and motivation in pursuing a Ph.D. degree.

I deeply thank my parents, Isamaria DeGuchy and Louis DeGuchy, for their sacrifice and unwavering support. Their patience throughout my academic career is something to marvel, and without it, none of my accomplishments would have been possible. I thank my brother, Alex DeGuchy, who always provides the laughter in my life when I need it the most. I would also like to thank the Bentson Family, especially Flint and Tyler, for always keeping me grounded as well as the rest of my family and friends for excusing me for all of the missed celebrations and gatherings.

Finally, I am grateful to my lovely wife Ré DeGuchy for her support, encouragement, and patience. Thank you for the home cooked meals when I didn't have time and understanding when I had to cancel our dates. Thank you for making me laugh when I was stressed and being there when I couldn't be. Thank you.

Curriculum Vitae

Education

2015–present PhD, Applied Mathematics
Advisor: Prof. Roummel F. Marcia
University of California, Merced. Merced, CA

2013–2015 MS, Mathematics
California State Polytechnic University, Pomona. Pomona, CA

2000-2006 BS, Business Management
California State University, Long Beach. Long Beach, CA

Publications

- L. Adhikari, R. Baikejiang, O. DeGuchy, and R.F. Marcia, "Non-Convex Shannon Entropy for Photon-Limited Imaging," *Proceedings of the SPIE Wavelets and Sparsity XVII, 2017*.
- L. Adhikari, O. DeGuchy, J.B. Erway, and R.F. Marcia, "Limited-Memory Trust-Region Methods for Sparse Relaxation," *Proceedings of the SPIE Wavelets and Sparsity XVII, 2017*.
- O. DeGuchy, L. Adhikari, A.D. Kim, R.F. Marcia, "Photon-Limited Fluorescence Lifetime Imaging Microscopy Signal Recovery With Known Bounds," *Proceedings of 2017 International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2017*.
- O. DeGuchy, J.B. Erway and R.F. Marcia, "Compact Representation of the Full Broyden Class of Quasi-Newton Updates." *Numerical Linear Algebra With Applications* 25.5 (2018): e2186.
- J. Rafati, O. DeGuchy, R.F. Marcia, "Trust-Region Minimization Algorithm for Training Responses (TRMinATR): The Rise of Machine Learning Techniques," *Proceedings of 26th European Signal Processing Conference, 2018*.
- B. Lewis, O. DeGuchy, J. Sebastian, and J. Kaminski, "Realistic SAR Data Augmentation Using Machine Learning Techniques," *Proceedings of SPIE Algorithms for Synthetic Aperture Radar Imagery XXVI, 2019*.
- O. DeGuchy, F. Santiago, M. Banuelos, and R.F. Marcia, "Deep Neural Networks for Low-Resolution Photon-Limited Imaging," *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing, 2019*.
- O. DeGuchy, A. Ho, and R.F. Marcia, "Image Disambiguation With Deep Neural Networks," *Proceedings of the SPIE Applications of Machine Learning, 2019*.

- O. DeGuchy, R.F. Marcia, "Parameter Tuning Using Asynchronous Parallel Pattern Search in Sparse Signal Reconstruction," *Proceedings of the SPIE Wavelets and Sparsity XVIII, 2019.*
- A. Rangantha, O. DeGuchy, M. Singhal, R.F. Marcia, "Second-Order Helping: A Hessian-Free Approach for Data-Hungry Inference," *Submitted, 2020.*
- O. DeGuchy, A.D. Kim, R.F. Marcia, C. Tsokga, "Forward and Inverse Scattering in Synthetic Aperture Radar Using Machine Learning," *Submitted, 2020.*
- J. Alvarez, O. DeGuchy, and R.F. Marcia, "Image Classification in Synthetic Aperture Radar Using Reconstruction From Learned Inverse Scattering," *Accepted to 2020 IEEE International Geoscience and Remote Sensing Symposium, 2020.*
- M. Banuelos, O. DeGuchy, S.S. Sindi, and R.F. Marcia, "Related Inference: A Supervised Learning Approach to Detect Signal Variation in Genome Data," *Submitted, 2020.*
- M. Mendez Jimenez, O. DeGuchy, and R.F. Marcia, "Machine Learning for Low-Resolution Blurred Denoising Signal Reconstruction," *Submitted, 2020.*

Chapter 1

Introduction

Over the course of this dissertation, we will present a variety of methods for data driven signal processing. As we become a data driven society, it is imperative to be able to extract the pertinent information from observed data so that we may use it to make informed decisions. This data takes shape as a variety of different types of signals. As such, the methods we use must be diverse and be able themselves to handle the diversity of the modalities of our observations.

1.1 Organization of the Dissertation

The overall structure of this dissertation is divided into two parts. In the first part of the dissertation (i.e. Chapters 2 and 3) will discuss the recovery of sparse signals and the large scale optimization algorithms suited to perform these signal reconstructions. In the second part of the dissertation, deep learning is presented as an alternative to numerical optimization as a signal recovery method. Chapters in this dissertation are organized as follows

Chapter 2: Large Scale Optimization

In this chapter, we propose two novel algorithms for large scale optimization applications: (1) A limited memory trust-region method for the recovery of sparse signals under a Gaussian assumption, (2) a compact representation of the full Broyden class of quasi-Newton updates.

Chapter 3: Sparse Optimization

In this chapter we lay the foundation for sparse signal recovery under an assumption of Poisson noise. The Poisson model is motivated by applications in photon-limited or low light imaging. We propose three novel methods for the recovery of sparse signals in this regime: (1) A bounded photon-limited image recovery algorithm for applications in fluorescence lifetime imaging microscopy (FLIM) where measurements are time

dependent, (2) a non-convex Shannon Entropy regularization method to promote sparsity in the recovery of sparse signals and (3) a parallel asynchronous pattern search method relying on derivative free optimization techniques in order to tune hyperparameters necessary to reconstruction in a photon-limited regime.

Chapter 4: Deep Learning

In this chapter we transition to the deep learning portion of the dissertation. We discuss the overall structure of the feed forward network as well as the different types of components that help build a neural network architecture. In particular, we discuss fully connected layers, convolutional layers, activation functions, cost functions and we lightly touch on the Back-propagation algorithm involved in the optimization routine necessary during the learning process.

Chapter 5: Optimization for Deep Learning Applications

In this chapter we discuss the current methods used in the gradient based training of neural networks. We also propose two novel algorithms for parameter learning in deep learning: (1) We extend beyond the gradient descent algorithm and use quasi-Newton approximations in a trust-region setting in order to estimate curvature information and improve the learning process, (2) a trust-region method which uses second order information without explicitly storing the Hessian and solves the trust-region subproblem using a conjugate gradient method.

Chapter 6: Applications in Deep Learning

In this chapter we focus on a variety of applications of deep learning for signal recovery. First, we revisit a photon-limited regime and show the ability of neural networks to recover signals from noisy downsampled observations. Second, we employ fully connected layers to perform blind source separation in the domain of images. Third, we switch modalities to remote sensing where we apply simple neural networks to recover the system matrix describing the physics in a synthetic aperture radar imaging system. We also utilize this architecture to recover an inverse approximation to solve the linear system recovery problem. We also investigate the effects of the quality of signal reconstruction in target classification. Finally, we investigate the use of neural networks as well as other machine learning algorithms in the detection of signal variations in the human genome.

Chapter 7: Conclusion

This chapter concludes the dissertation and makes suggestions for extending this work in the future.

Chapter 2

Large-Scale Optimization

As our society becomes increasingly data driven, the need for algorithms which implement efficient scalable numerical optimization routines have become increasingly more important. These types of algorithms have applications in machine learning, cancer research and power grid distribution [24, 26, 37]. One class of these types of algorithms known as quasi-Newton methods have become increasingly popular as the amount of available computing power increases. Initially discovered by William Davidon of Argonne National Laboratory in 1959 and left unpublished until 1963 as the first quasi-Newton method, these methods use gradient-based information in order to approximate second derivative information. The result is an algorithm with super linear convergence which is an improvement on most gradient based methods [33]. In this chapter we present two novel algorithms which take advantage of quasi-Newton methods. The first algorithm is motivated by applications in compressive sensing. We apply the quasi-Newton methods in a trust-region setting in order to recover a sparse signal from downsampled observations. In the second algorithm, we present a formulation which incorporates a broad family of commonly used quasi-Newton methods in a single representation.

2.1 Limited-Memory Trust-Region Methods for Sparse Relaxation

The work described in this section is based on the paper by Adhikari, DeGuchy, Erway, Lockhart and Marcia [1]. In this work, we focus on the sparse recovery problem

$$\underset{\mathbf{f}\in\mathbb{R}^{\tilde{n}}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{A}\mathbf{f}-\mathbf{b}\|_2^2 + \tau\|\mathbf{f}\|_1, \tag{2.1}$$

where $\mathbf{A} \in \mathbb{R}^{\tilde{m}\times\tilde{n}}$, $\mathbf{f} \in \mathbb{R}^{\tilde{n}}$, $\mathbf{b} \in \mathbb{R}^{\tilde{m}}$, $\tilde{m} \ll \tilde{n}$, and $\tau > 0$ is a constant regularization parameter. Under this regime we operate under a gaussian noise assumption where (2.1), incorporates a data fidelity term that is regularized by an ℓ_1-norm regularization term [14]. By letting $\mathbf{f} = \mathbf{u} - \mathbf{v}$, where $\mathbf{u}, \mathbf{v} \geq 0$, we can write (2.1) as the constrained but

differentiable optimization problem,

$$\begin{array}{ll} \underset{\mathbf{u},\mathbf{v}\in\mathbb{R}^{\tilde{n}}}{\text{minimize}} & \frac{1}{2}\|\mathbf{A}(\mathbf{u}-\mathbf{v})-\mathbf{b}\|_2^2+\tau\mathbb{1}_{\tilde{n}}^T(\mathbf{u}+\mathbf{v}) \\ \text{subject to} & \mathbf{u},\mathbf{v}\geq 0, \end{array}$$

where $\mathbb{1}_{\tilde{n}}$ is the $\tilde{n}$-vector of ones [19]. We transform the constrained differentiable problem into an unconstrained optimization problem using the change of variables $\mathbf{u}_i = \log(1+e^{\tilde{\mathbf{u}}_i})$ and $\mathbf{v}_i = \log(1+e^{\tilde{\mathbf{v}}_i})$, where $\tilde{\mathbf{u}}_i, \tilde{\mathbf{v}}_i \in \mathbb{R}$ for $1 \leq i \leq \tilde{n}$. With these definitions, $\mathbf{u}$ and $\mathbf{v}$ are guaranteed to be non-negative. Thus, (2.1) is equivalent to the following minimization problem:

$$\min_{\tilde{\mathbf{u}},\tilde{\mathbf{v}}\in\mathbb{R}^{\tilde{n}}} \boldsymbol{\Phi}(\tilde{\mathbf{u}},\tilde{\mathbf{v}}) \triangleq \frac{1}{2}\sum_{i=1}^{\tilde{m}}\left[\left\{\sum_{j=1}^{\tilde{n}}\mathbf{A}_{i,j}\log\left(\frac{1+e^{\tilde{\mathbf{u}}_j}}{1+e^{\tilde{\mathbf{v}}_j}}\right)\right\}-\mathbf{b}_i\right]^2+\tau\sum_{j=1}^{\tilde{n}}\log\left((1+e^{\tilde{\mathbf{u}}_j})(1+e^{\tilde{\mathbf{v}}_j})\right). \quad (2.2)$$

Notice that the gradient of $\boldsymbol{\Phi}(\tilde{\mathbf{u}},\tilde{\mathbf{v}})$ can be written as follows: Letting $\tilde{w} \in \mathbb{R}^{\tilde{n}}$ with $\tilde{w}_i = \log(1+e^{\tilde{\mathbf{u}}_i}) - \log(1+e^{\tilde{\mathbf{v}}_i})$, then

$$\begin{aligned} \nabla_{\tilde{\mathbf{u}}_i}\boldsymbol{\Phi}(\tilde{\mathbf{u}},\tilde{\mathbf{v}}) &= \frac{e^{\tilde{\mathbf{u}}_i}}{1+e^{\tilde{\mathbf{u}}_i}}\Big(\left(\mathbf{A}^T(\ \ \mathbf{A}\tilde{w}-y)\right)_i+\tau\Big), \\ \nabla_{\tilde{v}_i}\boldsymbol{\Phi}(\tilde{u},\tilde{v}) &= \frac{e^{\tilde{v}_i}}{1+e^{\tilde{v}_i}}\Big(\left(\mathbf{A}^T(-\mathbf{A}\tilde{w}+y)\right)_i+\tau\Big), \end{aligned}$$

and

$$\nabla\boldsymbol{\Phi}(x) = \begin{bmatrix}\nabla_{\tilde{\mathbf{u}}}\boldsymbol{\Phi}(\tilde{\mathbf{u}},\tilde{\mathbf{v}}) \\ \nabla_{\tilde{\mathbf{v}}}\boldsymbol{\Phi}(\tilde{\mathbf{u}},\tilde{\mathbf{v}})\end{bmatrix}.$$

We propose solving (2.2) using a limited-memory quasi-Newton trust-region optimization approach, which we describe in the next section.

Related Work

There are various methods for solving (2.1) (see e.g., Eldar and Kutyniok [15] and all the references therein), many of which use a gradient descent-type approach. Our proposed approach is based on quasi-Newton methods, which have been previously shown to be effective for sparsity recovery problems [25, 38, 40]. For example, Becker and Fadili [3] use a zero-memory rank-one quasi-Newton approach for proximal splitting. Trust-region methods have also been implemented for sparse reconstruction [22, 35]. Our approach is novel in the transformation of the sparse recovery problem to a differentiable unconstrained minimization problem and in the use of eigenvalues for efficiently solving the trust-region subproblem.

2.1.1 Trust-Region Methods

In this section, we outline the use of a trust-region method to solve (2.2). We begin by combining the unknowns $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ into one vector of unknowns $x = [\tilde{\mathbf{u}}^T\ \tilde{\mathbf{v}}^T]^T \in \mathbb{R}^n$,

where $n = 2\tilde{n}$. With this substitution, Φ can be considered as a function of x. Trust-region methods to minimize $\Phi(x)$ define a sequence of iterates $\{x_k\}$ that are updated as follows: $x_{k+1} = x_k + p_k$, where p_k is defined as the *search direction*. Each iteration, a new search direction p_k is computed from solving the following quadratic subproblem with a two-norm constraint:

$$p_k = \arg\min_{p \in \mathbb{R}^n} q_k(p) \triangleq g_k^T p + \frac{1}{2} p^T B_k p \ \text{ s.t. } \ \|p\|_2 \leq \delta_k, \tag{2.3}$$

where $g_k \triangleq \nabla \Phi(x_k)$, B_k is an approximation to $\nabla^2 \Phi(x_k)$, and δ_k is a given positive constant. In large-scale optimization, solving (2.3) represents the bulk of the computational effort in trust-region methods.

Methods that solve the trust-region subproblem to high accuracy are often based on the optimality conditions for a global solution to the trust-region subproblem given in the following theorem [11, 20, 31]:

Theorem 1. *Let δ be a positive constant. A vector p^* is a global solution of the trust-region subproblem (2.3) if and only if $\|p^*\|_2 \leq \delta$ and there exists a unique $\sigma^* \geq 0$ such that $B + \sigma^* I$ is positive semidefinite and*

$$(B + \sigma^* I)p^* = -g \quad \textit{and} \quad \sigma^*(\delta - \|p^*\|_2) = 0. \tag{2.4}$$

Moreover, if $B + \sigma^* I$ is positive definite, then the global minimizer is unique.

2.1.2 Quasi-Newton Methods

In this section we show how to build an approximation B_k of $\nabla^2 \Phi(x)$ using limited-memory quasi-Newton matrices.

Given the continuously differentiable function Φ and a sequence of iterates $\{x_k\}$, traditional quasi-Newton matrices are genererated from a sequence of update pairs $\{(s_k, y_k)\}$ where $s_k \triangleq x_{k+1} - x_k$ and $y_k \triangleq \nabla \Phi(x_{k+1}) - \nabla \Phi(x_k)$. In particular, given an initial matrix B_0, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [28, 32] generates a sequence of matrices using the following recursion:

$$B_{k+1} \triangleq B_k - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T B_k + \frac{1}{y_k^T s_k} y_k y_k^T, \tag{2.5}$$

provided $y_k^T s_k \neq 0$. In practice, B_0 is often taken to be a nonzero constant multiple of the identity matrix, i.e., $B_0 = \gamma I$, for some $\gamma > 0$. Limited-memory BFGS (L-BFGS) methods store and use only the m most-recently computed pairs $\{(s_k, y_k)\}$, where $m \ll n$. Often m may be very small (for example, Byrd et al. [9] suggest $m \in [3, 7]$).

The BFGS update is the most widely-used rank-two update formula that (i) satisfies the *secant condition* $B_{k+1} s_k = y_k$, (ii) has hereditary symmetry, and (iii) generates a sequence of positive-definite $\{B_k\}$, provided that $y_i^T s_i > 0$ for $i = 0, \ldots k$.

The L-BFGS matrix B_{k+1} in (2.5) can be defined recursively as follows:

$$B_{k+1} = B_0 + \sum_{i=0}^{k} \left\{ -\frac{1}{s_i^T B_i s_i} B_i s_i s_i^T B_i + \frac{1}{y_i^T s_i} y_i y_i^T \right\}.$$

Then B_{k+1} is at most a rank-$2(k+1)$ perturbation to B_0, and thus, B_{k+1} can be written as

$$B_{k+1} = B_0 + \begin{bmatrix} \\ \Psi_k \\ \\ \end{bmatrix} \begin{bmatrix} M_k \end{bmatrix} \begin{bmatrix} \quad \Psi_k^T \quad \end{bmatrix}$$

for some $\Psi_k \in \mathbb{R}^{n \times 2(k+1)}$ and $M_k \in \mathbb{R}^{2(k+1) \times 2(k+1)}$. Byrd et al. [9] showed that Ψ_k and M_k are given by

$$\Psi_k = \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix} \text{ and } M_k = -\begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1},$$

where $S_k \triangleq [\, s_0 \;\; s_1 \;\; s_2 \;\; \cdots \;\; s_k \,] \in \mathbb{R}^{n \times (k+1)}$, and $Y_k \triangleq [\, y_0 \;\; y_1 \;\; y_2 \;\; \cdots \;\; y_k \,] \in \mathbb{R}^{n \times (k+1)}$, and L_k is the strictly lower triangular part and D_k is the diagonal part of the matrix $S_k^T Y_k \in \mathbb{R}^{(k+1) \times (k+1)}$, i.e., $S_k^T Y_k = L_k + D_k + U_k$, where U_k is a strictly upper triangular matrix.

2.1.3 Solving the Trust-Region Subproblem

In this section, we show how to solve (2.3) efficiently. First, we transform (2.3) into an equivalent expression. For simplicity, we drop the subscript k. Let $\Psi = QR$ be the "thin" QR factorization of Ψ, where $Q \in \mathbb{R}^{n \times 2(k+1)}$ has orthonormal columns and $R \in \mathbb{R}^{2(k+1) \times 2(k+1)}$ is upper triangular. Then

$$B_{k+1} = B_0 + \Psi M \Psi^T = \gamma I + QRMR^T Q^T.$$

Now let $V \widehat{\Lambda} V^T = RMR^T$ be the eigendecomposition of $RMR^T \in \mathbb{R}^{2(k+1) \times 2(k+1)}$, where $V \in \mathbb{R}^{2(k+1) \times 2(k+1)}$ is orthogonal and $\widehat{\Lambda}$ is diagonal with $\widehat{\Lambda} = \text{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_{2(k+1)})$. We assume that the eigenvalues $\hat{\lambda}_i$ are ordered in increasing values, i.e., $\hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_{2(k+1)}$. Since Q has orthonormal columns and V is orthogonal, then $P_\parallel \triangleq QV \in \mathbb{R}^{n \times 2(k+1)}$ also has orthonormal columns. Let $P_\perp$ be a matrix whose columns form an orthonormal basis for the orthogonal complement of the column space of $P_\parallel$. Then, $P \triangleq [\, P_\parallel \;\; P_\perp \,] \in \mathbb{R}^{n \times n}$ is such that $P^T P = P P^T = I$. Thus, the spectral decomposition of B is given by

$$B = P \Lambda P^T, \text{ where } \Lambda \triangleq \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} = \begin{bmatrix} \hat{\Lambda} + \gamma I & 0 \\ 0 & \gamma I \end{bmatrix}, \tag{2.6}$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, $\Lambda_1 = \text{diag}(\lambda_1, \ldots, \lambda_{2(k+1)}) \in \mathbb{R}^{2(k+1) \times 2(k+1)}$, and $\Lambda_2 = \gamma I_{n-2(k+1)}$. Since the $\hat{\lambda}_i$'s are ordered, then the eigenvalues in Λ are also ordered, i.e., $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{2(k+1)}$. The remaining eigenvalues, found on the diagonal of Λ_2, are equal to γ. Finally, since B is positive definite, then $0 < \lambda_i$ for all i.

Defining $v = P^T p$, the trust-region subproblem (2.3), can be written as

$$\begin{aligned} v^* \quad = \quad & \underset{v \in \mathbb{R}^n}{\arg\min} \quad q_k(v) \triangleq \tilde{g}^T v + \frac{1}{2} v^T \Lambda v \\ & \text{subject to} \quad \|v\|_2 \leq \delta, \end{aligned} \tag{2.7}$$

where $\tilde{g} = P^T g$. From the optimality conditions in Theorem 1, the solution, v^*, to (2.7) must satisfy the following equations:

$$(\Lambda + \sigma^* I) v^* = -\tilde{g} \tag{2.8}$$

$$\sigma^* (\|v^*\|_2 - \delta) = 0 \tag{2.9}$$

$$\sigma^* \geq 0 \tag{2.10}$$

$$\|v^*\|_2 \leq \delta, \tag{2.11}$$

for some scalar σ^*. Note that the usual requirement that $\sigma^* + \lambda_i \geq 0$ for all i is not necessary here since $\lambda_i > 0$ for all i (i.e., B is positive definite). Note further that (2.9) implies that if $\sigma^* > 0$, the solution must lie on the boundary, i.e., $\|v^*\|_2 = \delta$. In this case, the optimal σ^* can be obtained by finding solving the so-called *secular equation*:

$$\phi(\sigma) = \frac{1}{\|v(\sigma)\|_2} - \frac{1}{\delta} = 0, \tag{2.12}$$

where $\|v(\sigma)\|_2 = \| - (\Lambda + \sigma I)^{-1} \tilde{g}\|_2$. Since $\lambda_i + \sigma > 0$ for any $\sigma \geq 0$, $v(\sigma)$ is well-defined. In particular, if we let

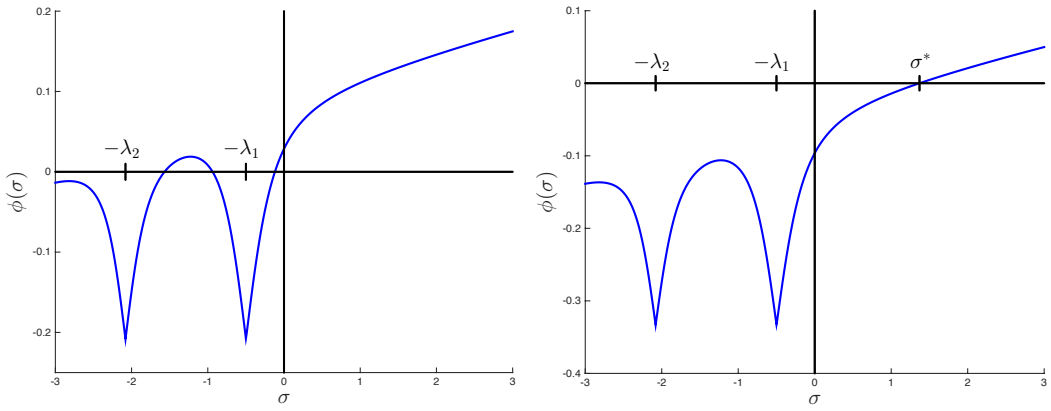
$$\tilde{g} = \begin{bmatrix} P_{\|}^T \\ P_{\perp}^T \end{bmatrix} g = \begin{bmatrix} P_{\|}^T g \\ P_{\perp}^T g \end{bmatrix} = \begin{bmatrix} g_{\|} \\ g_{\perp} \end{bmatrix},$$

then

$$\|v(\sigma)\|_2^2 = \left\{ \sum_{i=1}^{2(k+1)} \frac{(g_{\|})_i^2}{(\lambda_i - \sigma)^2} \right\} + \frac{\|g_{\perp}\|_2^2}{(\gamma - \sigma)^2}. \tag{2.13}$$

We note that $\phi(\sigma) \geq 0$ means $v(\sigma)$ is feasible, i.e., $\|v(\sigma)\|_2 \leq \delta$. Specifically, the unconstrained minimizer $v(0) = -\Lambda^{-1}\tilde{g}$ is feasible if and only if $\phi(0) \geq 0$ (see Fig. 1(a)). If $v(0)$ is not feasible, then $\phi(0) < 0$ and there exists $\sigma^* > 0$ such that $v(\sigma^*) = -(\Lambda + \sigma^* I)^{-1}\tilde{g}$ with $\phi(\sigma^*) = 0$ (see Fig. 1(b)). Since B is positive definite, the function $\phi(\sigma)$ is strictly increasing and concave down for $\sigma \geq 0$, making it a good candidate for Newton's method. In fact, it can be shown that Newton's method will converge monontonically and quadratically to σ^* with initial guess $\sigma^{(0)} = 0$ [11].

The method to obtain σ^* is significantly different that the one used by Burke et al. [8] in that we explicitly use the eigendecomposition within Newton's method to compute the optimal σ^*. That is, we differentiate the reciprocal of $\|v(\sigma)\|$ in (2.13) to compute the derivative of $\phi(\sigma)$ in (2.12), obtaining a Newton update that is expressed only in terms of $g_{\|}$, $g_{\perp}$, and the eigenvalues of B. In contrast to the method by Burke et al. [8] (specifically Alg. 2 in their paper), this approach eliminates the need for matrix solves for each Newton iteration.



(a) (b)

Figure 2.1: Plot of the secular function $\phi(\sigma)$ given in (2.12). (a) The case when $\phi(0) \geq 0$, which implies that the unconstrained minimizer of (2.7) is feasible. (b) When $\phi(0) < 0$, there exists $\sigma^* > 0$ such that $\phi(\sigma^*) = 0$, i.e., $v^* = -(\Lambda + \sigma^* I)^{-1}\tilde{g}$ is well-defined and is feasible.

Given σ^* and v^*, the optimal p^* is obtained as follows. Letting $\tau^* = \gamma + \sigma^*$, the solution to the first optimality condition, $(B + \sigma^* I)p^* = -g$, is given by

$$\begin{aligned} p^* &= -(B + \sigma^* I)g \\ &= -(\gamma I + \Psi M \Psi^T + \sigma^* I)^{-1} g \\ &= -\frac{1}{\tau^*}\left[I - \Psi(\tau^* M^{-1} + \Psi^T \Psi)^{-1}\Psi^T\right] g, \end{aligned} \tag{2.14}$$

using the Sherman-Morrison-Woodbury formula. Algorithm 1 details the proposed approach for solving the trust-region subproblem.

Algorithm 1: L-BFGS Trust-Region Subproblem Solver

Compute R from the "thin" QR factorization of Ψ;
Compute the spectral decomposition
$RMR^T = V\hat{\Lambda}V^T$ with $\hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_{2(k+1)}$;
Let $\Lambda_1 = \hat{\Lambda} + \gamma I$;
Define $P_\parallel = \Psi R^{-1} V$ and $g_\parallel = P_\parallel^T g$;
Compute $\|P_\perp^T g\|_2 = \sqrt{\|g\|_2^2 - \|g_\parallel\|_2^2}$;
if $\phi(0) \geq 0$ **then**
    $\sigma^* = 0$ and compute p^* from (5.2.2) with $\tau^* = \gamma$;
else
    Use Newton's method to find σ^*;
    Compute p^* from (5.2.2) with $\tau^* = \gamma + \sigma^*$;
end if

The method described here guarantees that the trust-region subpoblem is solved to high accuracy. Other L-BFGS trust-region methods that solve to high accuracy

include the Moré-Sorensen Sequential Method (MSS) [16], which uses a shifted L-BFGS approach, and the Limited-Memory Trust-Region Method [6], which uses a "shape-changing" norm in (2.3).

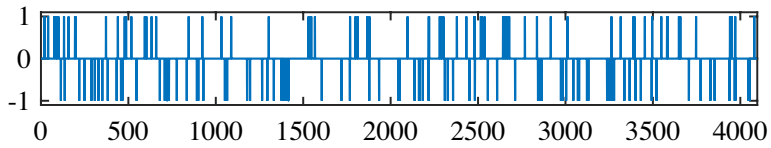
Convergence

Global convergence of Algorithm 2 can be proven by modifying the techniques found in [8, 34] that require that the following assumptions are satisfied: [A.1] There are constants l and u such that $l \leq \|B_k\| \leq u$ for all k. [A.2] $\nabla \boldsymbol{\Phi}$ is Lipschitz continuous. For Assumption A.1, since B_k is symmetric and positive definite, $\|B_k\|_2 = \lambda_{\max}$. Because we are able to explicitly compute the eigenvalues of B_k in (2.6), we can satisfy Assumption A.1 by accepting an update pair (s_k, y_k) only if $l \leq \lambda_{\max} \leq u$. For Assumption A.2, the gradient of the function $\boldsymbol{\Phi}(x)$ is continuously differentiable, and therefore, $\nabla \boldsymbol{\Phi}$ must be Lipschitz continuous.

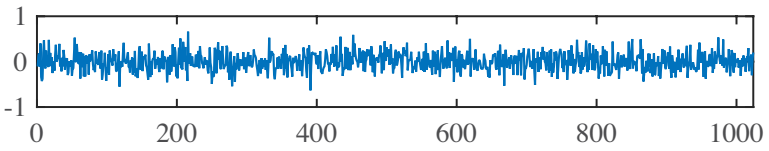
With these assumptions satisfied and noting that $\boldsymbol{\Phi}(x_k) \geq 0$ for all x_k (since each term in (2.2) is nonnegative), then by [8, Theorem 5.4] the sequence of iterates generated by Algorithm 2 converges to a critical point of $\boldsymbol{\Phi}$.

Algorithm 2: Trust-Spa: Limited-Memory BFGS Trust-Region Method for Sparse Relaxation

Define parameters: m, $0 < \tau_1 < 0.5$, $0 < \varepsilon$;
Initialize $x_0 \in \mathbb{R}^n$ and compute $g_0 = \nabla \boldsymbol{\Phi}(x_0)$;
Let $k = 0$;
while not converged do
  if $\|g_k\|_2 \leq \varepsilon$ **then done**
  Use Algorithm 1 to find p_k that solves (2.3);
  Compute $\rho_k = (\boldsymbol{\Phi}(x_k + p_k) - \boldsymbol{\Phi}(x_k))/q_k(p_k)$;
  Compute g_{k+1} and update B_{k+1};
  if $\rho_k \geq \tau_1$ **then**
    $x_{k+1} = x_k + p_k$;
  else
    $x_{k+1} = x_k$;
  end if
  Compute trust-region radius δ_{k+1};
  $k \leftarrow k + 1$;
end while



(a) Truth f ($\tilde{n}$ = 4096, number of nonzeros = 160)



(b) Measurements y ($\tilde{m}$ = 1024, noise level = 5%)

Figure 2.2: Experimental setup: (a) True signal **f** of size 4,096 with 160 spikes ($\pm$1), (b) Compressive measurements **y** with 5% Gaussian noise.

2.1.4 Numerical Experiments

We evaluate the performance of the proposed method (Trust-Spa) by solving 1D and 2D signal reconstruction problems. In particular, we compare the results with the widely-used GPSR method [19] and the more recent method, YALL1 [36]. All three methods were initialized using the same starting point, i.e., zero, and terminate if the relative objective values do not significantly change, i.e, $|\Phi(x^{k+1}) - \Phi(x^k)|/|\Phi(x^k)| \leq 10^{-8}$. The regularization parameter τ in (2.1) is optimized independently for each algorithm to minimize the mean-squared error (MSE = $\frac{1}{n}\|\widehat{\mathbf{f}} - \mathbf{f}\|_2^2$, where $\widehat{\mathbf{f}}$ is an estimate of **f**).

1D Signal Recovery

In this experiment, the true signal **f** is of size 4,096 with 160 randomly assigned nonzeros with amplitude $\pm$1 (see Fig. 2.2(a)). We obtain compressive measurements **y** of size 1,024 (see Fig. 2.2(b)) by projecting the true signal using a randomly generated system matrix (**A**) from the standard normal distribution with orthonormalized rows. In particular, the measurements are corrupted by 5% of Gaussian noise.

We use compressive measurements **b** of size 1,024 (with 5% noise) to recover the true signal **f** of size 4,096 made up of 160 nonzeros with values $\pm$1. On average over 10 trials, the proposed Trust-Spa method (average MSE = 9.827e-5) is shown to outperform GPSR (average MSE = 1.758e-4) and YALL1 (average MSE = 1.753e-4) in comparable computation time. Note that the Trust-Spa has fewer reconstruction artifacts (see Fig. 2.3). Because of the variable transformations used by Trust-Spa, the algorithm terminates with no zero components in its solution (even though the true signal has only 160 nonzero components); however, only a relatively few components of the Trust-Spa reconstruction have significant amplitudes. For example, in on e particular one, only 579 components are greater than 10^{-6} in absolute value. This has the effect of rendering most spurious solutions less visible.

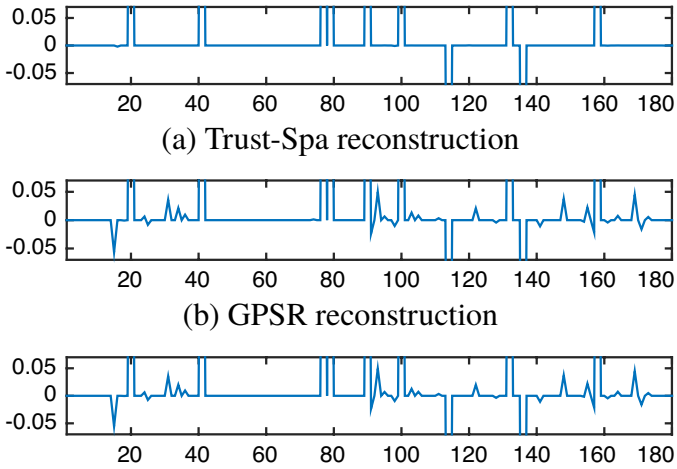


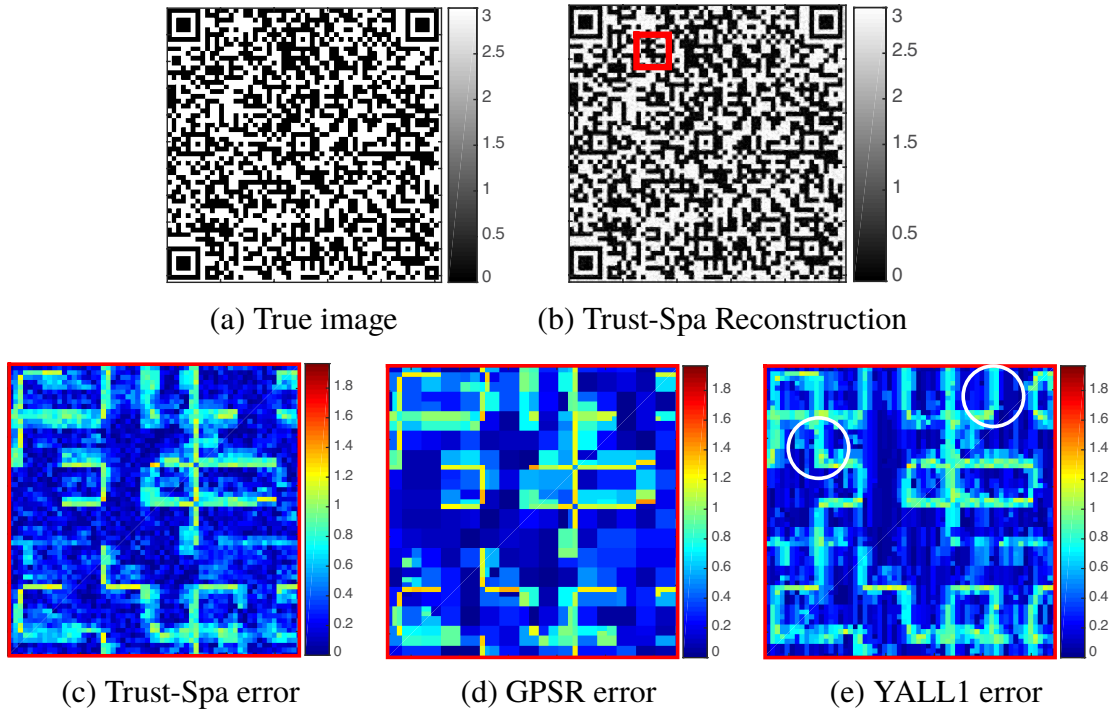
Figure 2.3: A Zoomed region of all reconstructions. Note the presence of more artifacts in the GPSR reconstruction (b) and the YALL1 recontrsuction (c) in comparison to the reconstruction from our proposed method Trust-Spa (a).

2D Signal Recovery

Here, we wish to deblur a Quick Response (QR) code of size 512×512 (see Fig. 2.4(a)) from a 3% zero-mean Gaussian noise corrupted blurry image. GPSR obtained MSE 5.3e-1 (20 sec) and YALL1 obtained MSE 4.03e-1 (40 sec). In contrast, the proposed Trust-Spa method took only 16 seconds to converge with MSE 3.9e-1. In the case of GPSR, there are very high amplitude artifacts around edges of the reconstruction (compare zoomed-in log-error plots Figs. 2.4(c) and 2.4(d) for orange areas). Even though YALL1 gives competitive results in MSE, it does not recover edges as well as Trust-Spa (compare circled areas in Fig. 2.4(e) with Figs. 2.4(c) and 2.4(d)).

2.1.5 Conclusion

In this work, we proposed an approach for solving the ℓ_2-ℓ_1 minimization problem that arises in compressed sensing and sparse recovery problems. Unlike gradient projection-type methods, our approach uses gradients from previous iterations to approximate a more accurate Hessian. Numerical experiments show that our proposed approach mitigates spurious solutions more effectively with a lower average MSE in a smaller amount of time.



(a) True image (b) Trust-Spa Reconstruction

(c) Trust-Spa error (d) GPSR error (e) YALL1 error

Figure 2.4: (a) True QR code image, (b) Trust-Spa reconstruction, (c) Log-error plot of the Trust-Spa reconstruction, (d) Log-error plot of the GPSR reconstruction, and (e) Log-error plot of the YALL1 reconstruction. Note the log-error of GPSR has higher amplitude, and the YALL1 reconstruction has more edge artifacts than the Trust-Spa reconstruction.

2.2 Compact Representation of the Full Broyden Class of Quasi-Newton Updates

In section 2.1 we relied on approximations of second derivative information to solve the sparse signal recovery problem. In particular, we used the compact representation of the popular L-BFGS (2.5) update which belongs to a broader class of approximations known as the Broyden class. This work, based on the paper by DeGuchy, Erway and Marcia [12], presents an algorithm which extends the notion of a compact representation to the full Broyden class of updates.

Quasi-Newton methods for minimizing a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ generate a sequence of iterates $\{x_k\}$ such that f is strictly decreasing at each iterate. Crucially, at each iteration a quasi-Newton matrix is used to approximate $\nabla^2 f(x_k)$ that is assumed to be either too computationally expensive to compute or unavailable. The approximation to the Hessian is updated each iteration using the most recently-computed iterate x_{k+1} by defining a new *quasi-Newton pair* (s_k, y_k) given by

$$s_k \triangleq x_{k+1} - x_k \quad \text{and} \quad y_k \triangleq \nabla f(x_{k+1}) - \nabla f(x_k).$$

The quasi-Newton Broyden family of updates is given by

$$B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T B_k + \frac{1}{y_k^T s_k} y_k y_k^T + \phi_k (s_k^T B_k s_k) w_k w_k^T, \tag{2.15}$$

where $\phi_k \in \Re$ and

$$w_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}.$$

For $\phi_k \in [0, 1]$, B_{k+1} is said to be in the *restricted* or convex Broyden class of updates. Setting $\phi = 0$ gives the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, arguably the most widely-used symmetric positive-definite update and a member of the restricted Broyden class. For $\phi \notin [0, 1]$, the sequence of quasi-Newton matrices generated by this update is not guaranteed to be positive definite. The most well-known update not in the restricted Broyden class is the symmetric rank-one (SR1) update, which is obtained by setting $\phi_k = (s_k^T y_k)/(s_k^T y_k - s_k^T B_k s_k)$.

There has been interest in the entire Broyden class of updates, and in particular, in negative values of ϕ. Research has shown that negative values of ϕ are desirable [10] and under some conditions, quasi-Newton methods based on negative values of ϕ exhibit superlinear convergence rates [10, 39]. There has also been empirical evidence that $\phi < 0$ may lead to more efficient algorithms than BFGS [27, 39]. More recently, the entire Broyden class of quasi-Newton methods has been generalized to solve minimization problems over Riemannian manifolds [23].

Compact representations of matrices from the Broyden class of updates were first described by Byrd et al [9] as matrix decompositions of the form

$$B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T,$$

where $\Psi_k \in \Re^{n\times l}$, $M_k \in \Re^{l\times l}$, and B_0 is the initial matrix. The size of l depends on the rank of the update; in the case of a rank-two update, $l = 2(k+1)$, and in the case of a rank-one update, $l = k+1$. Compact representations are known for members of the restricted Broyden class [9, 18]; however, outside the restricted Broyden class, the only known compact formulation is for the SR1 update (found in [9]). In this work, we present the compact representation for the full Broyden class of quasi-Newton matrices, allowing ϕ to be negative and to change each iteration. We also demonstrate how to efficiently solve linear systems with any member of the Broyden class using the compact representation of its inverse. This work can be viewed as an extension of the results found in [17, 18], which presented the compact representation for members of the restricted Broyden class and their inverses, as well as a practical method for solving linear systems involving only restricted Broyden class matrices (i.e., $\phi \in [0, 1]$).

One important application of the compact representation is the ability to efficiently compute the eigenvalues and a partial eigenbasis when the number of stored pairs is small [18], which is the case in large-scale optimization with so-called *limited-memory* quasi-Newton updates. In this setting, only the most recently-computed M quasi-Newton pairs $\{(s_k, y_k)\}$, $k = 0, 1, \ldots, M-1$, are stored and used to update B_{k+1} using the recursive application of (2.15). Typically, in large-scale applications $M \leq 10$ regardless of n, i.e., $M \ll n$ (see, e.g., [9]). With the eigenvalues it is now possible to compute condition numbers, compute singular values, and perform sensitivity analysis.

This work is organized in seven sections. In the second section, we review the compact formulation for the restricted Broyden class of updates ($\phi \in [0, 1]$) as well as overview the efficient computation of their eigenvalues. The main result of this work is in Section 3 where the compact representation is given for the entire Broyden class of updates that allows for ϕ to change each update. In this section, we also present a practical iterative method to compute the compact representation. In Section 4, we show how to perform linear solves with any member of the Broyden class using the compact representation of their inverse. Numerical experiments are reported in Section 5. Finally, Section 6 contains concluding remarks, and Section 7 includes acknowledgements for this work.

Notation and Assumptions

Throughout this work, we make use of the following matrices:

$$S_k \triangleq \begin{pmatrix} s_0 & s_1 & s_2 & \cdots & s_k \end{pmatrix} \in \Re^{n\times(k+1)}, \tag{2.16}$$

$$Y_k \triangleq \begin{pmatrix} y_0 & y_1 & y_2 & \cdots & y_k \end{pmatrix} \in \Re^{n\times(k+1)}. \tag{2.17}$$

Furthermore, we make use of the following decomposition of $S_k^T Y_k \in \Re^{(k+1)\times(k+1)}$:

$$S_k^T Y_k = L_k + D_k + R_k, \tag{2.18}$$

where L_k is strictly lower triangular, D_k is diagonal, and R_k is strictly upper triangular. We assume that the matrix B_k is nonsingular for each k. Finally, throughout the manuscript, I_j denotes the $j \times j$ identity matrix.

2.2.1 Compact Representation for the Restricted Broyden Class

The most widely-used member of the restricted Broyden class is the BFGS update (i.e., $\phi = 0$). In this case, Ψ_k and M_k are given in [9]:

$$\Psi_k \triangleq \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix} \quad \text{and} \quad M_k \triangleq -\begin{pmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{pmatrix}^{-1}, \tag{2.19}$$

where S_k and Y_k are defined in (2.16). The compact representation for any matrix in the restricted Broyden class (i.e, $\phi \in [0, 1]$) is given in [18]; in particular, for any matrix in the restricted Broyden class,

$$\Psi_k \triangleq \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix} \quad \text{and} \quad M_k = \begin{pmatrix} -S_k^T B_0 S_k + \phi \Lambda_k & -L_k + \phi \Lambda_k \\ -L_k^T + \phi \Lambda_k & D_k + \phi \Lambda_k \end{pmatrix}^{-1},$$

where L_k and D_k are given in (2.18) and $\Lambda_k \in \mathbb{R}^{(k+1)\times(k+1)}$ is the diagonal matrix $\Lambda_k = \text{diag}(\lambda_i)$, $(0 \leq i \leq k)$, given by

$$\lambda_i \triangleq \left(-\frac{1-\phi}{s_i^T B_i s_i} - \frac{\phi}{s_i^T y_i} \right)^{-1}. \tag{2.20}$$

To our knowledge, the only compact representation known for a member of the Broyden class of updates outside the restricted class is for the SR1 update (i.e., $\phi_k = (s_k^T y_k)/(s_k^T y_k - s_k^T B_k s_k)$.) As with the BFGS case, its compact representation is also given in [9]:

$$\Psi_k = Y_k - B_0 S_k \quad \text{and} \quad M_k = (D_k + L_k + L_k^T - S_k^T B_0 S_k)^{-1}.$$

Notice that Ψ_k in the compact representation for SR1 matrices is half the size of that of Ψ_k for the rank-two updates.

Applications of the Compact Representation

In this section, we briefly review how the eigenvalues of any quasi-Newton matrix that exhibits a compact representation can be efficiently computed. The first method to compute eigenvalues of limited-memory quasi-Newton matrices was proposed by Lu [29]. This method makes use of the singular value decomposition and an eigendecomposition of small matrices. An alternative approach, first described by Burdakov et al. [7], uses the QR factorization in lieu of the singular value decomposition. An overview of the method found in [7] follows below. For this section, we assume k is small, as in the case of limited-memory quasi-Newton matrices; moreover, we assume $\Psi_k \in \mathbb{R}^{n\times l}$ is full rank, where l is either $l = 2(k+1)$ or $l = k+1$. Finally, we assume $B_0 = \gamma I$, where $\gamma \in \mathbb{R}$.

Let QR be the "thin" QR decomposition of Ψ_k, where $Q \in \mathbb{R}^{n\times l}$ has orthonormal columns and $R \in \mathbb{R}^{l\times l}$ is upper triangular (see, e.g., [21]). Then,

$$B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T = B_0 + QRM_k R^T Q^T.$$

The matrix RM_kR^T is a real symmetric $l \times l$ matrix, whose spectral decomposition can be explicitly computed since l is small. Letting $V\Delta V^T$ be its spectral decomposition gives that

$$B_{k+1} = B_0 + QV\Delta V^T Q^T = \gamma I + QV\Delta V^T Q^T = QV(\gamma I + \hat{\Delta})V^T Q^T, \tag{2.21}$$

where $\hat{\Delta}$ is a diagonal matrix whose leading $l \times l$ block is Δ while the rest of the matrix is zeros. Thus, the spectral decomposition of B_{k+1} is given by (2.21). (Note that in practice, the matrices Q and V in (2.21) are not stored.) Note that the matrix B_{k+1} has an eigenvalue of γ with multiplicity $n - l$ and l eigenvalues given by $\gamma + \Delta_{i,i}$, where $1 \leq i \leq l$. It also turns out that it is also possible to efficiently compute the eigenvectors associated with the nontrivial eigenvalues and only one eigenvector associated with the trivial eigenvalue γ. (For more details, see [7, 18].)

Generally speaking, computing the eigenvalues of B_{k+1} directly is an $O(n^3)$ process. In contrast, the above decomposition requires the QR factorization of Ψ_k and the eigendecomposition of RM_kR^T, requiring $O(nl^2)$ flops and $O(l^3)$ flops, respectively. Since $l \ll n$, the proposed method's runtimes should increase only linearly with n. (For some details regarding updating the (full) QR factorization after a new quasi-Newton pair is computed, see [18].) This efficient computation of eigenvalues and a partial eigenbasis appears in new methods for large-scale optimization [1, 2, 4, 5, 7].

The compact representation is also useful for solving linear systems with quasi-Newton matrices. In [8], Burke et al. use the compact formulation of a BFGS matrix to solve a linear system involving a diagonally-shifted BFGS matrix. In [17], the compact representation for the inverse of any member in the restricted Broyden class is given as well as a practical method to solve linear systems involving these matrices using this representation.

2.2.2 Compact Representation for any Member of the Broyden Class

The main result for this section is a theorem giving the compact representation for any member of the Broyden class. The representation allows ϕ to change each iteration and to be negative. In this section, we also present a practical algorithm for computing the compact representation.

We begin by observing that B_{k+1} in (2.15) can be written as

$$B_{k+1} = B_k + \begin{pmatrix} B_k s_k & y_k \end{pmatrix} O_k \begin{pmatrix} (B_k s_k)^T \\ y_k^T \end{pmatrix}, \tag{2.22}$$

where

$$O_k = \begin{pmatrix} -\dfrac{(1-\phi_k)}{s_k^T B_k s_k} & -\dfrac{\phi_k}{y_k^T s_k} \\ -\dfrac{\phi_k}{y_k^T s_k} & \left(1 + \phi_k \dfrac{s_k^T B_k s_k}{y_k^T s_k}\right) \dfrac{1}{y_k^T s_k} \end{pmatrix}. \tag{2.23}$$

We now state two lemmas about O_k; specifically, we provide the condition for O_k when is singular as well as its inverse when it is nonsingular.

Lemma 1. *The* 2×2 *matrix* O_k *is singular if and only if* $\phi_k = (s_k^T y_k)/(s_k^T y_k - s_k^T B_k s_k)$.

Proof. The determinant of O_k is given by

$$\begin{aligned} \det(O_k) &= -\frac{(1-\phi_k)}{s_k^T B_k s_k}\left(1 + \phi_k \frac{s_k^T B_k s_k}{y_k^T s_k}\right)\frac{1}{y_k^T s_k} - \frac{\phi_k^2}{\left(y_k^T s_k\right)^2} \\ &= \frac{1}{y_k^T s_k}\left(-\frac{(1-\phi_k)}{s_k^T B_k s_k} - \frac{\phi_k}{y_k^T s_k}\right). \end{aligned}$$

Thus, O_k is singular if and only if

$$-\frac{(1-\phi_k)}{s_k^T B_k s_k} - \frac{\phi_k}{y_k^T s_k} = 0; \tag{2.24}$$

in other words, $\phi_k = (y_k^T s_k)/(y_k^T s_k - s_k^T B_k s_k)$. □

Lemma 1 states that O_k is singular if and only if the SR1 update is used. Special care will given to the SR1 case, since unlike other members of the Broyden class, this is a rank-one update. For the duration of this manuscript, we let $\phi_k^{SR1} \triangleq (y_k^T s_k)/(y_k^T s_k - s_k^T B_k s_k)$. For $\phi_k \neq \phi_k^{SR1}$, O_k is invertible and its inverse is given in Lemma 2. This result can be derived by using the formula for the inverse of a 2×2 matrix.

Lemma 2. *If* O_k *is invertible, then*

$$O_k^{-1} = \begin{pmatrix} -s_k^T B_k s_k + \dfrac{\phi_k}{\alpha_k + \beta_k} & \dfrac{\phi_k}{\alpha_k + \beta_k} \\ \dfrac{\phi_k}{\alpha_k + \beta_k} & y_k^T s_k + \dfrac{\phi_k}{\alpha_k + \beta_k} \end{pmatrix},$$

where $\alpha_k = -(1-\phi_k)/(s_k^T B_k s_k)$ and $\beta_k = -\phi_k/(y_k^T s_k)$.

We now state the main theorem of this manuscript that presents the compact representation for any member of the Broyden class, while allowing the parameter ϕ to vary at each iteration. After proving this theorem, we discuss several aspects of this compact representation as well as the key differences between the compact representation for the Broyden class of matrices (Theorem 1) and the compact representation of the *restricted* Broyden class reviewed in Section 2.

To present this theorem, we make use the following notation. Let $\Pi_k \in \mathbb{R}^{2(k+1)\times 2(k+1)}$ be the permutation matrix

$$\Pi_k = \begin{pmatrix} I_k & 0 & 0 & 0 \\ 0 & 0 & I_k & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.25}$$

with $\Pi_0 \triangleq I_2$. Additionally, let Ξ_k be defined recursively as

$$\Xi_k = \begin{pmatrix} \Pi_{k-1}^T \Xi_{k-1} & 0 \\ 0 & E_k \end{pmatrix}, \quad \text{where} \quad E_k = \begin{cases} (-1 \;\; 1)^T & \text{if } \phi_k = \phi_k^{SR1} \\ I_2 & \text{otherwise,} \end{cases} \tag{2.26}$$

with $\Xi_0 \triangleq E_0$. Finally, define $\Gamma_k \in \mathbb{R}^{(k+1)\times(k+1)}$ to be a diagonal matrix such that

$$\Gamma_k = \underset{0\le j\le k}{\operatorname{diag}}(\gamma_j), \qquad \text{where } \gamma_j = \begin{cases} \phi_j \left(-\frac{1-\phi_j}{s_j^T B_j s_j} - \frac{\phi_j}{s_j^T y_j} \right)^{-1} & \text{if } \phi_j \neq \phi_j^{SR1} \\ 0 & \text{otherwise.} \end{cases} \tag{2.27}$$

With these definitions, we state our main results.

Theorem 1. *Let $\Psi_k = (B_0 S_k \;\; Y_k) \in \Re^{n\times 2(k+1)}$. If B_{k+1} is a member of the Broyden class of updates, then*

$$B_{k+1} = B_0 + \widehat{\Psi}_k \widehat{M}_k \widehat{\Psi}_k^T, \tag{2.28}$$

where

$$\widehat{M}_k = \left(\Xi_k^T \Pi_k \begin{pmatrix} -S_k^T B_0 S_k + \Gamma_k & -L_k + \Gamma_k \\ -L_k^T + \Gamma_k & D_k + \Gamma_k \end{pmatrix} \Pi_k^T \Xi_k \right)^{-1}, \tag{2.29}$$

L_k and D_k are defined in (2.18), and

$$\widehat{\Psi}_k = \Psi_k \Pi_k^T \Xi_k. \tag{2.30}$$

Proof. This proof is by induction on k. For the base case ($k = 0$), $D_0 = y_0^T s_0$ with $L_0 = R_0 = 0$, and Γ_0 is the scalar γ_0. Thus, $\widehat{M}_0$ defined in (2.29) reduces to

$$\widehat{M}_0 = \left(\Xi_0^T \Pi_0 \begin{pmatrix} -s_0^T B_0 s_0 + \gamma_0 & \gamma_0 \\ \gamma_0 & d_0 + \gamma_0 \end{pmatrix} \Pi_0^T \Xi_0 \right)^{-1}. \tag{2.31}$$

By (2.22), B_1 is given by $B_1 = B_0 + \Psi_0 M_0 \Psi_0^T$ where $\Psi_0 = (B_0 s_0 \;\; y_0)$ and

$$M_0 \triangleq \begin{pmatrix} -\frac{(1-\phi_0)}{s_0^T B_0 s_0} & -\frac{\phi_0}{y_0^T s_0} \\ -\frac{\phi_0}{y_0^T s_0} & \left(1 + \phi_0 \frac{s_0^T B_0 s_0}{y_0^T s_0} \right) \frac{1}{y_0^T s_0} \end{pmatrix}. \tag{2.32}$$

It remains to show that $M_0 = \Pi_0^T \Xi_0 \widehat{M}_0 \Xi_0^T \Pi_0$. Since the initial permutation matrix is defined as $\Pi_0 = I_2$, we only need to show $M_0 = \Xi_0 \widehat{M}_0 \Xi_0^T$. For simplicity, M_0 can be written as

$$M_0 = \begin{pmatrix} \alpha_0 & \beta_0 \\ \beta_0 & \delta_0 \end{pmatrix}, \tag{2.33}$$

where

$$\alpha_0 = -\frac{(1-\phi_0)}{s_0^T B_0 s_0}, \quad \beta_0 = -\frac{\phi_0}{y_0^T s_0}, \quad \text{and} \quad \delta_0 = \left(1 + \phi_0 \frac{s_0^T B_0 s_0}{y_0^T s_0} \right) \frac{1}{y_0^T s_0}. \tag{2.34}$$

From Lemma 1, M_0 is nonsingular if and only if $\phi_0 \neq \phi_0^{SR1}$. Thus, we consider the following two cases separately: (a) $\phi_0 = \phi_0^{SR1}$ and (b) $\phi_0 \neq \phi_0^{SR1}$.

Case (a): If $\phi_0 = \phi_0^{SR1}$, then $\Xi_0 = E_0 = (-1 \;\; 1)^T$ by (2.26). By (2.24), $\alpha_0 + \beta_0 = 0$, and thus, M_0 can be simplified as

$$M_0 = \begin{pmatrix} -\beta_0 & \beta_0 \\ \beta_0 & -\beta_0 \end{pmatrix} = (-\beta_0) \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = -\beta_0 \Xi_0 \Xi_0^T.$$

Finally, since $\phi_0 = (y_0^T s_0)/(s_0^T y_0 - s_0^T B_0 s_0)$ and $\beta_0 = -\phi_0 / y_0^T s_0$, then

$$\widehat{M}_0 = \left(\Xi_0^T \begin{pmatrix} -s_0^T B_0 s_0 + \gamma_0 & \gamma_0 \\ \gamma_0 & s_0^T y_0 + \gamma_0 \end{pmatrix} \Xi_0 \right)^{-1} = \frac{1}{s_0^T y_0 - s_0^T B_0 s_0} = -\beta_0,$$

and thus, $M_0 = \Xi_0 \widehat{M}_0 \Xi_0^T$, as desired.

Case (b): If $\phi_0 \neq \phi_0^{SR1}$, then M_0 is nonsingular and $\Xi_0 = I_2$. Thus, it remains to show $M_0 = \widehat{M}_0$. By Lemma 1, $\alpha_0 + \beta_0 \neq 0$, making $\gamma_0 = \phi_0/(\alpha_0 + \beta_0)$ well defined. By Lemma 2, the inverse of M_0 is given by

$$M_0^{-1} = \begin{pmatrix} -s_0^T B_0 s_0 + \gamma_0 & \gamma_0 \\ \gamma_0 & s_0^T y_0 + \gamma_0 \end{pmatrix} = \widehat{M}_0^{-1}. \tag{2.35}$$

Note that the last equality in (2.35) follows since $\Pi_0 = I_2$.

For the induction step, assume

$$B_m = B_0 + \widehat{\Psi}_{m-1} \widehat{M}_{m-1} \widehat{\Psi}_{m-1}^T, \tag{2.36}$$

where $\widehat{M}_{m-1} = \left(\Xi_{m-1}^T \Pi_{m-1} \Omega_{m-1} \Pi_{m-1}^T \Xi_{m-1} \right)^{-1}$ and

$$\Omega_{m-1} = \begin{pmatrix} -S_{m-1}^T B_0 S_{m-1} + \Gamma_{m-1} & -L_{m-1} + \Gamma_{m-1} \\ -L_{m-1}^T + \Gamma_{m-1} & D_{m-1} + \Gamma_{m-1} \end{pmatrix}. \tag{2.37}$$

From (2.22), we have

$$B_{m+1} = B_0 + \widehat{\Psi}_{m-1} \widehat{M}_{m-1} \widehat{\Psi}_{m-1}^T + (B_m s_m \;\; y_m) \begin{pmatrix} \alpha_m & \beta_m \\ \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} (B_m s_m)^T \\ y_m^T \end{pmatrix}, \tag{2.38}$$

where

$$\alpha_m = -\frac{1 - \phi_m}{s_m^T B_m s_m}, \quad \beta_m = -\frac{\phi_m}{y_m^T s_m}, \quad \text{and} \quad \delta_m = \left(1 + \phi_m \frac{s_m^T B_m s_m}{y_m^T s_m} \right) \frac{1}{y_m^T s_m}.$$

Multiplying (2.36) by s_m on the right, we obtain

$$B_m s_m = B_0 s_m + \widehat{\Psi}_{m-1} \widehat{M}_{m-1} \widehat{\Psi}_{m-1}^T s_m. \tag{2.39}$$

Then, substituting this into (2.38) yields

$$\begin{aligned} B_{m+1} \;\; = \;\; & B_0 \; + \; \widehat{\Psi}_{m-1} \widehat{M}_{m-1} \widehat{\Psi}_{m-1}^T \; + \\ & \left(B_0 s_m + \widehat{\Psi}_{m-1} p_m \;\; y_m \right) \begin{pmatrix} \alpha_m & \beta_m \\ \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} (B_0 s_m + \widehat{\Psi}_{m-1} p_m)^T \\ y_m^T \end{pmatrix}, \end{aligned} \tag{2.40}$$

where $p_m \triangleq \widehat{M}_{m-1}\widehat{\Psi}_{m-1}^T s_m$. Equivalently,

$$B_{m+1} = B_0 + \left(\widehat{\Psi}_{m-1} \;\; B_0 s_m \;\; y_m\right) \mathcal{M}_m \begin{pmatrix} \widehat{\Psi}_{m-1}^T \\ (B_0 s_m)^T \\ y_m^T \end{pmatrix}, \tag{2.41}$$

where

$$\mathcal{M}_m = \begin{pmatrix} \widehat{M}_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\ \alpha_m p_m^T & \alpha_m & \beta_m \\ \beta_m p_m^T & \beta_m & \delta_m \end{pmatrix}. \tag{2.42}$$

Note that $\mathcal{M}_m$ has the following decomposition:

$$\mathcal{M}_m = \begin{pmatrix} I & p_m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \widehat{M}_{m-1} & 0 & 0 \\ 0 & \alpha_m & \beta_m \\ 0 & \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ p_m^T & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.43}$$

Thus, $\mathcal{M}_m$ is nonsingular if and only if $\alpha_m\delta_m - \beta_m^2 \neq 0$; that is, $\mathcal{M}_m$ is nonsingular if and only if $\phi_m \neq \phi_m^{SR1}$ (see Lemma 1). To complete the induction step, we will show that the last term in (2.41) is equal to $\widehat{\Psi}_m\widehat{M}_m\widehat{\Psi}_m^T$ by considering the following two cases separately: (i) $\phi_m = \phi_m^{SR1}$ and (ii) $\phi_m \neq \phi_m^{SR1}$.

Case (i): If $\phi_m = \phi_m^{SR1}$, then by Lemma 1, $\alpha_m = -\beta_m \neq 0$. Then

$$\mathcal{M}_m = \begin{pmatrix} I & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \widetilde{\mathcal{M}}_m \begin{pmatrix} I & 0 & 0 \\ 0 & -1 & 1 \end{pmatrix}, \tag{2.44}$$

where

$$\widetilde{\mathcal{M}}_m = \begin{pmatrix} \widehat{M}_{m-1} - \beta_m p_m p_m^T & \beta_m p_m \\ \beta_m p_m^T & -\beta_m \end{pmatrix}. \tag{2.45}$$

We now show that $\widetilde{\mathcal{M}}_m = \widehat{M}_m$. By the inductive hypothesis, $\widehat{M}_{m-1}$ is nonsingular. Together with the fact that $\beta_m \neq 0$, it can be checked directly that

$$\widetilde{\mathcal{M}}_m^{-1} = \begin{pmatrix} \widehat{M}_{m-1}^{-1} & \widehat{M}_{m-1}^{-1} p_m \\ p_m^T \widehat{M}_{m-1}^{-1} & -\beta_m^{-1} + p_m^T \widehat{M}_{m-1}^{-1} p_m \end{pmatrix}. \tag{2.46}$$

The (2,2)-entry of $\widetilde{\mathcal{M}}_m^{-1}$ can be simplified by substituting in for p_m and using the inductive step (2.36):

$$\begin{aligned} -\beta_m^{-1} + p_m^T \widehat{M}_{m-1}^{-1} p_m &= -\beta_m^{-1} + s_m^T(\widehat{\Psi}_{m-1}\widehat{M}_{m-1}\widehat{\Psi}_{m-1}^T)s_m \\ &= -\beta_m^{-1} - s_m^T B_0 s_m + s_m^T B_m s_m \\ &= -\beta_m^{-1} - s_m^T B_0 s_m + (1 - \phi_m)\beta_m^{-1} \\ &= -s_m^T B_0 s_m + s_m^T y_m. \end{aligned}$$

Substituting this into (2.46) and using the inductive hypothesis gives:

$$\begin{aligned}
\widetilde{\mathcal{M}}_m^{-1} &= \begin{pmatrix} \widehat{M}_{m-1}^{-1} & \widehat{\Psi}_{m-1}^T s_m \\ s_m^T \widehat{\Psi}_{m-1} & -s_m^T B_0 s_m + s_m^T y_m \end{pmatrix} \\
&= \begin{pmatrix} \left(\Xi_{m-1}^T \Pi_{m-1} \Omega_{m-1} \Pi_{m-1}^T \Xi_{m-1}\right) & \Xi_{m-1}^T \Pi_{m-1} \Psi_{m-1}^T s_m \\ s_m^T \Psi_{m-1} \Pi_{m-1}^T \Xi_{m-1} & -s_m^T B_0 s_m + s_m^T y_m \end{pmatrix} \\
&= \begin{pmatrix} \Xi_{m-1}^T \Pi_{m-1}^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Omega_{m-1} & \Psi_{m-1}^T s_m \\ s_m^T \Psi_{m-1} & -s_m^T B_0 s_m + s_m^T y_m \end{pmatrix} \begin{pmatrix} \Pi_{m-1}^T \Xi_{m-1} & 0 \\ 0 & 1 \end{pmatrix} \\
&= \Xi_m^T \begin{pmatrix} \Omega_{m-1} & -\Psi_{m-1}^T s_m & 0 \\ -s_m^T \Psi_{m-1} & -s_m^T B_0 s_m + \gamma_m & \gamma_m \\ 0 & \gamma_m & s_m^T y_m + \gamma_m \end{pmatrix} \Xi_m, \qquad (2.47)
\end{aligned}$$

where

$$\Xi_m \triangleq \begin{pmatrix} \Pi_{m-1}^T \Xi_{m-1} & 0 \\ 0 & E_m \end{pmatrix}, \quad \text{and} \quad E_m \triangleq \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Note that the middle matrix in (2.47) can be expressed as

$$\begin{pmatrix} \Omega_{m-1} & -\Psi_{m-1}^T s_m & 0 \\ -s_m^T \Psi_{m-1} & -s_m^T B_0 s_m + \gamma_m & \gamma_m \\ 0 & \gamma_m & s_m^T y_m + \gamma_m \end{pmatrix}, \qquad (2.48)$$

which is equivalent to

$$\begin{pmatrix} -S_{m-1}^T B_0 S_{m-1} + \Gamma_{m-1} & -L_{m-1} + \Gamma_{m-1} & -S_{m-1}^T B_0 s_m & 0 \\ -L_{m-1}^T + \Gamma_{m-1} & D_{m-1} + \Gamma_{m-1} & -Y_m^T s_m & 0 \\ -s_m^T B_0 S_{m-1} & -s_m^T Y_m & -s_m^T B_0 s_m + \gamma_m & \gamma_m \\ 0 & 0 & \gamma_m & y_m^T s_m + \gamma_m \end{pmatrix}.$$

Substituting this into (2.47), yields

$$\widetilde{\mathcal{M}}_m^{-1} = \Xi_m^T \Pi_m \Omega_m \Pi_m^T \Xi_m, \qquad (2.49)$$

where Π_m is defined in (2.25), replacing k with m, and Ω_m is defined in (2.37), replacing $m-1$ with m. Thus, $\widetilde{\mathcal{M}}_m = \widehat{M}_m$, as defined in (2.29).

We finish this case of the proof by showing that the last term in (2.41) is equal to $\widehat{\Psi}_m \widehat{M}_m \widehat{\Psi}_m^T$. Substituting in (2.44) gives that the last term in (2.41) can be written as

$$\begin{pmatrix} \Psi_{m-1} \Pi_{m-1}^T \Xi_{m-1} & B_0 s_m & y_m \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \widetilde{\mathcal{M}}_m \begin{pmatrix} I & 0 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \Xi_{m-1}^T \Pi_{m-1} \Psi_{m-1}^T \\ (B_0 s_m)^T \\ y_m^T \end{pmatrix}.$$

Using (2.49) this simplifies to

$$\Psi_m \Pi_m^T \begin{pmatrix} \Pi_{m-1}^T \Xi_{m-1} & 0 \\ 0 & E_m \end{pmatrix} \left(\Xi_m^T \Pi_m \Omega_m \Pi_m^T \Xi_m\right)^{-1} \begin{pmatrix} \Xi_{m-1}^T \Pi_{m-1} & 0 \\ 0 & E_m \end{pmatrix} \Pi_m \Psi_m^T,$$

or, in other words,

$$\Psi_m \Pi_m^T \Xi_m \left(\Xi_m^T \Pi_m \Omega_m \Pi_m^T \Xi_m\right)^{-1} \Xi_m^T \Pi_m \Psi_m^T,$$

which is exactly $\widehat{\Psi}_m \widehat{M}_m \widehat{\Psi}_m^T$. Thus, for $\phi_m = \phi_m^{SR1}$, the inductive step is proven.

Case (ii): We consider the case that $\phi_m \neq \phi_m^{SR1}$. We begin by showing that $\widehat{M}_m = \mathcal{M}_m$, given in (2.42). By Lemma 1, $\alpha_m + \beta_m \neq 0$. Second, $E_m = I_2$ (see (2.26)), and $\gamma_m = \phi_m/(\alpha_m + \beta_m)$ is well-defined (see (2.27)). Then, the inverse of $\mathcal{M}_m$ can be computed using arguments similar to those found in [18]:

$$\mathcal{M}_m^{-1} = \begin{pmatrix} \widehat{M}_{m-1}^{-1} & -\widehat{M}_{m-1}^{-1} p_m & 0 \\ -p_m^T \widehat{M}_{m-1}^{-1} & p_m^T \widehat{M}_{m-1}^{-1} p_m + \tilde{\alpha}_m & \tilde{\beta}_m \\ 0 & \tilde{\beta}_m & \tilde{\delta}_m \end{pmatrix}, \tag{2.50}$$

where

$$\tilde{\alpha}_m = \frac{\delta_m}{\alpha_m \delta_m - \beta_m^2}, \quad \tilde{\beta}_m = -\frac{\beta_m}{\alpha_m \delta_m - \beta_m^2} \quad \text{and} \quad \tilde{\delta}_m = \frac{\alpha_m}{\alpha_m \delta_m - \beta_m^2}. \tag{2.51}$$

Simplifying the expressions in (2.51), yields

$$\tilde{\alpha}_m = -s_m^T B_m s_m + \gamma_m, \quad \tilde{\beta}_m = \gamma_m, \quad \tilde{\delta}_m = y_m^T s_m + \gamma_m. \tag{2.52}$$

We now simplify the entries of (2.50) using the same approach as in [18]. Since $p_m = \widehat{M}_{m-1} \widehat{\Psi}_{m-1}^T s_m$, then $\widehat{M}_{m-1}^{-1} p_m = \widehat{\Psi}_{m-1}^T s_m$, giving us an expression for the (1,2) and (2,1) entries. The (2,2) block entry is simplified by first multiplying (2.39) by s_m^T on the left to obtain $s_m^T B_m s_m = s_m^T B_0 s_m + p_m^T \widehat{M}_{m-1}^{-1} p_m$. Then,

$$p_m^T \widehat{M}_{m-1}^{-1} p_m + \tilde{\alpha}_m = -s_m^T B_0 s_m + s_m^T B_m s_m + \tilde{\alpha}_m = -s_m^T B_0 s_m + \gamma_m.$$

Thus, using (2.48), (2.50) can be written as

$$\begin{aligned}
\mathcal{M}_m^{-1} &= \begin{pmatrix} \widehat{M}_{m-1}^{-1} & -\widehat{\Psi}_{m-1}^T s_m & 0 \\ -s_m^T \widehat{\Psi}_{m-1} & -s_m^T B_0 s_m + \gamma_m & \gamma_m \\ 0 & \gamma_m & y_m^T s_m + \gamma_m \end{pmatrix} \\
&= \begin{pmatrix} \Xi_{m-1}^T \Pi_{m-1} \Omega_{m-1} \Pi_{m-1}^T \Xi_{m-1} & -\Xi_{m-1}^T \Pi_{m-1} \Psi_{m-1}^T s_m & 0 \\ -s_m^T \Psi_{m-1} \Pi_{m-1}^T \Xi_{m-1} & -s_m^T B_0 s_m + \gamma_m & \gamma_m \\ 0 & \gamma_m & y_m^T s_m + \gamma_m \end{pmatrix} \\
&= \begin{pmatrix} \Xi_{m-1}^T \Pi_{m-1}^T & 0 \\ 0 & E_m \end{pmatrix} \begin{pmatrix} \Omega_{m-1} & \Psi_{m-1}^T s_m & 0 \\ s_m^T \Psi_{m-1} & -s_m^T B_0 s_m + \gamma_m & \gamma_m \\ 0 & \gamma_m & y_m^T s_m + \gamma_m \end{pmatrix} \begin{pmatrix} \Pi_{m-1}^T \Xi_{m-1} & 0 \\ 0 & E_m \end{pmatrix} \\
&= \Xi_m^T \begin{pmatrix} \Omega_{m-1} & \Psi_{m-1}^T s_m & 0 \\ s_m^T \Psi_{m-1} & -s_m^T B_0 s_m + \gamma_m & \gamma_m \\ 0 & \gamma_m & y_m^T s_m + \gamma_m \end{pmatrix} \Xi_m \\
&= \Xi_m^T \Pi_m \Omega_m \Pi_m^T \Xi_m,
\end{aligned} \tag{2.53}$$

proving that $\widehat{M}_m = \mathcal{M}_m$. Finally, using arguments similar to those in case (i), it can be shown that

$$B_{m+1} = B_0 + \Psi_m \Pi_m^T \Xi_m (\Xi_m^T \Pi_m \Omega_m \Pi_m^T \Xi_m)^{-1} \Xi_m^T \Pi_m \Psi_m^T = B_0 + \widehat{\Psi}_m \widehat{M}_m \widehat{\Psi}_m^T,$$

as desired. □

There are two main differences in the compact representation for the full Broyden class (Theorem 1) and the restricted Broyden class (Section 2). First, in Theorem 1, Ξ_k will always be the identity matrix for updates belonging to the restricted Broyden class. Second, in (2.22), the permutation matrices in the definitions of $\widehat{M}_k$ and $\widehat{\Psi}_k$, (equations (2.29) and (2.30), respectively) always cancel out in the restricted Broyden case. To emphasize that the permutation matrices do not cancel out for the general Broyden class updates, we use the notation $\widehat{M}_k$ and $\widehat{\Psi}_k$, in lieu of M_k and Ψ_k as in the restricted Broyden case.

Finally, we provide some insight regarding the permutation matrices (2.25). The permutation matrix Π_k acts in the following manner:

$$\Psi_k \Pi_k^T = \begin{pmatrix} B_0 s_0 & \cdots & B_0 s_{k-1} & y_0 & \cdots & y_{k-1} & B_0 s_k & y_k \end{pmatrix} = \begin{pmatrix} \Psi_{k-1} & B_0 s_k & y_k \end{pmatrix}.$$

Thus, $\hat{\Psi}_k$ in (2.22), which is given by $\hat{\Psi}_k = \Psi_k \Pi_k^T \Xi_k$ in (2.30), can be written as

$$\begin{aligned} \Psi_k \Pi_k^T \Xi_k &= \begin{pmatrix} \Psi_{k-1} & B_0 s_k & y_k \end{pmatrix} \begin{pmatrix} \Pi_{k-1}^T \Xi_{k-1} & 0 \\ 0 & E_k \end{pmatrix} \\ &= \begin{pmatrix} \Psi_{k-1} \Pi_{k-1}^T \Xi_{k-1} & (B_0 s_k \;\; y_k) E_k \end{pmatrix} \\ &\vdots \\ &= \begin{pmatrix} (B_0 s_0 \;\; y_0) E_0 & (B_0 s_1 \;\; y_1) E_1 & \cdots & (B_0 s_k \;\; y_k) E_k \end{pmatrix}. \end{aligned} \tag{2.54}$$

In other words, when applied on the right of Ψ_k, the product $\Pi_k^T \Xi_k$ permutes the columns of Ψ_k and, using the matrices $\{E_i\}$, combines columns of Ψ_k whenever the update is a rank-one update.

Unfortunately, computing $\widehat{M}_k$ is not straightforward. In particular, the diagonal matrix Γ_k in Eq. (2.27) involves $s_i^T B_i s_i$ for each $i \in \{0, \ldots, k\}$, which requires B_i for $0 \leq i \leq k$. In the next section, we propose a recursive method for computing $\widehat{M}_k$ that does not require storing the matrices B_i for $0 \leq i \leq k$.

Computing $\widehat{M}_k$

In this section, we propose a recursive method for computing $\widehat{M}_k$ from $\widehat{M}_{k-1}$. This method is based on the method proposed in [17, 18] for solving a linear system whose system matrix is generated using the restricted Broyden class of updates. We note that when ϕ is fixed, the recursive method here is similar to the one proposed in [30].

In the proof of Theorem 1, we showed that

$$\widehat{M}_k = \begin{cases} \begin{pmatrix} \widehat{M}_{k-1} - \beta_k p_k p_k^T & \beta_k p_k \\ \beta_k p_k^T & -\beta_k \end{pmatrix} & \text{if } \phi_k = \phi_k^{SR1} \\ \begin{pmatrix} \widehat{M}_{k-1} + \alpha_k p_k p_k^T & \alpha_k p_k & \beta_k p_k \\ \alpha_k p_k^T & \alpha_k & \beta_k \\ \beta_k p_k^T & \beta_k & \delta_k \end{pmatrix} & \text{otherwise,} \end{cases} \tag{2.55}$$

which are given in (2.43) and (2.45). We now relate some of the entries in $\widehat{M}_k$ with other stored or computable quantities involving the pairs $\{s_i, y_i\}, i = 0, \ldots, k$. The vector p_k can be computed as

$$p_k = \widehat{M}_{k-1}\widehat{\Psi}_{k-1}^T s_k = \widehat{M}_{k-1}\Xi_{k-1}^T \Pi_{k-1} \begin{pmatrix} S_{k-1}^T B_0 s_k \\ Y_{k-1}^T s_k \end{pmatrix}. \tag{2.56}$$

Note that in (2.56), the vector $S_{k-1}^T B_0 s_k$ is the first $k-1$ entries in the last column of $S_k^T B_0 S_k$, and the vector $Y_{k-1}^T s_k$ is the first k entries in the last column of $Y_k^T S_k$. Moreover, the entry α_k, given by $\alpha_k = -(1-\phi_k)/s_k^T B_k s_k$, can be computed from the following:

$$s_k^T B_k s_k = s_k^T \left(B_0 + \widehat{\Psi}_{k-1}\widehat{M}_{k-1}\widehat{\Psi}_{k-1}^T \right) s_k = s_k^T B_0 s_k + s_k^T \widehat{\Psi}_{k-1} p_k. \tag{2.57}$$

In (2.57), the quantity $s_k^T B_0 s_k$ is the kth diagonal entry in $S_k^T B_0 S_k$, and $s_k^T \widehat{\Psi}_{k-1} p_k$ is the inner product of p_k and $\widehat{\Psi}_{k-1}^T s_k$, the latter vector already having been computed in (2.56). Recall that the entry β_k is given by $\beta_k = -\phi_k / y_k^T s_k$, where $y_k^T s_k$ is the $(k+1)$st diagonal entry in $S_k^T Y_k$. Finally, $\delta_k = (1 + \phi_k s_k^T B_k s_k / y_k^T s_k)/y_k^T s_k$, which uses the previously computed quantities $s_k^T B_k s_k$ and $y_k^T s_k$.

For the initialization of $\widehat{M}_0$, notice that $\widehat{M}_0$ in (2.31) can be written as

$$\widehat{M}_0 = \begin{cases} -\beta_0 & \text{if } \phi_0 = \phi_0^{SR1} \\ \begin{pmatrix} \alpha_0 & \beta_0 \\ \beta_0 & \delta_0 \end{pmatrix} & \text{otherwise,} \end{cases} \tag{2.58}$$

where α_0, β_0, and δ_0 are defined as in (2.34).

In Algorithm 3, we use the recursions described above to compute $\widehat{M}_k$ given in (2.55).

Algorithm 3: This algorithm computes $\widehat{M}_k$ in (2.55).

Input: An initial ϕ_0 and B_0;
Define $\widehat{M}_0$ using (2.33);
Define $\Psi_0 = (B_0 s_0 \; y_0)$;
  for $j = 1 : k$ **do**
    $\widehat{\Psi}_{j-1}^T s_j \leftarrow \Xi_{j-1}^T \Pi_{j-1} \Psi_{j-1}^T s_j$;
    $p_j \leftarrow \widehat{M}_{j-1}(\widehat{\Psi}_{j-1}^T s_j)$;
    $s_j^T B_j s_j \leftarrow s_j^T B_0 s_j + (s_j^T \widehat{\Psi}_{j-1}) p_j$;
    $\alpha_j \leftarrow -(1-\phi_j)/(s_j^T B_j s_j)$;
    $\beta_j \leftarrow -\phi_j/(y_j^T s_j)$;
    $\delta_j \leftarrow (1 + \phi_j (s_j^T B_j s_j)/(y_j^T s_j))/(y_j^T s_j)$;
    Form $\widehat{M}_j$ using (2.55);
  end for

Note that the matrices Π_{j-1} and Ξ_{j-1} are not explicitly formed in Algorithm 3. Instead, (2.54) can be used to compute $\widehat{\Psi}_{j-1}$ in line 4 of Algorithm 3. Thus, at iteration j, only two vectors (p_j and $\widehat{\Psi}_{j-1}^T s_j$), both of which have length less than or equal to $2j$, need to be computed to form $\widehat{M}_j$.

2.2.3 Solving Linear Systems

Given the compact representation of B_{k+1}, we can solve

$$B_{k+1}r = z, \tag{2.59}$$

where $r, z \in \Re^n$, by computing the compact representation of the inverse of B_{k+1}. Intuitively speaking, computing the compact representation of the inverse is due to the fact that $H_{k+1} \triangleq B_{k+1}^{-1}$ can also be written using a recursion relation [13]:

$$H_{k+1} = H_k + \frac{1}{s_k^T y_k} s_k s_k^T - \frac{1}{y_k^T H_k y_k} H_k y_k y_k^T H_k + \Phi_k (y_k^T H_k y_k) v_k v_k^T, \tag{2.60}$$

where $H_k \triangleq B_k^{-1}$, $v_k = s_k/(y_k^T s_k) - (H_k y_k)/(y_k^T H_k y_k)$, and

$$\Phi_k = \frac{(1-\phi_k)(y_k^T s_k)^2}{(1-\phi_k)(y_k^T s_k)^2 + \phi_k (y_k^T H_k y_k)(s_k^T B_k s_k)}. \tag{2.61}$$

Note that when $\phi_k = \phi_k^{SR1}$, then the corresponding Φ_k is given by

$$\Phi_k^{SR1} = \frac{y_k^T s_k}{y_k^T s_k - y_k^T H_k y_k}.$$

In this section, we derive the compact representation of the inverse of a Broyden class member. This derivation is similar to the process of finding the inverse of a member of the restricted Broyden class presented in [17].

Applying the Sherman-Morrison-Woodbury formula (see, e.g., [21]) to the compact representation of B_{k+1} given in (2.22), gives that

$$B_{k+1}^{-1} = B_0^{-1} + B_0^{-1} \widehat{\Psi}_k \left(-\widehat{M}_k^{-1} - \widehat{\Psi}_k^T B_0^{-1} \widehat{\Psi}_k \right)^{-1} \widehat{\Psi}_k^T B_0^{-1}.$$

For quasi-Newton matrices it is conventional to let H_i denote the inverse of B_i for each i; with this notation, the inverse of B_{k+1}^{-1} is given by

$$H_{k+1} = H_0 + H_0 \widehat{\Psi}_k \left(-\widehat{M}_k^{-1} - \widehat{\Psi}_k^T H_0 \widehat{\Psi}_k \right)^{-1} \widehat{\Psi}_k^T H_0. \tag{2.62}$$

Using the definition of $\widehat{\Psi}_k$ in (2.30) gives that

$$\widehat{\Psi}_k^T H_0 \widehat{\Psi}_k = \Xi_k^T \Pi_k \begin{pmatrix} S_k^T B_0 S_k & S_k^T Y_k \\ Y_k^T S_k & Y_k^T H_0 Y_k \end{pmatrix} \Pi_k^T \Xi_k,$$

and thus,

$$-\widehat{M}_k^{-1} - \widehat{\Psi}_k^T H_0 \widehat{\Psi}_k = -\Xi_k^T \Pi_k \begin{pmatrix} \Gamma_k & D_k + R_k + \Gamma_k \\ D_k + R_k^T + \Gamma_k & D_k + \Gamma_k + Y_k^T H_0 Y_k \end{pmatrix} \Pi_k^T \Xi_k.$$

Substituting $H_0 \widehat{\Psi}_k = H_0 (B_0 S_k \; Y_k) \Pi_k^T \Xi_k = (S_k \; H_0 Y_k) \Pi_k^T \Xi_k$ into (2.62) gives the compact representation for the inverse of any member of the full Broyden class:

$$H_{k+1} = H_0 + \widetilde{\Psi}_k \widetilde{M}_k \widetilde{\Psi}_k^T, \tag{2.63}$$

where $\widetilde{\Psi}_k = (S_k \; H_0 Y_k)\, \Pi_k^T \Xi_k$ and

$$\widetilde{M}_k \equiv \left(-\Xi_k^T \Pi_k \begin{pmatrix} \Gamma_k & D_k + R_k + \Gamma_k \\ D_k + R_k^T + \Gamma_k & D_k + \Gamma_k + Y_k^T H_0 Y_k \end{pmatrix} \Pi_k^T \Xi_k \right)^{-1}. \tag{2.64}$$

Computing $\widetilde{M}_k$. Using an approach similar to how $\widehat{M}_k$ is computed, $\widetilde{M}_k$ can be computed as follows:

$$\widetilde{M}_k = \begin{cases} \begin{pmatrix} \widetilde{M}_{k-1} - \tilde{\beta}_k \tilde{p}_k \tilde{p}_k^T & -\tilde{\beta}_k \tilde{p}_k \\ -\tilde{\beta}_k \tilde{p}_k^T & -\tilde{\beta}_k \end{pmatrix} & \text{if } \Phi_k = \Phi_k^{SR1} \\ \begin{pmatrix} \widetilde{M}_{k-1} + \tilde{\delta}_k \tilde{p}_k \tilde{p}_k^T & \tilde{\beta}_k \tilde{p}_k & \tilde{\delta}_k \tilde{p}_k \\ \tilde{\beta}_k \tilde{p}_k^T & \tilde{\alpha}_k & \tilde{\beta}_k \\ \tilde{\delta}_k \tilde{p}_k^T & \tilde{\beta}_k & \tilde{\delta}_k \end{pmatrix} & \text{otherwise,} \end{cases} \tag{2.65}$$

where

$$\tilde{\alpha}_k = \frac{1}{s_k^T y_k} + \Phi_k \frac{y_k^T H_k y_k}{(s_k^T y_k)^2}, \quad \tilde{\beta}_k = -\frac{\Phi_k}{y_k^T s_k}, \quad \tilde{\delta}_k = -\frac{1 - \Phi_k}{y_k^T H_k y_k}, \tag{2.66}$$

and $\tilde{p}_k = \widetilde{M}_{k-1} \widetilde{\Psi}_{k-1}^T y_k$. The initial matrix $\widetilde{M}_0$ is given by the following:

$$\widetilde{M}_0 = \begin{cases} -\tilde{\beta}_0 & \text{if } \Phi_0 = \Phi_0^{SR1} \\ \begin{pmatrix} \tilde{\alpha}_0 & \tilde{\beta}_0 \\ \tilde{\beta}_0 & \tilde{\delta}_0 \end{pmatrix} & \text{otherwise,} \end{cases} \tag{2.67}$$

where $\tilde{\alpha}_0$, $\tilde{\beta}_0$, and $\tilde{\delta}_0$ are defined as in (2.66) with $k = 0$. A practical iterative method to solve equations of the form (2.59) is given in Algorithm 4. At each iteration j, only the four vectors p_j, $\widehat{\Psi}_{j-1}^T s_j$, $\tilde{p}_j$, and $\widetilde{\Psi}_{j-1}^T y_j$, (all of which have length less than or equal to $2j$), need to be computed to form $\widetilde{M}_j$. In addition, the two previous matrices $\widehat{M}_{j-1}$ and $\widetilde{M}_{j-1}$ must be stored; both of these have dimensions less than or equal to $2j \times 2j$.

2.2.4 Numerical Experiments

In this section we test the accuracy of Algorithm 3 to compute the compact representation by comparing it with the matrix obtained using the Broyden update formula (2.15). In addition, we demonstrate that solves with B_{k+1} in (2.59) can be done efficiently using Algorithm 4 with respect to both accuracy and time. For these experiments, we used five (limited-memory) quasi-Newton pairs to compute B_{k+1}. To generate quasi-Newton pairs, we simulated a line-search method where the iterates are updated as follows:

$$x_{j+1} = x_j - \alpha_j B_j^{-1} g_j, \quad \text{for } 1 \le j \le 4,$$

where α_j is drawn from a uniform distribution between 0 and 1. To initialize the process, the vectors x_0 and x_1 were drawn from a Gaussian distribution with mean 0 and standard deviation 1 so that $s_0 = x_1 - x_0$. The corresponding gradients, $g_j = \nabla f(x_j)$ for $0 \le j \le 5$, were also drawn from a Gaussian distribution in order to form $y_j = g_{j+1} - g_j$ for $0 \le j \le 4$. The matrix B_0 was initially defined as $B_0 = \gamma I$, where $\gamma > 0$ was drawn from a Gaussian distribution with mean 0.

We considered four experiments where we vary the value of ϕ_i at each iteration i. In particular, we chose values of ϕ_i according to the scheme given in Table 2.1. We ran each experiment ten times with $n = 10^2, 10^3$, and 10^4 and report results.

Algorithm 4 : This algorithm solves $B_{k+1}r = z$.

Input: An initial ϕ_0, B_0, and H_0;
Define $\widehat{M}_0$ using (2.32) and $\widetilde{M}_0$ = using (2.67);
  for $j = 1 : k$ **do**
    Compute $s_j^T B_j s_j$ using Algorithm 3;
    $\widetilde{\Psi}_{j-1}^T y_j \leftarrow \Xi_{j-1}^T \Pi_{j-1} \Psi_{j-1}^T H_0 y_j$;
    $\tilde{p}_j \leftarrow \widetilde{M}_{j-1}(\widetilde{\Psi}_{j-1}^T y_j)$;
    $y_j^T H_j y_j \leftarrow y_j^T H_0 y_j + (y_j^T \widetilde{\Psi}_{j-1})\tilde{p}_j$;
    $\Phi_j \leftarrow (1-\phi_j)(y_j^T s_j)^2 / ((1-\phi_j)(y_j^T s_j)^2 + \phi_j (y_j^T H_j y_j)(s_j^T B_j s_j))$;
    $\tilde{\alpha}_j \leftarrow (1 + \Phi_j (y_j^T H_j y_j)/(y_j^T s_j))/(y_j^T s_j)$;
    $\tilde{\beta}_j \leftarrow -\Phi_j/(y_j^T s_j)$;
    $\tilde{\delta}_j \leftarrow -(1-\Phi_j)/(y_j^T H_j y_j)$;
    Form $\widetilde{M}_j$ using (2.65);
  end for
$\widetilde{\Psi}_k \leftarrow H_0 \Psi_k \Pi_k^T \Xi_k$;
$r = H_0 z + \widetilde{\Psi}_k \widetilde{M}_k \widetilde{\Psi}_k^T z$;

Experiment	ϕ_0	ϕ_1	ϕ_2	ϕ_3	ϕ_4
1	$\phi_0 < 0$	1	$0 < \phi_2 < 1$	0	$\phi_4 > 1$
2	$\phi_0 < 0$	1	ϕ_2^{SR1}	0	$\phi_4 > 1$
3	$\phi_0 < 0$	1	ϕ_2^{SR1}	ϕ_3^{SR1}	$\phi_4 > 1$
4	ϕ_0^{SR1}	1	ϕ_2^{SR1}	0	$\phi_4 > 1$

Table 2.1: The values of ϕ_i for $0 \le i \le 4$ for each experiment. The choice of $\phi_1 = 1$ corresponds to the BFGS update while $\phi_3 = 0$ corresponds to the DFP update. Note that Experiment 1 does not use SR1 updates.

Accuracy of the Compact Representation

To test the accuracy of the compact representation, we form each B_{k+1} using (2.22) together with the proposed compact formulation given in Theorem 1. (In particular, we use Algorithm 3 to form $\widehat{M}_k$.) We denote the resulting matrix by B_{k+1}^{CR}. In Table 2, we report the average relative error of the compact representation in the Frobenius norm:

$$\text{Relative error } = \frac{\|B_{k+1} - B_{k+1}^{\mathrm{CR}}\|_F}{\|B_{k+1}\|_F},$$

where B_{k+1} is computed using (2.15). In addition, we report the maximum relative error for each set of ten trials.

The small relative errors in Table 2.2 reflects the fact that the proposed compact representation for the full Broyden class of quasi-Newton matrices is correct; moreover, the relative errors suggest that Algorithm 3 provides a method to compute the compact representation to high accuracy.

Experiment 1		
n	Average Error	Maximum Error
10^2	1.3067e-11	7.6229e-11
10^3	4.6184e-15	1.5131e-14
10^4	2.4791e-13	2.1677e-12

Experiment 2		
n	Average Error	Maximum Error
10^2	1.1665e-11	3.8207e-11
10^3	4.7083e-15	1.5442e-14
10^4	1.5523e-13	8.0258e-13

Experiment 3		
n	Average Error	Maximum Error
10^2	4.1647e-12	1.9090e-11
10^3	4.5646e-14	3.7051e-13
10^4	5.9815e-14	4.2493e-13

Experiment 4		
n	Average Error	Maximum Error
10^2	1.8618e-14	1.3834e-13
10^3	1.0335e-15	3.1745e-15
10^4	7.6799e-16	4.3197e-15

Table 2.2: Average and maximum relative errors over ten different trials for each experiment with $n = 10^2, 10^3$, and 10^4.

Accuracy of the Compact Representation of the Inverse

In these experiments, we test the accuracy of Algorithm 4 to solve linear systems of the form $B_{k+1}r = z$, where $r, z \in \Re^n$ and B_{k+1} is a quasi-Newton matrix. The matrix B_{k+1} is generated using five quasi-Newton pairs as described in the beginning of this section. Moreover, the righthand side z is randomly generated for each experiment. In Table 2.3, we present the average residual error using the two-norm:

$$\text{Relative error} = \frac{\|B_{k+1}r^{\text{ICR}} - z\|_2}{\|z\|_2},$$

where r^{ICR} is the solution to $B_{k+1}r = z$ using the inverse compact representation computed by Algorithm 4. These results suggest that the compact representation of the inverse can be used to solve linear systems to high accuracy.

Experiment 1		
n	Average Error	Maximum Error
10^2	1.6839e-10	1.3668e-09
10^3	1.8737e-14	9.5484e-14
10^4	3.3781e-12	2.6962e-11

Experiment 2		
n	Average Error	Maximum Error
10^2	3.1829e-10	1.9051e-09
10^3	2.9169e-14	1.3481e-13
10^4	2.2182e-12	1.5569e-11

Experiment 3		
n	Average Error	Maximum Error
10^2	1.1744e-10	4.6894e-10
10^3	8.9801e-14	3.5093e-13
10^4	6.2890e-12	5.2380e-11

Experiment 4		
n	Average Error	Maximum Error
10^2	4.5225e-14	3.1923e-13
10^3	4.8059e-15	3.1537e-14
10^4	3.7784e-15	5.1305e-15

Table 2.3: Average and maximum relative errors over ten different trials for each experiment with $n = 10^2, 10^3$, and 10^4.

In addition, during the experiments, the computational time of the proposed method was recorded and compared to a similar solve using the MATLAB "backslash". In particular, with

the same quasi-Newton pairs, the backslash command was used to solve $B_{k+1}r = z$, where B_{k+1} was formed using (2.15). The times required were averaged for each experiment and for each value of n. These results, as well as the maximum time for each set of ten trials, are given in Table 4 and do not include the time MATLAB required to form B_{k+1}. Note that the average computational times in Table 2.4 indicate that as n increases using Algorithm 4 becomes significantly less computationally expensive than using the backslash command. Finally, we performed similar experiments for larger systems of equations with $n = 10^5$ and 10^6. These results are reported in Table V. Due to the size of n, the matrix B_{k+1} was too large to store in memory to test the MATLAB backslash command; for this reason, only results using the inverse compact representation are presented the table.

Experiment 1				
	ICR		MATLAB backslash	
n	Avg. Time	Max. Time	Avg. Time	Max. Time
10^2	4.8e-03	1.6e-02	1.7e-03	1.1e-02
10^3	1.3e-03	6.2e-03	1.5e-02	1.9e-02
10^4	2.0e-03	3.4e-03	5.3e+00	5.6e+00

Experiment 2				
	ICR		MATLAB backslash	
n	Avg. Time	Max. Time	Avg. Time	Max. Time
10^2	3.1e-03	1.1e-02	7.5e-04	4.3e-03
10^3	9.5e-04	2.4e-03	1.7e-02	3.4e-02
10^4	1.9e-03	2.8e-03	5.4e+00	5.8e+00

Experiment 3				
	ICR		MATLAB backslash	
n	Avg. Time	Max. Time	Avg. Time	Max. Time
10^2	2.5e-03	4.4e-03	5.3e-04	1.4e-03
10^3	7.3e-04	2.1e-03	1.6e-02	2.3e-02
10^4	1.8e-03	3.0e-03	5.3e+00	5.5e+00

Experiment 4				
	ICR		MATLAB backslash	
n	Avg. Time	Max. Time	Avg. Time	Max. Time
10^2	2.6e-03	6.8e-03	8.2e-04	2.6e-03
10^3	8.6e-04	2.2e-03	1.7e-02	3.3e-02
10^4	1.7e-03	3.0e-03	5.3e+00	5.7e+00

Table 2.4: Average and maximum computational times for solving $B_{k+1}r = z$ using the inverse compact representation (ICR) and the MATLAB "backslash" command with $n = 10^2, 10^3$, and 10^4.

Finally, in Figure 2.5, each of the ten trials that were summarized in Tables 2.4 and 2.5 are displayed. For smaller n (e.g., $n = 1000$), we do not have confidence that the computational times were recorded correctly by MATLAB because the times required for $n = 10^3$ were, on average, smaller than those required for $n = 10^2$. We suspect this could be because the computations are

performed so quickly. However, for larger n, the computational time appears to increase linearly with respect to n.

	Experiment 1	
n	Avg. Time	Max. Time
10^5	1.9e-02	4.2e-02
10^6	2.6e-01	3.0e-01

	Experiment 2	
n	Avg. Time	Max. Time
10^5	1.5e-02	2.0e-02
10^6	2.5e-01	2.9e-01

	Experiment 3	
n	Avg. Time	Max. Time
10^5	1.8e-02	4.0e-02
10^6	2.0e-01	2.4e-01

	Experiment 4	
n	Avg. Time	Max. Time
10^5	1.6e-02	3.0e-02
10^6	1.9e-01	2.2e-01

Table 2.5: Average and maximum computational times for solving $B_{k+1}r = z$ using the inverse compact representation (ICR) with $n = 10^5$ and 10^6.

2.2.5 Conclusion

We derived the compact formulation for members of the full Broyden class of quasi-Newton updates. The compact representation allows for different ϕ_k at each iteration as well as different ranks of updates. With this compact formulation, we demonstrated how to solve linear systems defined by these limited-memory quasi-Newton matrices. Numerical results suggest that the compact representation can be computed to high accuracy and that we can solve (2.59) efficiently and accurately using the compact representation of the inverse of B_{k+1}.

2.3 Summary of Contribution

In this chapter we proposed a novel approach for solving the sparse ℓ_2-ℓ_1 problem in a trust-region setting. By using an approximation of second-derivative information known as the L-BFGS update we improve the quality of the optimization routine and provide improved reconstructions from noisy observations. Finally, we extended the formulation of the compact representation typically used to compute the L-BFGS to include the Full Broyden class of quasi-Newton updates allowing the quasi-Newton method to update at each iteration. We demonstrate the ability to accurately and efficiently use the formulation to solve large scale linear systems often involved in solving large scale optimization problems.

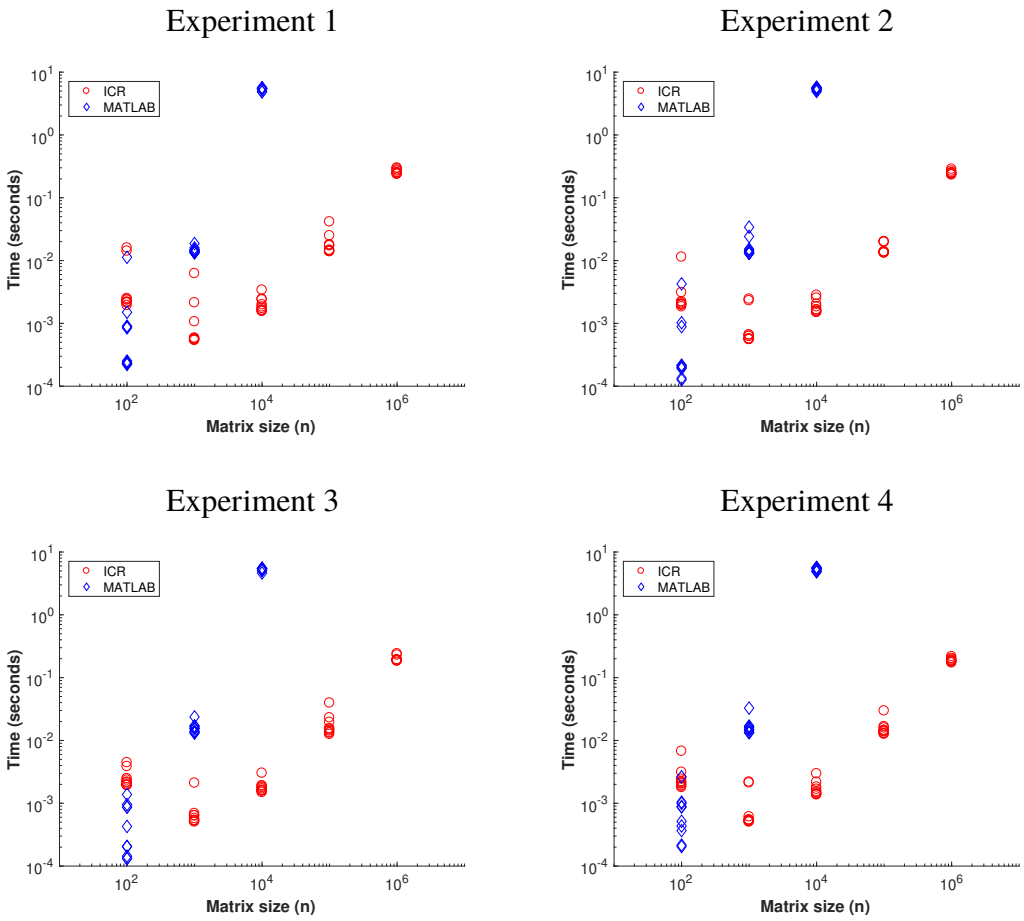


Figure 2.5: Comparison of computational times for solving $B_{k+1}r = z$ using the inverse compact representation (ICR) and the MATLAB "backslash" command. Each experiment displays 10 tests each for $n = 10^k$, where $k = 2, \ldots 6$. The MATLAB "backslash" command could not be used for $n = 10^5$ and 10^6 due to memory limitations.

Bibliography

[1] L. Adhikari et al. “Limited-memory trust-region methods for sparse relaxation”. *Proc.SPIE* 10394 (2017), pp. 10394 - 10394 - 8. DOI: 10.1117/12.2271369.

[2] L. Adhikari et al. “Trust-region methods for nonconvex sparse recovery optimization”. *The 2016 International Symposium on Information Theory and Its Applications*. 2016, pp. 275–279.

[3] S. Becker and J. Fadili. “A quasi-Newton proximal splitting method”. *Advances in Neural Information Processing Systems*. 2012, pp. 2618–2626.

[4] J. Brust, J. B. Erway, and R. F. Marcia. “On solving L-SR1 trust-region subproblems”. *Computational Optimization and Applications* 66.2 (2016), pp. 1–22. ISSN: 1573-2894. DOI: 10.1007/s10589-016-9868-3.

[5] J. Brust et al. *Shape-changing L-SR1 trust-region methods*. Technical Report 2016-2. Wake Forest University, 2016.

[6] O. Burdakov et al. *On Efficiently Combining Limited Memory and Trust-Region Techniques*. Tech. rep. 2013:13. Linkping University, Optimization, 2015, p. 33.

[7] O. Burdakov et al. “On Efficiently Combining Limited Memory and Trust-Region Techniques”. *Mathematical Programming Computation* 9 (2016), pp. 101–134.

[8] J. V. Burke, A. Wiegmann, and L. Xu. *Limited Memory BFGS Updating In A Trust-Region Framework*. Technical Report. University of Washington, 1996.

[9] R. H. Byrd, J. Nocedal, and R. B. Schnabel. “Representations of quasi-Newton matrices and their use in limited-memory methods”. *Math. Program.* 63 (1994), pp. 129–156.

[10] R. H. Byrd, D. C. Liu, and J. Nocedal. “On the behavior of Broyden’s class of quasi-Newton methods”. *SIAM Journal on Optimization* 2.4 (1992), pp. 533–557.

[11] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2000, pp. xx+959. ISBN: 0-89871-460-5.

[12] O. DeGuchy, J. B. Erway, and R. F. Marcia. “Compact representation of the full Broyden class of quasi-Newton updates”. *Numerical Linear Algebra with Applications* 25.5 (2018), e2186.

[13] J. E. Dennis Jr and J. J. Moré. “Quasi-Newton methods, motivation and theory”. *SIAM review* 19.1 (1977), pp. 46–89.

[14] D. L. Donoho and M. Elad. “Optimally sparse representation in general (nonorthogonal) dictionaries via $\ell 1$ minimization”. *Proceedings of the National Academy of Sciences* 100.5 (2003), pp. 2197–2202.

[15] Y. C. Eldar and G. Kutyniok. *Compressed sensing: theory and applications*. Cambridge Uni. Press, 2012.

[16] J. B. Erway and R. F. Marcia. “Algorithm 943: MSS: MATLAB Software for L-BFGS Trust-Region Subproblems for Large-Scale Optimization”. *ACM Transactions on Mathematical Software* 40.4 (June 2014), 28:1–28:12. URL: http://doi.acm.org/10.1145/2616588.

[17] J. B. Erway and R. F. Marcia. “On solving large-scale limited-memory quasi-Newton equations”. *Linear Algebra and its Applications* 515 (2017), pp. 196–225. ISSN: 0024-3795. DOI: 10.1016/j.laa.2016.11.003.

[18] J. B. Erway and R. F. Marcia. “On efficiently computing the eigenvalues of limited-memory quasi-Newton matrices”. *SIAM Journal on Matrix Analysis and Applications* 36.3 (2015), pp. 1338–1359.

[19] M. Figueiredo and et al. “Gradient Projection for Sparse Reconstruction”. *IEEE J. of Selected Top.in Sig. Proc.* 1.4 (2007), pp. 586–597.

[20] D. M. Gay. “Computing optimal locally constrained steps”. *SIAM J. Sci. Stat. Com.* 2.2 (1981). ISSN: 0196-5204.

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Third. Baltimore, Maryland: The Johns Hopkins University Press, 1996. ISBN: 0-8018-5414-8.

[22] M. Hintermüller and T. Wu. “Nonconvex TV^q-Models in Image Restoration: Analysis and a Trust-Region Regularization–Based Superlinearly Convergent Solver”. *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1385–1415.

[23] W. Huang, K. A. Gallivan, and P.-A. Absil. “A Broyden class of quasi-Newton methods for Riemannian optimization”. *SIAM Journal on Optimization* 25.3 (2015), pp. 1660–1685.

[24] M. Langer et al. “Large scale optimization of beam weights under dose-volume restrictions”. *International Journal of Radiation Oncology* Biology* Physics* 18.4 (1990), pp. 887–893.

[25] J. Lee, Y. Sun, and M. Saunders. “Proximal Newton-type methods for convex optimization”. *Advances in Neural Information Processing Systems*. 2012, pp. 836–844.

[26] J. D. Lee et al. “Practical large-scale optimization for max-norm regularization”. *Advances in neural information processing systems*. 2010, pp. 1297–1305.

[27] C. Liu and S. A. Vander Wiel. “Statistical quasi-Newton: A new look at least change”. *SIAM Journal on Optimization* 18.4 (2007), pp. 1266–1285.

[28] D. C. Liu and J. Nocedal. “On the limited memory BFGS method for large scale optimization”. *Math. Program.* 45 (1989), pp. 503–528.

[29] X. Lu. “A study of the limited memory SR1 method in practice”. PhD thesis. Department of Computer Science, University of Colorado at Boulder, 1996.

[30] L. Lukšan and J. Vlček. “Recursive form of general limited memory variable metric methods”. *Kybernetika* 49.2 (2013), pp. 224–235.

[31] J. J. Moré and D. C. Sorensen. “Computing a trust region step”. *SIAM J. Sci. and Stat. Com.* 4 (1983).

[32] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[33] J. Nocedal and S. J. Wright. *Numerical Optimization*. New York: Springer-Verlag, 1999, pp. xxii+636. ISBN: 0-387-98793-2.

[34] M. J. D. Powell. “Convergence properties of a class of minimization algorithms”. *Nonlinear programming* 2.0 (1975), pp. 1–27.

[35] Y. Wang, J. Cao, and C. Yang. “Recovery of seismic wavefields based on compressive sensing by an l1-norm constrained trust region method and the piecewise random subsampling”. *Geophysical Journal International* 187.1 (2011), pp. 199–213.

[36] J. F. Yang and Y. Zhang. “Alternating direction algorithms for L1 problems in compressive sensing”. *Sci. Comput* 33.1 (2011), p. 250278.

[37] R. Yokoyama, Y. Hasegawa, and K. Ito. “A MILP decomposition approach to large scale optimization in structural design of energy supply systems”. *Energy Conversion and Management* 43.6 (2002), pp. 771–790.

[38] J. Yu et al. “A quasi-Newton approach to nonsmooth convex optimization problems in machine learning”. *The Journal of Machine Learning Research* 11 (2010), pp. 1145–1200.

[39] Y. Zhang and R. Tewarson. “Quasi-Newton algorithms with updates from the preconvex part of Broyden’s family”. *IMA Journal of Numerical Analysis* 8.4 (1988), pp. 487–509.

[40] G. Zhou, X. Zhao, and W. Dai. “Low rank matrix completion: A smoothed l0-search”. *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2012, pp. 1010–1017.

Chapter 3

Sparse Optimization

Signal recovery methods which seek to reconstruct high-dimensional sources from low-dimensional observations are particularly difficult in that they are typically formulated as nonlinear optimization problems. This is especially true in the application known as photon-limited imaging. Under this regime, measurements at the detector contain low photon counts and are corrupted with Poison noise. In this chapter we present three algorithms with a focus on photon-limited applications. The first method incorporates more sophisticated bounds in the optimization problem to increase the accuracy of the solution. The second utilizes a generalized Shannon entropy function as a regularizer to promote sparsity. The third method uses asynchronous parallel pattern searches to determine the intensity of the regularization term. In the next section, we define the conditions under which this modality is established as well as the associated optimization problem used to reconstruct their associated signals.

3.1 Sparse Signal Recovery

Sparse signal recovery is fundamental in the theory of compressed sensing, which seeks to recover a signal from a series of low dimensional observations. The signal is obtained as the solution of an underdetermined linear system where we rely on the a priori knowledge that the signal of interest has a high level of sparsity. In this context, a high level of sparsity corresponds to the fact that the signal can be represented by only a few crucial non-zero elements. It is the a priori knowledge of sparsity which allows us to recover the signal from a lower number of samples than the minimum requirement established by the Nyquist-Shannon sampling theorem [17, 22]. The assumption of sparsity is not limited to a signal's canonical basis. For instance, applications of compressive sensing with natural images leverage the sparsity of the images in the wavelet or discrete cosine basis [5]. In magnetic resonance imaging (MRI), sparsity in the Fourier domain has been used to decrease the number of readings required and therefore increase the speed of image processing [50]. Beyond these applications, sparse signal recovery has been proven effective in a variety of applications such as astronomy, radar imaging and communication [14, 16, 56]

The signal recovery problem described previously, can be mathematically formulated as the following: If we let $\mathbf{f}^* \in \mathbb{R}^n$ be the sparse signal of interest, then we can model the observational process as the linear equation

$$\mathbf{y} = \mathbf{A}\mathbf{f}^*, \tag{3.1}$$

where $\mathbf{y} \in \mathbb{R}^m$ is a vector of the recorded measurements and $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a matrix, with

(a) (b) (c)

Figure 3.1: (a) The ”woman” image. (b) The Haar wavelet coefficients of the ”woman” image in (a). (c) The magnitude of the discrete cosine transform. Note that most of the power lies in the upper left coefficients.

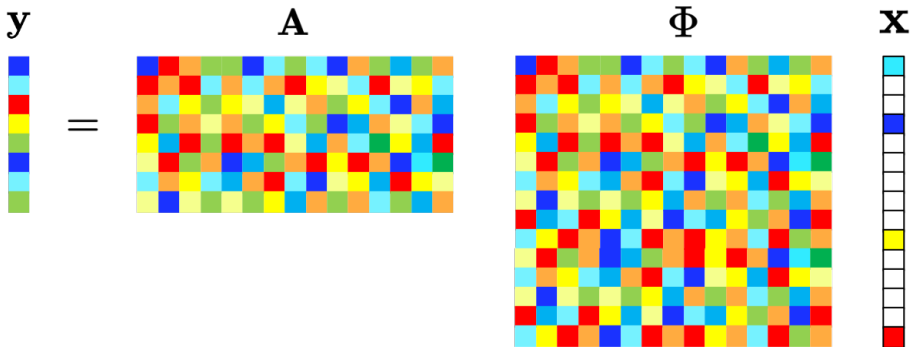


Figure 3.2: The sparse signal recovery problem where $\mathbf{y}$ is the low dimensional observation, $\mathbf{A}$ is the projection matrix dependent on the imaging system, $\boldsymbol{\Phi}$ is a sparsifying basis and $\mathbf{x}$ is the vector of basis coefficients.

$n > m$, describing the observational processes. As described previously, $\mathbf{f}^*$ need not be sparse in the canonical basis. In order to accommodate this scenario, let us consider the sparsifying basis $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$, then (3.1) can be reformulated as $\mathbf{y} = \mathbf{A}\boldsymbol{\Phi}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$ contains the sparse coefficients. The setup is shown in Figure 3.2, note that $\mathbf{f}^* = \boldsymbol{\Phi}\mathbf{x}$, thus if $\mathbf{f}^*$ is sparse in the canonical basis then $\boldsymbol{\Phi}$ is the identity matrix and $\mathbf{x} = \mathbf{f}^*$. During this explanation we have assumed that the collection of measurements is a noiseless action. In actuality, noise often effects the quality of the observations used to reconstruct the signal. In the next chapter we will be discussing applications of signal recovery in a photon-limited regime, as such, it is necessary to discuss the recovery of sparse signals in the setting of a Poisson process.

3.2 The Inhomogeneous Poisson Process Model

In order to create a realistic model of the signal recovery process, we must consider the fact that the observations used during the process are likely corrupted by noise. Factors such as faulty equipment, transmission errors or even human error may be culpable for the presence of corrupted

signals. Typically, this type of noise is assumed to be Gaussian in nature, which is suitable for most applications in signal recovery. On the other hand we are interested in applications of photon-limited imaging where we assume that the number of photons hitting the detectors or charged-coupled devices (CCD), is relatively low. Beyond the low number of photons being recorded at the detector, we also assume that the measurements are corrupted by Poisson noise and model the process using a Poisson distribution [61]. Under the inhomogenous Poisson process model, we can describe the observational process as

$$\mathbf{y} \sim \text{Poisson}(\mathbf{A}\mathbf{f}^*), \tag{3.2}$$

where $\mathbf{y} \in \mathbb{Z}_+^m$ is the vector of observed photon counts, $f* \in \mathbf{R}_+^n$ is the vector of true signal intensity and $A \in \mathbb{R}_+^{m \times n}$ is the system matrix.

The estimation of the true signal $\mathbf{f}^*$ from the model described in (3.2) is obtained by minimizing the negative Poisson log-likelihood function

$$F(\mathbf{f}) = \mathbb{1}^T\mathbf{A}\mathbf{f} - \sum_{i=1}^{m} y_i \log(\mathbf{e}_i^T\mathbf{A}\mathbf{f}), \tag{3.3}$$

where $\mathbb{1}$ is an m-vector of ones and $\mathbf{e}_i$ is the i-th column of the canonical basis.The likelihood function (3.3) corresponds to observing $\mathbf{y}$ given $\mathbf{A}\mathbf{f}$. Because we are operating under the assumption that the true signal of interest is sparse, many components of the estimation $\mathbf{f}$ are assumed to be zero. When this occurs, (3.3) contains a singularity. In order to avoid this complication we introduce the parameter $0 < \beta \ll 1$ within the log function [28]. The result is the convex function

$$F(\mathbf{f}) = \mathbb{1}^T\mathbf{A}\mathbf{f} - \sum_{i=1}^{m} y_i \log(\mathbf{e}_i^T\mathbf{A}\mathbf{f} + \beta). \tag{3.4}$$

For the applications presented in this Chapter, the true signal $\mathbf{f}^*$ is non-negative, furthermore (3.4) is ill-posed as a result of the underdetermined nature of (3.2). In order to make the problem more tractable, penalization terms are incorporated into the optimization process. The resultant constrained optimizaion problem takes the form:

$$\begin{aligned} \hat{\mathbf{f}} \quad = \quad & \underset{\mathbf{f}\in\mathbb{R}^n}{\arg\min} \quad F(\mathbf{f}) + \tau\,\text{pen}(\mathbf{f}) \\ & \text{subject to } \ \mathbf{f} \geq 0, \end{aligned} \tag{3.5}$$

where $\tau > 0$ is the weight of the penalization term and pen is a typically non-convex penalty function promoting sparsity in the solution. Over the course of the work in this Chapter, we will describe optimization techniques used to solve (3.5).

3.3 Photon-Limited Fluorescence Lifetime Imaging Microscopy Signal Recovery with Known Bounds

The work described in this section is based on the paper by DeGuchy, Adhikari, Kim, and Marcia [20]. Fluorescence microscopy is a technique that offers a sensitive, specific and versatile approach to studying *in vivo* cellular and molecular dynamics in real time [46, 62]. In particular, fluorescence lifetime imaging (FLIM) is emerging as an increasingly important improvement in fluorescence microscopy that allows for the analysis of complex events within biological

specimens [25, 29, 54]. The lifetime of a fluorophore provides vital information about the local environment as a result of its sensitivity to factors such as pH levels, ion presence and oxygen concentration. Even though the fluorescence lifetime is sensitive to the previously mentioned factors, measurements are independent of local fluorophore concentration or absorption in the sample [39]. In fluorescence lifetime imaging, the fluorescence spatial distribution and decay rates within a tissue sample are recovered. The population of fluorophores within the tissue domain is typically low. Thus, many of the recently developed methods for sparse signal recovery can potentially be applied to FLIM applications. However, these methods typically minimize a least-squares cost function (e.g., [7, 15]), thereby implicitly assuming the measurement intensity is sufficiently high and the system noise can be modeled using Gaussian statistics. Here, we operate under the assumption presented in Section 3.2 in that the number of photons hitting the CCD is relatively low, and thus the measurements are corrpupted by Poisson noise [10]. In addition, we also assume the maximum concentration of fluorophore is known, information that we can exploit by incorporating bound constraints to the optimization problem to improve the reconstruction. In this work, we propose a method for solving the FLIM inverse problem in this setting with time-dependent measurements.

3.3.1 Fluoresence-Lifetime Model

For the fluorescence-lifetime imaging problem, we recover the time-dependent fluorescence emitted by fluorophores from time-dependent measurements due to pulsed excitation of a strongly scattering medium. This involves determining the location of the fluorophores as well as their corresponding concentration and lifetime. The optical properties of the medium are assumed to be known to sufficient precision. In this section, we review our partial differential equation model (see [2, 9, 68] for details).

Forward Model

Let Ω denote the domain of a uniform absorbing and scattering medium with boundary denoted by $\partial\Omega$. A pulse of exciting light is injected through $\partial\Omega$ and propagates into Ω. Let $S(\mathbf{r}, t)$ denote an exterior time-dependent source of exciting light, where $\mathbf{r} \in \partial\Omega$ is a prescribed location on the boundary and $t > 0$. We denote the intensity of the exciting light source by $I^e(\mathbf{r}, t)$, where $\mathbf{r} \in \Omega$ is a position within the interior domain and $t \in [0, T]$ for some final time T. The propagation and scattering of light in the domain is governed by the following initial-boundary value problem for the diffusion equation:

$$\begin{aligned} \frac{1}{c}\frac{\partial I^e}{\partial t} - \nabla\cdot(\kappa^e \nabla I^e) + \mu_a^e I^e &= 0 \quad \text{in } \Omega \times (0, T], \\ I^e(\mathbf{r}, 0) &= 0 \quad \text{in } \Omega \\ I^e + \alpha^e \kappa^e \frac{\partial I^e}{\partial n} &= \begin{cases} \gamma^e S(\mathbf{r}, t) & \text{on } \mathbf{r} \in \mathbf{r}_s, \\ 0 & \text{on } \mathbf{r} \in \partial\Omega \backslash \mathbf{r}_s, \end{cases} \end{aligned} \tag{3.6}$$

where κ^e denotes the diffusion coefficient, μ_a^e denotes the absorption coefficient at the exciting wavelength, $\partial I^e / \partial n$ denotes the outward normal derivative of I^e, the constants α^e and γ^e are defined in terms of μ_a^e and κ^e as part of the diffusion approximation and $\mathbf{r}_s$ denotes the set of source locations on the boundary.

Portions of the exciting light source $I^e(\mathbf{r},t)$ are absorbed by the fluorophores and are then re-emitted. The transportation of the re-emitted light $I^f(\mathbf{r},t)$ is then modeled by

$$\begin{aligned} \frac{1}{c}\frac{\partial I^f}{\partial t} - \nabla\cdot(\kappa^f\nabla I^f) + \mu_a^f I^f = Q(\mathbf{r},t) \quad \text{in } \Omega\times(0,T], \\ I^f(\mathbf{r},0) = 0 \ \text{ in } \Omega, \\ I^f + \alpha^f\kappa^f\frac{\partial I^f}{\partial n} = 0 \ \text{ on } \partial\Omega, \end{aligned} \tag{3.7}$$

where κ^f denotes the diffusion coefficient and μ_a^f denotes the absorption coefficient at the exciting wavelength. Here, the emission of fluorescent light is due to the excited interior fluorescence source

$$Q(\mathbf{r},t) = \chi(\mathbf{r})h(\mathbf{r})\int_0^t e^{-(t-t')/\tau(\mathbf{r})}I^e(\mathbf{r},t')dt', \tag{3.8}$$

where $\chi(\mathbf{r})$ is an indicator function for the sparse spatial distribution of the fluorophores, $h(\mathbf{r})$ is the fluorophore concentration, and $\tau(\mathbf{r})$ is the fluorescence lifetime (see [10] for details).

The measurements, $u(\mathbf{r},t)$, of scattered light leaving the boundary of the domain are modeled using

$$u(\mathbf{r},t) = -\kappa^f\frac{\partial I^f}{\partial n} = \frac{1}{\alpha^f}I^f \quad \text{on } \partial\Omega\times(0,T] \tag{3.9}$$

using the boundary condition in (3.7). In our approach, we make use of time-averaged measurements defined by

$$\bar{u}(\mathbf{r}) = \frac{1}{\alpha^f}\bar{I}^f(\mathbf{r}) = \frac{1}{\alpha^f}\frac{1}{T}\int_0^T I^f(\mathbf{r},t)\mathrm{d}t \quad \text{on } \partial\Omega. \tag{3.10}$$

The steady-state optical fluence rate for emission light, $\bar{I}^f$, satisfies the steady-state diffusion equation

$$\begin{aligned} -\kappa^f\nabla^2\bar{I}^f + \mu_a\bar{I}^f &= \bar{Q} \quad \text{in } \Omega, \\ \bar{I}^f + \alpha^f\kappa^f\partial_n\bar{I}^f &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{3.11}$$

Assuming that the location of the fluorophores do not vary temporally, we can estimate the support of the fluorophores from these time-averaged data.

Inverse Problem

At M distinct locations denoted by $\mathbf{r}_m \in \partial\Omega$ for $m = 1,\cdots,M$, we take CCD measurements of the scattered light leaving the boundary of the domain. Furthermore, we take N samples of these measurements in time, which are collected with sampling rate $\Delta t = T/N$. We denote the observed collection of data by the vector $\mathbf{u}\in\mathbb{R}^{MN}$ with $\mathbf{u} = [u(\mathbf{r}_1,t_1),\cdots,u(\mathbf{r}_M,t_1),u(\mathbf{r}_1,t_2),\cdots,u(\mathbf{r}_M,t_N)]$. The objective of the inverse problem is to recover the sparse spatial distribution of fluorescence lifetime $Q(\mathbf{r},t)$ in (3.8) from the set of noisy measurements $\mathbf{u}$ given by (3.9).

3.3.2 Fluorescence Source Recovery

In this section, we describe our numerical approach for recovering the parameters that define the fluorescence source $Q(\mathbf{r},t)$, namely its support $\chi(\mathbf{r},t)$, concentration $h(\mathbf{r})$, and lifetime $\tau(\mathbf{r})$. We assume that the fluorophores are concentrated only in a small area (i.e., the spatial support

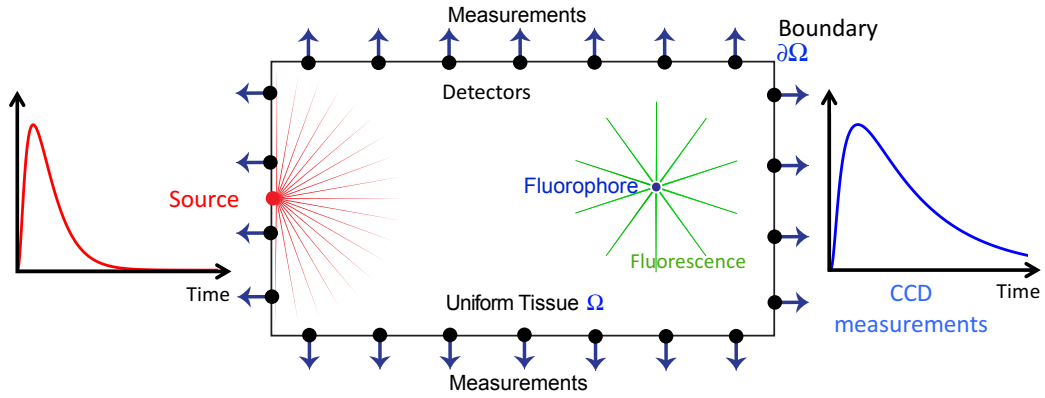


Figure 3.3: A schematic diagram of our time-dependent fluorescence lifetime imaging microscopy setup. Here a fluorophore within the interior of the domain Ω is excited by a pulse of light from a source $S(\mathbf{r},t)$. CCD measurements $u(\mathbf{r},t)$ of the time-resolved fluorescence from a fluorophore source $Q(\mathbf{r},t)$ are obtained along the boundary $\partial\Omega$.

$\chi(\mathbf{r},t)$ is small) and that the fluorophore locations do not vary in time (see [66]). Furthermore, we assume that the optical properties of the medium for excitation and emission (i.e., the constants κ^e, κ^f, μ_a^e, and μ_a^f) are known. Finally, we assume that we are in a photon-limited regime, where the number of photons hitting the detector is relatively low.

Rather than recovering $\chi(\mathbf{r},t)$, $h(\mathbf{r})$, and $\tau(\mathbf{r})$ simultaneously (which is computationally prohibitive), we make use of the following 3-stage approach for recovering $Q(\mathbf{r},t)$.

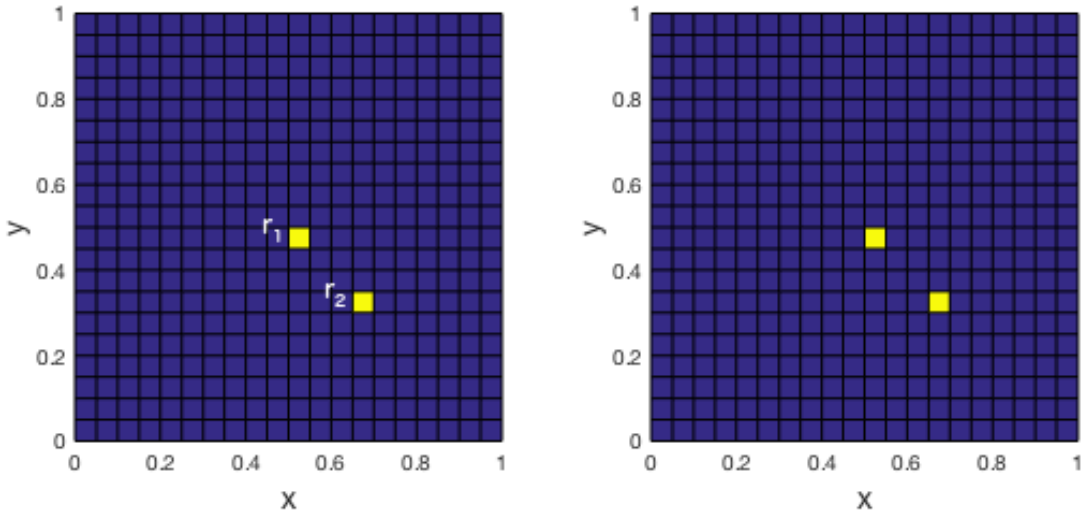
Stage I: Recover $\chi(\mathbf{r},t)$. In this stage, we determine the locations at which $Q(\mathbf{r},t)$ is non-zero, i.e., the support of the characteristic function $\chi(\mathbf{r},t)$. Since the fluorophore locations do not vary in time, we exploit this a priori information by using the *time-averaged* observations, which dramatically reduces the problem size.

Stage II: Determine the values of $Q(\mathbf{r},t)$ at specific locations $\mathbf{r}$ and times t. We recover $Q(\mathbf{r},t)$ by restricting the solution to the support $\chi(\mathbf{r})$ from Stage I. This restriction allows us to concentrate our reconstruction efforts to regions in the medium that directly produce the measurements.

Stage III: Recover $h(\mathbf{r})$ and $\tau(\mathbf{r})$. Having computed from Stage II the values of $Q(\mathbf{r},t)$ at specific locations $\mathbf{r}$ and times t, we recover $h(\mathbf{r})$ and $\tau(\mathbf{r})$ using (3.8) and a nonlinear fitting algorithm.

Related Methods

We note that this work is similar to the approach we had proposed previously [2]. The primary difference is that the new approach allows us to incorporate known bounds on the signal in Stage 2 using a *bounded* Poisson recovery algorithm. Previous work for solving Poisson inverse problems include statistical multiscale modeling and analysis frameworks [52], nonparametric estimators using wavelet decompositions [8], and combination expectation-maximization algorithms with a total variation-based regularization [58]. The approach in [27] offers a different approach for minimizing Poisson log-likelihood functions using the alternating direction method of multipliers. For a recent review article on inverse problems with Poisson data, see [63] and all the references therein. Our approach is different from existing methods through our use of (i) a p-norm



(a) True Support (b) Reconstructed support

Figure 3.4: 2D signal support. (a) True fluorophore locations $\mathbf{r_1}$ and $\mathbf{r_2}$. (b) Reconstructed support provided by thresholding and computing the mode of the reconstructions for the five excitation sources.

regularization to promote sparsity in the reconstruction, (ii) a multi-stage approach to handle the high-dimensional time-dependent measurements, and (iii) constraints to exploit known bounds on the signals.

3.3.3 Numerical Approach

Finite Difference Discretization

In our setting, we solve the initial-boundary value problems (3.6) and (3.7) at equally-spaced nodes using the Crank-Nicolson method [48]. We note that the size of the spatial and temporal discretization depends on available computational resources since the total size of the problem is $N_x N_y N$, where N_x and N_y are the number of nodes in each spatial dimension and N is the number of time-level samples, which can easily become prohibitively very large. As defined in (3.9), in the discrete setting, the measurements are obtained by restricting the numerical solution of emission light, say $\mathbf{I}^f$, to the boundary:

$$\mathbf{u} = \frac{1}{\alpha^f}\mathbf{R}\mathbf{I}^f = \frac{1}{\alpha^f}\mathbf{R}\mathbf{L}^{-1}\tilde{\mathbf{Q}}, \tag{3.12}$$

where $\mathbf{R}$ is a boundary restriction operator, $\mathbf{L}$ is the finite difference operator and $\tilde{\mathbf{Q}}$ is the averaged value of Q between consecutive time steps. We denote the matrix product of operators $\frac{1}{\alpha^f}\mathbf{R}\mathbf{L}^{-1}$ by the system matrix $\mathbf{A}$ for the inverse algorithm. Note that the system matrix $\mathbf{A}$ is not computed explicitly; rather we implicitly compute the action $\mathbf{A}(\mathbf{x})$ and $\mathbf{A}^T(\mathbf{x})$ on-the-fly using the forward and backward substitution techniques. The actions of the steady-state boundary value problem in (3.11) and (3.12) are similarly defined.

Photon-Limited Recovery Method

Because we are assuming that we are in a photon-limited regime, we model the arrival of photons at the detector using Poisson statistics as outlined by (3.2). As in (3.5), we formulate the Poisson reconstruction problem as a constrained optimization formulation. This is accomplished using the negative log likelihood function in (3.3) given by

$$F(\mathbf{f}) = \mathbb{1}^T\mathbf{A}\mathbf{f} - \sum_{i=1}^{m} y_i \log(\mathbf{e}_i^T\mathbf{A}\mathbf{f}).$$

The associated constrained Poisson reconstruction problem is given by the following

$$\begin{aligned} \widehat{\mathbf{f}} \;=\; & \underset{\mathbf{f}\in\mathbb{R}^n}{\arg\min} \quad \boldsymbol{\Phi}(\mathbf{f}) \equiv F(\mathbf{f}) + \beta \,\|\mathbf{f}\|_p^p \\ & \text{subject to} \quad \mathbf{b_L} \preceq \mathbf{f} \preceq \mathbf{b_U}. \end{aligned} \tag{3.13}$$

where $\|\mathbf{f}\|_p^p$ $(0 \le p < 1)$ is a penalty function that promotes sparsity in our solution, $\beta > 0$ is a scalar regularization parameter, and $\mathbf{b_L}$ and $\mathbf{b_U}$ are vectors of known lower and upper bounds, respectively, on the true signal. Typically, $\mathbf{b_L}$ is a vector of zeros since it ensures that the solution, which corresponds to the fluorescence sources, is nonnegative. Our optimization problem formulation is different from the more commonly used least-squares minimization problem [10] in three ways: (1) we use a negative log-likelihood function, instead of the widely used least-squares data-fidelity term, to model the noise statistics more accurately; (2) instead of using sparsity-promoting ℓ_1-norm, we use a non-convex p-norm, where $0 \le p < 1$, to bridge the convex ℓ_1-norm and the ℓ_0 counting semi-norm; and (3) we enforce known bound constraints on our solution.

In Stage I, we are more interested in estimating the support of $\chi(\mathbf{r}, t)$. Enforcing the bounds in (3.13) at each iteration slows down the algorithm computationally, so we relax the bounds to simple non-negativity constraints.

The function $\boldsymbol{\Phi}(f)$ is nonconvex due to the nonconvexity of the p-norm penalty term. Although the SPIRAL-ℓ_p algorithm is not guaranteed to converge to a global solution, it can be shown that any accumulation point of the algorithm will be a critical point [1].

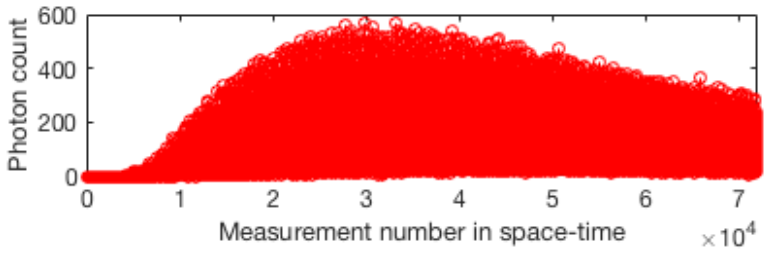
Having obtained the support $\chi(\mathbf{r})$ from Stage I, we define $\mathbf{b_L}$ and $\mathbf{b_U}$. Specifically, in the known locations i where the signal $\mathbf{f}$ is zero, we enforce $(\mathbf{b_L})_i = (\mathbf{b_U})_i = 0$. In Stage II, we solve (3.13) using B-SPIRAL-ℓ_1, a method we previously developed [3]. We use a negligible regularization parameter β and the conventional ℓ_1-norm, i.e., $p = 1$, since the sparsity of the solution is already known. In this stage, we solve a sequence of quadratic subproblems of the form

$$\begin{aligned} \mathbf{f}^{k+1} \;=\; & \underset{\mathbf{f}\in\mathbb{R}^n}{\arg\min} \quad \tfrac{1}{2} \,\|\, \mathbf{f} - \mathbf{s}^k \,\|_2^2 \\ & \text{subject to} \quad \mathbf{b_L} \preceq \mathbf{f} \preceq \mathbf{b_U}, \end{aligned}$$

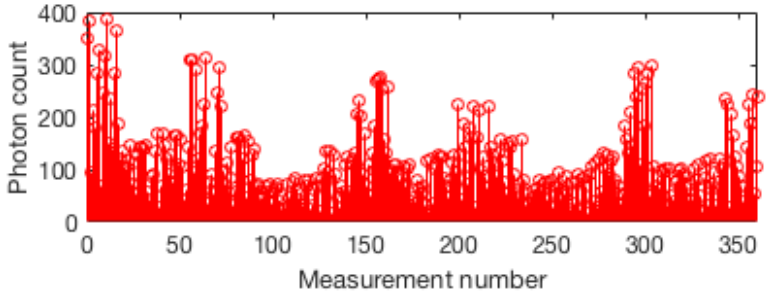
where $\mathbf{s}^k = \mathbf{f}^k - \frac{1}{\alpha_k}\nabla F(\mathbf{f}^k)$. Here, α_k refers to the step size in the direction of the gradient which is chosen by a modified Barzilai-Borwein (BB) method [13]. These subproblems are significantly easier to solve than (3.13).

3.3.4 Numerical Experiments

In this section, we implement the previously discussed three-stage reconstruction method in order to solve the 2D fluorescence lifetime imaging problem. The simulations are performed in



(a)

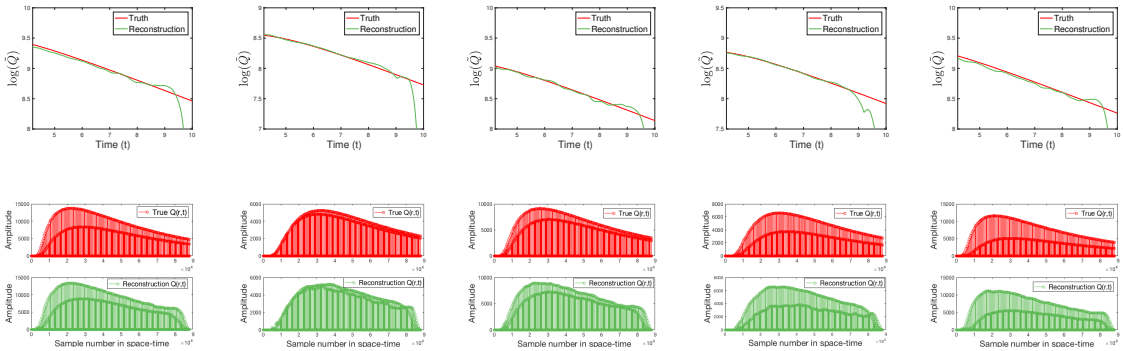


(b)

Figure 3.5: Experiment measurements: (a) Time-dependent measurements $\mathbf{u}$ (from Eq. (3.9)) corrupted by 7.5% Poisson noise. (b) Time-averaged measurements $\overline{\mathbf{u}}$ (from Eq. (3.10)). Measurements are taken from 72 boundary detectors per exterior source.

MATLAB using a unit square domain $\Omega = (0,1) \times (0,1)$ in conjunction with the following non-dimensionalized optical properties: the absorption coefficient $\mu_a = 0.05$ and the diffusion coefficient $\kappa = 0.0476$ [7]. The experiments required $N = 200$ time-level samples with the sampling rate $\triangle t = 0.05$ recorded by $M = 72$ boundary detectors using 5 exterior near-infrared source points. The fluorescence-lifetime and fluorophore concentration are set to 5.7 and 2000, respectively [64]. MATLAB's `poissrnd` function is used to corrupt the simulated boundary measurements with Poisson noise. The noise level(%) is computed by $100 * \|\mathbf{A}\mathbf{f}^* - \mathbf{y}\|_2 / \|\mathbf{y}\|_2$. The SPIRAL-$\ell_p$ algorithm used in Stages I and II are initialized using $\mathbf{A}^T\mathbf{y}$. The termination criteria for the algorithm is triggered if the relative objective values do not produce a significant change, i.e., $|\Phi(\mathbf{f}^{k+1}) - \Phi(\mathbf{f}^k)|/|\Phi(\mathbf{f}^k)| < 10^{-8}$. The regularization parameters (β and p) are manually optimized to get the minimum RMSE (RMSE(%) $= 100 \cdot \|\widehat{\mathbf{f}} - \mathbf{f}^*\|_2 / \|\mathbf{f}^*\|_2$, where $\widehat{\mathbf{f}}$ is an estimate of $\mathbf{f}^*$).

This work considers an experiment which consists of a two fluorophore point source fluorescence-lifetime reconstruction problem (see Fig. 3.4a) with 5 exterior sources. The time dependent observations $\mathbf{u}$ are corrupted by Poisson Noise (see Fig. 3.5a). In Step 1 of our approach, we use the time averaged measurements $\overline{\mathbf{u}}$ to recover an estimate of the fluorophore support for each of the 5 exterior sources. Exploiting the fact that the location of the fluorophores is the same for each of the sources, the final support reconstruction is obtained by thresholding and computing the mode of the SPIRAL-ℓ_p reconstruction (see Fig. 3.4). Using the recovered support estimation from Step 1, we reconstruct $\tilde{\mathbf{Q}}$ in (3.12) using B-SPIRAL-ℓ_1, where the upper bound on the fluorescence is $\mathbf{b_U} = 1.6 \times 10^4$, which we assume is known for each fluorophore (see Fig. 3.6). The reconstruction requires negligible regularization as the support has been identified and the sparsity of the support is reflected in the lower and upper bounds $\mathbf{b_L}$ and $\mathbf{b_U}$, respectively.



(a) Excitation Source 1 (b) Excitation Source 2 (c) Excitation Source 3 (d) Excitation Source 4 (e) Excitation Source 5

Figure 3.6: (Top) Approximations (in log scale) of the fluorescence lifetime from each of the five excitation sources. (Bottom) Time-dependent fluorescence reconstruction $\tilde{\mathbf{Q}}$ for each of the five excitation sources.

Finally, in Step 3 we use the integrated MATLAB nonlinear least squares function `lsqnonlin` to compute the estimate for the fluorophore concentration $\hat{h}$ at the two source locations as well as the fluorescence-lifetime $\hat{\tau}$. The estimates are computed using the initial concentration value of $\hat{h}_0$=1.0 for both locations and $\hat{\tau} = 1.0$ for the initial fluorescence lifetime value. The results are presented in Table 1.

	Ground Truth	Estimate
$h(r_1)$	2.00×10^3	2.08×10^3
$h(r_2)$	2.00×10^3	1.95×10^3
τ	5.70	5.63

Table 3.1: The comparison between the true and computed concentrations and fluorescence lifetimes. Fluorophore concentrations are denoted by h at locations r_1 and r_2. The fluorescence lifetimes are given by τ.

3.3.5 Conclusion

In this work, we used a three-stage method to solve the fluorescence lifetime imaging problem from Poisson noise corrupted boundary measurements. This imaging problem required us to model signal measurements through the solutions of a coupled initial-boundary value problem for light scattering and absorption inside our simulated tissue sample. Furthermore, as an alternative to the typical Gaussian model, we explicitly model Poisson noise in the inverse problem and use a non-convex sparse recovery method (SPIRAL-ℓ_p) to determine the support of the fluorophores and promote sparsity in the signal recovery. To improve on the accuracy of our reconstruction, we incorporate known bounds on the signal as constraints in the optimization formulation of the problem. A numerical experiment for a small scale problem demonstrates that the proposed method accurately solves the FLIM problem.

3.4 Non-Convex Shannon Entropy for Photon-Limited Imaging

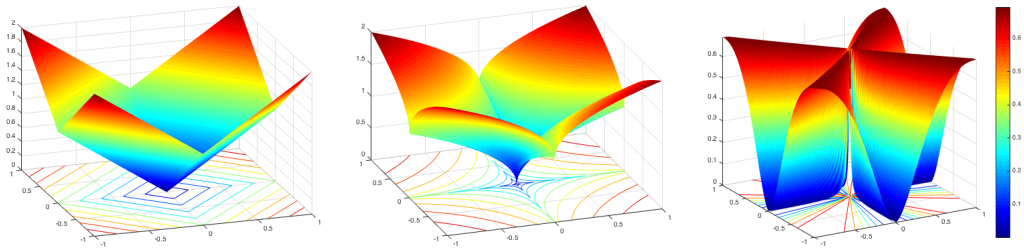
The work described in this section is based on the paper by Adhikari, Baikejiang, DeGuchy, and Marcia [6]. We again address sparse signal reconstruction in a photon-limited regime. As stated in Section 3.1, if the signal $\mathbf{f}^*$ is known to only have a few non-zero components, a sparsity-promoting penalty term, e.g., ℓ_1-norm [27, 33] (see Fig. 3.7(a)) or ℓ_p-norm [4] (see Fig. 3.7(b)) can be used to regularize the Poisson log-likelihood to recover $\mathbf{f}^*$. In this work, we propose to regularize the Poisson log-likelihood by the generalized nonconvex Shannon entropy function:

$$H_p(\mathbf{f}) = \sum_{i=1}^{n} \psi_i(\mathbf{f}), \quad \text{where} \quad \psi_i(\mathbf{f}) = \begin{cases} -\dfrac{|f_i|^p}{\|\mathbf{f}\|_p^p} \log\left(\dfrac{|f_i|^p}{\|\mathbf{f}\|_p^p}\right) & \text{if } f_i \neq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{3.14}$$

where $p > 0$ (see Fig. 3.7(c)). Recently, this function $H_p(\mathbf{f})$ in (3.14) was proposed by Huang et al. [38] as a sparsity-promoting regularizer in the Gaussian noise context, where data fidelity is imposed using a least-squares objective function. The Shannon entropy function has been used in various applications as well. For instance, a Bayesian image reconstruction method based on a Shannon entropy form [59] has been studied by Nunez et al. [53] for the Hubble space telescope data. Skilling et al. [60] maximized the Shannon entropy

$$S(\mathbf{f}) = -\sum_{j=1}^{n} p_j \log(p_j), \quad \text{where } p_j = \frac{f_j}{\sum_{i=1}^{n} f_i},$$

for image recovery in astronomy. Moreover, Donoho et al. [23] recovered nearly-black objects by minimizing the Shannon entopy regularized least-squares function. Our approach is novel in regularizing the Poisson log-likelihood using the sparsity promoting generalized Shannon entropy function (3.14) within a photon-limited context.



(a) (b) (c)

Figure 3.7: Penalties in 2D space for promoting sparsity. (a) Convex ℓ_1 norm, (b) p norm, where $0 < p < 1$, and (c) the *generalized* nonconvex Shannon entropy function $H_p(\mathbf{f})$.

3.4.1 Shannon Poisson Intensity Reconstruction

The generalized Shannon sparsity-promoting Poisson intensity reconstruction problem has the following constrained minimization form

$$\begin{aligned} \widehat{\mathbf{f}} \;=\; & \underset{\mathbf{f}\in\mathbb{R}^n}{\arg\min} \quad \boldsymbol{\Phi}(\mathbf{f}) \equiv F(\mathbf{f}) + \tau H_p(\mathbf{f}) \\ & \text{subject to } \; \mathbf{f} \succeq 0, \end{aligned} \tag{3.15}$$

where

$$F(\mathbf{f}) \;=\; \mathbb{1}^\top \mathbf{A}\mathbf{f} - \sum_{i=1}^{m} y_i \log(e_i^\top \mathbf{A}\mathbf{f} + \beta) \tag{3.16}$$

is the negative Poisson log-likelihood function, where $\mathbb{1}$ is an m-vector of ones, e_i is the ith canonical basis unit vector, and $\beta > 0$ (where typically $\beta \ll 1$). In this work, we follow the Sparse Poisson Intensity Reconstruction ALgorithm (SPIRAL) framework [33] and the regularization technique from the Entropy Minimization approach by Huang et al. [38]. Specifically, we define a sequence of quadratic subproblems of the form

$$\begin{aligned} \mathbf{f}^{k+1} \;=\; & \underset{\mathbf{f}\in\mathbb{R}^n}{\arg\min} \quad \boldsymbol{\Phi}^k(\mathbf{f}) \equiv F^k(\mathbf{f}) + \tau H_p^k(\mathbf{f}) \\ & \text{subject to } \; \mathbf{f} \succeq 0, \end{aligned} \tag{3.17}$$

that have closed-form solutions so that the sequence of iterates $\{\mathbf{f}^k\}$ are easily obtained. We define $\boldsymbol{\Phi}^k(\mathbf{f})$ as follows.

The first term, $F^k(\mathbf{f})$, is obtained using the second-order Taylor series approximation $F^k(\mathbf{f})$ to $F(\mathbf{f})$ around the current iterate $\mathbf{f}^k$:

$$F(\mathbf{f}) \approx F(\mathbf{f}^k) + (\mathbf{f} - \mathbf{f}^k)^\top \nabla F(\mathbf{f}^k) + \frac{1}{2}(\mathbf{f} - \mathbf{f}^k)^\top \nabla^2 F(\mathbf{f}^k)(\mathbf{f} - \mathbf{f}^k). \tag{3.18}$$

A further simplification can be made by approximating the Hessian matrix $\nabla^2 F(\mathbf{f}^k)$ with a scalar multiple of the identity, $\alpha_k \mathbf{I}$, where α_k is chosen by the modified Barzilai-Borwein method [12] as

$$\alpha_k \;=\; \frac{\left\| \sqrt{\mathbf{y}} \cdot (\mathbf{A}\delta^k) / (\mathbf{A}\mathbf{f}^k + \beta\, \mathbf{1}) \right\|_2^2}{\| \delta^k \|_2^2},$$

where $\delta^k = \mathbf{f}^k - \mathbf{f}^{k-1}$ and the operators $\sqrt{\cdot}, \cdot$, and $/$ are understood to be component-wise operators. This Hessian approximation yields

$$F^k(\mathbf{f}) = F(\mathbf{f}^k) + (\mathbf{f} - \mathbf{f}^k)^\top \nabla F(\mathbf{f}^k) + \frac{\alpha_k}{2} \| \mathbf{f} - \mathbf{f}^k \|_2^2 . \tag{3.19}$$

Note that

$$F^k(\mathbf{f}) = F(\mathbf{f}^k) + \frac{\alpha_2}{2} \left\| \mathbf{f} - \left(\mathbf{f}^k - \frac{1}{\alpha_k} \nabla F(\mathbf{f}^k) \right) \right\|_2^2 - \frac{2}{\alpha_k} \|\nabla F(\mathbf{f}^k)\|_2^2 = \frac{\alpha_k}{2} \left\| \mathbf{f}^k - \mathbf{s}^k \right\|_2^2 + c_k,$$

where $\mathbf{s}^k = \mathbf{f}^k - \frac{1}{\alpha_k} \nabla F(\mathbf{f}^k)$ and $c_k = F(\mathbf{f}^k) - \frac{2}{\alpha_k} \|\nabla F(\mathbf{f}^k)\|_2^2$ is a constant.

The second term, $H_p^k(\mathbf{f})$, is obtained by linearizing $H_p(\mathbf{f})$ at the current iterate $\mathbf{f}^k$, i.e., using its first-order Taylor series approximation:

$$H_p^k(\mathbf{f}) \equiv H_p(\mathbf{f}^k) + \nabla H_p(\mathbf{f}^k)^\top (\mathbf{f} - \mathbf{f}^k)$$

where the gradient of $H_p^k(\mathbf{f})$ is given by

$$\nabla H_p(\mathbf{f}) = \begin{bmatrix} \frac{\partial H_p(\mathbf{f})}{\partial |f_1|} \\ \frac{\partial H_p(\mathbf{f})}{\partial |f_2|} \\ \vdots \\ \frac{\partial H_p(\mathbf{f})}{\partial |f_n|} \end{bmatrix},$$

where

$$\frac{\partial H_p(\mathbf{f})}{\partial |f_i|} = -\frac{p|f_i|^{p-1}}{\|\mathbf{f}\|_p^p} \log |f_i|^p + \frac{p|f_i|^{p-1}}{\|\mathbf{f}\|_p^{2p}} \sum_{\ell=1}^{n} |f_\ell|^p \log |f_\ell|^p, \qquad \text{for } i = 1 \dots n.$$

Manipulating these approximations and ignoring constant terms yield a sequence of subproblems of the form

$$\mathbf{f}^{k+1} = \underset{\mathbf{f} \geq 0}{\arg\min} \quad \frac{\alpha_k}{2} \sum_{i=1}^{n} (f_i - s_i^k)^2 + \sum_{i=1}^{n} \tau (\nabla H_p(\mathbf{f}^k))_i f_i, \tag{3.20}$$

where s_i^k is the i^{th} element of the vector $\mathbf{s}^k$. Note that we can uncouple (3.20) into scalar minimization problems that have quadratic objective functions with non-negativity constraint. Thus, we can compute the minimizer $\mathbf{f}^{k+1}$ analytically. In particular, the minimizer $\mathbf{f}^{k+1}$ can be computed by solving each scalar function of the form

$$f^* = \underset{f \geq 0}{\arg\min} \quad \frac{1}{2}(f - s)^2 + \gamma f, \tag{3.21}$$

where f and s denote i-th element of the vectors $\mathbf{f}$ and $\mathbf{s}^k$ respectively and $\gamma = \frac{\tau}{\alpha_k}(\nabla H_p(\mathbf{f}^k))_i$. Then the minimum of (3.21) is given by

$$f^* = [s - \gamma]_+,$$

where a thresholding operator $[\,\cdot\,]_+ = \max\{0, \cdot\}$ is employed to get a nonnegative solution. We call this proposed approach SPIRAL-Shannon method. We note that unlike the SPIRAL-ℓ_p subproblem solution, which is obtained iteratively, the proposed SPIRAL-Shannon approach has a closed-form subproblem solution.

3.4.2 Numerical Experiments

In this study, we investigate the effectiveness and efficiency of the proposed method in two different experiments.

Experiment I

We compare our proposed method (SPIRAL-Shannon) to the existing SPIRAL-ℓ_1 [33] and SPIRAL-ℓ_p [4] methods on a one-dimensional signal. Here, our goal is to recover the true signal $\mathbf{f}$ of length 1.0×10^5 consisting of $1,500$ nonzero entries of intensity 1.5×10^4, i.e., $\mathbf{f}$ has 1.5% sparsity (see Fig. 3.8(a)). The locations of the nonzero entries were randomly generated. The observation $\mathbf{y}$ is of length 4×10^4 (see Fig. 3.8(b)) and is corrupted with 16% Poisson noise (see Fig. 3.8(c)). The system matrix $\mathbf{A}$ is a $40,000 \times 100,000$ matrix obtained from the SPIRAL toolbox [32]. All algorithms are initialized with $\mathbf{A}^\top(\mathbf{y})$ and terminate if the relative change in the iterates

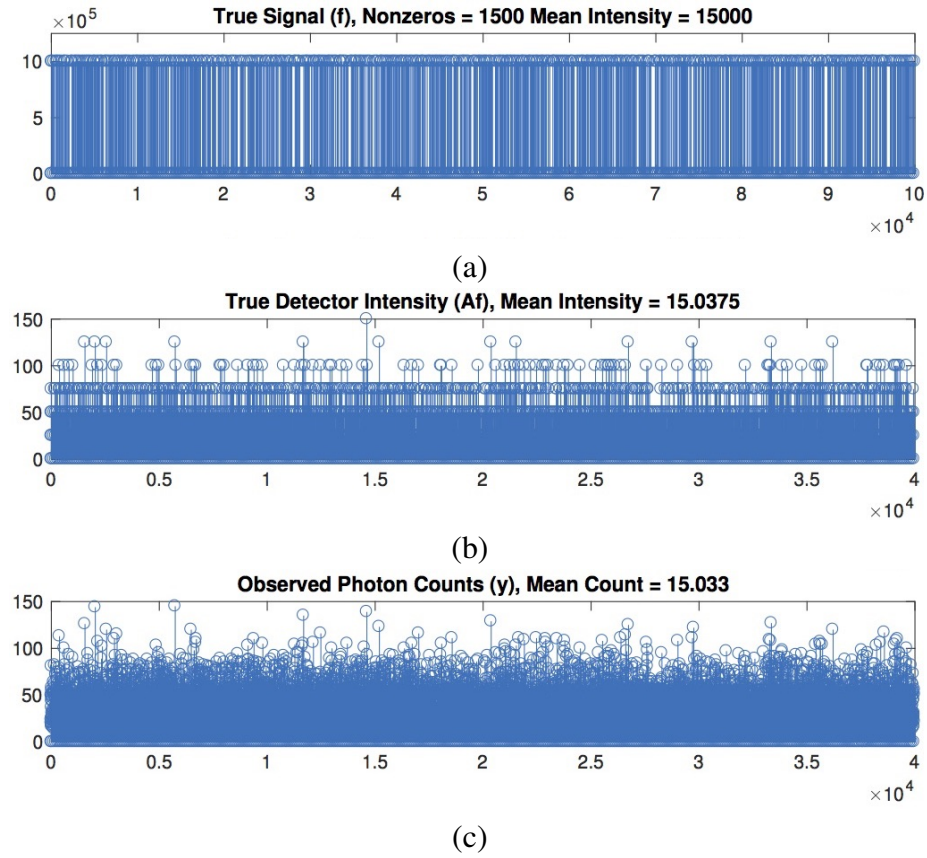


Figure 3.8: (a) The true signal **f**. (b) The true detector intensity **Af**. (c) One realization of the observed photon count **y**.

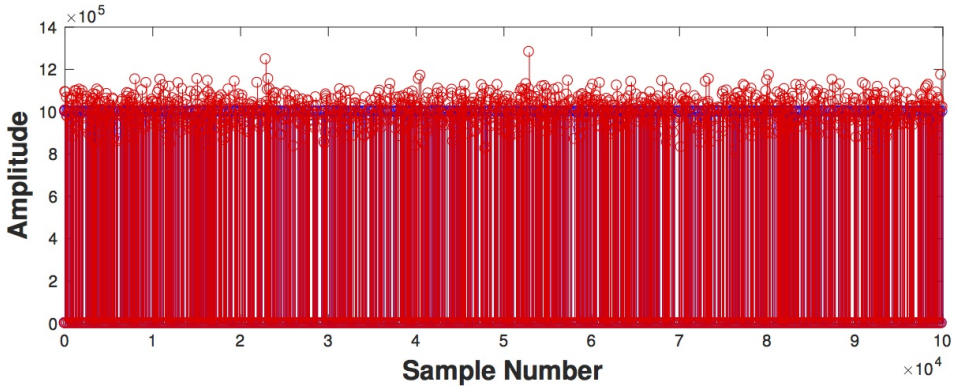


Figure 3.9: SPIRAL-Shannon reconstruction for $p = 0.2$.

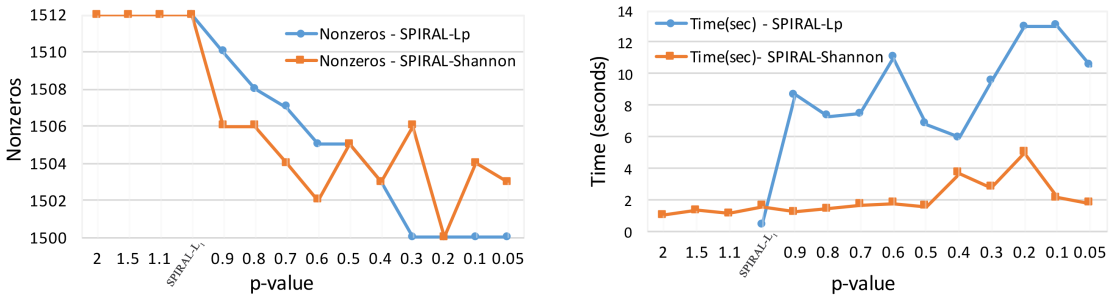


Figure 3.10: (a) Number of nonzeros in the reconstructions of ℓ_p-norm method and the proposed SPIRAL-Shannon over the p-values.(b) Computation time of the ℓ_p-norm method and SPIRAL-Shannon over the p-values.

falls below a tolerance of 1.0×10^{-8}. We ran this simulation 10 times with different Poisson noise realizations and investigated the performance of the existing SPIRAL-ℓ_p method with our proposed SPIRAL-Shannon approach.

Analysis

The SPIRAL-Shannon reconstruction is shown in Fig. 3.9. The number of nonzeros in the reconstruction of SPIRAL-Shannon (see Fig. 6.19(a)) decreases drastically after $p < 1$ and recover the exact sparsity (i.e., $1,500$) at $p = 0.2$. Even though the SPIRAL-ℓ_p converges monotonically to the exact sparsity at $p = 0.3$, it requires more computational effort than the SPIRAL-Shannon method (see Fig. 6.19(b)). On average, SPIRAL-Shannon and SPIRAL-ℓ_p methods recover the true signal with root mean square error 0.059, where the root mean square error (RMSE) is given by RMSE= $\|\widehat{\mathbf{f}} - \mathbf{f}\|_2 / \|\mathbf{f}\|_2$, where $\widehat{\mathbf{f}}$ is an estimate of the true signal $\mathbf{f}$.

Experiment II

The next application of the SPIRAL-Shannon algorithm is performed within the regime of Positron Emission Tomography (PET) image reconstruction. PET is an *in vivo* imaging tool that allows the user to record spacial and dynamical biological processes [11]. Images are reconstructed by recording rapid changes in radio labelled tracers. The quality of these observations is limited by the resolution of the detector as well as the low photon count density that is inherent to the PET projection data [11, 57, 67]. The result is an ill-posed inverse problem where in the arrival of photons at the detectors are more appropriately modeled by Poisson statistics. The quality of PET images have been greatly improved by using penalized maximum likelihood functions [57, 67].

In this experiment, PET scans were simulated for a General Electric Discovery Seek and Treat (GE DST) whole-body PET scanner using a Zubal head phantom with 111×111 pixels in the axial plane and a 1mm isotropic pixel size [72]. The 2D sinogram contains 249 radial bins and 210 angular projections. The PET phantom image was forward projected using the system matrix provided by the Michigan Image Reconstruction Toolbox (MIRT) to generate

	Number of iterations	CPU Time (sec.)
EM Algorithm	84	1.03
SPIRAL-ℓ_p	386	6.83
SPIRAL-Shannon	565	10.42

Table 3.2: A comparison of the number of iterations and the total CPU time required to converge to the results in Fig. 3.11.

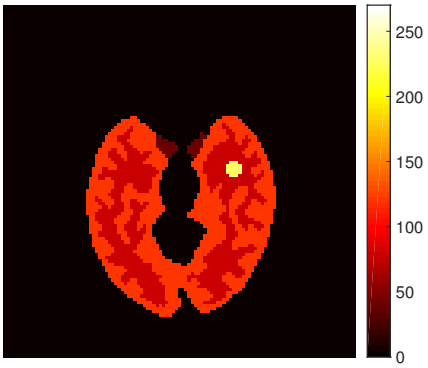
the noise-free projection data [26]. A uniform sinogram with a value equal to 20% of the mean of the noise-free sinogram was added to simulate background events (i.e., randoms and scatters). Along with brain tissue matter the Zubal phantom includes a 15mm diameter tumor (Fig. 3.11a). Finally, a noisy realization was generated by introducing Poisson noise to the sinogram with the expected total number of events set to 200 thousand. We show the effectiveness of SPIRAL-Shannon by comparing it to the previously mentioned SPIRAL-ℓ_p and the conventional Expectation-Maximization (EM) algorithm, which is an iterative method that is suited to emission and transmission image reconstruction under the assumption of a Poisson statistic [47]. Our metric for comparison is the Mean Squared Error (MSE) $= \|\widehat{\mathbf{f}} - \mathbf{f}\|_2^2/n$, where $\widehat{\mathbf{f}}$ is an estimate of $\mathbf{f}$ and $n = 12,321$ is the number of elements in $\mathbf{f}$.

Analysis

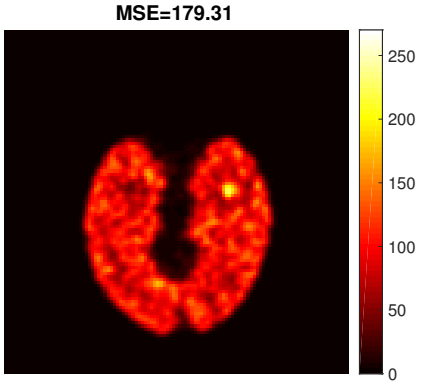
In order to obtain reconstructions for the SPIRAL-ℓ_p and SPIRAL-Shannon method it is necessary to tune τ and p to obtain the optimum MSE. For the SPIRAL-ℓ_p method $\tau = 1 \times 10^{-6}$ and $p = 0.84$ produced the results in Fig. 3.11c. The SPIRAL-Shannon method required values of $\tau = 0.001$ and $p = 1.99$ to produce the results in Fig. 3.11d. Comparing the previously mentioned algorithms with the commonly used EM algorithm (Fig. 3.11b), the SPIRAL-ℓ_p and the SPIRAL-Shannon algorithms were more effective at reconstructing the truth. The images also show that SPIRAL-Shannon is comparable to the SPIRAL-ℓ_p method. Table 3.2 shows a comparison of the efficiency of the algorithms. Although the EM Algorithm converges at a faster rate when compared to the other two algorithms, it appears to be less accurate than the SPIRAL algorithms.

3.4.3 Conclusion

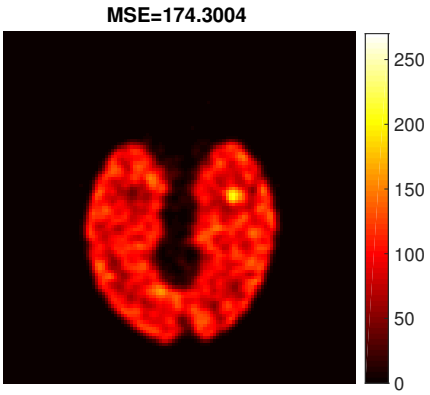
In this work, we proposed a novel Poisson intensity reconstruction method by introducing a sparsity promoting Shannon entropy penalizer to the photon-limited imaging problem. Unlike previous nonconvex methods, the proposed method has a closed-form expression for the subproblem solution. The proposed SPIRAL-Shannon algorithm demonstrated its accuracy in solving a 1D signal and a 2D PET image reconstruction problem. Its performance is comparable to recent methods.



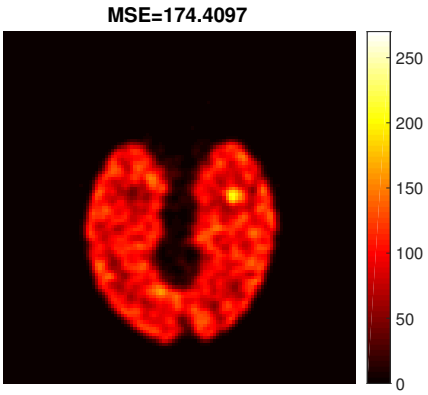
(a) **Truth**



(b) **EM-Reconstruction**



(c) **SPIRAL-ℓ_p Reconstruction**



(d) **SPIRAL-Shannon Reconstruction**

Figure 3.11: Reconstructions of the simulated PET scan data. The Truth is taken as the Zubal head phantom (Fig. 3.11a). The reconstructions for each respective algorithm are shown in Fig. 3.11b - Fig. 3.11d along with the Mean Squared Error (MSE).

3.5 Parameter Tuning Using Asynchronous Parallel Pattern Search in Sparse Signal Reconstruction

As we have seen in sections 3.3 and 3.4, parameter tuning in signal recovery problems involves the judicious selection of parameters that influence not only the problem model, but the solution as well. For example, in compressed sensing and in general sparse signal recovery problems, regularization parameters balance least-squares data fidelity terms with sparsity-promoting penalty terms. More recently, there has been evidence that non-convex ℓ_p quasi-norm minimization, where $0 < p < 1$, leads to an improvement in reconstruction over existing methods that use convex regularization. In this work, we propose a computationally efficient method for parameter tuning in a parallel but asynchronous environment using derivative-free optimization. To demonstrate its effectiveness, we apply the proposed approach to existing methods for photon-limited imaging. This work is based on the paper by DeGuchy and Marcia [19].

Related Methods

The problem of estimating multiple parameters arises in many other fields. For example, kernel parameters of a support vector machine often have to be estimated [18, 34, 69]. Hyper-parameter tuning arises in stochastic gradient descent (SGD) methods for machine learning [24, 41, 71]. Automatic parameter selection methods have been previously proposed [43, 49, 70], and pattern search methods are well-known in literature [21, 35, 45, 51, 65]. This work explores *asynchronous* parallel derivative-free approaches for parameter tuning.

3.5.1 Poisson Problem Formulation

In this section we will motivate the need for parameter tuning within the context of photon-limited imaging. Specifically, we will review the formulation of the Poisson reconstruction problem summarized in Section 3.2 as the ℓ_p-norm penalized Poisson log-likelihood function. This will provide the bases for the SPIRAL-ℓ_p algorithm [4].

We begin with the inhomogeneous Poisson process

$$\mathbf{y} \sim \text{Poisson}(\mathbf{A}\mathbf{f}^*),$$

where $\mathbf{y} \in \mathbb{Z}_+^m$, $\mathbf{f}^* \in \mathbb{R}_+^n$ is the true signal and $\mathbf{A}_+^{m\times n}$ is the system matrix projecting the true signal to the detector space [61]. Our interest is in the recovery of $\mathbf{f}^*$ and since the Poisson parameter is unknown, we use the maximum likelihood principle to maximize the probability of observing the vector y. Specifically, if $\mathbf{f}^*$ is known to contain a few non-zero entries, we can use a sparsity promoting penalty term to regularize the Poisson likelihood. The introduction of a penalty term such as the ℓ_1-norm [33] or the ℓ_p-norm [4] requires the tuning of parameters prior to optimization in order to control the severity of the sparsity. The generalized ℓ_p-norm sparsity-promoting Poisson reconstruction problem can be written as the following constrained minimization problem:

$$\begin{aligned} \hat{\mathbf{f}} \;=\; & \underset{\mathbf{f}\in\mathbb{R}^n}{\arg\min} \quad \Phi(\mathbf{f}) \equiv F(\mathbf{f}) + \tau\|\mathbf{f}\|_p^p \\ & \text{subject to } \mathbf{f} \geq 0, \end{aligned} \tag{3.22}$$

where

$$F(\mathbf{f}) = \mathbb{1}^\top \mathbf{A}\mathbf{f} - \sum_{i=1}^{m} y_i \log(e_i^\top \mathbf{A}\mathbf{f} + \beta) \tag{3.23}$$

is the negative Poisson log-likelihood function, $\mathbb{1}$ is the m-vector of ones, e_i is the i-th column of the $m \times m$ identity matrix and $\beta > 0$ (typically $\beta \ll 1$). We take particular note of the penalty term in (3.22). Here the parameter $\tau > 0$ serves to control the weight of the penalty and $0 < p < 1$ acts as a bridge from the ℓ_0 counting norm to the convex ℓ_1-norm. The solution to (3.22) is computed from a sequence of quadratic models using a second-order Taylor approximation to $F(\mathbf{f})$ at each iteration [4]. However, the authors assume that p and τ are known *a priori*.

3.5.2 Cross-Validation

In order to determine the pair of parameters p and τ which return the optimal $\hat{\mathbf{f}}$ that well approximates $\mathbf{f}^*$, it is necessary to define the criteria by which the optimal pair is chosen. This is difficult in the absence of information about the true signal $\mathbf{f}^*$. Typically the observation vector $\mathbf{y} \in \mathbb{R}^m$ and the system matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ are known elements of the problem. Our approach partitions $\mathbf{y}$ and $\mathbf{A}$ in the following manner:

$$\underset{m\times 1}{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_{\text{train}} \\ \mathbf{y}_{\text{test}} \end{bmatrix} \begin{matrix} \}r \\ \}m-r \end{matrix} \quad \text{and} \quad \underset{m\times n}{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{\text{train}} \\ \mathbf{A}_{\text{test}} \end{bmatrix} \begin{matrix} \}r \\ \}m-r \end{matrix}$$

where $\mathbf{y}_{\text{train}} \in \mathbb{R}^r$, $\mathbf{y}_{\text{test}} \in \mathbb{R}^{m-r}$, $\mathbf{A}_{\text{train}} \in \mathbb{R}^{r \times n}$ and $\mathbf{A}_{\text{test}} \in \mathbb{R}^{(m-r)\times n}$, with $r < m$. For a candidate pair of parameters (p, τ), we use $\mathbf{y}_{\text{train}}$ and $\mathbf{A}_{\text{train}}$ to compute $\hat{\mathbf{f}}_{\text{test}} \in \mathbb{R}^n$ using the SPIRAL-ℓ_p algorithm. The signal reconstruction is then used to compute $\hat{\mathbf{y}}_{\text{test}} = \mathbf{A}_{\text{test}}\hat{\mathbf{f}}_{\text{test}}$. Finally, we compute

$$\text{RMSE}_{\text{test}} = \frac{\|\hat{\mathbf{y}}_{\text{test}} - \mathbf{y}_{\text{test}}\|_2}{\|\mathbf{y}_{\text{test}}\|_2}.$$

The choice of $\text{RMSE}_{\text{test}}$ as a metric for choosing the parameters p and τ is motivated by the fact that it behaves similarly to the $\text{RMSE}(\hat{\mathbf{f}})$ describing the difference between the true signal $\mathbf{f}^*$ and the reconstruction $\hat{\mathbf{f}}$ (see Fig. 3.12). We formulate the parameter search as the following constrained optimization problem:

$$\begin{aligned} \underset{p,\tau\in\mathbb{R}}{\text{minimize}} \quad & \text{RMSE}_{\text{test}}(p, \tau) \\ \text{subject to} \quad & 0 < p < 1, \\ & b_L \leq \tau \leq b_U, \end{aligned} \tag{3.24}$$

where $\text{RMSE}_{\text{test}}(p, \tau)$ is computed as previously stated for a given parameter pair and $0 < b_L < b_U$ are reasonable lower(b_L) and upper bounds(b_U) for the regularization parameter τ. This approach is based on cross-validation techniques used in statistical inference and machine learning to test how well a predictive model performs [42]. Typically the data set used in these regimes is randomly divided into a test set and a validation set. In the Poisson reconstruction model we must maintain the structure of the data $\mathbf{y}$ limiting the random nature of traditional cross-validation. These techniques also involve input data used to train the model and output data used to test the model. Our problem uses the observation vector $\mathbf{y}$ as both the input data and the output data.

3.5.3 Asynchronous Parallel Pattern Search

Solving (3.24) using fast gradient-based optimization techniques is not practical because how to compute (or even estimate) the gradient of $\text{RMSE}_{\text{test}}(p, \tau)$ is not clear. Instead, we propose

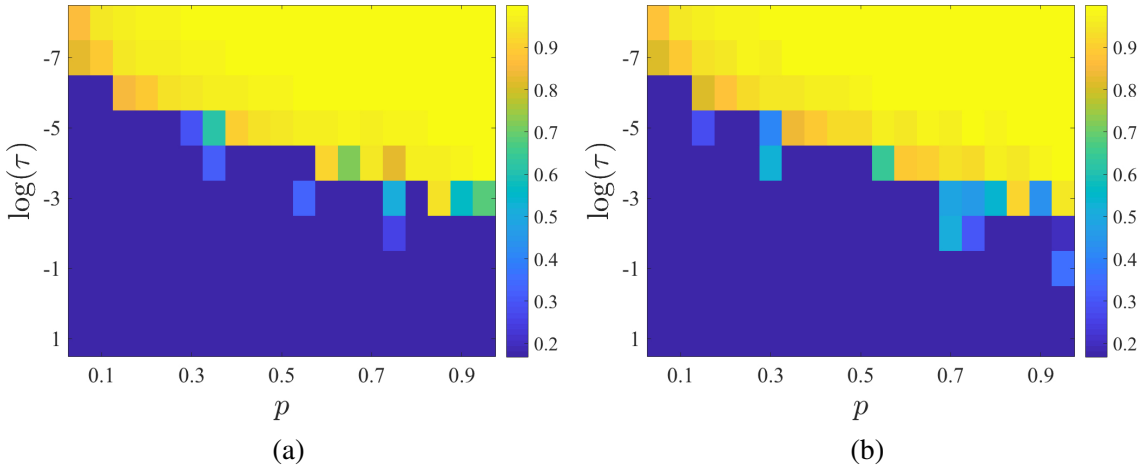


Figure 3.12: A comparison of the behavior of (a) the RMSE for $\hat{\mathbf{y}}_{\text{test}}$ ($\text{RMSE}_{\text{test}}$) and (b) the RMSE for $\widehat{\mathbf{f}}$ ($\text{RMSE}(\widehat{\mathbf{f}})$) for various combinations of the parameters p and τ. Note the near-consistent agreement between $\text{RMSE}_{\text{test}}$ and $\text{RMSE}(\widehat{\mathbf{f}})$.

using a derivative-free approach to minimizing the problem. Instead of an exhaustive grid-search approach or even standard derivative-free methods, we take advantage of the availability of multiple processors that *asynchornously* searches for the optimal parameters. Initially we create a coarse grid of possible values for p and τ [37]. As an alternative to computing $\text{RMSE}_{\text{test}}(p,\tau)$ for every possible combination on the grid (which can be computationally exhaustive) we randomly sample a percentage of combinations from the original grid (see Fig. 3.13) and compute $\text{RMSE}_{\text{test}}(p,\tau)$ for the resultant candidates [40]. We compare the $\text{RMSE}_{\text{test}}$ values and choose the pairs (p,τ) with the lowest associated values. The number of pairs chosen is based on the order of magnitude of the $\text{RMSE}_{\text{test}}$ with a minimum of 5 pairs and a maximum of 10. These candidates will be further refined as they form the initialization points for the HOPSPACK framework, which we explain next.

HOPSPACK (Hybrid Optimization Parallel Search PACKage) [55] is an open source frame

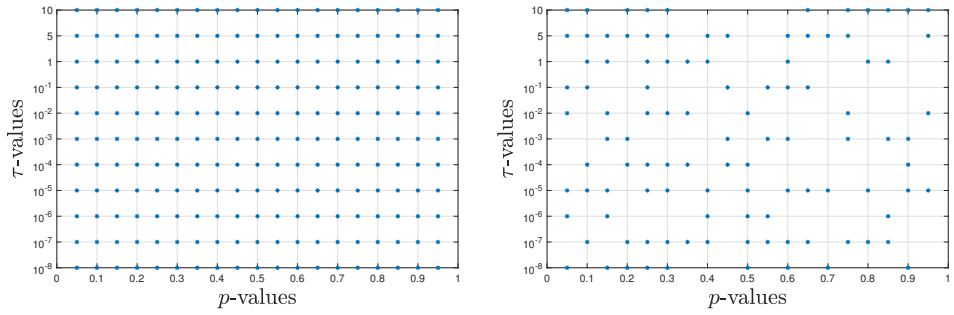
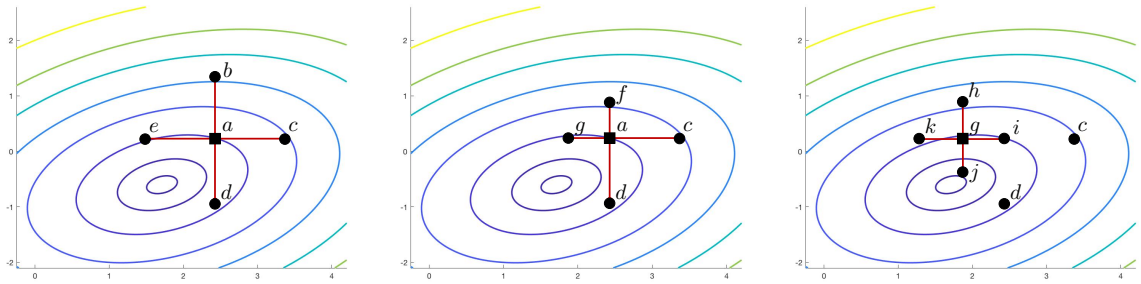


Figure 3.13: Grid of potential parameter pairs (p,τ). (a) The full grid of 190 potential parameter values where $0 < p < 1$ and $10^{-8} \leq \tau \leq 10$. (b) The reduction of candidates by randomly choosing 50% of the initial pairs.



(a) (b) (c)

Figure 3.14: An illustration of an asynchronous parallel search implementation of the Generating Set Search method. (a) An initial (parent) point a and points ($\{b, c, d, e\}$) within a trial step along the standard basis directions are generated. (b). Suppose b and e required less time to evaluate than c and d and both b and e did not improve the objective function value at a, then the trial step is decreased and new trial points f and g are generated while c and d continue to be evaluated. (c) From the contour plots, the objective value at g is lower than at a; therefore g becomes the new parent and new trial points are generated based on the current trial step.

work that enables users to implement derivative-free algorithms in parallel [30, 31, 36, 44]. This work takes advantage of its asynchronous pattern search implementation of GSS (Generating Set Search) [45]. The GSS algorithm begins with an initial point and step size and generates trial points along the available axes in the negative and positive direction. The trial points are evaluated with two possible outcomes. If a trial point improves upon the value returned by the parent (current best point), then it becomes the parent and the process is reimplemented. If a trial point does worse, then the step size is decreased and a new trial point is determined. The algorithm terminates when the step size in all directions has reached a user defined step tolerance. (See Fig. 3.14 for an illustration.)

Implementing GSS with the HOPSPACK framework provides us with two advantages. The first is that the implementation is asynchronous, meaning that for a given iteration it does not wait for all of the trial point evaluations to be completed. HOPSPACK allows the algorithm to continue testing trial points with a partial set of results from the previous iteration. This characteristic makes the implementation ideal for parallelism and beneficial in the case where function evaluations in certain regions take longer. The latter is of significant interest when computing $\text{RMSE}_{\text{test}}$ as the SPIRAL-ℓ_p algorithm computational time may vary based on the choice of p and τ. The implementation will then continue to search for better trial points in the faster region while the slower region computes its value. The second advantage is that the asynchronous implementation of GSS inherits the properties of convergence of the synchronous implementation [45].

Using the previously computed candidates from the randomized grid search as initial points in the HOPSPACK implementation of GSS, we further refine our choice of parameters until we have the optimal pair (p, τ). The pair is then used along with $\mathbf{y}$ and $\mathbf{A}$ in the SPIRAL-ℓ_p algorithm to compute the signal reconstruction $\hat{\mathbf{f}}$.

3.5.4 Numerical Experiments

In this section we implement the previously discussed method for finding the optimal parameter pair (p, τ). We perform 10 similar experiments where we generated measurements from 10 different signals and 10 different noise realizations for each signal. In each experiment, the true signal $\mathbf{f}^*$ has a length of 100,000 with 1,500 nonzero entries and a mean intensity of 15,000. The observation vector $\mathbf{y}$ has a length of 40,000 with a mean intensity much lower than that of $\mathbf{f}^*$ (see Fig. 3.15 for an example of one Poisson realization of one particular signal).

The goal of each experiment is to construct the best approximation of the true signal $\mathbf{f}^*$ from the observation vector $\mathbf{y}$. Specifically we seek to capture the support (number of non zeros and their location) of $\mathbf{f}^*$ without any prior information on $\mathbf{f}^*$. We initialize the method by creating the vector $\mathbf{y}_{\text{train}}$ using 90% of the length of $\mathbf{y}$. This results in the vector $\mathbf{y}_{\text{train}}$ having a length of 36,000 and $\mathbf{y}_{\text{test}}$ having a length of 4,000. The initial coarse grid was created using the bounds $0 < p < 1$ and $1.0 \times 10^{-8} < \tau < 1.0 \times 10^{3}$. The grid points of p values were uniformly spaced in increments of .05 while the τ values were logarithmically spaced. The result was a grid of 285 possible combinations. Fifty percent of the combinations were then randomly chosen. All computations were performed on the MERCED Cluster which consists of 84 compute nodes with a total of 1876 cores at 2301 MHz with a total capacity of approximately 60 TFLOPS. The $\text{RMSE}_{\text{test}}$ values were computed using the MATLAB implementation of the SPIRAL-ℓ_p algorithm. The best candidates from the coarse grid search were chosen by finding the pairs with lowest $\text{RMSE}_{\text{test}}$ values. The pairs with the same initial two significant digits were chosen resulting in a group of 5-10 candidates that were used to initialize the HOPSPACK framework. Searches were performed around each parameter pair by the HOPSPACK multi-threaded executable. Each computation was implemented on a cluster node across 20 cores. The function calls (computation of $\text{RMSE}_{\text{test}}$) were passed to MATLAB and the evaluations were parsed by the HOPSPACK framework. On average each node took approximately 10-20 minutes of CPU time to perform approximately 70-90 function evaluations before converging to a solution. Finally, the optimal pair from all nodes was determined and used to calculate $\hat{\mathbf{f}}$.

In all 100 experiments the true support or location of the nonzero elements (1500) were recovered. In some of the experiments slightly more nonzero elements than the support were also recovered. These extra entries are reflected in the averages shown in Table 1. In the most extreme case 1510 nonzero entries were recovered while most reconstructions captured the true support exactly. This consistency was exhibited across different realizations as well as across different signals. Table 1 also shows the range of average RMSE values for $\hat{\mathbf{f}}$. These values are consistent with those presented in literature [4] where the true signal $\mathbf{f}^*$ was used to tune the parameters. We also present in Table 1 the average computational times and the average number of pair (p, τ) evaluations per experiment.

3.5.5 Conclusion

In this work, we created the metric $\text{RMSE}_{\text{test}}$ as a means for validating the parameter choice in the ℓ_p-norm Poisson reconstruction problem. The creation of $\text{RMSE}_{\text{test}}$ is particularly useful in that it assumes that nothing is known about the true signal $\mathbf{f}^*$. This measure of parameter choice coupled with the asynchronous parallelization of GSS implemented by HOPSPACK, accurately, efficiently and consistently produced the support of the true signal and RMSE values comparable to those produced by manual tuning where information about the true signal is known.

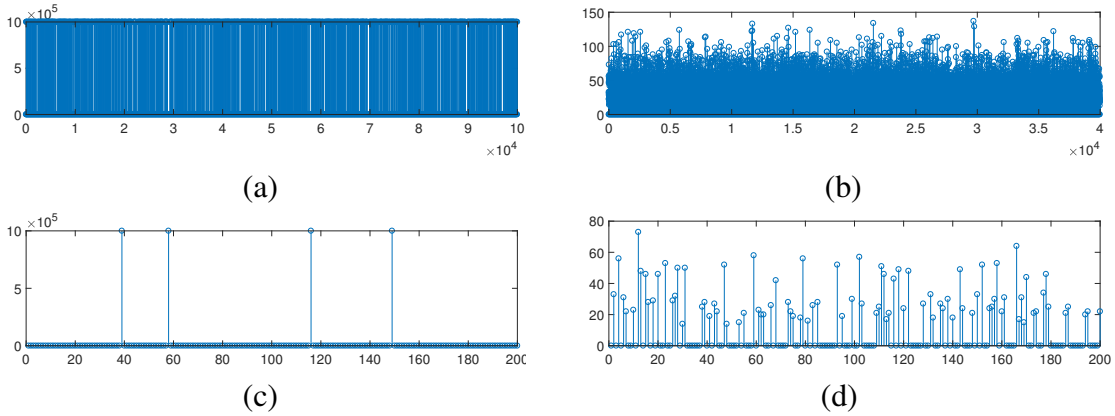


Figure 3.15: Experimental setup: (a) True signal $\mathbf{f}^*$ of size 100,000 with a mean intensity of 15,000. (b) Observation vector $\mathbf{y}$ of size 40,000 corrupted with Poisson noise. (c) The first 200 entries of $\mathbf{f}^*$. (d) The first 200 entries of $\mathbf{y}$.

Exp.	RMSE	Non-zeros	CPU	Evals.
1	0.0600	1501.0	842.48	164.91
2	0.0602	1500.4	595.03	144.42
3	0.0602	1500.4	708.59	167.74
4	0.0602	1501.7	483.54	92.48
5	0.0604	1501.7	484.32	138.31
6	0.0611	1501.7	382.20	116.71
7	0.0597	1501.1	397.94	108.62
8	0.0600	1500.6	433.11	121.29
9	0.0602	1500.5	315.77	96.01
10	0.0600	1500.9	326.40	100.22

Table 3.3: Average RMSE ($\|\mathbf{f}^* - \hat{\mathbf{f}}\|_2/\|\mathbf{f}^*\|$), number of non-zeros, CPU time (in seconds), and number of pairs (p, τ) evaluated for 10 different realizations of 10 different signals.

3.6 Summary of Contribution

In this chapter, we proposed three novel algorithms addressing sparse signal recovery in the context of photon-limited imaging: (1) An approach to fluorescence lifetime imaging where in we use aprori knowledge about the sparsity and intensity of the intended signal in order to speed up and improve signal estimation, (2) A non-convex Shannon entropy regularization method. This work introduced the weighted Shannon Entropy function in order to promote sparsity in the reconstructed signal, (3) an asynchronous parallel pattern search method in order to tune the hyperparamters involved in the photon-limited recovery problem. The objective of the pattern search was to find the optimal parameter p for the p-norm being used as well as the intensity of the sparsity controlled by τ.

Bibliography

[1] L. Adhikari. “Nonconvex Sparse Recovery Methods.” *UC Merced: Applied Mathematics* http://escholarship.org/uc/item/2099g1s6 (2017).

[2] L. Adhikari, A. D. Kim, and R. F. Marcia. “Sparse reconstruction for fluorescence lifetime imaging microscopy with Poisson noise”. *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Dec. 2016, pp. 262–266. DOI: 10.1109/GlobalSIP.2016.7905844.

[3] L. Adhikari and R. F. Marcia. “Bounded sparse photon-limited image recovery”. *2016 IEEE International Conference on Image Processing (ICIP)*. Sept. 2016, pp. 3508–3512.

[4] L. Adhikari and R. F. Marcia. “Nonconvex relaxation for Poisson intensity reconstruction”. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 1483–1487.

[5] L. Adhikari. “Nonconvex sparse recovery methods”. PhD thesis. UC Merced, 2017.

[6] L. Adhikari et al. “Non-convex Shannon entropy for photon-limited imaging”. *Wavelets and Sparsity XVII*. Vol. 10394. International Society for Optics and Photonics. 2017, p. 103940L.

[7] D. Álvarez, P. Medina, and M. Moscoso. “Fluorescence lifetime imaging from time resolved measurements using a shape-based approach”. *Opt. Express* 17.11 (May 2009), pp. 8843–8855.

[8] A. Antoniadis and J. Bigot. “Poisson inverse problems”. *The Annals of Statistics* (2006), pp. 2132–2158.

[9] S. R. Arridge. “Optical tomography in medical imaging”. *Inverse Problems* 15.2 (1999), R41.

[10] S. R. Arridge and J. C. Schotland. “Optical tomography: forward and inverse problems”. *Inverse Problems* 25.12 (2009), p. 123010.

[11] B. Bai, Q. Li, and R. M. Leahy. “Magnetic resonance-guided positron emission tomography image reconstruction”. *Seminars Nucl Med.* (2013), pp. 30–44.

[12] J. Barzilai and J. M. Borwein. “Two-Point Step Size Gradient Methods”. *IMA J. Numeri. Anal* 8.1 (1988), pp. 141–148.

[13] J. Barzilai and J. M. Borwein. “Two-point step size gradient methods.” *IMA J. Numeri. Anal* (1988), 8(1):141–148.

[14] C. R. Berger et al. “Application of compressive sensing to sparse channel estimation”. *IEEE Communications Magazine* 48.11 (2010), pp. 164–174.

[15] S. Bloch et al. “Whole-body fluorescence lifetime imaging of a tumor-targeted near-infrared molecular probe in mice”. *Journal of Biomedical Optics* 10.5 (2005), pp. 054003-054003-8.

[16] R. E. Carrillo, J. D. McEwen, and Y. Wiaux. “PURIFY: a new approach to radio-interferometric imaging”. *Monthly Notices of the Royal Astronomical Society* 439.4 (2014), pp. 3591–3604.

[17] R. E. Carrillo et al. “Robust compressive sensing of sparse signals: a review”. *EURASIP Journal on Advances in Signal Processing* 2016.1 (2016), p. 108.

[18] O. Chapelle et al. “Choosing multiple parameters for support vector machines”. *Machine Learning* 46.1 (2002), pp. 131–159.

[19] O. DeGuchy and R. F. Marcia. “Parameter tuning using asynchronous parallel pattern search in sparse signal reconstruction”. *Wavelets and Sparsity XVIII*. Vol. 11138. International Society for Optics and Photonics. 2019, p. 111381I.

[20] O. DeGuchy et al. “Photon-Limited fluorescence lifetime imaging microscopy signal recovery with known bounds”. *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE. 2017, pp. 1–5.

[21] J. E. Dennis Jr and V. Torczon. “Direct search methods on parallel machines.” *SIAM Journal on Optimization* 1.4 (1991), pp. 448–474.

[22] D. L. Donoho. “Compressed sensing”. *IEEE Transactions on information theory* 52.4 (2006), pp. 1289–1306.

[23] D. L. Donoho et al. “Maximum entropy and the nearly black object”. *Journal of the Royal Statistical Society. Series B (Methodological)* (1992), pp. 41–81.

[24] J. C. Duchi, E. Hazan, and Y. Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159. URL: `http://dl.acm.org/citation.cfm?id=2021068`.

[25] M. Elangovan, R. N. Day, and A. Periasamy. “Nanosecond fluorescence resonance energy transfer-fluorescence lifetime imaging microscopy to localize the protein interactions in a single living cell”. *Journal of Microscopy* 205.1 (2002), pp. 3–14.

[26] J. Fessler. *Michigan Image Reconstruction Toolbox*. `https://web.eecs.umich.edu/~fessler/code/`.

[27] M. A. T. Figueiredo and J. M. Bioucas-Dias. “Restoration of Poissonian images using alternating direction optimization”. *IEEE transactions on Image Processing* 19.12 (2010), pp. 3133–3145.

[28] M. Figueiredo and et al. “Gradient Projection for Sparse Reconstruction”. *IEEE J. of Selected Top.in Sig. Proc.* 1.4 (2007), pp. 586–597.

[29] T. W. J. Gadella, T. M. Jovin, and R. M. Clegg. “Fluorescence lifetime imaging microscopy (FLIM): spatial resolution of microstructures on the nanosecond time scale”. *Biophysical chemistry* 48.2 (1993), pp. 221–239.

[30] G. A. Gray and T. G. Kolda. “Algorithm 856: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization”. *ACM Transactions on Mathematical Software (TOMS)* 32.3 (2006), pp. 485–507.

[31] J. D. Griffin, T. G. Kolda, and R. M. Lewis. “Asynchronous parallel generating set search for linearly-constrained optimization”. *SIAM Journal on Scientific Computing* 30 (2008), pp. 1892–1924.

[32] Z. T. Harmany, R. F. Marcia, and R. M. Willett. *The Sparse Poisson Intensity Reconstruction ALgorithms (SPIRAL) Toolbox*. `http://drz.ac/code/spiraltap/`.

[33] Z. T. Harmany, R. F. Marcia, and R. M. Willett. “This is SPIRAL-TAP: Sparse Poisson Intensity Reconstruction ALgorithms; Theory and Practice”. *IEEE Trans. on Image Processing* 21.3 (2012), pp. 1084–1096.

[34] T. Hastie et al. “The entire regularization path for the support vector machine”. *Journal of Machine Learning Research* 5.Oct (2004), pp. 1391–1415.

[35] R. Hooke and T. A. Jeeves. ““Direct Search” Solutiton of Numerical and Statistical Problems”. *Journal of the ACM* 8.2 (2010), pp. 212–229.

[36] P. D. Hough, T. G. Kolda, and V. J. Torczon. “Asynchronous parallel pattern search for nonlinear optimization”. *SIAM Journal on Scientific Computing* 23.1 (2001), pp. 134–156.

[37] C. Hsu, C. Chang, and C. Lin. *A practical guide to support vector classification*. Tech. rep. Department of Computer Science, National Taiwan University, 2003.

[38] S. Huang, D. N. Tran, and T. D. Tran. “Sparse signal recovery based on nonconvex entropy minimization”. *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 3867–3871.

[39] H. Ishikawa-Ankerhold, R. Ankerhold, and G. Drummen. “Advanced Fluorescence Microscopy Techniques – FRAP, FLIP, FLAP, FRET and FLIM”. *Molecules* 17.4 (2012), p. 4047.

[40] J. Bergstra and Y. Bengio. “Random search for hyper-paramter optimization”. *Journal of Machine Learning Research* 9 (2012), pp. 281–305.

[41] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. *CoRR* abs/1412.6980 (2014). URL: `http://arxiv.org/abs/1412.6980`.

[42] R. Kohavi. “A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection”. *Proceedings of the 14th International Joint Conference on Articial Intelligence - Volume 2*. Aug. 1995, pp. 1137–1143.

[43] R. Kohavi and G. H. John. “Automatic parameter selection by minimizing estimated error”. *Machine Learning Proceedings 1995*. 1995, pp. 304–312.

[44] T. G. Kolda. “Revisiting asynchronous parallel pattern search for nonlinear optimization”. *SIAM Journal on Optimization* 16.2 (2005), pp. 563–586.

[45] T. G. Kolda, R. M. Lewis, and V. Torczon. “Optimization by direct search: New perspectives on some classical and modern methods”. *SIAM Review* 45.3 (2003), pp. 385–482.

[46] J. R. Lakowicz. *Principles of fluorescence spectroscopy*. Springer Science & Business Media, 2013.

[47] K. Lange and R. Carson. “EM reconstruction algorithms for emission and transmission tomography”. *Journal of Computer Assisted Tomography* 8 (1984), pp. 306–316.

[48] R. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Classics in Applied Mathematics. SIAM, Society for Industrial and Applied Mathematics, July 2007. ISBN: 9780898716290.

[49] W.-Y. Loh. “Classification and regression trees”. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1 (2011), pp. 14–23.

[50] M. Lustig, D. Donoho, and J. M. Pauly. “Sparse MRI: The application of compressed sensing for rapid MR imaging”. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine* 58.6 (2007), pp. 1182–1195.

[51] J. A. Nelder and R. Mead. “A simplex method for function minimization”. *The Computer Journal* 7.4 (1965), pp. 308–313.

[52] R. D. Nowak and E. D. Kolaczyk. “A statistical multiscale framework for Poisson inverse problems”. *IEEE Transactions on Information Theory* 46.5 (Aug. 2000), pp. 1811–1825. ISSN: 0018-9448. DOI: 10.1109/18.857793.

[53] J. Nunez and J. Llacer. “A general Bayesian image reconstruction algorithm with entropy prior. Preliminary application to HST data”. *Publications of the Astronomical Society of the Pacific* 105.692 (1993), p. 1192.

[54] T. Oida, Y. Sako, and A. Kusumi. “Fluorescence lifetime imaging microscopy (flimscopy). Methodology development and application to studies of endosome fusion in single cells”. *Biophysical Journal* 64.3 (1993), pp. 676–685.

[55] T. D. Plantenga. *HOPSPACK 2.0 User Manual*. Tech. rep. SAND2009-6265. Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Oct. 2009.

[56] L. C. Potter et al. “Sparsity and compressed sensing in radar imaging”. *Proceedings of the IEEE* 98.6 (2010), pp. 1006–1020.

[57] J. Qi and R. Leahy. “Iterative reconstruction techniques in emission computed tomography”. *Physics in Medicine and Biology* (2006), R541–R578.

[58] A. Sawatzky et al. “EM-TV methods for inverse problems with Poisson noise”. *Level Set and PDE Based Reconstruction Methods in Imaging*. Springer, 2013, pp. 71–142.

[59] C. E. Shannon. “A mathematical theory of communication”. *Bell system technical journal* 27 (1948).

[60] J. Skilling and R. K. Bryan. “Maximum entropy image reconstruction: general algorithm”. *Monthly notices of the royal astronomical society* 211.1 (1984), pp. 111–124.

[61] D. L. Snyder and M. Miller. *Random Point Processes in Time and Space*. Springer, 1991.

[62] S. R. Swift and L. Trinkle-Mulcahy. “Basic principles of FRAP, FLIM and FRET”. *Proc R Microsc Soc.* 39 (2004), pp. 3–10.

[63] T. Hohage and F. Werner. “Inverse problems with Poisson data: statistical regularization theory, applications and algorithms”. *Inverse Problems* 32.9 (2016), p. 093001. URL: http://stacks.iop.org/0266-5611/32/i=9/a=093001.

[64] E. Terpetschnig and D. M. Jameson. “Fluorescence lifetime”. *ISS Technical Note* (2005).

[65] V. Torczon. “On the Convergence of Pattern Search Algorithms”. *SIAM Journal on Optimization* 7.1 (1997), pp. 1–25. DOI: 10.1137/S1052623493250780. URL: https://doi.org/10.1137/S1052623493250780.

[66] P. J. Verveer, A. Squire, and P. I. H. Bastiaens. “Global analysis of fluorescence lifetime imaging microscopy data”. *Biophysical Journal* 78.4 (2000), pp. 2127–2137.

[67] G. Wang and J. Qi. “PET image reconstruction using kernal method”. *IEEE transactions on medical imaging 34.1*. IEEE. 2015, pp. 61–71.

[68] L. V. Wang and H. Wu. *Biomedical Optics: Principles and Imaging*. John Wiley & Sons, 2012.

[69] K.-P. Wu and S.-D. Wang. “Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space”. *Pattern Recognition* 42.5 (2009), pp. 710–717.

[70] L. Yu and H. Liu. “Feature selection for high-dimensional data: A fast correlation-based filter solution”. *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 856–863.

[71] S. Zhang, A. E. Choromanska, and Y. LeCun. “Deep learning with elastic averaging SGD”. *Advances in Neural Information Processing Systems*. 2015, pp. 685–693.

[72] I. Zubal et al. “Computerized three-dimensional segmented human anatomy”. *Medical Physics* 21 (1994), pp. 299–302.

Chapter 4

Deep Learning

The algorithms presented in previous chapters rely on *a priori* knowledge of the signal being processed e.g. sparsity and Poisson noise. This motivated the formulation of the associated optimization problem as a means of signal recovery. In the remaining chapters we deviate from this regime and look towards machine learning algorithms as a recovery method. While we will revisit the photon-limited imaging problem in Chapter 6, using machine learning techniques opens the scope of usability to a variety of problems in signal processing. In the following chapters we will present models suited to different types of signals and applications as well as numerical optimization algorithms that aid in the implementation of those models.

While the definition of *machine learning* can be subjective, a generalized understanding is that the field of machine learning seeks to leverage the power of computers in order to learn from data [5]. The field itself is as broad as the definition, including statistical models, regression algorithms, decision tree algorithms, etc [4, 17]. In this chapter we will mainly be focusing on the subsection of machine learning known as deep learning. We will define deep learning, as well as establish the building blocks involved in creating deep learning algorithms. Most of the explanations in this section is established by Goodfellow et al. in [6].

4.1 Feedforward Neural Network

Deep learning techniques use a combination of mathematical operations and optimization routines in order to learn internal parameters with the intent of extracting features from large data sets and drawing inference from data [10]. These models have already been established as the state of the art in computer vision and natural language processing applications [7, 9]. At the heart of deep learning algorithms is the feed forward neural network or multilayer perceptron (MLP). The goal of the MLP is to infer an unknown function f, based on a series of inputs x_i and outputs y_i. The process can be described mathematically as $y = \hat{f}(x, W)$ where W is a set of learned parameters that define the mapping from x to y. The complexity or depth of an MLP is defined by chaining together successive functions, each containing their own set of parameters. For example, the most basic MLP contains three functions or layers. The process is described by the equation

$$\hat{f} = \hat{f}^{(3)}(\hat{f}^{(2)}(\hat{f}^{(1)}(x))), \tag{4.1}$$

where $\hat{f}^{(1)}$ is referred to as the input layer starting the forward process [6]. The last layer, known as the output layer, provides the intended output of the inference. In a classification application

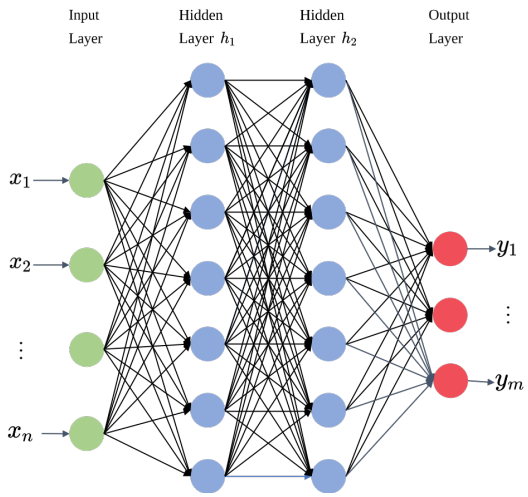
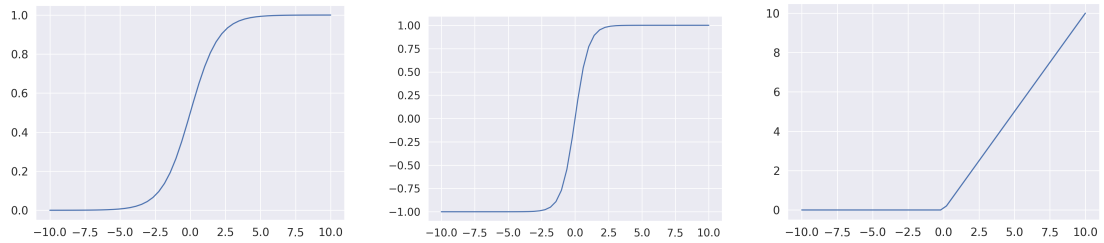


Figure 4.1: Multi-layer fully connected network

this could be a binary indicator or in a regression setting this could be an approximate value. In (4.1) we note that the output of $\hat{f}^{(2)}$ remains "hidden", as such, these types of layers are known as hidden layers. Deep learning models typically contain multiple hidden layers giving them the notion of depth, motivating the name deep learning. Each layer in a "vanilla" or simple neural network has a user defined number of neurons (this will be discussed in the following section), thus there is a notion of depth (the number of hidden layers) versus width (the number of neurons). The universal approximation theorem states that these types of networks can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$ given that there is at least one hidden layer with an activation function and that there are enough neurons present in the network [3]. The idea of having enough neurons in this layer allows the possibility of having an intractable amount of units. To reduce the number of necessary neurons in any given layer, it is common practice to increase the depth of the network [20]. Now that we have the general framework for a deep feedforward network, we can explain the building blocks of a given network's architecture.

4.2 Fully Connected Layers

In section 4.1 we referred to an MLP as a series of chained functions that approximate the mapping f. In this section we describe the most basic block of the MLP, the perceptron also known as a fully connected layer [15]. In order to discuss this piece of architecture, we follow the forward process through the network shown in figure 4.1. The input layer contains the input $x \in \mathbb{R}^n$. In an application where your data is tabular this would be a row of features. In a computer vision type of classification problem this would be a vectorization of an image in your data set. We define the following hidden layers as the transformation $h_i = g(W_i^T h_{i-1} + b_i)$ where $W_i \in \mathbb{R}^{n \times m}$ is a weight matrix of trainable parameters corresponding to the number of neurons in that layer, $b_i \in \mathbb{R}^m$ is a bias term and g is an activation function adding non-linearity to the output of the layer. The output of the current layer is h_i while the input h_{i-1} is from the previous layer. We can



(a) (b) (c)

Figure 4.2: Behavior of activation functions used in neural networks (a) sigmoidal function, (b) hyperbolic tangent and (c) rectified linear unit (ReLU).

then describe the network in figure 4.1 as the following series of equations

$$\begin{aligned} h_1 &= g(W_1^T x + b_1), \\ h_2 &= g(W_2^T h_1 + b_2), \\ y &= s(W_o^T h_2 + b_o). \end{aligned} \tag{4.2}$$

There are a few things to note about this system. The first, is that the dimension of the output of each hidden layer need not be the same dimension for all i. Depending on the application, it may be beneficial to have smaller hidden layers in the center of the network. This means that it is necessary to have weight matrices W_i and biases b_i of different dimensions corresponding to the intended dimension of the input and output of those layers. Furthermore, in (4.2) the activation functions are noted as g and s where g corresponds to the activation function of the hidden layers and s corresponds to the activation of the final layer. Typically the same activation function is used for all hidden layers, but this is not a necessary restriction. The activation function of the final layer or output layer is typically different than that of the hidden layers.

4.2.1 Activation Functions

Until now we have only referred to the activation functions in name. These non-linear functions are crucial to a neural network's ability of function estimation. This is not only stated by the universal approximation theorem, but intuition would tell us that without the activation function we would only be able to produce a linear representation of our inputs. In this section we define two of the most commonly used activation functions which will be used in applications in Chapters 5 & 6. These functions are not the only ones available to those building neural networks. This is an active field of research and we guide the reader to the following references for alternatives [2, 8, 22].

Rectified Linear Unit

The rectified linear unit or ReLU activation function is the most commonly used activation function and is defined by the function $g(z) = \max\{0, z\}$ where $z = W^T x + b$ is the linear transformation defined in section 4.2 [13]. The function is pictured in figure 4.2 (c). The choice of the rectified linear unit is motivated by the empirical evidence presented in Krizhevsky et al. in [9]. In addition, the rectified linear unit also promotes sparsity in the number of active units. In

half of its domain the activation outputs zero and the other half of the domain is linear. This keeps the derivatives consistent that are necessary for optimization during the learning process.

Hyperbolic Tangent

An alternative to the ReLU activation function is the hyperbolic tangent function seen in figure 4.2 (b). The use of the function is an alternative to the logistic sigmoid function (see Fig. 4.2 (a)) which was used quite heavily before ReLU. The function is given as

$$\sigma(z) = \frac{1}{1 + \exp(-z)}. \tag{4.3}$$

The sigmoid function maps to values between $(0, 1)$ which can be viewed as a probability that the input belongs to a labeled class. However, the output is not zero centered, making the optimization step during training difficult. For this reason the sigmoid function is typically relegated to the output layer. Instead the hyperbolic tangent $\tanh(z)$ is typically used as an activation. The hyperbolic tangent seen in figure 4.2 (b) resembles a linear model close to 0 which aids in the learning process [14].

4.3 Convolutional Layers

Introduced by Yann LeCun and his team in [12] and later brought back in the spotlight by the success of AlexNet [9] in the field of computer vision, the convolutional layer can be thought of as an alternative to the fully connected layer. Convolutional layers are typically used in a feedforward network much like the fully connected layers of the previous section. The input is processed by the convolution and then an elementwise activation function is applied to the output. These types of networks, which consist of at least one convolutional layer are known as convolutional neural networks (CNNs). While fully connected layers rely on dense connections represented by large weight matrices, each layer of a CNN has a few parameters which are shared over each input. Each layer consists of an input I from a previous layer or the raw data itself, the kernal or filter K with $K \ll I$ and the output or feature map S. The dimensions of these components may vary based on the application, but in general we are convolving the kernal with the image in order to create the feature map, see Fig. 4.3 (a). The operation that is actually implemented by most neural network libraries is described for the two dimensional case as

$$S(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n), \tag{4.4}$$

where m and n denote the size of your kernal [6]. If we describe the input as a tensor of dimensions $D \times W \times H$, then (4.4) considers an input with a channel of 1 (consider for example a grayscale image). Typically in CNNs, the filter must be described in three dimensions, with the third dimension matching the depth of your data see Fig.4.3(b). In order to acheive the desired depth D of your output, D kernals are applied in parallel to the input. For example, in an image classification problem where the images have a dimension of 32×32, if the images are 3-channel images, red, green, blue (RGB), then the depth of your kernal must be 3. The remaining dimensions H and W of the feature map are controlled by using padding (see Fig.4.3 (a)), manipulating stride and choosing the appropriate kernal size. These features are particular to the application and will be discussed in later sections. As mentioned before, one major advantage

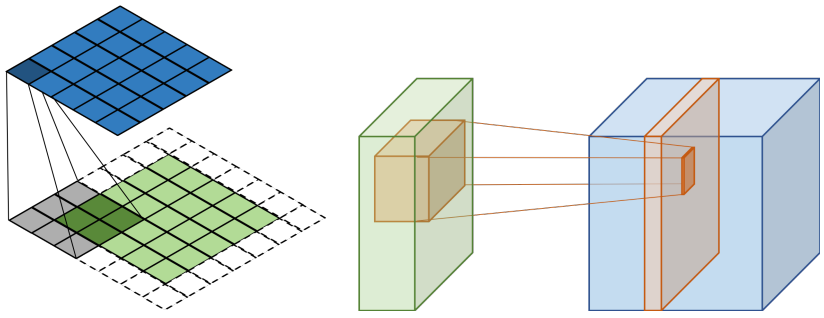


Figure 4.3: Behavior of the convolutional layer. (a) Using padding to obtain the same dimensions of output (H × W) as the input. (b) The convolutional filter considers all channels in the input and reduces the information to a single channel in the output.

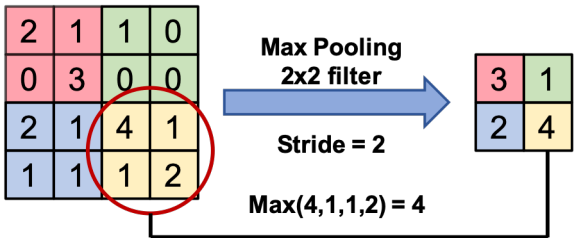


Figure 4.4: Max pooling layer applied to the output of the convolutional layer. The filter is a 2 × 2 filter with a stride of 2.

that CNNs have over their fully connected layer counterparts are the decrease in the number of trainable parameters. By creating kernals or filters that are substantially smaller than their inputs, the architecture can still detect subtle features while cutting down on training expense. One thing to note is that CNNs typically still incorporate fully connected layers into their architecture. In fact, this is necessary when performing classification using CNNs in that the feature map of the last convolutional layer is vectorized and processed through fully connected layers in order to arrive at the probability that the input belongs to a certain class.

Max Pooling

We have seen that the convolutional layer is similar to the fully connected layer in that we provide a transformation on the input to the layer and then provide a non-linearity to the output of that transformation. The convolutional layer goes one step beyond and adds a pooling layer to the resultant feature map. The pooling layer, in particular a max pooling layer, significantly improves the performance of convolutions in object recognition applications [18]. The maximum pooling layer reports the maximum number within a certain window. This window is similar to that of the kernal in the convolution except that the there are no learned parameters. The motivation behind this layer is that by performing this pooling, the output is invariant or less sensitive to perturbations in the input. This is helpful in the learning and inference process in that the architecture will see many instances of the same type of data under slight translations, pooling helps the architecture identify the same features during this process [6].

4.4 Parameter Learning

In previous sections, we discussed the building blocks for the architectures we will use in the next few chapters. Whether it be a fully connected layer network or a CNN, in either case the network relies on a combination of learned parameters from each layer in order to approximate the intended function. The key here is that the parameters are learned and not provided by the user. In this section we will discuss the process by which these parameters are learned using gradient based learning. First, we will define the different metrics that are commonly used for most applications in deep learning and then we will discuss the learning or training process.

4.4.1 Cost Functions

The cost function, sometimes referred to as the loss function, is part of a feedback loop that indicates how well your model is performing during the learning process. The training procedure involves subsampling a portion of your data, running it through the network and then comparing the output of the model to the true corresponding data. This process is known as supervised learning [1] and requires a metric to determine the performance of your model. It should be no surprise that the application influences the choice of the cost function. The applications in this work will fall under two categories: (1) classification and (2) image reconstruction. This will limit our choice of cost function to a cross entropy function or the mean squared error (MSE).

Cross Entropy Loss Function

The most basic case of utilizing deep neural networks (DNNs) for classification is being able to discern if, given some information about a data point e.g. features, image, etc., the instance belongs to a class or not. This is known as a binary classification problem. Under this scenario, the output layer of neural network produces a single scaler value. Furthermore, by using the sigmoid activation function in equation (4.3) we can assure that the value of the output lies between 0 and 1. The value then acts as a probability of the point being in the class or excluding it from the class. If we let θ represent the trainable parameters of our neural network and $p(y_i)$ represent the probability of y_i belonging to the positive class ($y = 1$) given input x_i and θ, then the cost function over m instances is

$$\mathcal{L}(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)). \tag{4.5}$$

Equation (4.5) often called the binary cross entropy function and is designed to increase the cost when the probability estimate is close to zero and the truth is close to one as well as if the truth is close to zero and the probability estimate is close to one [5]. Our goal then becomes to tune the parameters θ in order to minimize the function in (4.5). We will talk about the optimization routines that are typically used to accomplish this in the next chapter, but as this is a convex function we can use variants of gradient descent in order to find a minima.

In many cases we are interested in problems with more than one class. In these cases, the output from the final layer is a vector $z \in \mathbb{R}^K$ where K corresponds to the number of classes. The neural network assigns a raw score to each class and by using the softmax function

$$\hat{p}_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}, \tag{4.6}$$

we can convert the k-th raw score into $\hat{p}_k$ the probability that the instance belongs to the k-th class. The denominator in (4.6) is performing a normalization across all values and thus the probabilities in $\hat{p} \in \mathbb{R}^K$ add up to one. As an extension to (4.5) and drawing on the cross entropy function originating in information theory [19], the cross entropy cost function is given by

$$\mathcal{L}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_{(i)k}\log(\hat{p}_{(i)k}), \tag{4.7}$$

where $y_{(i)k}$ is the k-th component of the i-th instance. The vector y_i is a one-hot encoding of the target where the k-th component is equal to 1 if the instance belongs to the k-th class and zero everywhere else. In equations (4.5) & (4.7) we seek to minimize the cost function over the parameters in order to improve classification.

Mean Squared Error

In Chapters 2 & 3 we used the mean squared error or MSE to evaluate the reconstruction of our signal. In the context of deep learning the MSE is also used as the cost function

$$\mathcal{L}(\theta) = \frac{1}{m}\left(\hat{y}_i - y_i\right)^2, \tag{4.8}$$

where $\hat{y}_i$ is the estimate of y_i. Unlike applications in classification, the estimate and target typically refer to an image or a vector. In chapter 6 we will see that MSE will be used to train the neural network to perform our signal reconstruction tasks as well as a metric to evaluate how well the algorithm has performed the intended task. In a similar manner to the cross entropy cost functions, we seek to minimize the MSE with respect to the model parameters θ.

4.4.2 Back-Propagation Algorithm

In this section we draw the connection between the cost function and the training of the neural network's parameters. In order to do so we provide a high-level overview of the back-propagation algorithm. We refer the reader to [6] for a more thorough explanation.

Although the back-propagation algorithm was developed years earlier, the algorithm made a resurgence in the 1980s appearing in papers as research in neural networks began to bloom [21, 16, 11]. The algorithm addresses the increase in the number of layers incorporated in a deep learning architecture and the problem presented with trying to minimize the cost function in relation to weights or parameters in earlier layers. In order to explain the learning process (also known as the training process) of a feed forward network, let us consider a simple binary classificaion problem where (x_i, y_i) are a data pair from our data set. In the first stage we feed the input x_i into our model and receive the probability estimate $p(y_i)$, this is known as the forward pass. We evaluate the output using the cost function $\mathcal{L}(\theta)$, in this case (4.5). Back-propagation or backprop allows us to use this information in order to compute the gradient $\nabla_\theta \mathcal{L}(\theta)$ where the parameters θ are dispersed among the layers. The algorithm amounts to applying the chain rule recursively in order to obtain gradients with respect to all parameters. With the gradients calculated, an optimization routine (discussed in chapter 5) is used to tune or learn the correct parameters. The process is repeated over many data points until the algorithm learns the correct parameters and an approximation to the underlying function is made.

Bibliography

[1] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[2] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". *arXiv preprint arXiv:1511.07289* (2015).

[3] B. C. Csáji et al. "Approximation with artificial neural networks". *Faculty of Sciences, Etvs Lornd University, Hungary* 24.48 (2001), p. 7.

[4] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[5] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press, 2016.

[7] G. Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

[8] G. Klambauer et al. "Self-normalizing neural networks". *Advances in neural information processing systems*. 2017, pp. 971–980.

[9] A. Krizhevsky, I. Sutskever, and G. Hinton. "Imagenet classification with deep convolutional neural networks". *Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 1097–1115.

[10] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". *nature* 521.7553 (2015), pp. 436–444.

[11] Y. LeCun et al. "A theoretical framework for back-propagation". *Proceedings of the 1988 connectionist models summer school*. Vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann. 1988, pp. 21–28.

[12] Y. LeCun et al. "Backpropagation applied to handwritten zip code recognition". *Neural computation* 1.4 (1989), pp. 541–551.

[13] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[14] C. Nwankpa et al. “Activation functions: Comparison of trends in practice and research for deep learning”. *arXiv preprint arXiv:1811.03378* (2018).

[15] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review* 65.6 (1958), p. 386.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. *nature* 323.6088 (1986), pp. 533–536.

[17] S. R. Safavian and D. Landgrebe. “A survey of decision tree classifier methodology”. *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), pp. 660–674.

[18] D. Scherer, A. Müller, and S. Behnke. “Evaluation of pooling operations in convolutional architectures for object recognition”. *International conference on artificial neural networks*. Springer. 2010, pp. 92–101.

[19] J. Shore and R. Johnson. “Properties of cross-entropy minimization”. *IEEE Transactions on Information Theory* 27.4 (1981), pp. 472–482.

[20] M. Telgarsky. “Benefits of depth in neural networks”. *arXiv preprint arXiv:1602.04485* (2016).

[21] P. J. Werbos. “Backpropagation through time: what it does and how to do it”. *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[22] B. Xu et al. “Empirical evaluation of rectified activations in convolutional network”. *arXiv preprint arXiv:1505.00853* (2015).

Chapter 5

Optimization for Deep Learning Applications

In chapter 4 we established an overview of the components that compose a deep learning model. In particular, we discussed the parameters that are an essential component of a network's ability to provide an inference of some kind. The obvious question becomes how do we "learn" these parameters? We briefly touched on the fact that backprop in section 4.4.2 calculates the gradients of the loss function with respect to the parameters. In this chapter, we will discuss the method by which the loss function is minimized using those gradients. Initially, we will summarize two of the most commonly used techniques in the field of deep learning, stochastic gradient descent (SGD) and Adam, a derivative of SGD. In the following sections, we propose two novel algorithms for the training of a deep neural network. The first algorithm uses a quasi-Newton approximation in a trust-region setting, while the second algorithm uses an efficient computation of the second derivative in order to provide curvature information in the descent process.

5.1 Related Methods

First-derivative or gradient-based algorithms have emerged as the standard optimization techniques used for training deep neural networks [4, 29, 31, 34]. They are preferred for their ease of implementation and their relatively low computational costs. Many of these methods are based on a variant of gradient descent known as stochastic gradient descent (SGD). This method differs from a classic gradient descent approach in that the gradient is replaced by an estimation calculated from a subset of the data as opposed to the entire dataset. This lends itself to the training of deep neural networks in that we typically process only a few instances of a data set at a time in a subsample known as a batch. This reduces the computational efforts involved in the forward pass at a cost of the quality of the solutions [5].

If we continue to describe the parameters of our network as θ and we recognize that the values of θ evolve during the training process, at iteration k, a sample batch $S_k \in \{1, 2, \dots, n\}$ is randomly chosen and the current iterate θ_k is updated using

$$\theta_{k+1} = \theta_k - \alpha_k \frac{1}{|S_k|} \sum_{j \in S_k} \nabla \mathcal{L}_j(\theta_k),$$

where α_k is a step length parameter (known as the *learning rate*). In SGD methods, an approximate gradient, $\sum_{j \in S_k} \nabla \mathcal{L}_j(\theta_k)$, is used instead of the full gradient, $\nabla \mathcal{L}(\theta_k)$, because evaluating the full gradient is computationally expensive, especially for large data sets, i.e., when n is very large.

However, these methods are not without their drawbacks. Although they are easily implemented, gradient decent-based methods are extremely sensitive to weight initialization and parameter tuning. As such, they require hyperparamter tuning schedules which can be expensive in terms of time and computational resources. Due to the highly non-convex nature of the objective function associated with training neural networks and the stochastic behavior of these techniques, it is likely that the algorithm will rest at local minima for sometime before continuing to ascend. One way researchers have tried to address these issues is by developing adaptive algorithms in which the learning rate is adjusted during the training process [31, 32]. By far, the most popular evolution of SGD is Adam [14]. Adam seeks to improve on SGD by integrating features from a few other SGD algorithms into a single algorithm. For starters it takes advantage of an adaptive learning rate method featured in AdaGrad [10] which computes different learning rates for different parameters. It determines the learning rates by using the mean and variance of the gradient of the cost function. The cost function can be thought of as a random variable because it is being calculated using batches. It also incorporates a feature of Root Mean Square Propagation (RMSProp) [32] which keeps an average of the magnitude of the gradients. Although Adam tries to address these issues, it is still beholden to the shortcomings of gradient descent. Mainly that they are using first order information. In the next chapters we propose methods which either use first derivative information to approximate curvature or second derivative information itself.

5.2 Quasi-Newton Trust-Region Method

While architecture and computational resources play an important role in the effectiveness of neural networks, the methodology used in training these networks leaves an opportunity to increase efficiency. As we have seen in Section 5.1, gradient based optimization methods are used in order to minimize what is commonly referred to as the loss function. The goal here is to minimize the disparity between the outcome provided by the neural network and the intended result. In this section we propose the Trust-Region Minimization Algorithm for Training Responses (TRMinATR) as an alternative to gradient descent methods. This work is based on the paper by Rafati, DeGuchy, and Marcia [27].

Looking towards quasi-Newton methods as an alternative to gradient decent is an active area of research. Limited memory quasi-Newton algorithms with line search have been implemented in a deep learning setting [15]. These methods approximate second derivative information improving the quality of each training iteration and circumvent the need for application specific parameter tuning. The novelty of TRMinATR is in the use of the L-BFGS quasi-Newton method in a trust-region setting to train deep neural networks. TRMinATR solves the associated trust-region subproblem, which can be computationally intensive in large scale problems, by efficiently computing a closed form solution at each iteration.

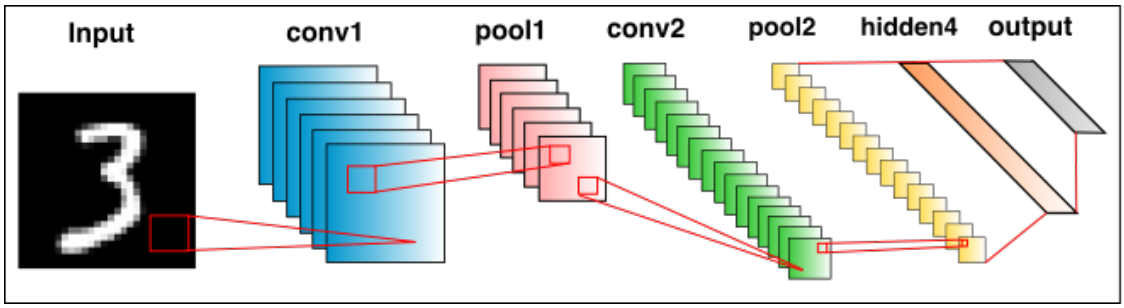


Figure 5.1: A LeNet deep learning network inspired by the architecture found in [16] . The neural network is used in the classification of the MNIST dataset of hand written digits. The convolutional neural network (CNN) uses convolutions followed by pooling layers for feature extraction. The final layer transforms the information into the required probability distribution.

5.2.1 Training Deep Neural Networks

In this section we briefly review the formulation of the optimization problem presented in Chapter 4 that must be solved in order to train a neural network. We present the architecture as a general framework where each layer of a feed forward network is defined as $h_i = g(W_i, h_{i-1})$, where W_i represents the trainable parameters of the layer, h_{i-1} is the input from the previous layer and g is an activation function. This differs with the notation provided in section 4.1 in that the layer is allowed to be either convolutional or fully connected. During training of the neural network we pass the input $h^{(0)} = x$ through the layers and obtain the network's approximation of the desired output $h^{(n)} = \hat{y}$ where n is the number of layers in the neural network. We seek to adjust the weights $W^{(i)}$ in order to improve the quality of the approximate output $h^{(n)}$. Learning is accomplished by minimizing the loss function $\mathcal{L}(h, h^{(n)})$ where $h = y$ is the true response for the input x. The output $h^{(n)}$ is in fact a function of the parameters $W = (W^{(1)}, W^{(2)}, \dots, W^{(n)})$, thus we seek to

$$\min_{W} \mathcal{L}(h, h^{(n)}). \tag{5.1}$$

5.2.2 Methods

In this section, we outline two methods used to solve the following unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \boldsymbol{\Phi}(x). \tag{5.2}$$

Both methods seek to minimize the objective function $\boldsymbol{\Phi}(x)$ by defining a sequence of iterates $\{x_k\}$ which are governed by the search direction p_k. Each respective method is defined by its approach to computing the search direction p_k with minimizing the quadratic model of the objective function defined by

$$q_k(p) \triangleq g_k^T p + \frac{1}{2} p^T B_k p, \tag{5.3}$$

where $g_k \triangleq \nabla \boldsymbol{\Phi}(x_k)$ and B_k is an approximation to $\nabla^2 \boldsymbol{\Phi}(x_k)$.

Algorithm 5: Line search method pseudo-code.

Input: starting point x_0, tolerance $\epsilon > 0$
  $k \leftarrow 0$
  repeat
    compute $g_k = \nabla \boldsymbol{\Phi}(x_k)$
    update L-BFGS matrix B_k
    compute search direction p_k by solving (5.4)
    find α_k that satisfies Wolfe Conditions in (5.6)
    $k \leftarrow k + 1$
  until $\|g\| < \epsilon$ or k reached to max number of iterations

Line Search Method

Each iteration of the line search method computes search direction p_k by solving optimization subproblem

$$p_k = \arg\min_{p \in \mathbb{R}^n} q_k(p), \tag{5.4}$$

and then decides how far to move along p_k by choosing a *step length* α_k. The iteration x_k updates by following relation:

$$x_{k+1} = x_k + \alpha_k p_k. \tag{5.5}$$

Usually p_k is required to be a descent direction and $\alpha_k \in (0, 1]$ is chosen to satisfy the sufficient decrease and curvature conditions, e.g. Wolfe conditions [23]:

$$\boldsymbol{\Phi}(x_k + \alpha_k p_k) \leq \boldsymbol{\Phi}(x_k) + c_1 \alpha_k \nabla \boldsymbol{\Phi}_k^T p_k, \tag{5.6a}$$

$$\nabla \boldsymbol{\Phi}(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla \boldsymbol{\Phi}(x_k)^T p_k, \tag{5.6b}$$

with $0 < c_1 < c_2 < 1$. The general pseudo-code for line search method is given in Algorithm 5 (see [23] for details).

Trust-Region Method

The trust-region method solves (5.2) using the localized quadratic approximation of the objective function q_k defined in (5.3) at each iteration.

$$p_k = \arg\min_{p \in \mathbb{R}^n} q_k(p) \quad \text{subject to} \quad \|p\|_2 \leq \delta_k, \tag{5.7}$$

where δ_k denotes the radius of the trust region. There is a computational bottleneck associated with solving (5.7) in large-scale optimization. This is the type of problem associated with training neural networks. These computational costs will be addressed in a later section.

Solving the trust-region subproblem to high accuracy requires consideration of the problem's optimality conditions for a global solution. Methods such as those presented in [12, 22, 8] make use of the following theorem:

Algorithm 6: Trust region method pseudo-code.

Input: starting point x_0, tolerance $\epsilon > 0$, $\hat{\delta} > 0$,
$\delta_0 \in (0, \hat{\delta})$, $\eta \in [0, 1/4)$

$k \leftarrow 0$
repeat
  compute $g_k = \nabla \Phi(x_k)$
  update L-BFGS matrix B_k
  compute search direction p_k by solving (5.7)
  $\rho_k \leftarrow (\Phi(x_k) - \Phi(x_k + p_k))/(q_k(0) - q_k(p_k))$
  update trust-region radius δ_k
  if $\rho_k > \eta$ **then**
    $x_{k+1} = x_k + p_k$
  else
    $x_{k+1} = x_k$
  end if
  $k \leftarrow k + 1$
until $\|g_k\| < \epsilon$ or k reached to max number of iterations

Theorem 2. *Let δ be a positive constant. A vector p^* is a global solution of the trust-region subproblem* (5.7) *if and only if $\|p^*\|_2 \leq \delta$ and there exists a unique $\sigma^* \geq 0$ such that $B + \sigma^* I$ is positive semidefinite and*

$$(B + \sigma^* I)p^* = -g \quad \text{and} \quad \sigma^*(\delta - \|p^*\|_2) = 0. \tag{5.8}$$

Moreover, if $B + \sigma^* I$ is positive definite, then the global minimizer is unique.

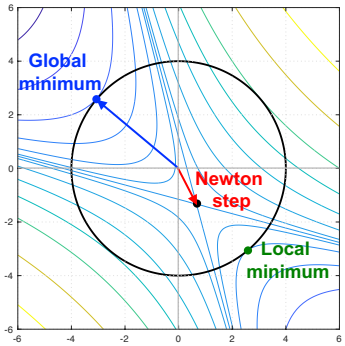


Figure 5.2: An illustration of trust-region methods. For indefinite matrices, the Newton step (in red) leads to a saddle point. The global minimizer (in blue) is characterized by the conditions in Eq. (5.8) with $B + \sigma^* I$ positive semidefinite. In contrast, local minimizers (in green) satisfy Eq. (5.8) with $B + \sigma^* I$ not positive semidefinite.

The general pseudo-code for trust region method is given in Algorithm 6. (see [8, 23] for details).

Quasi-Newton Methods

Methods that use $B_k = \nabla^2 \Phi(x_k)$ for the Hessian in the quadratic model in (5.3) typically exhibit quadratic rates of convergence. However, there are several assumptions needed to ensure this approach is computationally feasible. First, solves with the matrix $\nabla^2 \Phi(x_k)$ must be done efficiently. For large-scale problems, unless $\nabla^2 \Phi(x_k)$ has structure that can easily be exploited, this is generally not the case. Second, $\nabla^2 \Phi(x_k)$ must have the positive eigenvalues so that the resulting search direction p_k is guaranteed to be a descent direction. Third, $\nabla^2 \Phi(x_k)$ must be computationally available. In applications such as inverse problems, computing $\nabla^2 \Phi(x_k)$ requires solving a system of partial differential equations, which may or may not have an expression for the Hessian. In cases where using the Hessian matrix $\nabla^2 \Phi(x_k)$ is not practical, quasi-Newton methods are viable alternatives because they exhibit superlinear convergence rates while maintaining memory and computational efficiency.

Perhaps the most well-known among all of the quasi-Newton methods is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [19, 23], given by

$$B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T B_k + \frac{1}{y_k^T s_k} y_k y_k^T, \tag{5.9}$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla \Phi(x_{k+1}) - \nabla \Phi(x_k)$. The matrices are defined recursively with the initial B_0 taken to be a $B_0 = \gamma I$, where the scalar $\gamma > 0$. In practice, only the m most-recently computed pairs $\{(s_k, y_k)\}$ are stored, where $m \ll n$, typically $m \leq 100$ for very large problems. This approach is often referred to as *limited-memory* BFGS, or L-BFGS. Because these updates are low-rank, the matrix B_{k+1} can be compactly represented as $B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T$, for some $\Psi_k \in \Re^{n \times 2(k+1)}$ and $M_k \in \Re^{2(k+1) \times 2(k+1)}$. In particular,

$$\Psi_k = \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix} \text{ and } M_k = -\begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1},$$

where $S_k = [\, s_0 \;\; s_1 \;\; s_2 \;\; \cdots \;\; s_k \,] \in \Re^{n \times (k+1)}$, and $Y_k = [\, y_0 \;\; y_1 \;\; y_2 \;\; \cdots \;\; y_k \,] \in \Re^{n \times (k+1)}$, and L_k is the strictly lower triangular part and D_k is the diagonal part of the matrix $S_k^T Y_k \in \Re^{(k+1) \times (k+1)}$, i.e., $S_k^T Y_k = L_k + D_k + U_k$, where U_k is a strictly upper triangular matrix (see [6] for details).

Given the compact representation of B_{k+1}, then both the line search problem (5.4) and the trust-region subproblem (5.7) can be efficiently solved when the L-BFGS matrix is used as the Hessian approximation. In particular, the solution to (5.3) is given by $p_k^* = -\frac{1}{\gamma}\left[I - \Psi_k(\gamma M_k^{-1} + \Psi_k^T \Psi_k)^{-1} \Psi_k^T\right] g_k$, using the well-known Sherman-Morrison-Woodbury formula. Similarly, the solution to the trust-region subproblem is obtained efficiently. First, the QR factorization of $\Psi_k = QR$ is formed. Then the eigendecomposition of the $2(k+1) \times 2(k+1)$ matrix $RM_kR^T = V \Lambda V^T$ is computed so that the partial eigendecomposition of $B_{k+1} = \gamma I + QV \Lambda V^T Q^T$ is obtained. This allows for a change in variables in (5.7) that yields a closed form expression for the solution p_k^* (see Algorithm 1 in [1] for details).

5.2.3 Numerical Experiments

In this section we compare the line search L-BFGS optimization method with our proposed Trust-Region Minimization Algorithm for Training Responses (TRMinATR). The goal of the experiment is to perform the optimization necessary for neural network training. Both methods are implemented to train the LeNet-5 architecture with the purpose of image classification of the

Table 5.1: Structure of the LeNet5 convolutional neural network trained on the MNIST dataset.

LeNet-5	
Layer	NVIDIAConnectivity
0: input	28 × 28 image
1	convolutional, 20 5 × 5 filters (stride=1), total 11520 neurons, followed by ReLU
2	max pool, 2 × 2 window (stride=2), total 2280 neurons
3	convolutional, 50 5 × 5 filters (stride=1), total 3200 neurons, followed by ReLU
4	max pool, 2 × 2 window (stride=2), total 800 neurons
5	fully connected, 500 neurons without dropout followed by ReLU
6: output	fully connected, 10 neurons without dropout followed by softmax
total of 431080 trainable parameters	

MNIST dataset. All simulations were performed on an AWS EC2 p2.xlarge instance with 1 Tesla K80 GPU, 64 GiB memory, and 4 Intel 2.7 GHz Broadwell processors. For the scalars c_1 and c_2 in the Wolfe line search condition, we used the typical values of $c_1 = 10^{-4}$ and $c_2 = 0.9$ [23].

LeNet-5

The convolutional neural network known as LeNet-5 was mainly used for character recognition tasks such as reading zip codes and digits [16]. The architecture is given in Table 5.1. The convolutional layers extract features from the input image and preserve spatial relationships between pixels using the learned information.

MNIST Dataset

The convolutional neural network was trained and tested using the MNIST Dataset [17]. The dataset consists of 70,000 examples of handwritten digits with 60,000 examples used as a training set and 10,000 examples used as a test set. The digits range from 0 - 9 and their sizes have been normalized to 28x28 pixel images. The images include labels describing their intended classification.

5.2.4 Results

The line search algorithm and TRMinATR perform comparably in terms of loss and accuracy. This remains consistent with different choices of the memory parameter m (see Fig. 5.4). The more interesting comparison is that of the training accuracy and the test accuracy, the two metrics follow each other closely. This is unlike the typical results using common gradient descent based

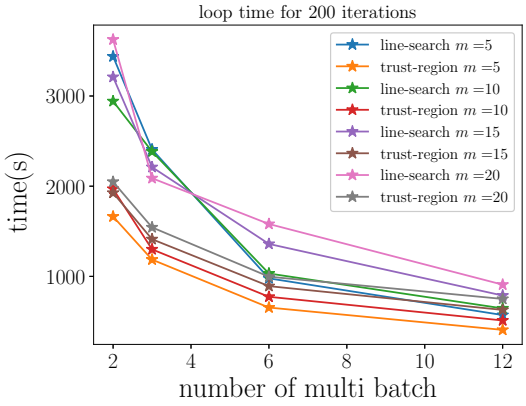


Figure 5.3: We compare the loop time for 200 iterations of the line-search and trust-region quasi-Newton algorithms for different batch sizes. As the number of multi batches increase, the size of each batch decreases. Both methods were tested using different values of the memory parameter m.

optimization. Typically the test accuracy is delayed in achieving the same results as the train accuracy. This would suggest that the model has a better chance of being generalized beyond the training data.

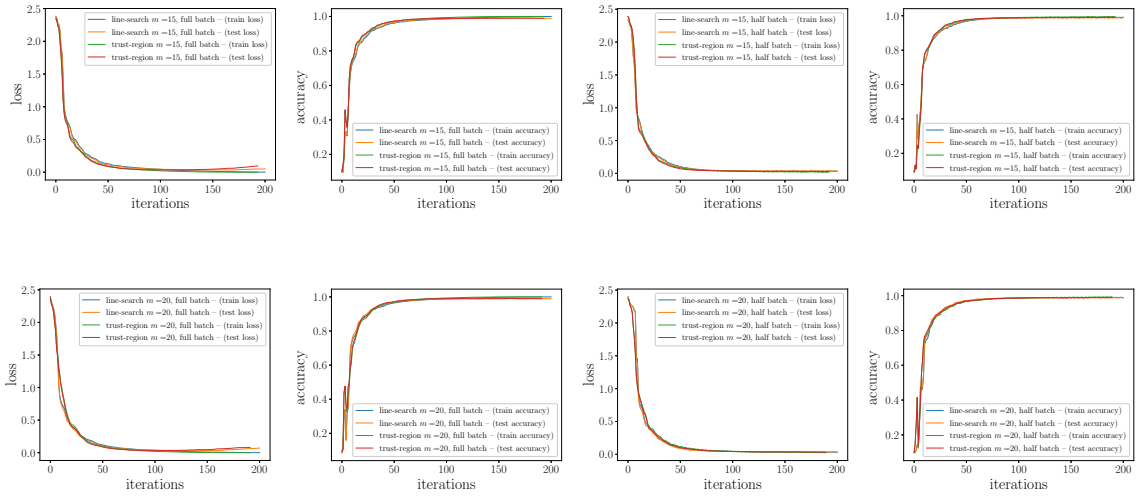
We also report that the TRMinATR significantly improves on the computational efficiency of the line-search method when using larger batch sizes. This could be the result of the line-search method's need to satisfy certain wolfe conditions at each iteration. There is also an associated computational cost when verifying that the conditions for sufficient decrease are being met. When the batch size decreases, the trust-region method continues to outperform the line-search method. This is especially true when less information is used in the Hessian approximation (see Fig. 5.3).

5.2.5 Conclusion

In this work we presented the limited memory quasi-Newton method known as L-BFGS as an alternative to the gradient descent methods used to train deep neural networks. In particular we develop the algorithm known as TRMinATR which minimizes the cost function of the neural network by efficiently solving a sequence of trust-region subproblems using low-rank Hessian approximations. The benefit of the method is that the algorithm is free from the constraints of data specific parameters seen in traditionally used methods. TRMinATR also improves on the computational efficiency of a similar line search implementation.

5.3 Second-Order Trust-Region Method

In section 5.2 we presented an algorithm which addressed the need to solve the optimization problem presented in (5.1). The problem is motivated by the need to minimize the cost function during training of the neural network. The algorithm extended beyond the typically used gradient descent method and approximated second derivative information in order to improve the quality of the decent direction. In this work, based on the paper by Ranganath, DeGuchy, Marcia, and Singhal [28], we provide a solution to the aforementioned problem by exploiting second-derivative



(a) (b)

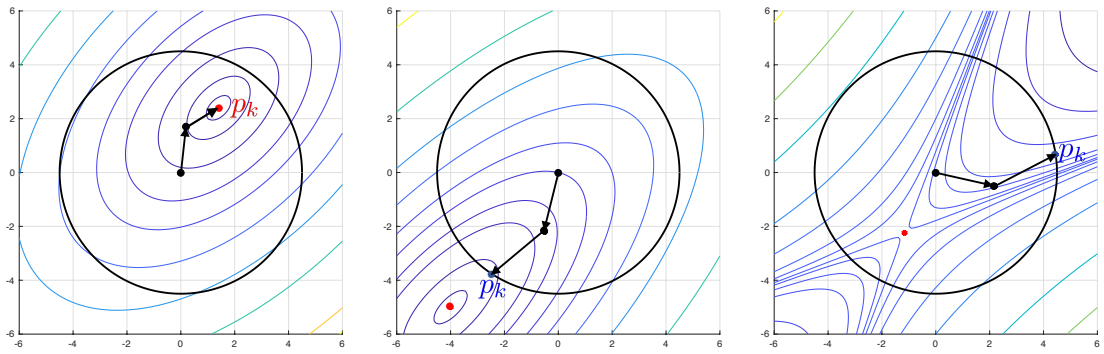
Figure 5.4: The behavior of the loss and accuracy for the training and test sets. (a) Training and testing using the full training set as a batch size. (b) Training using batch sizes that are half of the size of the full set. Results are shown using typical memory settings ($m = 15$ and $m = 20$) in an L-BFGS setting.

information to improve the predictive capabilities of artificial neural networks for data-hungry inference. The novelty in our method combines the efficient computation of a true Hessian-vector product and a trust region setting, thus allowing us to solve the trust-region subproblem using a conjugate based method.

5.3.1 Related Methods

Quasi-Newton Methods

As stated before, quasi-Newton methods have been explored as a possible alternative to gradient descent based algorithms for training neural networks [3, 7, 9, 15]. In section 5.2 we presented an algorithm using the L-BFGS in a trust-region setting. Although they are an improvement on traditional gradient-descent methods, quasi-Newton methods are still only approximations of the second derivative; as such they are beholden to certain complications. The L-BFGS method makes the assumption that the underlying Hessian is always positive definite, which given the non-convexity of the objective function, we know is not always the case [35]. Recently the SR-1 update has been shown to provide more accurate representations of second derivative information in a machine learning context [11]. In either case, the methods are still limited by the amount of data in the training set, particularly in a stochastic setting. Finite difference approximations have also been proposed as a way to approximate second derivative information, but these methods have undesirable qualities in that they can be unstable and still require special attention when the Hessian approximation is not positive definite [20].



(a) (b) (c)

Figure 5.5: Illustration of the CG-Steihaug approach in two dimensions. (a) When $\mathcal{Q}_k(p)$ is convex and its unconstrained minimizer lies within the trust-region radius, then the CG iterates will converge to the unconstrained minimizer. (b) When $\mathcal{Q}_k(p)$ is convex but its unconstrained minimizer is outside the trust region, then the minimizer p_k is defined where the CG iterate crosses the boundary. (c) When $\mathcal{Q}_k(p)$ is not convex, i.e., H_k is not positive definite, then the CG-Steihaug method terminates when a direction of curvature is detected and the minimizer p_k is defined where $\mathcal{Q}_k(p)$ is minimized along the last computed CG iterate.

Hessian-Free Methods

Like quasi-Newton methods, Hessian-free methods look to improve on gradient descent-based algorithms by capitalizing on approximations of second-derivative information or in some cases using the true Hessian itself. In either regime, the explicit storage of the associated matrix can be quite expensive when considering the large-scale optimization problem associated with training a neural network. In order to minimize this cost, Hessian-free methods focus on the matrix vector product of the Hessian or Hessian approximation (H) and an n-dimensional vector (d). In [20], the Hessian-free optimization method is implemented using the finite difference approximation of the matrix vector product:

$$Hd = \lim_{\epsilon \to 0} \frac{\nabla f(x + \epsilon d) - \nabla f(x)}{\epsilon}, \tag{5.10}$$

where the operation is used in a conjugate gradient (CG) setting in order to provide the descent direction to the next iterate. The method presented in [25] improves on the finite difference approximation and computes the actual Hessian vector product as

$$Hd = \frac{\partial}{\partial \epsilon} \nabla f(x + \epsilon d)\Big|_{\epsilon = 0}. \tag{5.11}$$

To calculate Hd for a simple backpropagation network, see [33, 21], and for recurrent backpropagation networks, see [26, 2].

One requirement when using either operation in the context of typical CG methods is that H be positive definite. This is a requirement to ensure that the CG method or any linesearch methods result in a descent direction. As they are stated, the matrix-vector products in (5.10) and (5.11) do not provide any guarantees that the matrix H is positive definite. One commonly used technique to guarantee to avoid the problems associated with an indefinite matrix is to shift or "dampen" the eigenvalues of H as $B = H + \lambda I$, where $\lambda \geq 0$. The Hessian-vector product is now expressed as $Bd = Hd + \lambda d$, where Hd is evaluated using the previously described techniques.

Gauss-Newton approximations have also been proposed as an alternative to the previous two methods. The Gauss-Newton approximation, G, also known as the squared Jacobian, is an outer product of the Hessian, typically used in least-squares problems. In [30], they extend the approximation for use with a cross-entropy loss using a similar framework to that presented in [25]. The author notes that as long as G is positive semi-definite, descent is guaranteed, but to overcome the possibility of indefiniteness, a similar dampening approach must be taken. In later sections, we will make use of Pearlmutter's method while relaxing the requirement that the Hessian be positive definite.

5.3.2 Proposed Approach

The proposed method presents a novel optimization routine designed to minimize the loss function. Unlike the previous methods described, the goal is computing fast Hessian-vector products within a trust-region setting. This allows us to approximately solve the trust-region subproblem using CG while allowing for negative curvature. By incorporating second-derivative information, we improve the impact of each iteration and avoid certain local minima and saddle points. The increase in the quality of the optimization routine will increase the value of each data point and allow us to reach better optima with fewer training instances. In the following subsections we describe the proposed approach in more detail.

Fast Exact Hessian-Vector Products

As stated in the previous section, computing the second-derivative or Hessian can be computationally intensive. Furthermore, in the context of neural networks, storing the Hessian can be infeasible. Using exact second derivative information allows us to create an accurate localized model of the true objective function which is used in the trust-region setting described in the next section. At the same time we require this information to be available for use in the CG method outlined later in this section. In both cases, a Hessian-vector multiplication is required, and so we chose to use Pearlmutter's algorithm, commonly referred to as Rop [25]. While we are not the first to use Rop in a CG setting, our approach is novel in that we no longer require dampening of the Hessian to guarantee positive definiteness.

The motivation for avoiding damped Hessian approximations comes in two parts. The first is that it requires the choice of another hyperparameter, λ. The choice of λ can greatly affect the convergence of the optimization routine and should be chosen for each update of the Hessian vector product. The second motivation is that the perturbation to the true Hessian imposed by λ results in an approximation of the second derivative and thus less accurate curvature information. The proposed algorithm relaxes the requirement of the Hessian to be positive-definite and compensates for the possibility of negative curvature by using a trust-region setting.

Trust-Region Subproblem

Trust-region methods are alternative approaches to line-search methods for solving optimization problems. While line-search methods first compute a search direction and then determine a step length along that direction at each iteration, trust-region methods determine a quadratic model to the true objective function and a corresponding region over which the quadratic model can be trusted to be accurate. Specifically, trust-region methods solve a sequence

Algorithm 7 CG-Steihaug

Given: Tolerance $\epsilon_k > 0$
Set: $p_0 = 0, r_0 = g_k, d_0 = -r_0 = -g_k$
if $\|r_0\|_2 < \epsilon_k$ **then**
  return $p_k = z_0 = 0$
end if
for $j \in 0,1,2,3,4..$ **do**
  if $d_j^\top H_k d_j \leq 0$ **then**
    Find τ such that $p_k = z_j + \tau d_j$ minimizes
      $\mathcal{Q}_k(p)$ in (5.12) with $\|p_k\|_2 = \Delta_k$
    return p_k
  end if
  Set $\alpha_j = r_j^\top r_j / d_j^\top H_k d_j$
  Set $z_{j+1} = z_j + \alpha_j d_j$
  if $\|z_{j+1}\| \geq \Delta_k$ **then**
    Find $\tau \geq 0$ such that $p_k = z_j + \tau d_j$ satisfies
      $\|p_k\|_2 = \Delta_k$
    return p_k
  end if
  Set $r_{j+1} = r_j + \alpha_j H_k d_j$
  if $\|r_{j+1}\|_2 < \epsilon_k$ **then**
    return $p_k = z_{j+1}$
  end if
  Set $\beta_{j+1} = r_{j+1}^\top r_{j+1} / r_j^\top r_j$
  Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$
end for

of quadratic subproblems with a single constraint of the following form:

$$p_k = \underset{p \in \Re^m}{\arg\min} \quad \mathcal{Q}_k(p) \equiv g_k^\top p + \frac{1}{2} p^\top H_k p \quad \text{subject to } \|p\|_2 \leq \Delta_k \tag{5.12}$$

where $g_k = \nabla f(w_k)$ is the gradient of f at the current iterate w_k, $H_k = \nabla^2 f(w_k)$ is the Hessian of f (or an approximation), and Δ_k is a scalar parameter referred to as the *trust-region radius*.

Conjugate Gradient (CG)-Steihaug Approach

The final component of the algorithm facilitates solving the trust-region subproblem in (5.12) in order to provide the directional step p_k to the next iteration. In a similar approach to [20] we use a conjugate gradient method in concert with the fast exact Hessian-vector product in (5.11). Our algorithm uses a modified CG method known as the CG-Steihaug approach [24], or as the Steihaug-Toint truncated conjugate gradient method [8], which is outlined in Algorithm 1. Typically used to solve linear systems, CG requires that the system matrix be positive definite. Steihaug's implementation uses a predefined trust region as a means to accommodate non positive

definiteness of the Hessian. If the minimizer lies within the trust-region radius, then the CG iterates converge to the unconstrained minimizer. If the minimizer is outside of the trust region, then the minimizer is located at the point where the iterate crosses the boundary. Finally, if the Hessian is not positive definite, then CG-Steihaug terminates the algorithm and the minimizer is detected at the boundary of the trust region in the direction of the last computed CG iterate (see Fig. 5.5). Once again it is important to re-iterate that the proposed method does not require the Hessian to be positive definite. We can therefore use the exact Hessian and no longer have to choose the dampening scalar λ. As a result, the method allows directions of negative curvature to be detected and used to avoid saddle points. We note that the final iterate p_k in Algorithm 1 satisfies the following decrease in the quadratic model:

$$\mathcal{Q}_k(0) - \mathcal{Q}_k(p_k) \geq c_1 \|g_k\|_2 \min\left(\Delta_k, \frac{\|g_k\|_2}{\|H_k\|_2}\right), \tag{5.13}$$

where c_1 is a constant with $c_1 \in (0, 1]$. Also, we note that the direction p_k satisfies the trust-region constraint, i.e.,

$$\|p_k\|_2 \leq \Delta_k, \tag{5.14}$$

which will be used for convergence results.

Summary of Proposed Approach

In summary the proposed approach has three major components: We use (1) a trust-region method (outlined in Algorithm 2), which allows for indefinite Hessians and Hessian approximations and defines iterates that avoid saddle points; (2) a fast and exact Hessian-vector product which efficiently provides true second-derivative information at the current iterate (cf. (5.11)); and (3) the CG-Steihaug approach, which solves the trust-region subproblem without storing the Hessian matrix. In order to guarantee that the search direction provided by the CG-Steihaug method sufficiently decreases the value of the quadratic subproblem, we implement a Wolfe line search in the direction of p_k [24]. Furthermore, we evaluate the accuracy of the quadratic model of the objective function within the trust region using the Levenberg-Marquardt ratio to determine whether the update is acceptable or not. The trust region is then relaxed if the step is acceptable and constricted if the step is not acceptable. The result is an algorithm that considers second derivative information and allows for the possibility of negative curvature. As such, the impact of each iteration is more valuable when compared to those of first-order methods.

We conclude with the following convergence guarantee, which can be found in [24]. We first define the level set $S = \{w \colon f(w) \leq f(w_0)\}$, where w_0 is the initial point, and an open neighborhood of S by $S(R_0) = \{w \colon \|w - y\| < R_0 \text{ for some } y \in S\}$, where R_0 is a positive constant. Then we have the following result.

Theorem. Let $\eta = 0$ in Algorithm 2. Suppose that $\|H_k\| \leq \beta$ for some constant $\beta > 0$, that $f(w)$ is bounded below on the level set S and Lipschitz continuously differentiable in the neighborhood $S(R_0)$ for some $R_0 > 0$, and that all solutions p_k of (5.12) satisify (5.13) and (5.14). Then we have

$$\liminf_{k \to \infty} \|g_k\|_2 = 0.$$

For proof, see Theorem 4.5 in [24].

Algorithm 8 Proposed second-order trust-region method

Given: For some $\Delta_{\max} > 0$, $\Delta_0 \in (0, \Delta_{\max})$, $\eta \in [0, \frac{1}{4})$, $\epsilon > 0$
while $\|\nabla f(w_k)\|_2 > \epsilon$ **do**
  Obtain p_k from CG-Steihaug
  Perform line search using **Wolfe conditions**
  Evaluate the **Levenberg-Marquardt** ratio given by
    $\rho_k = (f(w_k) - f(w_{k+1}))/(\mathcal{Q}_k(0) - \mathcal{Q}_k(p_k))$
  if $\rho_k < \frac{1}{4}$ **then**
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  else if $\rho_k > \frac{3}{4}$ **and** $\|p_k\|_2 = \Delta_k$ **then**
    $\Delta_{k+1} = \min(2\Delta_k, \Delta_{\max})$
  else
    $\Delta_{k+1} = \Delta_k$
  end if
  if $\rho > \eta$: **then**
    $w_{k+1} = w_k + p_k$
  else
    $w_{k+1} = w_k$
  end if
end while

5.3.3 Experimental Setup

In order to validate the effectiveness of the proposed algorithm, we implemented the optimization routine with the intent to train a neural network to perform a well established classification task. However, we imposed a limitation on the amount of data available for training in order to simulate data-starvation. The following section provides the guidelines under which we performed the associated experiments.

Neural Network Architecture

In each of these experiments, we use a Multi-Layer Perceptron (MLP). MLPs are a class of feed-forward artificial neural networks with the ability to separate data with a non-linear decision boundary [13].

The MLP used for experimentation was implemented using the deep learning package Theano and consisted of three layers of nodes/neurons. The input layer consisted of the vectorized version of the input image (784 nodes), where each node corresponds to each pixel in the sample image. The hidden layer contained 500 neurons and was followed by the non-linear hyperbolic tangent function (tanh) used as a non-linear activation on the output of the layer. Finally, the output layer consisted of 10 neurons corresponding to the 10 classes present in the the dataset. This layer was also followed by the same activation function as the previous layer. Finally, the softmax activation function was applied to the output of the neural network in order to provide a probability distribution of the possible classes. The probability distribution was compared to the true one-hot encoding of the target class using a cross entropy loss function. The architecture was kept simple

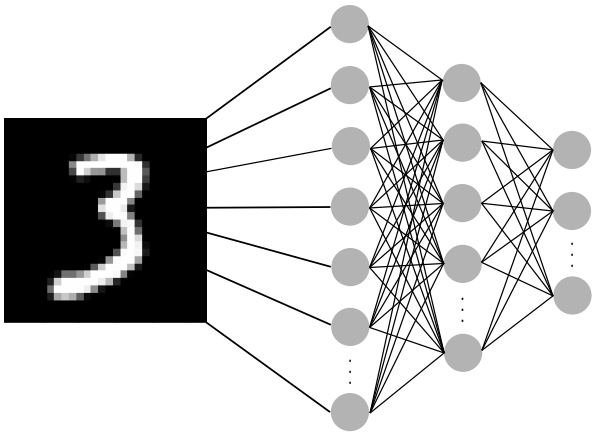


Figure 5.6: The architecture used to test the proposed method. First, the MNIST image is reshaped into a vector of 782×1. The hidden layer consists of 500 neurons and the output consists of a layer of 10 neurons.

to demonstrate the effectiveness of the optimization method on the intended task rather than the complexity of the MLP (see Fig. 5.6).

Dataset

The dataset used for these experiments is known as the MNIST dataset, which consists of rasterized 28×28 pixel images of handwritten digits from 0-9 (10 classes). The collection of images is partitioned in a training set of 60,000 images and a test set of 10,000 images. It is a subset of a larger set available from NIST [18].

Hardware

Experiments were carried out on GPUs housed in a high-performance cluster. The cluster consists of 95 computer nodes with a total of 2116 cores and 2301 Mhz processing power. The GPUs include 4 NVIDIA Tesla K20 graphics cards and 2 Nvidia p100s with a total operating capacity of 62 TFLOPS.

Testing Procedure

The goal of our experiments is to show that the proposed approach can improve on SGD when training a neural network with **limited data**. The 70,000 images in the MNIST dataset were partitioned in the following manner: 50,000 images in the training set, 10,000 images in the validation set and 10,000 images in the testing set. From the training set, we randomly sampled subsets with sizes of 20, 100, 500, 1000 and 10,000 images. When implementing our algorithm, the entire subset is used in a single batch over a single epoch. The algorithm terminates when the norm of the gradient of the objective function reaches a sufficient tolerance (in our case, this was set to 10^{-5}). When using SGD, we approached training using the standard practice of using minibatches. We chose a minibatch size of 20 images for all data subsets and tried various run times for the algorithm.

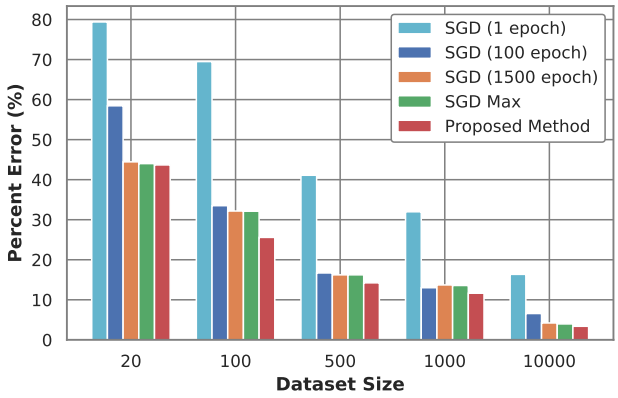


Figure 5.7: A comparison of our method to Stocastic Gradient Descent (SGD) with limited datasets. Here we report the accuracy error for datasets of 20, 100, 500, 1000 and 10000 images. Results are also shown for various numbers of epochs. The proposed approach was only trained for 1 epoch while SGD was allowed to run for 1 epoch, 100 epochs, 1500 epochs and what is referred to as SGD Max. SGD Max refers to the number of epochs of SGD allowed to run within the time our proposed method runs 1 epoch.

5.3.4 Results

Fig. 5.7 displays the results of our experiments. It is clear that when compared to SGD for various training times and various dataset sizes, the proposed method improves on the accuracy of the classification task. In particular, after a single pass (1 epoch) through the available data the proposed method is almost twice as accurate as SGD. This confirms the notion that the iterations of the proposed method have a greater impact than that of the first order method. As we continued to allow the network to train we can see that after almost 1500 epochs of SGD (which may be considered overtraining) we do not see a great improvement in the level of accuracy. **In fact, we even allowed the SGD algorithm to run in the same GPU time as our method, we still improve on the performance of SGD.** The results in Table 1 show that the improvement on SGD in the case of a 20 image dataset is less than 2%. It appears that in this case 20 images might approach the lower limit on the minimum size of the data set you need in order for the network to learn. We can see that as we increase the number of images in the dataset, SGD still underperforms in comparison to our method.

5.3.5 Conclusion

In this work we propose a novel algorithm for training neural networks using second order information for data-hungry inference. In particular, our algorithm improves on first order methods by allowing the use of curvature information in order to improve the quality of each iteration in the optimization method. This is accomplished by using a fast exact Hessian operation, which allows us to efficiently compute matrix-vector multiplication with the exact second derivative matrix. Unlike previous algorithms that have used this type of algorithm in a conjugate gradient setting, by using the CG-Steihaug approach in a trust-region setting it allows us to relax the requirement that the Hessian be positive definite. This allows the algorithm to consider negative curvature information and avoid saddle points. We proved and provided numerical results in a standard

Dataset size	Percent Error				
	SGD 1 epoch	**SGD 100 epochs**	**SGD 1500 epochs**	**SGD Max**	**Proposed Method**
20	79.39	58.45	44.45	44.01	**43.68**
100	69.49	33.50	32.17	32.12	**25.58**
500	41.10	16.70	16.23	16.23	**14.23**
1000	31.98	13.01	13.71	13.57	**11.64**
10000	16.35	6.57	4.21	3.97	**3.38**

Table 5.2: Error table corresponding to the testing error/loss of the neural network for our proposed method in comparison to SGD over various epochs and for different dataset sizes. For our proposed method, the dataset is fed as a batch to the network. For SGD, the data are fed in minibatches of 20 images. SGD Max corresponds to SGD trained over the same GPU run time as our proposed algorithm.

implementation of an MLP classification problem where the training dataset was limited. In all cases, the proposed method improved on a standard implementation of Stochastic Gradient Descent trained over various epochs.

5.4 Summary of Contribution

In this chapter we focused on the extension of optimization methods for training neural networks beyond gradient descent methods. We proposed two new algorithms incorporating curvature information about the objective function being minimized: (1) A Quasi-Newton method where we approximate second derivative information using using the gradient within a trust-region method. The trust region method improves our approximation of the objective function and promotes generality of the network to data outside of the training set, (2) a method which incorporates second derivative information without explicitly storing or computing the hessian. This was also presented in a trust-region setting where the associated subproblem was solved using a conjugate based method. The benefit of the method is that the quality of the optimization is improved and requires less iterations and less data to train the neural network.

Bibliography

[1] L. Adhikari et al. “Limited-memory trust-region methods for sparse relaxation”. *Proc.SPIE*. Vol. 10394. 2017, pp. 10394 - 10394 - 8. DOI: 10.1117/12.2271369. URL: https://doi.org/10.1117/12.2271369.

[2] L. B. Almeida. “A learning rule for asynchronous perceptrons with feedback in a combinatorial environment”. *Artificial neural networks: concept learning*. IEEE Press, 1990, pp. 102–111.

[3] L. Bottou, F. Curtis, and J. Nocedal. “Optimization Methods for Large-Scale Machine Learning”. *SIAM Review* 60.2 (2018), pp. 223–311. DOI: 10.1137/16M1080173.

[4] L. Bottou. “Large-scale machine learning with stochastic gradient descent”. *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[5] L. Bottou and O. Bousquet. “The tradeoffs of large scale learning”. *Advances in neural information processing systems*. 2008, pp. 161–168.

[6] R. H. Byrd, J. Nocedal, and R. B. Schnabel. “Representations of quasi-Newton matrices and their use in limited-memory methods”. *Math. Program.* 63 (1994), pp. 129–156.

[7] R. H. Byrd et al. “A stochastic quasi-Newton method for large-scale optimization”. *SIAM Journal on Optimization* 26.2 (2016), pp. 1008–1031.

[8] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2000, pp. xx+959. ISBN: 0-89871-460-5.

[9] F. Curtis. “A Self-Correcting Variable-Metric Algorithm for Stochastic Optimization”. *Proceedings of The 33rd International Conference on Machine Learning*. 2016, pp. 632–641.

[10] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. *Journal of machine learning research* 12.Jul (2011), pp. 2121–2159.

[11] J. B. Erway et al. “Trust-region algorithms for training responses: machine learning methods using indefinite Hessian approximations”. *Optimization Methods and Software* 0.0 (2019), pp. 1–28. DOI: 10.1080/10556788.2019.1624747.

[12] D. M. Gay. “Computing optimal locally constrained steps”. *SIAM Journal on Scientific and Statistical Computing* 2.2 (1981), pp. 186–197.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press Cambridge, 2016.

[14] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).

[15] Q. V. Le et al. “On optimization methods for deep learning”. *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress. 2011, pp. 265–272.

[16] Y. LeCun et al. “LeNet-5, convolutional neural networks”. *URL: http://yann. lecun. com/exdb/lenet* (2015), p. 20.

[17] Y. LeCun. “The MNIST database of handwritten digits”. *http://yann. lecun. com/exdb/mnist/* (1998).

[18] Y. LeCun, C. Cortes, and C. Burges. “MNIST handwritten digit database”. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010), p. 18.

[19] D. C. Liu and J. Nocedal. “On the limited memory BFGS method for large scale optimization”. *Math. Program.* 45 (1989), pp. 503–528.

[20] J. Martens. “Deep learning via Hessian-free optimization.” *ICML*. Vol. 27. 2010, pp. 735–742.

[21] M. F. Møller. “Exact calculation of the product of the hessian matrix of feed-forward network error functions and a vector in $O(n)$ time”. *DAIMI Report Series* 432 (1993).

[22] J. J. Moré and D. C. Sorensen. “Computing a trust region step”. *SIAM Journal on Scientific and Statistical Computing* 4.3 (1983), pp. 553–572.

[23] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. New York: Springer, 2006.

[24] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[25] B. A. Pearlmutter. “Fast exact multiplication by the Hessian”. *Neural computation* 6.1 (1994), pp. 147–160.

[26] F. J. Pineda. “Generalization of back-propagation to recurrent neural networks”. *Physical Review Letters* 59.19 (1987), p. 2229.

[27] J. Rafati, O. DeGuchy, and R. F. Marcia. “Trust-Region Minimization Algorithm for Training Responses (TRMinATR): The Rise of Machine Learning Techniques”. *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE. 2018, pp. 2015–2019.

[28] A. Ranganath et al. “Second-Order Helping: A Hessian-Free Approach for Data-Hungry Inference”. *Submitted* (2020).

[29] B. Recht et al. “Hogwild: A lock-free approach to parallelizing stochastic gradient descent”. *Advances in neural information processing systems*. 2011, pp. 693–701.

[30] N. N. Schraudolph. “Fast curvature matrix-vector products for second-order gradient descent”. *Neural computation* 14.7 (2002), pp. 1723–1738.

[31] I. Sutskever et al. “On the importance of initialization and momentum in deep learning”. *International conference on machine learning*. 2013, pp. 1139–1147.

[32] T. Tieleman and G. Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[33] P. Werbos. “Backpropagation: Past and future”. *Proceedings of the Second International Conference on Neural Network*. Vol. 1. IEEE. 1988, pp. 343–353.

[34] T. Zhang. “Solving large scale linear prediction problems using stochastic gradient descent algorithms”. *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 116.

[35] C. Zhu et al. “Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization”. *ACM Transactions on Mathematical Software (TOMS)* 23.4 (1997), pp. 550–560.

Chapter 6

Applications in Deep Learning

In this Chapter, we use the ideas presented in Chapter 4 in order to create deep neural networks for signal recovery in a variety of applications. We revisit an application explored in Chapter 3 in the context of using neural networks as opposed to classical optimization techniques. We also include models which solve problems in blind source separation, remote sensing and genomics. The common characteristic in all of these applications is that we are interested in recovering signals using neural networks.

6.1 Deep Neural Networks for Low-Resolution Photon-Limited Imaging

In this section we revisit the photon-limited problem presented in chapter 3. Under the process of photon-limited imaging, we seek to reconstruct signals from noisy low-dimensional observations. These signals – often sparse in some basis – allows for the use of penalty based algorithms to promote sparsity in its reconstruction [37, 64]. The key characteristic of this modality is that the observations contain low photon or light levels. This type of imaging process is typically found in applications such as medical imaging and night vision where measurements at the photon detector are corrupted by Poisson noise and thus modeled using the Poisson distribution [87]. Previous methods for solving the signal reconstruction problem include its reformulation into an optimization problem and use iterative methods in order to arrive at a solution [42, 38, 43, 49, 2, 13]. Deep neural network architectures have been used to effectively extract features from similar signals through the use of autoencoders and convolutional neural networks (CNN) [**krizhevsky2012imagenet**, 92, 85]. In this work, based on the paper by DeGuchy, Santiago, Banuelos and Marcia [36], we explore the use of various architectures in deep learning techniques as applied to the area of compressed sensing and establish their effectiveness in the field of photon-limited imaging.

6.1.1 Problem Formulation

In the context of photon limited imaging, the arrival of photons at the detector are modeled by the following inhomogeneous Poisson process

$$\mathbf{y} \sim \text{Poisson}(\mathbf{A}\mathbf{f}^*),$$

where $\mathbf{y} \in \mathbb{Z}_+^m$ is the observation vector whose entries consists of photon counts, $\mathbf{f}^* \in \mathbb{R}_+^n$ is the true signal and $\mathbf{A}_+^{m\times n}$ is the system matrix projecting the true signal to the observation space with $m \ll n$. Our interest is in recovering the higher dimensional signal $\mathbf{f}^*$ given the lower dimensional observation vector $\mathbf{y}$. Existing recovery methods use the maximum likelihood principle to maximize the probability of observing the vector $\mathbf{y}$. Furthermore, under the assumption that $\mathbf{f}^*$ is sparse, a sparsity promoting penalty term is incorporated in the reconstruction $\hat{\mathbf{f}}$. These iterative algorithms require tuning parameters associated with the choice and enforcement of penalties and also require a substantial number of iterations to recover the signal [48, 66, 11].

We seek to avoid the complications associated with the current iterative optimization methods by solving the sparsity-promoting Poisson reconstruction problem using a variety of deep learning architectures. We accomplish this by training neural networks to process the low dimensional input $\mathbf{y}$ and provide a reconstruction of the true signal $\mathbf{f}^*$. Recently, deep learning techniques have been implemented separately for image reconstruction from downsampled observations and for Poisson denoising problems [75, 74, 81]. The novelty of the proposed architecture is that it solves both problems simultaneously as is required in many photon-limited applications.

6.1.2 Deep Learning Architectures

We propose three different neural network configurations with the purpose of recovering data from noisy low dimensional observations. One architecture employs the use of a convolution neural network (CNN) while the other two take advantage of the structure of autoencoders. As is common in these types of networks, all three implementations were trained using backpropagation and the mean squared error (MSE) was used as a loss function.

Poisson Inverting ConvolutionS (PICS)

The first implementation is a CNN based on the neural network known as DeepInverse [74]. The DeepInverse architecture was not intended to denoise the signal, but rather recover signals from compression. In [74] the disparity between the dimension of the observation and the true signal is circumvented by the use of the proxy signal $\hat{x} = \mathbf{A}^T y$. The proxy is fed into the neural network and padding is used to keep the signal dimension consistence throughout. Instead of assuming explicit access to a measurement matrix, we added an extra fully connected layer in place of the proxy signal. This transformation serves two purposes. First, it increases the size of the observation vector $\mathbf{y}$ to the dimensions of the original signal. Secondly, it allows us to learn the transformation to the compressed space. There are three primary layers to this CNN architecture. The first consists of 64 filters of size $11 \times 11 \times 1$ with the last dimension pertaining to the depth of the filter . The next layer consists of 32 filters each with a dimension of $11 \times 11 \times 64$. The final layer consists of one $11 \times 11 \times 32$ filter to output the original image. After each layer a ReLU nonlinearity is applied to the output. The network was originally trained in the literature using 64×64 natural images and the preliminary results showed that this method was not accommodating to our database. Alternatively, the structure was modified using 2×2 filters while the number of filters was modified to reflect the scale of the input data and the required final output. This resulted in a slightly different architecture that was allowed to scale with the size of the input vector. Because these changes were not reflected in the literature, we will refer to the modified network as Poisson Inverting ConvolutionS (PICS).

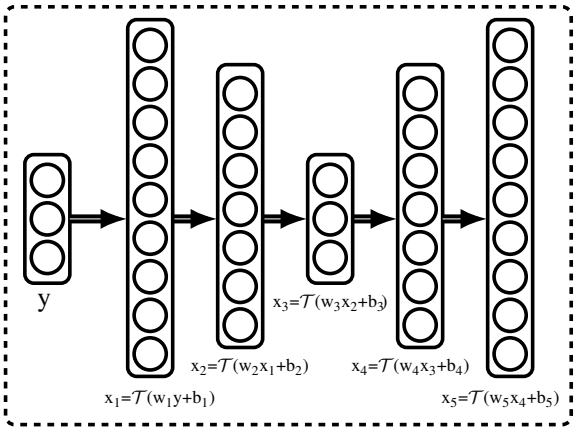


Figure 6.1: Our proposed Poisson Autoencoder Inverting Network (PAIN). Each encoder and decoder consists of two layers. This framework also incorporates the sigmoidal activation function $\mathcal{T} = 1/(1 + e^{-x})$.

Stacked Denoising Autoencoders (SDA)

The authors in [75] use stacked denoising autoencoders to learn and recover the structure of sparse signals, again the intention of the method did not include a denoising component. This type of architecture was introduced in [93] as a way to make the learning capabilities more robust by introducing noise to the input before each autoencoding layer. The architecture involves stacking a decoder before encoding and decoding once again to arrive at the dimension of the true signal. Layers are differentiated by the dimensions of the weight matrix and bias vectors. After the dimension has been increased or decreased, the sigmoid function is applied to the output. The Stacked Denoising Autoencoder (SDA) structure implicitly resolves the increase in dimension eliminating the need for modifications.

Poisson Autoencoder Inverting Network (PAIN)

The final architecture we propose in this work is Poisson Autoencoder Inverting Network (PAIN). Similar networks have been effective in the application of image compression [8]. The architecture is similar to SDA, the difference being that each decoding and encoding layer consists of multiple layers. Under this structure, compression and decompression is done gradually. The intuition is that the image must past through more layers and therefore the compression is refined at each layer, making the encoding more impervious to noise. In order to adapt the architecture to the sparse Poisson reconstruction problem, we required two modifications. The first change involves initializing the weight matrix using a truncated normal distribution instead of random samples from a normal distribution. The truncated normal initialization eliminates values more than 2 standard deviations from the mean. This immediately improved the architecture's ability to denoise the given signal. The network also utilizes a single layer decoder to initialize the process. The network starts with a layer that boosts the dimension of the observation to the dimension of the true signal. The next layer is a double encoder that reduces the dimension of the true signal to a length of 256 and then to the dimension of the observation. The dual layer decoder then brings the dimension back to 256 and finally to the dimension of the true signal (see Fig. 6.1).

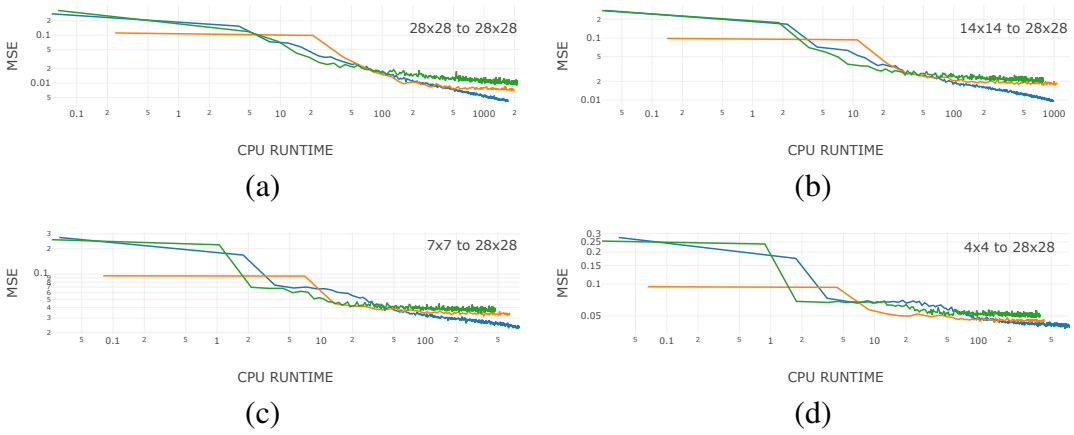


Figure 6.2: The evolution of the MSE versus CPU runtime for PAIN (our proposed method, in blue), PICS (in orange), and SDA (in green) on a log-log scale for the purpose of reconstructing 28×28 images from (a) 28×28, (b) 14×14, (c) 7×7, and (d) 4×4 Poisson realizations. All CPU time is recorded in seconds.

6.1.3 Results

The proposed architectures PICS, PAIN and SDA were all implemented using the open source machine learning language Tensorflow. Training and testing of the neural networks was performed using a quad core Intel i7-6700 CPU on a local PC with 64 GB of ram. The networks were trained using the stochastic gradient descent method known as RMSprop [82, 83].

MNIST Dataset

The MNIST data set first used in [65] was altered in order to create pairs of signals to fit the Poisson model. The dataset consists of 70,000 28×28 images of handwritten numbers from 0-9 and their associated classification labels. From this dataset, 60,000 examples are used for training and 10,000 examples are used for testing. The reconstruction problem does not make use of the labels because we are not concerned with classification. For the purposes of this work, we consider the original 28×28 image as the true signal $\mathbf{f}^*$. The associated observation vectors are created by taking the mean average of blocks of pixels, taking care that the size of the blocks reduce the size of the image without the need for padding. We then impose Poisson noise on the downsampled signal. Data sets were created by pairing true signals with observational signals of varying size. Under this structure the neural network is expected to train on a set of noisy observations with a fixed dimension ($n \times n$ with $n \in \{4, 7, 14, 28\}$) and reconstruct the full image (28×28).

Performance

The proposed architectures were able to perform suitable reconstructions of the test data sets. Both the MSE and the CPU runtime were used to quantify the effectiveness of the architectures during training (Fig. 6.2). Both PICS and PAIN achieved a lower MSE over the training sets than the SDA architecture for observation vectors in all dimensions. Furthermore, PICS completed 10,000 iterations (10,000 iterations are reported since the MSE does not show significant

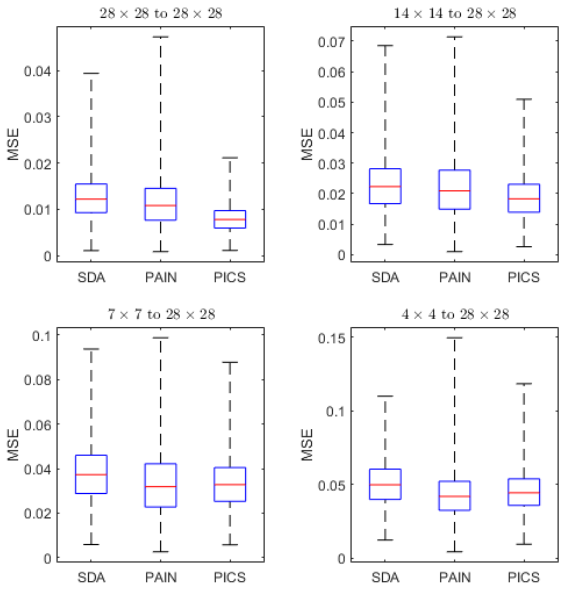


Figure 6.3: Boxplots comparing mean squared error (MSE) computed using the MNIST validation images (5,000) and their reconstructions for the three architectures proposed (SDA, PAIN, and PICS). We observe that all three architectures behave similarly in terms of their MSE as the amount of compression increases.

improvement after this number) in the same computational time required for 45,000 iterations of PAIN and SDA. This is expected as PICS is a CNN and is therefore a more computationally intensive process than the other two methods. Although the PICS architecture takes longer to complete a given number of training iterations, it reaches a lower MSE faster than SDA or PAIN. This can been seen in the restoration of 28 × 28 and 14 × 14 images. This shows that PICS is learning at a faster rate, making each training step more valuable when compared to the other architectures. While the CNN structure is initially more accurate than the autoencoder structures, a decrease in input dimension results in competitive MSE from the less complex PAIN structure. The output for a set of input digits is displayed in Fig. 6.9. As the amount of compression increases, the quality of the reconstruction decreases for all three types of architecture. The PAIN and PICS architectures clearly outperform the SDA when it comes to the 4 × 4 compression. The surprising observation is that PAIN seems to have a higher intensity and smoothing effect when it comes to recreating the initial image. This is counterintuitive considering that we expect higher intensity from PICS since it uses the ReLU activation function.

The models were also tested on the 5,000 entries of the MNIST validation set. All MSE scores comparing the original signal and the reconstructed signal were computed and the results are presented in Fig. 6.3. We note that the MSE for all three architectures is skewed towards a lower MSE value. Also, we observe that as the amount of compression increases, the SDA and PAIN architectures become comparable to the more computationally intensive PICS architecture.

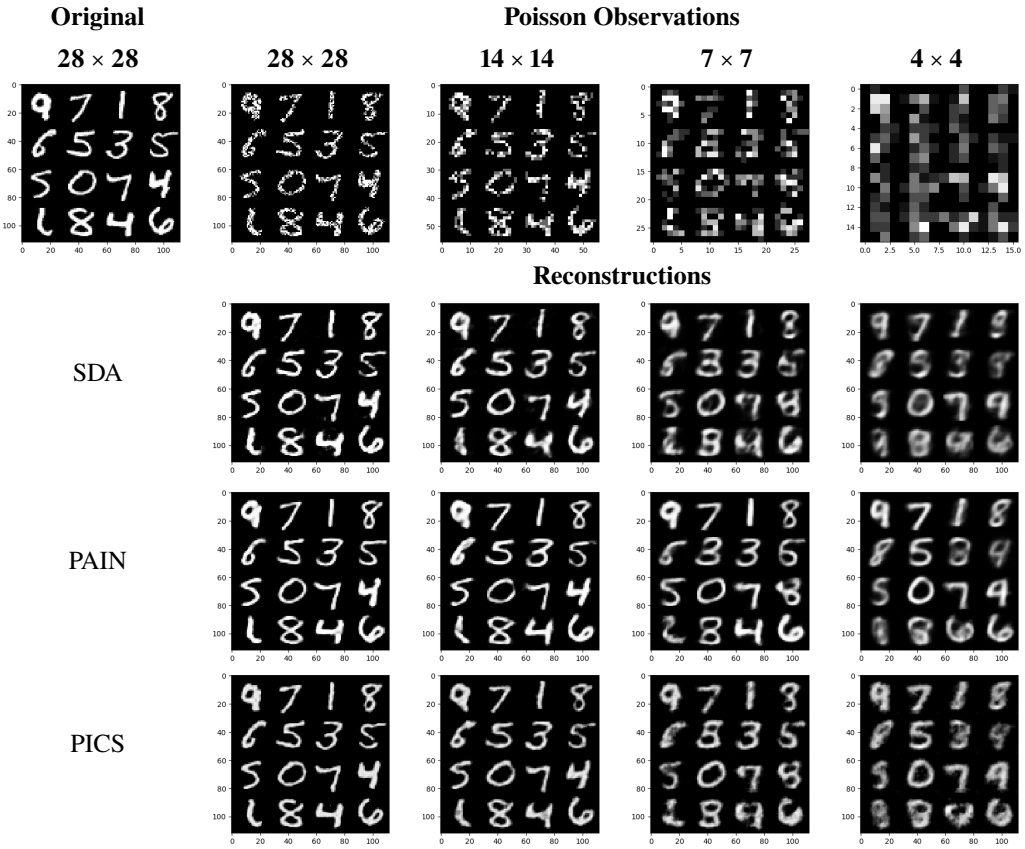


Figure 6.4: The first row is comprised of 16 original MNIST images and their corresponding downsampled Poisson realizations. Given these input images, we present the reconstructions using Stacked Denoising Autoencoders (SDA), our proposed methods Poisson Autoencoder Inverting Network (PAIN) and Poisson Inverting ConvolutionS (PICS) for each dimension of images.

6.1.4 Conclusion

In this work we implemented three deep learning architectures to solve the Poisson inverse problem. These neural networks have proven to be very effective in the reconstruction of images under the same modality. The first two networks involved modified autoencoders, while the third used a convolutional neural network. The results show that the stacked denoising autoencoder did not perform as well as the PAIN and PICS networks during training using the Mean Squared Error (MSE) as a performance metric. PAIN also has the benefit of being less computationally intensive, which could suggest that it will scale better with larger image sizes. Furthermore, PAIN has a smoothing characteristic in the reconstructions which is not reflected in the MSE. While the image smoothing could negatively effect the MSE, this property could have advantages during classification.

6.2 Image Disambiguation with Deep Neural Networks

Applications such as blind source separation [32] and multiplexed imaging [89] involve measurement data that are composed of multiple signals that have been observed simultaneously and combined at the detector stage. For example, in blind source separation settings, a microphone might pick up several conversations concurrently, or perhaps there might be one predominant signal mixed with ambient or latent sounds. In multiplexed imaging, optical systems might utilize beam splitters and mirrors to superimpose different scene components onto a single focal plane array (FPA). This type of architecture is particularly useful in settings where a wide field-of-view is needed but the FPA size is limited. The main challenge in these applications is to separate the combined measurements into distinct signals or perhaps simply isolate a particular signal. These problems are highly underdetermined and ill-posed, and, as such, they require sophisticated numerical methods. The methods we propose in this work use machine learning techniques,specifically autoencoders, and is based on the paper by DeGuchy, Ho, and Marcia [35].

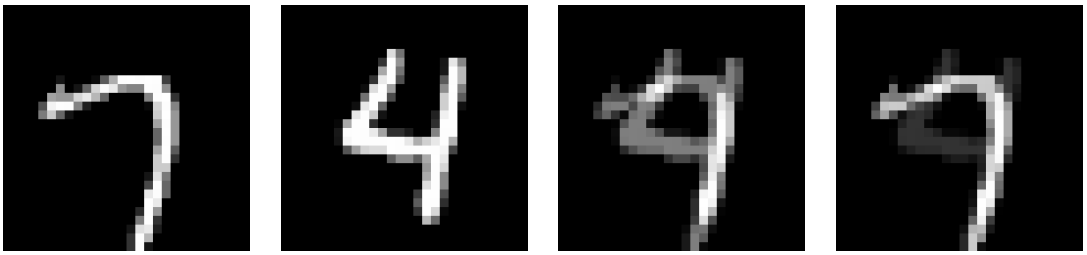
Related Methods

The blind source separation problem has been well studied [32] and has been analyzed from a statistical perspective [16, 21]. Algorithms include leveraging sparse decomposition [100] and on-line learning [5]. Multiplexed optical systems have been physically implemented [22, 91, 53, 84], and algorithms for multiplexed imaging include those that use nonnegative matrix factorization [30] and those that exploit a priori knowledge about the multiplexed images, such as sparse representation [18, 70]. In contrast, the approaches proposed in this work incorporate existing datasets to train deep neural networks for disambiguating the superimposed images.

6.2.1 Methods

Neural Network Architecture

We propose the following neural network configuration for the purpose of recovering two images from a mixed signal measurement. The architecture is based on the deep learning building block known as an autoencoder [93]. In its simplest form, an autoencoder is composed of two parts. The first is the encoder whose objective is to reduce the dimensionality of the input by providing a latent space representation of the most pertinent information. The second



(a) (b) (c) (d)

Figure 6.5: Image disambiguation setup. Two 28 × 28 pixel images (a) and (b) are randomly chosen from the MNIST dataset. Images (a) and (b) are superimposed to yield measurement image (c). Image (d) is the result of superimposing images (a) and (b) with image (b) at 25% intensity.

element, referred to as a decoder, is tasked with interpreting the resultant latent space variable and ultimately recovering the original input [47]. In practice, more complex evolutions of the autoencoder such as the variational autoencoder and stacked denoising autoencoders are typically implemented [59, 92]. Beyond modifications of the overall structure of the autoencoder, there is also a choice between the commonly used fully connected layer and the convolutional layer as the basis of the encoder and decoder substructures [47, 71]. The method presented in this work utilizes stacked denoising autoencoders composed of fully connected layers. The motivation for this implementation is two fold. Stacked denoising autoencoders (SDAs) have been successfully implemented in a variety of other image processing tasks [68, 75, 94, 96, 97]. As an extension to their autoencoder ancestor, SDAs continually encode and decode the information until the intended output is obtained. The intuition is that as they are forced to compress and decompress the input, they become impervious to noise during the process (see Fig. 6.6). Our intent was to take advantage of this property during the image extracting process, resulting in smoother, noiseless reconstructions. Fully connected layers were chosen based on the configuration of the problem. Because the goal of the application is multiple image extraction from a single source, the architecture needs to be accommodating to an output of two images. Initially, convolutional layers were tested with a single channel input image and a dual channel output tensor. The network often returned the same image for both channels. As an alternative, the images were vectorized necessitating the use of fully connected layers. This allowed the network to produce a single vector containing both reconstructions.

Network Parameters

The network used in this work begins by reshaping the single channel $n \times n$ combined images into the vector $p = [p_1\ p_2\ p_3\ \dots\ p_{n^2}]$ where p_i represents the pixel intensity at a given location (see Fig. 6.6). The input layer is followed by 3 stacked autoencoders where the input from the previous layer is either compressed to length m or decompressed to the size n^2. The dimension of compression (m) is a hyperparameter which requires tuning, but the dimension n is constrained to the size of the image being processed. The final layer produces the vector $o = [o_1\ o_2\ o_3\ \dots\ o_{2n^2}]$ containing the separated images. The two images are recovered by dissociating the output vector into the components $\tilde{p}_1 = [o_1\ \dots\ o_{n^2}]$ and $\tilde{p}_2 = [o_{n^2+1}\ \dots\ o_{2n^2}]$ and reshaping each vector into the original $n \times n$ dimensions. After all but the final layer, the rectified linear unit (ReLU) activation function was applied to the output before being passed as the input of the next layer.

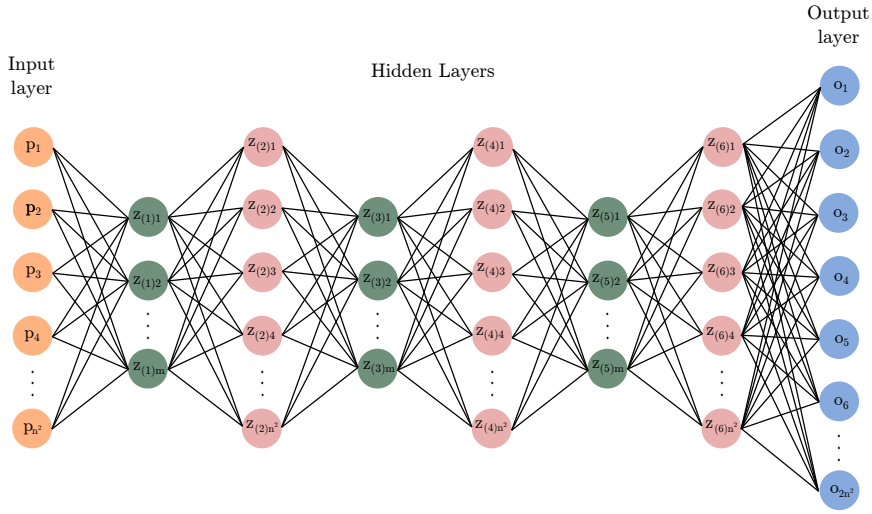


Figure 6.6: Deep neural network for image disambiguation. The network processes the n^2 input p through six fully connected layers $z_{(1)} - z_{(6)}$. The output layer o doubles the size of the input to allow for the extraction of the two images.

After the output layer the, sigmoidal activiation function is applied to ensure that the output pixel intensities are within the range 0 to 1. The network was implemented using two different cost functions which will be discussed in a later section. In either case the network was trained using the backpropagation algorithm.

6.2.2 Numerical Experiments

The architecture was developed using the open source machine learning library for Python, PyTorch. Training and testing were done using an NVIDIA Tesla P100 PCI-E GPU on the MERCED cluster. The loss functions were minimized using the PyTorch implementation of the Adam optimizer [60].

MNIST Dataset

The dataset used to test and train the proposed architecture is a modified form of the original MNIST data set [65]. The data set consists of 70,000 28 × 28 images of handwritten numbers from 0-9. The data is then partitioned into 60,000 training examples and 10,000 testing examples. For the purposes of this work, classification is not the intended objective and therefore all of the labels were disregarded. Each training and testing sample were created by first randomly selecting two images from the dataset without replacement. The images were normalized to assure that the pixel intensities ranged from 0 to 1 and then converted to a single channel tensor. The two tensors were then added together and the result was normalized and paired with the original images creating a triplet consisting of a combined image and its two sources (see Fig. 6.5). Because of the nature of the previously described image selection, the new training set consisted of 30,000 training instances and 5,000 testing images where both the combinations and the individual images are unique. Multiple datasets were created following a similar protocol, the difference being the intensity of one of the two target images was reduced to 75%, 50% or 25%. When describing these

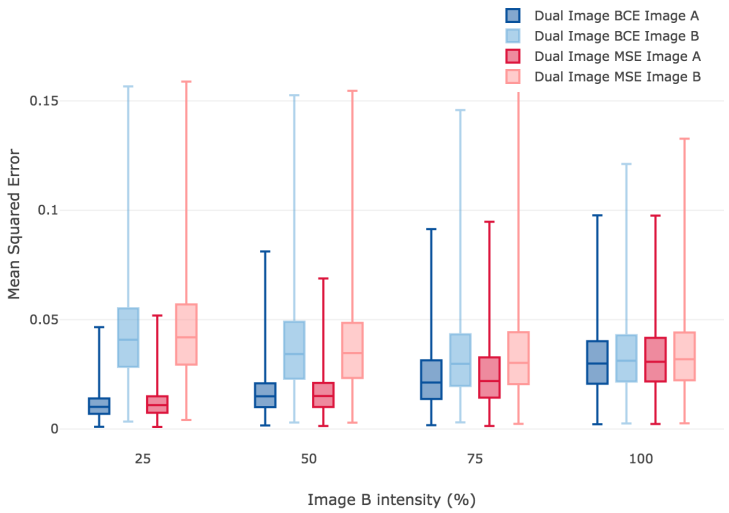


Figure 6.7: Boxplots of the Mean Squared Error (MSE) for 5,000 MNIST test images using the Dual Image Recovery method with the Binary Cross Entropy (BCE) [in blue] and the MSE [in red] loss functions. The MSEs are reported for both Images A and B under varying intensities of Image B in the measurements. As the intensity of Image B weakens, the accuracy of its recovery naturally worsens while the accuracy of Image A's recovery improves.

data sets the image with the full intensity is noted as Image A and the image with the reduced intensity will be known as Image B.

6.2.3 Performance

The proposed architecture was tested for a variety of configurations. In what will be referred to as the Single Image Recovery experiment, the intent was to recover only one of the two images. The image target was chosen randomly from the two potential candidates and the output layer in Fig. 6.6 was removed so that $z_{(6)}$ would provide an extraction of the appropriate size. The ReLU activation function implemented in $z_{(6)}$ was also substituted for the sigmoidal activation function. The Dual Image Recovery Experiment seeks to extract both images using the architecture described in Sec. 6.2.1. In both experiments, the dimension of the latent space was chosen to be $m = 256$ after hyperparameter tuning revealed this to be the optimal setting. All test images were compared to their targets using the Mean Squared Error (MSE) after the model was trained.

The normalization of the pixel intensities within a range of 0 to 1 allows us to use the Binary Cross Entropy (BCE) function as a loss function during training. Experiments were performed to determine if there was an advantage in the choice of using either the BCE or the MSE in a learning capacity. The performance of the two loss functions are comparable when using the MSE as a metric of validation on the test set (see Fig. 6.7). The reconstructions using the BCE had a slightly sharper appearance. This is to be expected as the MSE tends to average the pixel intensities resulting in a smoother image. For the remainder of this work we present only the results using the BCE loss function. The Dual Image Recovery experiment indicates that image extraction of both images are comparable when they are at full intensity. As the image intensity of Image B decreases, the architecture improves on its ability to extract Image A at the cost of

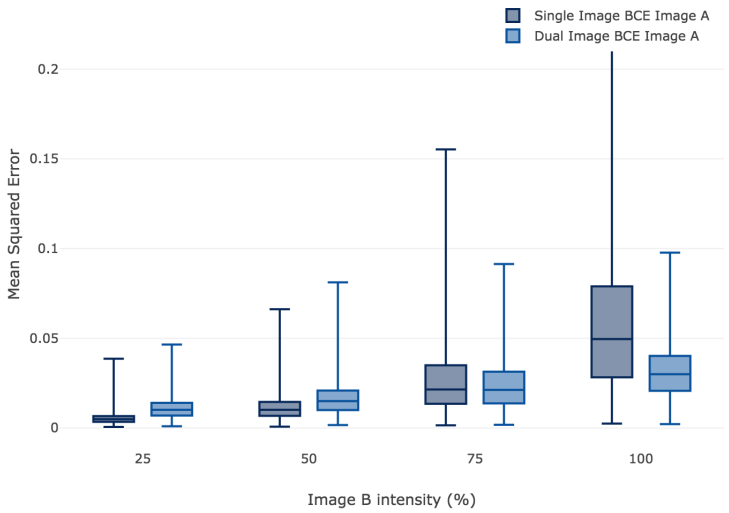


Figure 6.8: Boxplots comparing the Dual and Single Image Recovery methods to recover Image A using the Binary Cross Entropy (BCE) loss function. The MSE is reported for the recovered Image A from measurements with varying intensities of Image B.

an accurate reconstruction of Image B (see Fig. 6.7). This is to be expected as the intensity of Image B is weaker. The Single Image Recovery approach underperforms in comparison to Dual Image Recovery approach when both images have the same intensity. The Single Image Recovery approach is prone to recovering more elements from Image B. As the intensity of Image B decreases, the extraction of Image A improves, ultimately outperforming the Dual Image Recovery method (see Fig. 6.8). It is under the previously stated conditions that the Single Image Recovery approach views Image B as noise and performs its intended task as that of denoising.

6.2.4 Conclusion

In this work we implemented a deep neural network in order to solve the ill-posed image separation problem. By relying on the denoising properties of the stacked autoencoder structure, we have shown that these types of networks can be effective in the recovery of two superimposed images. While the choice of the loss function between the mean squared error and the binary cross entropy function is negligible in terms of MSE validation, the binary cross entropy function provides a slightly sharper advantage. The results show that when the objective is the recovery of both images, the reconstruction of the secondary image with a weaker intensity (Image B) slightly suffers while the recovery of the primary image (Image A) improves. By shifting the focus to a single image extraction the quality of the reconstruction of the primary image improves on the dual image extraction method.

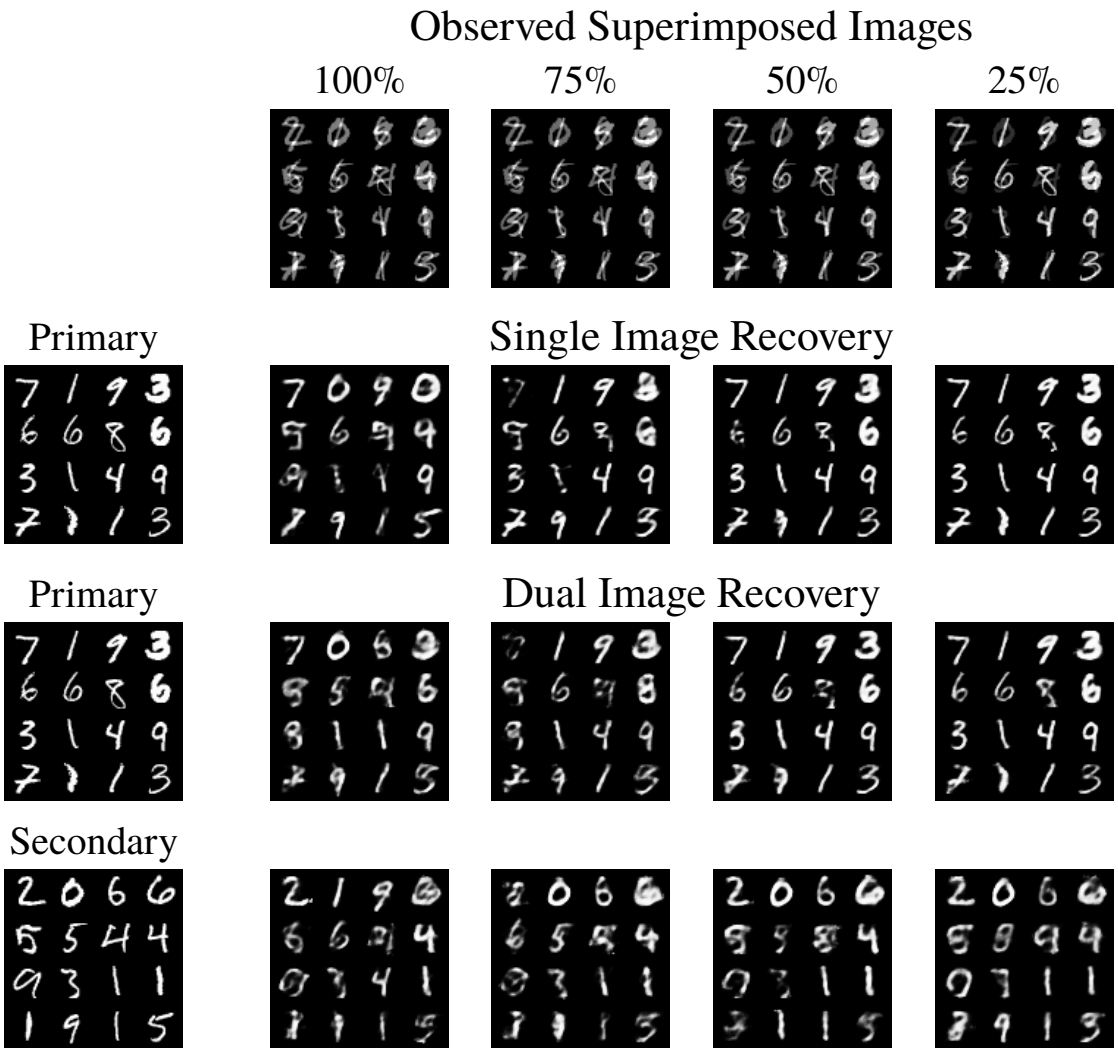


Figure 6.9: The first row contains the superimposed images which are composed of the primary and secondary images. They vary in that the intensity of the secondary image was altered to the percentage noted above. Given these input images, we present the reconstructions for the dual image recovery experiments and the single image recovery experiments.

6.3 Machine Learning for Direct and Inverse Scattering in Synthetic Aperture Radar

When considering the success of deep learning in applications such as image classification [63], segmentation [46] and object detection [67], it is no surprise that it has also made its way in remote sensing applications. We refer to [99] and references therein for a recent overview on the subject.

Typically, the primary use of deep learning in remote sensing has been for image classification. In this work, based on the paper by DeGuchy, Kim, Marcia, and Tsogka [34], we follow a different perspective and consider machine learning for solving the forward and inverse scattering problems associated with the remote sensing application of synthetic aperture radar (SAR). The inverse scattering problem consists in recovering the unknown reflectivity by probing the imaging scene with several excitations. Using deep learning in this context has been considered in [98] where the deep network plays the role of a non-linear forward model that captures the process that transforms reflectivities to measurements. Moreover, in [98] a recurrent auto-encoder is designed using a recurrent neural network (RNN) architecture that allows for image reconstruction. The proposed architecture uses the data directly and does not require any ground truth image knowledge. It assumes however partial knowledge about the imaging system. Specifically, the passive SAR problem is considered where the illuminations are opportunistic and assumed unknown while the receiver flight path and characteristics are known.

In a different setting with applications in seismic imaging, Switchnet was designed in [58]. This is a neural network that was specifically designed for solving forward and inverse scattering problems for the Helmholtz equation. Building on certain low-rank property that arises in the linearized operators, the network desinged in [58] replaces the use of fully-connected layers by a low complexity switch layer. Supervised learning assuming a large number of ground truth data was used to train Switchnet and good reconstruction results for noise free data were reported in [58].

The approach that we follow here is based on the observation that in most applications the forward map that transforms reflectivities to measurements can be accurately represented using a linearized single scattering model. This allows us to restrict the network architecture and simplify the learning process. More precisely, the first step in our approach consists in formulating the inverse problem as a linear system of the form

$$A\varrho = d$$

where d is the measured data, ϱ the unknown reflectivity and A is the sensing matrix that depends on the characteristics of the imaging system, such as the measurement process, the excitations used and the geometric configuration of the imaging scene. In all generality the sensing matrix is unknown and the reflectivity is reconstructed by solving instead the system

$$A_m\varrho = d + \delta d$$

where δd is the noise and A_m is our model for the sensing matrix. The quality of the reconstruction depends of course on the noise level but it also depends critically on how close the model sensing matrix is to the true one. The idea that we pursue in this work is to use machine learning so as to recover the sensing matrix A or its inverse from a set of data corresponding to known reflectivities. To keep the machine learning algorithm simple and interpretable we use a

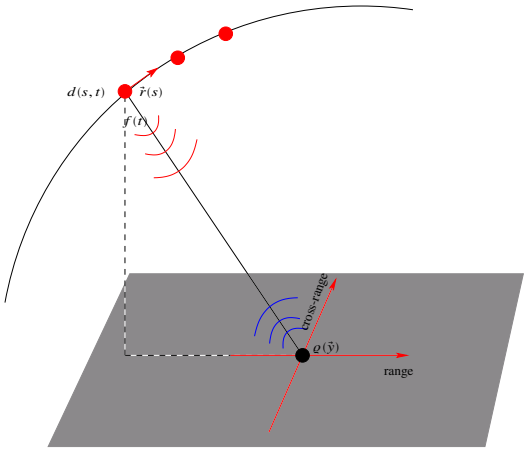


Figure 6.10: Setup for synthetic aperture radar imaging.

single, multiple connected layer. This way the weights of the network correspond to the matrix A or its inverse that we seek to recover. We show with numerical simulations using CIFAR-10 data set [62] that the learned version of the sensing matrix is close to the true one while its learned inverse leads to very good reconstructions of the reflectivity. Moreover the performance is robust over a large range of SNR values.

In this work, we assume that ground truth is known for a quite large set of images while nothing is known about the imaging system. In the future we need to consider the interplay between increasing our knowledge about the imaging system while decreasing the available data set of ground truth images.

6.3.1 Synthetic Aperture Radar

In this section we introduce the synthetic aperture radar (SAR) imaging problem. In SAR typically a single transmitter/receiver is used to collect the scattered electromagnetic field over a synthetic aperture that is created by a moving platform [27, 28, 72]. At each measurement location the transmitter emits a broadband pulse $f(t)$ and the corresponding echoes are recorded. The SAR data $d(s,t)$ depend on the slow time s that parametrizes the location $\vec{r}(s)$ of the platform at time s and the fast time t in which the round-trip travel time between the platform and the imaging scene on the ground is measured.

Although SAR uses a single transmit/receive element, high resolution images of the probed scene can be obtained because the data are coherently processed over a large synthetic aperture created by the moving platform. As illustrated in Figure 6.10, the platform is moving along a trajectory probing the imaging scene by sending a pulse $f(t)$ and collecting the corresponding echoes. We call range the direction that is obtained by projecting on the imaging plane the vector that connects the center of the imaging region to the central platform location. Cross-range is the direction that is orthogonal to the range. Denoting by a the length of the synthetic aperture and by B the available bandwidth, the typical resolution of the imaging system is c/B in range and $\lambda L/a$ in cross-range. Here c is the speed of light and λ the wavelength corresponding to the central frequency while L denotes the distance between the platform and the imaging region.

The start-stop approximation is typically made in SAR which consists in neglecting the targets' and the platfrom's displacement during the travel time between the emission and the reception

of the echoes. This approximation is true in radar since the speed of light is orders of magnitude larger than the speed of the targets and the platform.

Direct Scattering Problem

Denoting by $\varrho(\vec{y})$ the unknown reflectivity of the imaging scene, we use the following scalar wave equation model to describe the SAR data in the frequency domain,

$$d(s,\omega) = f(\omega) \int d\vec{y}\, \varrho(\vec{y}) \frac{e^{2i\omega\tau(s,\vec{y})}}{(4\pi|\vec{r}(s)-\vec{y}|)^2}. \tag{6.1}$$

Here $d(s,\omega)$ and $f(\omega)$ denote the Fourier transform of $d(s,t)$ and $f(t)$ respectively. The following definition for the Fourier transform of any finite energy signal $g(t)$ is used

$$g(t) = \int \frac{d\omega}{2\pi} g(\omega) e^{-i\omega t}, \tag{6.2}$$

where ω is the angular frequency. In (6.1), $\tau(s,\vec{y})$ is the travel time between the platform and the point $\vec{y}$, at slow time s

$$\tau(s,\vec{y}) = \frac{|\vec{r}(s)-\vec{y}|}{c}. \tag{6.3}$$

The model (6.1) is derived under the Born approximation and neglects multiple scattering. It assumes that the reflectivity is isotropic and frequency independent. Although quite simple, this scalar wave equation model is frequently used in SAR [28, 20] since it captures the main features of the scattering problem. To estimate the reflectivity, a discretization of the imaging window (IW) is considered, composed of the grid points $\vec{y}_j, j = 1,\dots,K$. By discretizing (6.1), the SAR imaging problem reduces to solving the following linear system of equations,

$$A\varrho = d. \tag{6.4}$$

Inverse Scattering Problem

The goal for the inverse scattering problem is to recover the reflectivity vector $\varrho \in \mathbb{C}^K$ whose components are the values of the reflectivity on the grid points $\varrho(\vec{y}_j), j = 1,\dots,K$. By discretizing the SAR data $d(s,\omega)$ we form the vector $d \in \mathbb{C}^N$ defined as

$$d = (d_i)_{i=1,\dots,N} = \left(d(s_m,\omega_l)\right)_{m=1,\dots N_s, l=1,\dots,N_\omega}, \tag{6.5}$$

$$i = i(m,l) = (l-1)N_s + m, N = N_s N_\omega, \tag{6.6}$$

with $s_m, m = 1,\dots,N_s$ the slow time samples and $\omega_l, l = 1,\dots,N_\omega$ the frequencies in the available bandwidth. The entries of the sensing matrix $A \in \mathbb{C}^{N\times K}$ follow from (6.1) using (6.6)

$$A_{ij} = f(\omega_l) \frac{e^{2i\omega_l\tau(s_m,\vec{y}_j)}}{(4\pi|\vec{r}(s_m)-\vec{y}_j|)^2}, \tag{6.7}$$

for $i = 1,\dots,N$ with $i = i(m,l)$ as defined in (6.6) and $j = 1,\dots,K$.

In SAR the system (6.4) is usually overdetermined with $N \gg K$, and the least squares solution can be obtained as

$$\hat{\varrho}_{\ell_2} = Bd, \tag{6.8}$$

with B denoting the pseudo-inverse of A, defined as $B = (A^H A)^{-1} A^H$, when A is full column rank and where A^H denotes the complex conjugate transpose of A. It is well known that the choice of the discretization of the IW plays an important role in the inversion of (6.4). The common choice consists in discretizing the IW in steps of the order of the expected resolution of the imaging system (cf. [20]). In this case the matrix $A^H A$ is close to a diagonal matrix which justifies the SAR inversion formula,

$$\hat{\varrho}_{\text{SAR}} = A^H d. \tag{6.9}$$

When the reflectivity is sparse, i.e., the imaging scene consists of a small number of localized scatterers, a better estimate of ϱ can be obtained using ℓ_1 optimization. We refer to [14, 41, 79] for studies of SAR imaging with ℓ_1 optimization. We also refer to [20] where an ℓ_1 formulation for a frequency and direction dependent reflectivity was proposed.

6.3.2 Machine Learning for Synthetic Aperture Radar

The exact sensing matrix A appearing in (6.4) for the direct scattering problem in SAR and in (6.8) and (6.9) for the inverse scattering problem in SAR is not known in general. The expression in (6.7) is a physical model that is commonly used in SAR applications (cf. [28, 98]). Nonetheless, it gives only an approximation of the sensing matrix based on the Born approximation. Alternatively, we consider here using machine learning to recover A from training data.

In what follows, we assume a flat imaging scene on the ground. Suppose that we have a set of reflectivities $\{\varrho_p\}$ and associated measurements $\{d_p\}$ corresponding to the same imaging system and settings. Assuming that there exists a linear mapping from each ϱ_p to the corresponding d_p, we use machine learning methods to recover an approximation of the sensing matrix, which we denote here by A_L. With that learned sensing matrix, we form images using measurements outside of this training set by replacing A in (6.9) with A_L. Alternatively, using the same machine learning methodology, we can recover an approximate "inverse," which we denote by B_L, by seeking a linear mapping from each d_p to the corresponding ϱ_p in the training set. This approximate inverse can then be used to form images using measurements outside of this training set. We give the details of this procedure below.

SAR Sensing Matrix

We give below the precise parameters used for the model matrix A in the numerical simulations. The flight path follows a circular trajectory with radius $R = 7.1$km at height $H = 7.3$km. The platform is moving with a speed of 70m/s and the total length covered is $a = 325.5$m corresponding to a slow time interval of 4.65s. The probing pulse has a flat spectrum over the available bandwidth of 1.2GHz and a central frequency of 9.6GHz. These parameters result to an imaging system with resolution of 0.24m in range and 0.97m in cross-range. The image dimension is 32×32 pixels with a pixel size of 25cm $\times$25cm. The matrix A is square with $K = N = 1024$ corresponding to data for $N_s = 32$ slow time samples and $N_\omega = 32$ frequencies in the available bandwidth.

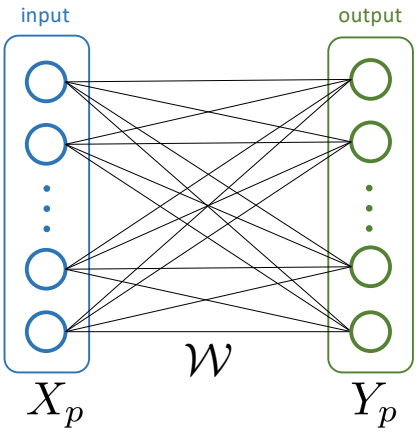


Figure 6.11: The network configuration used to recover the mapping between the reflectivity data $\tilde{\varrho}_p$ and the measurement data $\tilde{\mathrm{d}}_p$ is a single, fully connected layer network. When $\tilde{\varrho}_p$ is the input data, *i.e.* $X_p = \tilde{\varrho}_p$ and $\tilde{\mathrm{d}}_p$ is the output data, *i.e.* $Y_p = \tilde{\mathrm{d}}_p$, then the learned weights in $\mathcal{W}$ correspond to the sensing matrix A_L. When $X_p = \tilde{\mathrm{d}}_p$ and $Y_p = \tilde{\varrho}_p$, $\mathcal{W}$ corresponds to the approximate inverse B_L.

Training Setup

Using machine learning to recover the sensing matrix and its inverse, is a somewhat unusual application. The objective here is to evaluate the potential use of machine learning to solve the direct and inverse scattering problems in SAR. In doing so, we develop a fully data-driven method without explicit knowledge about the imaging system (*i.e.*, bandwidth, platform trajectory and pulse). Thus, it is important that we are able to interpret each and all aspects of the results from the machine learning methods. For this reason, we restrict our attention to a neural network arranged as a single, fully-connected layer.

We use images from the CIFAR-10 dataset [62] as the set of reflectivity data $\{\varrho_p\}$ used for training. The CIFAR-10 dataset consists of 60,000 images which we partitioned into a training set of 50,000 images and a test set of 10,000 images. Each image is 32×32 pixels which is vectorized so that $\varrho_p \in \mathbb{R}^K$, where $K = 1024$. We generate the set of SAR data $\{d_p\}$, with $d_p \in \mathbb{C}^K$, corresponding to these reflectivities using the model matrix A in (6.7). To accommodate for the complex-valued entries of the SAR data in the neural network training regime, we rewrite (6.7) as the equivalent real 2×2 block linear system:

$$\underbrace{\begin{bmatrix} A_{\mathrm{Re}} & -A_{\mathrm{Im}} \\ A_{\mathrm{Im}} & A_{\mathrm{Re}} \end{bmatrix}}_{\tilde{\mathrm{A}}} \underbrace{\begin{bmatrix} \varrho_{\mathrm{Re}} \\ \varrho_{\mathrm{Im}} \end{bmatrix}}_{\tilde{\varrho}} = \underbrace{\begin{bmatrix} d_{\mathrm{Re}} \\ d_{\mathrm{Im}} \end{bmatrix}}_{\tilde{\mathrm{d}}}, \tag{6.10}$$

where $A = A_{\mathrm{Re}} + \mathrm{i}A_{\mathrm{Im}}$, $\varrho = \varrho_{\mathrm{Re}} + \mathrm{i}\varrho_{\mathrm{Im}}$, and $d = d_{\mathrm{Re}} + \mathrm{i}d_{\mathrm{Im}}$ are written in terms of their real and imaginary parts. In (6.10), $\tilde{\mathrm{A}} \in \mathbb{R}^{2K \times 2K}$ and $\tilde{\varrho}, \tilde{\mathrm{d}} \in \mathbb{R}^{2K}$. Note that since the reflectivity data are real, $\varrho_{\mathrm{Im}} = 0$. We will denote the real 2×2 block matrices corresponding to A_L and B_L by $\tilde{\mathrm{A}}_L$ and $\tilde{\mathrm{B}}_L$, respectively.

Each of the SAR data d_p is corrupted with additive Gaussian white noise using MATLAB's `awgn` function to obtain a desired signal-to-noise ratio (SNR). These data were then imported into a Python coding environment where we used PyTorch, a Python deep learning platform built on the torch library [31] and requiring the NVIDIA platform, CUDA for GPU processing [77].

We denote the input data in the training by $X_p \in \mathbb{R}^{2K}$ and the output data by $Y_p \in \mathbb{R}^{2K}$, and the neural network produces a square weight matrix $\mathcal{W} \in \mathbb{R}^{2K\times 2K}$ that maps X_p to Y_p (see Fig. 6.11). During the training process the 50,000 inputs are partitioned into non-overlapping batches $\mathcal{B}$, each containing 32 images. The inputs are passed through the layer with a weight matrix of initial values drawn randomly from a zero-mean Gaussian distribution, resulting in 32 output data. Each output is compared to the target using the loss function defined by the mean squared error (MSE) given by

$$\mathrm{MSE}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|}\sum_{p\in\mathcal{B}}\left\{\tfrac{1}{2K}\|Y_p - \mathcal{W}X_p\|_2^2\right\}.$$

The MSE is then used to adjust the values of the weight matrix using the Adam optimization routine (a form of stochastic gradient descent – see [60] for details) in the torch package. Our results are presented using a learning rate of either 0.001 or 0.0001. Training continues until all 50,000 images are evaluated in batches of 32. Each iteration is one epoch, and the network was trained for 500 epochs.

Learning the SAR Sensing Matrix A_L

To recover A_L, we use the set of reflectivity data $\{\varrho_p\}$ as the input data to the neural network and the corresponding SAR data $\{d_p\}$ as the target data. As stated previously, the algorithm uses a simple neural network consisting of a single layer. The network is created using the fully connected torch `nn.Linear()` network component. Both the input and output dimensions are 2048. The layer bias is turned off and no activation function is applied to the output. Under this configuration, the fully connected layer amounts to a left multiply by a square weight matrix.

Here we do not worry about over-trainning since we know that for this SAR system, the transform is linear and will be reflected in the weight matrix. This differs from traditional machine learning methods in that our goal is the recovery of the transformation matrix and not just the output of the network. Once training is complete we can detach the weight matrix from the network and store it as a Python numpy array. Because the input and target vectors are separated into their real and imaginary components, the recovered matrix has a 2×2 block matrix structure. The upper left corner of the matrix corresponds to $(A_L)_{\mathrm{Re}}$ and the lower left corner corresponds to $(A_L)_{\mathrm{Im}}$. We test the effectiveness of the recovered SAR matrix's ability to recover the reflectivity from the measurements by scaling the learned forward transformation A_L by a factor of $1/K$, taking its Hermitian, and performing left multiplication with the measurements $\{d_p\}$ from the test set. We then take the upper half of the resultant vector corresponding to the real values and reshape the vector in order to compare it to the original image. We denote the reflectivity recovered using this approach by

$$\hat{\varrho}_L = \tfrac{1}{K}A_L^H d, \tag{6.11}$$

where $A_L = (A_L)_{\mathrm{Re}} + i(A_L)_{\mathrm{Im}}$ and the matrices $(A_L)_{\mathrm{Re}}$ and $(A_L)_{\mathrm{Im}}$ are from the learned real 2×2 block matrix $\tilde{\mathrm{A}}_L$.

Learning the Approximate "Inverse" B_L

The method to learn the approximate "inverse" B_L is similar to that used for learning the forward sensing matrix A_L. The network is composed identically to that in the forward direction. The inverse training regime also uses a learning rate of 0.0001 in the optimization routine for 500 epochs with a batch size of 32. Unlike the forward training regime, recovering the inverse

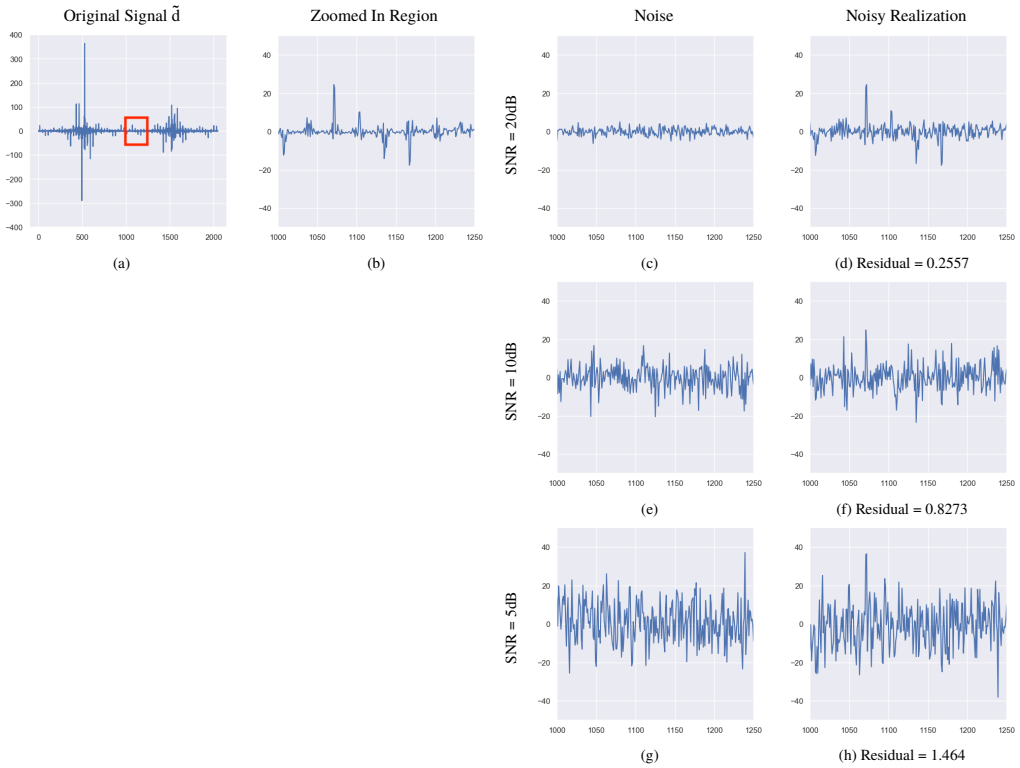


Figure 6.12: An example of the noisy realization used in our numerical experiments. (a) The measurement data $\tilde{\mathrm{d}} \in \mathbb{R}^{2K}$. (b) A particular region of $\tilde{\mathrm{d}}$ magnified. Different noise levels (c)-(e)-(g) are added to $\tilde{\mathrm{d}}$ to achieve the desired SNR in (d)-(f)-(h). Here, residual = $\|\tilde{\mathrm{d}}_{\text{noisy}} - \tilde{\mathrm{d}}\|_1 / \|\tilde{\mathrm{d}}\|_1$.

matrix requires different inputs and targets. In this case the inputs are the measurements $\{d_p\}$ and the targets are the set of reflectivities $\{\varrho_p\}$. In addition to switching the inputs and targets, the inputs are also multiplied by a scaling factor of $1/K$. The scaling term improves the robustness of the optimization routine and results in a more accurate representation of the inverse. Once the inverse weight matrix is recovered the testing process involves multiplying the weight matrix by the scaling factor of $1/K$ performing a left multiply on the measurements. The real values again are extracted from the result and the vector is reshaped and compared to the original image. We denote the reflectivity recovered using this approach by

$$\hat{\varrho}_I = \frac{1}{K} B_L d, \tag{6.12}$$

where $B_L = (B_L)_{\text{Re}} + i(B_L)_{\text{Im}}$ and the matrices $(B_L)_{\text{Re}}$ and $(B_L)_{\text{Im}}$ are from the learned real 2×2 block matrix $\tilde{\mathrm{B}}_L$.

6.3.3 Results

We present three numerical experiments to demonstrate our proposed approach. In our numerical experiments, we consider measurements $\{d_p\}$ with three different noise levels in addition to the noiseless measurements. Fig. 6.12 depicts an example of different noisy realizations of the measurement data $\{\tilde{\mathrm{d}}_p\}$. In Experiment I, we show how well we recover the

Table 6.1: The mean squared error (MSE) of the learned sensing matrix A_L trained on the CIFAR-10 dataset with varying signal-to-noise (SNR) levels. Here, MSE = $\frac{1}{4K^2}\|\tilde{\mathrm{A}} - \tilde{\mathrm{A}}_L\|_F^2$.

SNR	MSE
5dB	5.232e-01
10dB	1.639e-01
20dB	4.333e-02
Noiseless	6.000e-09

sensing matrix from the measurement data. In Experiment II, we reconstruct the reflectivity data from measurement data with noise level at 20dB. In Experiment III, we present results from the reflectivity reconstructions at the three different noise levels.

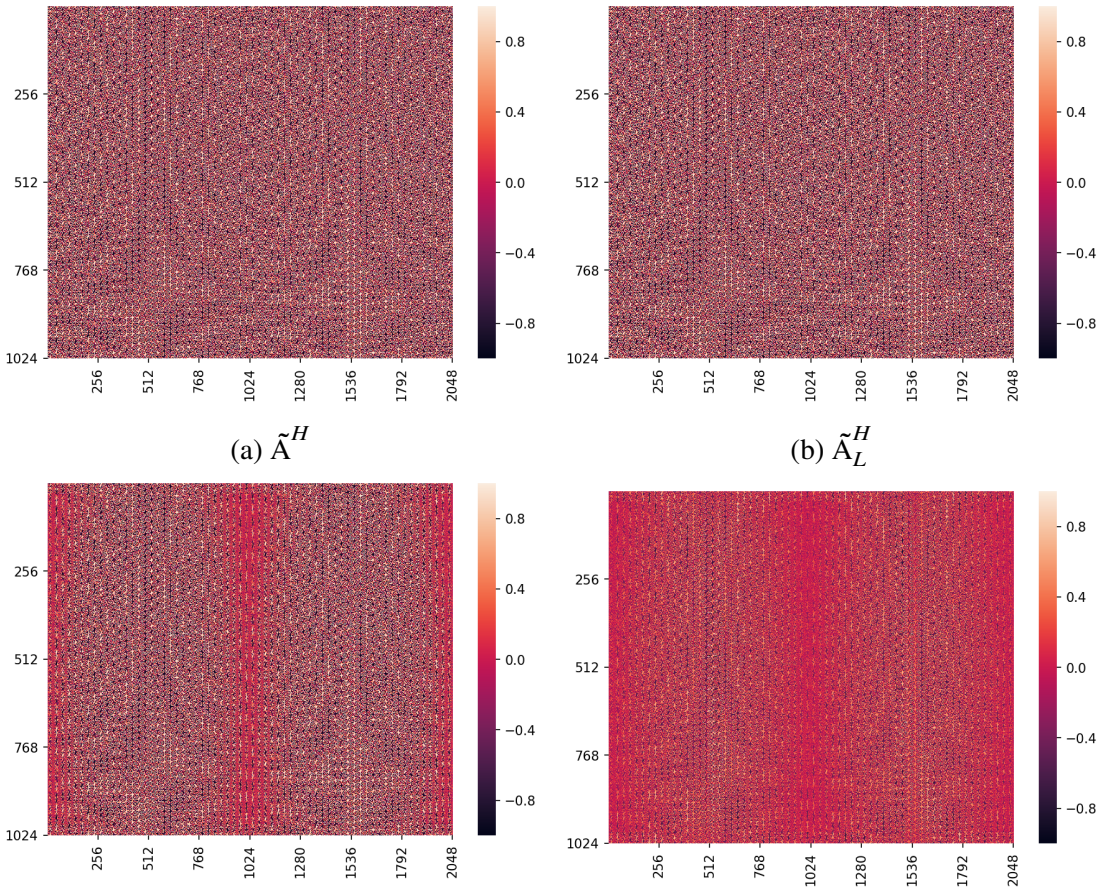
Experiment I

We first show results of the matrices recovered using the machine learning procedure described above. Because of the 2×2 block structure of the matrix in (6.10), we only present the upper half heat map representations of the recovered matrices. We show the upper half heat map representations of $\tilde{\mathrm{A}}^H$ in Fig. 6.13(a), of $\tilde{\mathrm{A}}_L^H$ recovered from noiseless data in Fig. 6.13(b), of $\tilde{\mathrm{A}}_L^H$ recovered from noisy data with SNR $=$ 20dB in Fig. 6.13(c), and of $\tilde{\mathrm{B}}_L$ also recovered from noisy data with SNR $=$ 20dB in Fig. 6.13(d). The results shown in Fig. 6.13 show that the learned matrices possess a qualitatively similar structure to $\tilde{\mathrm{A}}^H$. In Table 6.1, we present the accuracy of the learned matrices quantitatively using the mean squared error (MSE = $\frac{1}{K^2}(\|A_{\mathrm{Re}} - (A_L)_{\mathrm{Re}}\|_F^2 + \|A_{\mathrm{Im}} - (A_L)_{\mathrm{Im}}\|_F^2)$. We did not quantitatively compare B_L to A^H since that comparison is not necessarily interpretable.

Experiment II

Next, we use the learned matrices to reconstruct images of reflectivities for data in the testing set (*i.e.* those not used for training). Fig. 6.14 shows three different example reconstructions for SNR $=$ 20dB. The left column of Fig. 6.14 shows the target image representing "ground truth." The second column of images in Fig. 6.14 are the reconstructions $\hat{\varrho}_{SAR}$ from using the Hermitian A^H of the exact known SAR matrix A given in (6.7). The third column of images in Fig. 6.14 are the reconstructions $\hat{\varrho}_L$ from using A_L^H, with A_L denoting the learned sensing matrix. The fourth column of images in Fig. 6.14 are reconstructions using B_L, the learned approximate "inverse" of the sensing matrix. The corresponding MSE for each of these reconstructions appears below the image.

The reconstructions appearing in Fig. 6.14 are reasonable estimates of the target image. The results from using A^H and A_L^H are comparable, but qualitatively, the reconstructions from using A^H appear to have less imaging artifacts. The learned inverse B_L gives the best qualitative and quantitative reconstructions. For the three examples shown here, the learned inverse yields MSE's that are two orders of magnitude smaller. Moreover, those reconstructions appear to be less adversely affected by noise corruption in the measurement data.



(a) $\tilde{\mathrm{A}}^H$ (b) $\tilde{\mathrm{A}}_L^H$

(c) $\tilde{\mathrm{A}}_L^H$ (SNR = 20dB) (d) $\tilde{\mathrm{B}}_L$ (SNR = 20dB)

Figure 6.13: Comparison of the mappings from the measurements d to the reflectivity estimates $\tilde{\varrho}$. (a) The upper half of the Hermitian $\tilde{\mathrm{A}}^H$ of the exact sensing matrix. (b) The upper half of the Hermitian of the learned sensing matrix from noiseless measurements. (c) The upper half of the Hermitian of the learned sensing matrix from noisy measurements with SNR = 20dB. (d) The upper half of the learned approximate "inverse" of A from noisy measurements with SNR = 20dB. Note that we present only the upper half of each mapping because that is only what is needed in (6.10) to estimate $\tilde{\varrho} = \tilde{\varrho}_{\mathrm{Re}}$.

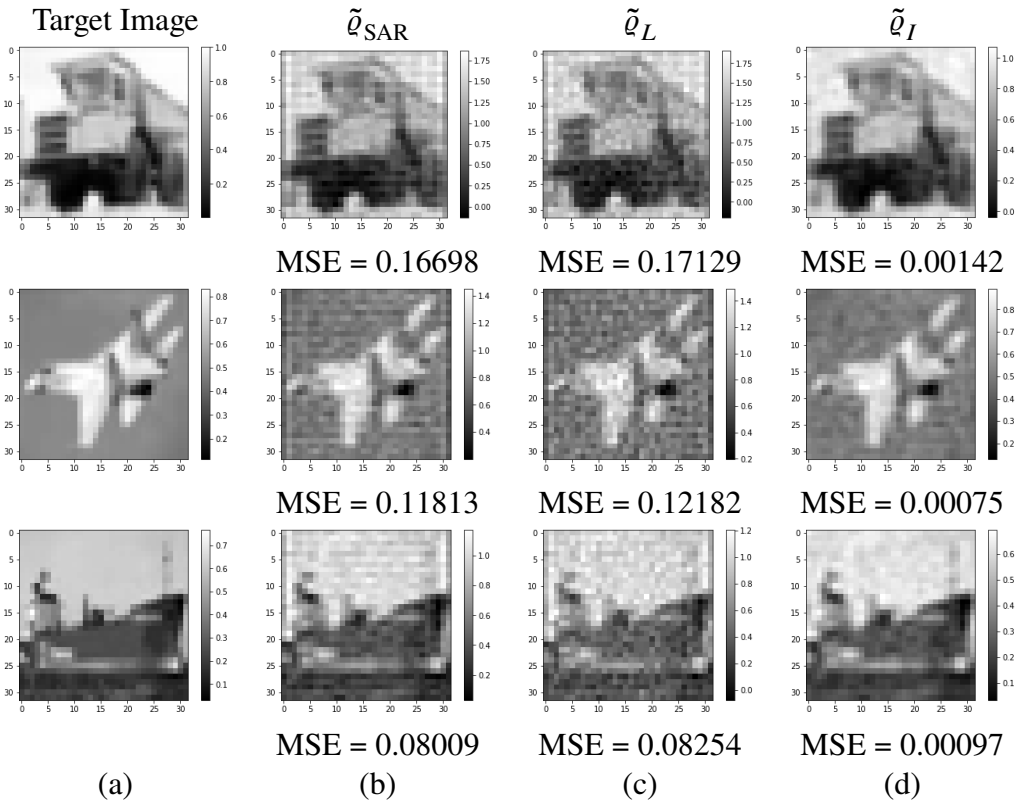


Figure 6.14: Numerical experiments using images from the CIFAR-10 dataset. Gaussian noise was added to the measurements so that $\{d_p\}$ have SNR = 20dB. (a) Ground truth. (b) Reconstruction $\tilde{\varrho}_{\text{SAR}}$ from (6.9) using the SAR inversion formula. (c) Reconstruction $\tilde{\varrho}_L$ from (6.11) using the learned SAR sensing matrix A_L. (d) Reconstruction $\tilde{\varrho}_I$ from (6.12) using the learned approximate "inverse" B_L. The mean squared error (MSE) is reported for each reconstruction. Note the significant improvement in MSE from using B_L to recover the target images over the SAR inversion formula. The slight decrease in the performance of using A_L compared to the SAR inversion formula may be attributed to the training on noisy data.

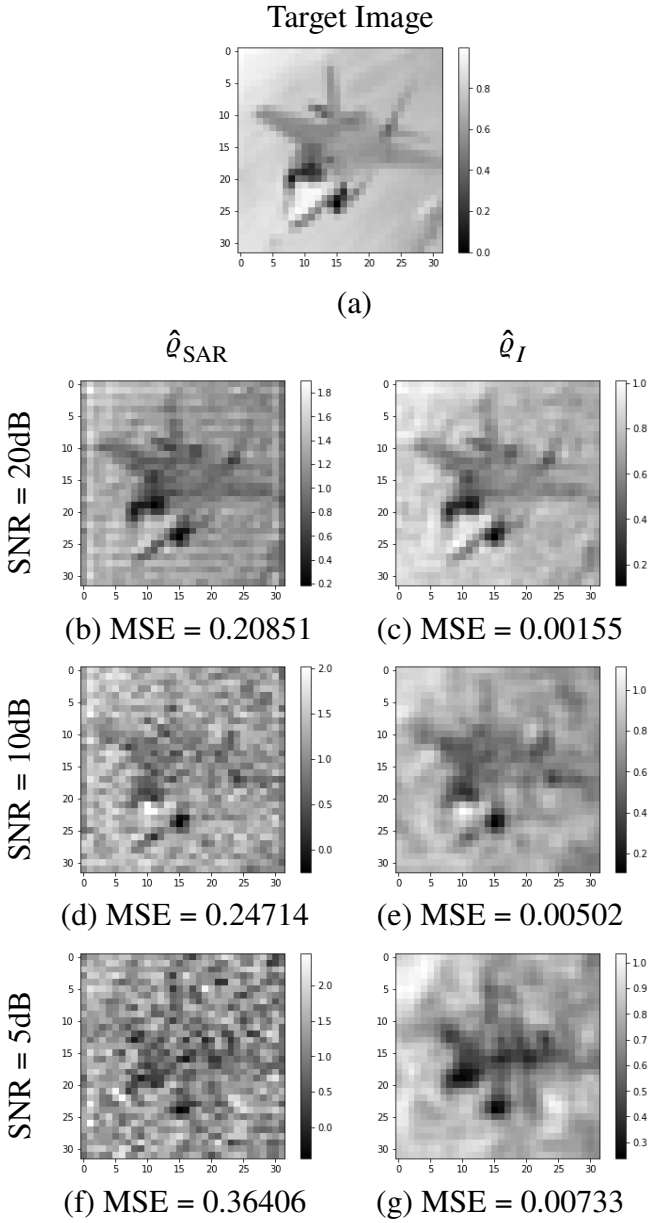


Figure 6.15: Image reconstructions for various SNR levels. (a) Ground truth target image. Reconstructions $\hat{\varrho}_{\text{SAR}}$ using the Hessian of the exact SAR sensing matrix A^H from noisy measurements with SNR = 20db, 10db, and 5db are presented in (b), (d) and (f), respectively with corresponding MSEs. Reconstructions $\hat{\varrho}_I$ using the learned approximate "inverse" sensing matrix B^L from noisy measurements with SNR = 20db, 10db, and 5db are presented in (c), (e) and (g), respectively with corresponding MSEs.

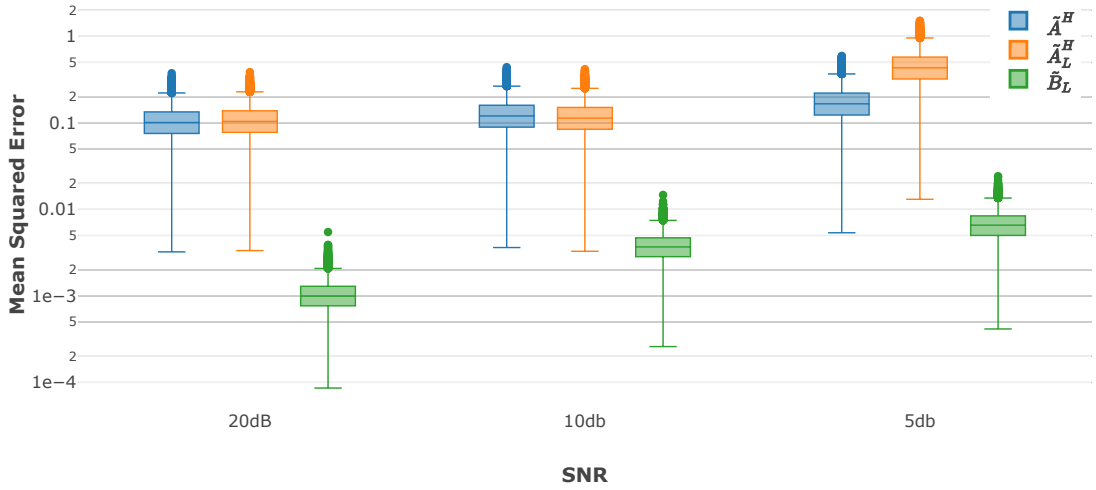


Figure 6.16: Performance of the reconstruction methods on the 10,000 CIFAR-10 images making up test set. These box and whisker plots show the MSE for each of the reconstruction methods as well as each of the SNR values.

Experiment III

To study how our image reconstruction approach is affected by noise, we show in Fig. 6.15 reconstructions of a target image (Fig. 6.15(a)) with different SNR levels. The left column of reconstructions $\hat{\varrho}_{SAR}$ use the Hermitian A^H of the exact SAR matrix and the right column of reconstructions $\hat{\varrho}_I$ use the learned approximate "inverse" B_L. Figs. 6.15(b) and 6.15(c) are for SNR = 20dB, Figs. 6.15(d) and 6.15(e) are for SNR = 10dB, and Figs. 6.15(f) and 6.15(g) are for SNR = 5dB.

Overall, we find the reconstructions that used B_L to be better than those that used A^H. The reconstructions using B_L appear to have an inherent low-pass filtering operation that smooths the images and reduces imaging artifacts. In contrast, the images reconstructed using A^H suffer from significant reconstructions artifacts, especially as the SNR decreases.

To summarize the quantitative errors we have found in using A^H, A_L^H and B_L for image reconstructions, we show in Fig. 6.16 the MSE statistics computed for all image reconstructions for the 10,000 images making up the test set. These box and whisker plots show the sample distribution of MSE values computed for the different reconstruction methods at different SNRs. These statistics clearly show that reconstructions using B_L yield substantially lower MSEs in comparison to reconstructions using A^H and A_L^H. The reconstructions using A^H and A_L^H are comparable except for low SNR in which the reconstructions using A^H yield slightly lower MSEs.

6.3.4 Conclusion

We have used a simple neural network consisting of a single, fully-connected layer to learn the sensing matrix and its inverse from a training set of data comprised of reflectivities and their corresponding measurements computed using a standard model in SAR. This simple machine learning setup has allowed us to evaluate the potential of a fully data-driven method for solving

the direct and inverse scattering problems in SAR. By computing image reconstructions for an out-of-sample set of test data using these learned matrices, we evaluate the effectiveness of this approach.

Our results show that the learned inverse of the sensing matrix, denoted here by B_L outperforms the standard SAR inversion formula (6.9). These results are rather remarkable given the fact that the training and test data sets are generated using the sensing matrix, A. Overall, we have found that images reconstructed using this learned inverse are qualitatively better and yield a substantially smaller MSE than those computed using the standard SAR inverse formula (6.9). These results are consistent over the large range of SNR we have considered here.

These results strongly indicate the potential use of machine learning methods to solve the direct and inverse scattering problems in SAR. The key is to provide this machine learning algorithm a meaningful set of training data. Provided such data are available, this method opens the possibility of developing fully data-driven methods that do not require explicit consideration or knowledge of the physical setting or details of the imaging system. Here ground truth is assumed known for a quite large set of images. To be able to develop machine learning methods for practical SAR imaging systems, we will need to consider that ground truth is known for a smaller set of data, assuming eventually some knowledge about the imaging system (pulse form and/or flight path). Evaluating the performance of the algorithm and modifying it accordingly to account for this interplay between available ground truth data and knowledge about the sensing system is a challenging problem and will be considered in a forthcoming work.

Even though this work is an initial study of the use of machine learning for solving direct and inverse scattering for SAR, we have found the proposed approach to be remarkably effective. Because this method presents an initial indication for a fully data-driven method, it should be very useful for SAR imaging. Moreover, the proposed methodology has the capacity to be extended to a broad variety of other inverse scattering problems.

6.4 Image Classification in Synthetic Aperture Radar Using Reconstruction from Learned Inverse Scattering

In this section we extend our work in the domain of synthetic aperture radar and machine learning. As opposed to the application in section 6.3, we are interested in the capabilities of deep learning algorithms in the field of computer vision. Specifically, we are interested in classifying targets within the modality of synthetic aperture radar (SAR). Typically, classification of SAR images requires that they are reconstructed from noisy measurements. Thus, the quality of reconstruction will affect the classification of each image. We investigate the effect of the quality of image reconstruction in relation to the accuracy of classification of SAR images. We do this using different data acquisitions as well as different deep learning architectures. This work is based on the paper by Alvarez, DeGuchy, and Marcia [3].

Related Work

Automatic target recognition applications have traditionally relied on SAR data to make informed decisions. With the emmergence of deep learning algorithms as the state of the art in

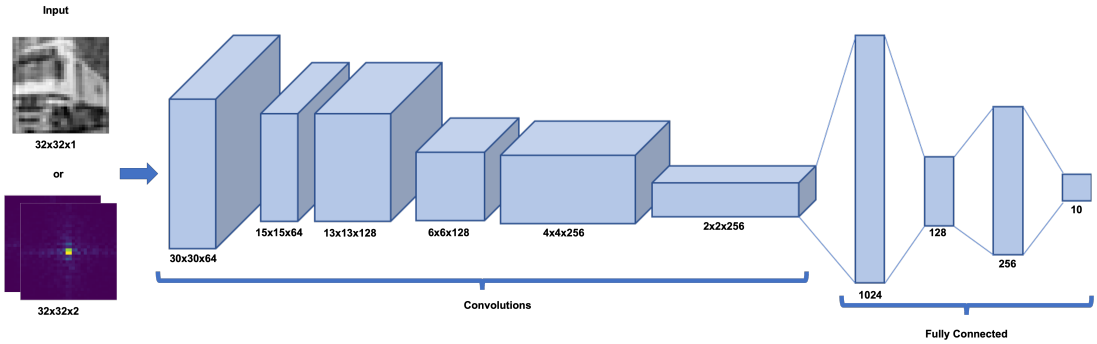


Figure 6.17: Convolutional Neural Network (CNN) for SAR image classification. The input to the the network is either a single channel grayscale reconstructed image or SAR data represented as a two channel tensor where each channel contains the real and imaginary components of the complex SAR data d.

computer vision, there is a natural extension of neural networks in the field of automatic target recognition. SAR image classification requires that the reflectivity from SAR data be recovered for use as inputs in a deep learning regime [24, 73, 23].

6.4.1 Problem Formulation

In this section, we review the SAR remote sensing modality explained in Section 6.3. SAR uses a single transmitter/receiver in concert with a moving platform to create a synthetic aperture. The process involves the emission of a broadband pulse and the collection of the corresponding echoes in the form of SAR data d at various points along the path of the platform. (For an overview, see [27, 28, 72]). The SAR data is dependent on two time scales: (1) the slow time that parameterize the location of the platform, and (2) the fast times which measures the round-trip travel time between the platform and the reflectivity window on the ground. The goal is to recover the unknown reflectivity of interest ϱ in the imaging window.

The SAR process can be modeled using a scalar wave equation, which in turn is discretized, reducing the SAR imaging problem to the following system of equations:

$$A\varrho = d, \tag{6.13}$$

where $A \in \mathbb{C}^{N\times K}$ is the sensing matrix which maps the reflectivity vector $\varrho \in \mathbb{C}^K$ to the SAR measurement $d \in \mathbb{C}^N$ (see [34] for further details). Recent improvements on traditional reflectivity recovery in [34] provide an opportunity to increase the quality of the data used in training neural networks for classification. We posit that the increase in image quality will translate to an increase in classification accuracy. Here we present two possible reconstruction methods.

Kirchoff Migration

In general, the SAR system is overdetermined with $N \gg K$ such that N is the number of points on the moving platform and K is the number of grid points in the imaging window. A least-squares solution may be obtained as $\hat{\varrho}_{\ell_2} = Bd$ with B as the pseuso-inverse of A where $B = (A^HA)^\dagger A^H$ such that A is full column rank and A^H denotes the complex conjugate-transpose of A. A common

choice in discretizing the imaging window leads to the matrix $A^H A$ being close to diagonal [20], which justifies the SAR inversion formula, $\hat{\varrho}_{\text{SAR}} = A^H d$ (see [19] for details).

Learned Inverse

Reconstruction of the reflectivity can also be achieved using a machine learning framework. The network configuration used to recover the mapping between the SAR measurements d and the true reflectivities ϱ is a single, fully-connected layer without a bias or an activation function [34]. The inputs to the network are the SAR measurements and the output are approximations of the reflectivities. As the network is trained, the matrix of weights associated with the fully-connected layer converge to an approximate inverse of the SAR linear system. Our reconstruction is then governed by $\hat{\varrho}_I = B_L d$, where B_L is the final weight matrix from the previously described network.

6.4.2 Proposed Approach

Current methods for classifiying SAR images require that they be reconstructed from the measurements before classification takes place. In this section, we introduce three methods for classifying SAR images. In addtition, we also describe the two different architectures used in the next section.

Proposed Methods

The three proposed methods begin with SAR measurements. However Methods II and III reconstruct the images from the measurements.

Method I:

We consider classifying using the raw SAR measurements as input data. These measurements are composed of complex values, i.e., real and imaginary parts. We take the real and imaginary components and consider them as separate channels of an image.

Method II:

We reconstruct the images from the SAR measurements using Kirchoff migration which is descibed in Section 6.4.1. In practice, the sensing matrix is unknown. For the purpose of this project we have access to the true sensing matrix, thus we use the exact Hermitian for reconstruction.

Method III:

We reconstruct the images using the learned approximate inverse described in Sec. 6.4.1. We could also reconstruct the images using the Hermitian of a learned sensing matrix, however the learned approximate inverse achieves better quality reconstructions [34].

Architectures

As mentioned in the previous section, we have multiple data inputs. The two reconstructions are images, thus they are easily used in a deep neural network. However, the SAR data is composed of complex values which are not easily used in deep neural networks. We must consider the real and imaginary parts separately. Thus, we consider two approaches to structure the data, each using a different classification architecture, the first being a multi-layer perceptron and second a convolutional neural network based on [90].

A. Multi-Layer Perceptron.

For the multi-layer perceptron (MLP) architecture, the input is a vector. Thus, for each data input mentioned above we restructure the data as vectors. However, the main concern is how to handle the complex values of the SAR data. In this case, we stack the real and imaginary values in a vector with real above imaginary. This results with a vector that is doubled in dimension compared to the reconstructions. The neural network is composed of three fully-connected layers, each using dropout.

B. Convolutional Neural Network.

For the convolutional neural network (CNN), the inputs are three-dimensional arrays, typically images, where the last dimension refers to the number of channels an image has. For the two reconstruction types we only have one channel. However in the case of the SAR data, we have two channels such that the first channel contains the real elements of the SAR measurements and the second channel contains the imaginary elements. The neural network, depicted in Fig. 6.21, is composed of three convolutional layers, each followed by a maxpool layer, then three fully-connected layers at the end.

6.4.3 Numerical Experiments

In this section, we present the dataset and the experimental setup.

Dataset

We use the CIFAR-10 dataset as the reflectivity data. The CIFAR-10 dataset is comprised of 60,000 images, which are partitioned into 50,000 images for training and 10,000 for testing. The dataset includes 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image is 32×32 pixels and is grayscale, resulting in one channel. The SAR measurement data is then generated by applying the sensing model used in [34]. Gaussian noise was added to the SAR measurements, d, to have signal-to-noise ratios (SNR) of 20db, 10db and 5db.

Architecture Parameters

The MLP and CNN were both trained using the three methods from Sec. 6.4.2 applied to the CIFAR-10 dataset with varying SNR levels and noiseless data. In both architectures, each output

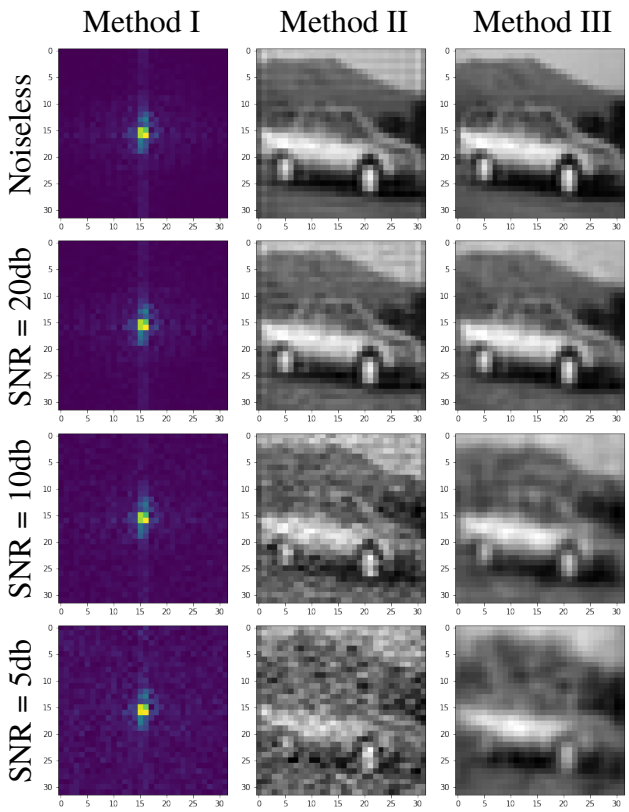
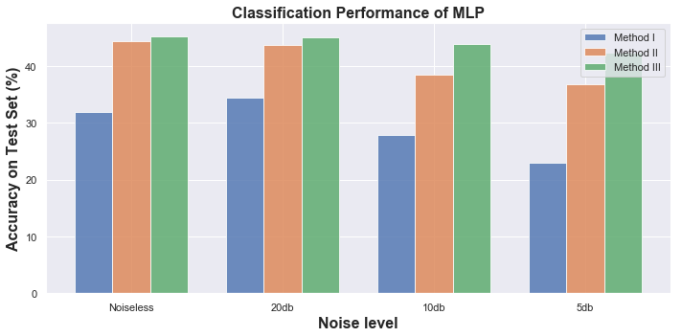


Figure 6.18: Method I uses raw SAR data, which consist of complex frequenices. Pictured above is the magnitude of the SAR data for an image of an automobile. Method II uses reconstructions of the SAR data using Kirchoff migration. Method III also uses reconstructions of the SAR data, but using the learned approximate inverse.

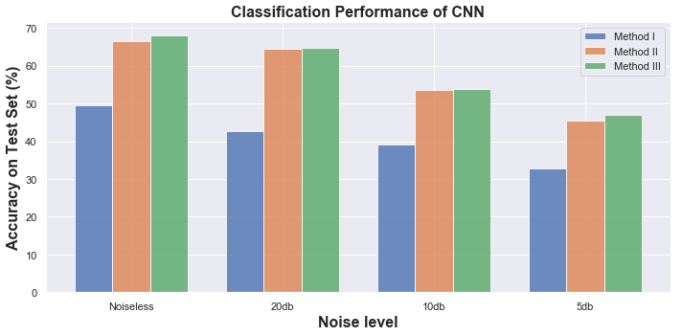
label is measured against the target label using a loss function described by cross entropy (see [50] for details).

For the MLP, the 50,000 inputs are partitioned into batches each containing 32 images. We utilize the ADAM optimizer, which is a form of stochastic gradient descent, with a learning rate of 0.0001 to update the weights at each iteration. One iteration includes the evaluation of each batch. This network was trained for 300 epochs. Additionally, a validation set was used during training to track progress. The validation set contains 5,000 of the images from the original test set. As a result, the test set used was composed of the remaining 5,000 images.

For the CNN, the 50,000 inputs were partitioned into batches containing 4 images. However, stochastic gradient descent was used with a learning rate of 0.001 and momentum set to 0.9. Here, we trained the network for 20 epochs. As a baseline, this network was trained on the true reflectivities using color and grayscale images. When the network was trained on the original 3-channel images the accuracy rate was 75% [90]. Due to a loss of information in the number of channels, training on greyscale images lowers our baseline to an accuracy of 67.5%.



(a) Accuracy using a multi-layer perceptron



(b) Accuracy using a convolutional neural network

Figure 6.19: Classification performance comparison between the proposed methods (Methods I, II, and III) using two different architectures ((a) MLPs and (b) CNNs) on data with four different noise levels.

6.4.4 Results

We present results from our numerical experiments to compare the classification accuracy between Methods I, II, and III using the two different architectures (MLP and CNN). All cases consider data with varying noise levels: noiseless, 20db, 10db, and 5db.

The results using an MLP for classifying SAR images show that the learned approximate inverse provides a higher classification accuracy than that of the Krichhoff migration or raw SAR data. This holds true even as we increase the noise on the images, which is depicted in Fig. 6.19(a). In the case of using a CNN, the results are similar and show that the learned approximate inverse provides the highest classification accuracy. In the case of noiseless data, the results are comparable to using the original images which had an accuracy of 67.50%. These results are shown in Fig. 6.19(b). We note that using a CNN architecture achieves a much higher classification than using an MLP.

6.4.5 Conclusion

In this work, we compared three methods for classifying SAR images using two different machine learning architectures. The first method uses raw SAR measurement data, which consist of complex frequencies. The other two methods are reconstructions of the SAR data using Kirchoff migration and a more recent machine learning based reconstruction. Futhermore, we

considered two deep learning architectures, namely a multi-layer perceptron and convolutional neural network. The results show that using the learned approximate inverse (Method III) on both network architectures classifies the images with the highest accuracy. This holds true when using data of varying noise levels. We note that Method II produces classification results better than those from using raw SAR data (Method I), it cannot be directly applicable when the sensing matrix A is not available. In contrast, Method III provides the best classification results without explicitly knowing the SAR sensing matrix. These results support that using the learned approximate inverse aids in achieving higher classification accuracy of SAR images. Furthermore, combined with a more sophisticated machine learning architecture (CNN), the proposed methods can improve classification accuracy over standard MLP architectures.

6.5 Signal Variation Detection in Genome Data

The genome, the complete DNA sequence, of an organism is a long sequence of nucleotides represented by the letters {A,C,G,T}. For mammals, the length of the genome is approximately 3 billion letters whereas for the single celled yeast (*S. cerevisiae*), the length is around 12 million letters. Most individuals within the same species have highly-similar genomes, and differences between the genome of two individuals in the same species are characterized by their lengths. Single-nucleotide variants (SNVs) correspond to a single letter difference. In addition, there are also short regions where multiple letters are inserted or deleted termed In/Dels ($\leq$ 50 letters). Structural variants (SVs) are genomic regions ($>$ 50bp) that vary between members of the same species. SVs may be insertions, deletions, inversions or more general and complex exchanges of DNA segments between regions of the genome [51, 95].

While DNA sequencing costs continue to decline, it is still cost prohibitive to determine the complete DNA sequence for humans. However, because we have a high-quality reference genome for humans and a variety of other species, genomic variants can detected by comparing samples of DNA sequence to the reference. It is far easier to assess the presence of single-letter differences (SNVs) than SVs. Indeed, such technology is readily available to the general public from companies like Ancestry [6] or 23andMe [1]. The dominant method of SV detection is to take samples (fragments) of DNA from an unknown genome and compare them to a high-quality reference. The resulting configuration of mapped fragments is analyzed, and structural differences between the unknown and reference genome should conform to arrangements of mappings that are discordant (with respect to order, length, orientation, etc) or regions of the reference with higher or lower than expected numbers of fragments [51]. For example, a fragment that has portions matching two distant regions of the reference genome, a split-alignment, indicates a potential SV. The problem of SV detection is complicated by errors in the sequencing and mapping process which can create observations that look like true SVs.

One approach to improve SV detection is to simply take more samples from the test genome to separate the true from false predictions, but this approach will result in an increase in cost. An alternative approach is to make better predictors which explicitly incorporate known biases and multiple signals from related individuals [80, 86]. We follow in this spirit; but rather than predefining the signals or features of interest, we use a deep-learning approach by building a feed-forward neural network. Machine-learning approaches are becoming more common in genomics and have been previously used for variant detection [69, 76, 7, 4]. However, our work is distinguished from these methods by the fact that we consider the ability of simultaneously

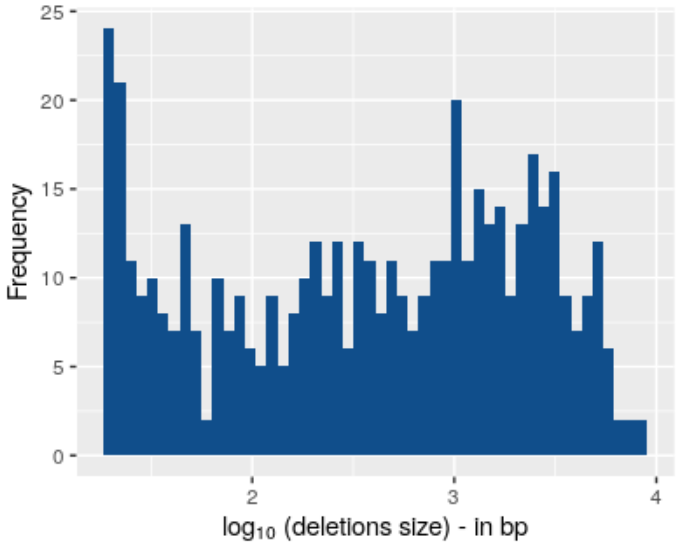


Figure 6.20: Size distribution of 500 simulated deletions, which is based on deletion sizes from the Database of Genomic Variants. We note that the majority of deletions will be less than 1000bp in length.

considering sequencing data from a parent and a child. Because the *de novo* formation rate of SVs is low, all but a very small minority of variants present in a child's genome will have been inherited from the parent. We have previously used parent-child trios to improve SV detection but not in a deep-learning framework (see e.g., [88]).

In this work, we propose two different approaches for reducing the number of false positive SV predictions from a popular SV tool delly [80]. For simplicity, we focus on deletions (see Fig. 6.20 for the estimated size distribution of these variants in humans). We compare the performance of each method on both simulated and real parent-child sequencing data. Our results on both simulated and real data demonstrate that deep-learning and gradient boosting are powerful tools for SV detection but that including related individuals in the data set greatly boost the ability to recover true SVs. This work is based on the paper by Banuelos, DeGuchy, Sindi, and Marcia [12].

6.5.1 Methods

In this section we describe the two machine learning approaches implemented for SV detection. The first method uses a feed forward neural network of fully connected layers. While well established as the state of the art in the world of computer vision, neural networks are emerging as powerful tools for processing tabular data [10, 17, 57]. Unlike images, time series, or text datasets, tabular data consists of a columnar format where each column contains variable information for a given number of data points. The second method, XGBoost, has already been established as a workhorse for classification applications in the tabular data domain. XGBoost is an ensemble method that uses decision-trees in concert with gradient descent in order to improve performance. In the following section we describe the parameters as well as the preprocessing to the data for each method.

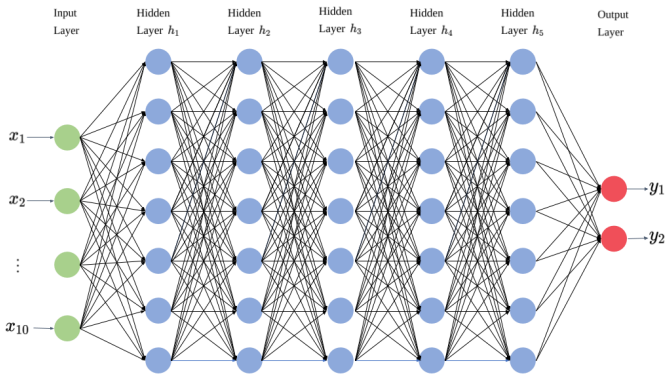


Figure 6.21: Neural network architecture used for identifying structural variation in genomic data. The inputs $x_1 \dots x_{10}$ represent the features corresponding to the delly deletion calls. Each entry of the output $y \in \mathbb{R}^2$ is the probability corresponding to the absence or presence of a signal variation.

Neural Network

We adopt the same framework for a feedforward network established in Chapter 4. Since our network consists of fully connected layers, the process can be described by the equation

$$h_i = g(W_i^T h_{i-1} + b_i),$$

where $h_{i-1} \in \mathbb{R}^m$ is the output of the previous layer and $h_i \in \mathbb{R}^n$ is the output of the current. Each data pair can be described as (X_j, y_j) for $j = 1 \dots q$, where q is the number of training points. The input $X_j \in \mathbb{R}^{10}$ consists of the features provided by delly and the target $y_j \in \mathbb{R}$ is the true binary label indicating the absence (0) or presence (1) of a structural variant. The architecture consists of one input layer $h_0 = g(W_0^T X_j + b_o)$, five hidden layers $h_1 \dots h_5$ and an output layer $o = s(W_o^T h_4 + b_o)$. All hidden layers contain 120 neurons and use the ReLU activation function with the exception of the output layer which consists of two neurons and uses a log softmax activation function [44]. Before each layer we apply batch normalization to improve the performance and stability of our network [55].

Given the two possible classes (the presence or absence of an SV), the output of the neural network is a distribution of probabilities $\hat{p_k}$ with $k = 1 \dots K$ with K equal to the number classes (in this case $K = 2$). We seek to minimize the Cross Entropy cost function

$$\mathcal{L}(W) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_{(i)k} \log(\hat{p}_{(i)k}), \tag{6.14}$$

where $y_{(i)k}$ is the true binary label of the k^{th} class and m is the number of training samples for a given batch [45]. We use the Adam optimization algorithm to minimize (6.14) with a batch size of 16 over 100 epochs. Before training and testing, the data is normalized so that all columns of the features have zero mean and unit variance.

6.5.2 XGBoost

Extreme gradient boosting or XGBoost is an optimized implementation of the Gradient Boosting Trees Algorithm. It has been shown to be highly effective in classification problems

Simulated Data Experiment	
Parameter	**Value**
Estimators	800
Min. Child Weight	0.5
Max. Depth	7
Gamma	1
Subsample Ratio	1.0
Column Subsampling	1.0
Platinum Genomes Data Experiment	
Parameter	**Value**
Estimators	400
Min. Child Weight	20
Max. Depth	3
Gamma	0.5
Subsample Ratio	0.8
Column Subsampling	0.6

Table 6.2: Parameters used for training XGBoost Algorithm.

across a variety of disciplines [26, 9, 40]. XGBoost is an ensemble machine learning algorithm built using decision trees. Given our data set X_j, y_j, XGBoost creates an ensemble of K additive Classification and Regression Trees (CART) which we express as $T_1(X_i, y_i) \dots T_K(X_i, y_i)$ in order to predict the class label y_i. The prediction scores for each CART are summed up as a final score expressed by the equation

$$\hat{y_i} = \sum_{k=1}^{K} f_k(X_i), f_k \in F \tag{6.15}$$

where f_k is a tree structure and F is the space of all trees [25, 9]. Given the final score, we seek to minimize the cost function

$$J(\Theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k}^{K} \Omega(f_k) \tag{6.16}$$

where the first term l in (6.16) is a data fidelity term between the prediction $\hat{y_i}$ and the target y_i. The regularization term Ω penalizes the model complexity to avoid overfitting. We refer the reader to [25] for the optimization routine for (6.16) which cannot be optimized using traditional optimization methods in Euclidean space.

Hyperparameter tuning is as essential to XGBoost as it is to the neural networks. We use grid search methods in order to perform parameter sweeps using k fold cross validation in order to find the optimal model parameters. The optimal hyperparameters for both models are shown in Table 6.2. In either case the learning rate was set to 0.02 and a binary logistic objective function was used as a cost function. All parameters not listed in Table 6.2 were set to the default values. Both datasets were preprocessed using a min-max normalization.

6.5.3 Numerical Experiment

Simulated Data

Using the first twelve chromosomes of the hg19 build of the human reference genome, we introduced 500 deletions using RSVSim [33, 54, 15]. We simulated corresponding reads with read lengths $L = 75$bp and $L = 150$bp using `dwgsim` and aligned reads with `speedseq` [52, 29]. We follow a similar approach to simulate 2 different individuals, one offspring derived from the

Features	Description
Chr	Chromosome
Start	Predicted start position of deletion
End	Predicted end position of deletion
FILTER	PE/SR support < 3 or mapping quality < 20
IMPRECISE	SR support > 0
PE	Number of paired-end reads supporting deletion
MAPQ	Median mapping quality of paired-ends
CIPOS	Paired-end confidence interval around Start
CIEND	Paired-end confidence interval around End
SR	Number of split-reads supporting deletion

Table 6.3: Features corresponding to the delly deletion calls and a description of each feature.

Table 6.4: Performance metrics for both experiments.

Simulated Data				
Method	Precision	Recall	F1	AUC
Neural Network	0.66	0.67	0.65	0.75
XGBoost	0.76	0.77	0.75	0.85
Platinum Genomes Data				
Method	Precision	Recall	F1	AUC
Neural Network	0.59	0.50	0.48	0.78
XGBoost	0.65	0.53	0.53	0.853

Table 6.5: Performance metrics for both experiments.

mutated parent and one unrelated individual. Since the rate of *de novo* variations is less than one per generation, the offspring only had 3 novel deletions not present in the parent [61, 56].

To obtain candidate genomic variant locations, we call deletions with delly [80]. We incorporated variant call format (vcf) files into our Python workflow using `cyvcf2` to extract a total of 10 features corresponding to the delly deletion calls [78]. We summarize these in Table 6.3. Since we know the truth signal for each individual, we apply our proposed methods to reduce the number of false positives predicted by delly. For both methods, we use the mutated parent as the training data, and the offspring as the testing data.

Platinum Genomes Data

Following a similar framework as the simulated data, we apply our proposed methods to delly calls for individual NA12878 (child) and NA12891 (mother) from the CEU population [39]. Both individuals are from a 17-member pedigree, where the true genomic variants have been experimentally validated. We filter out deletions smaller than 50bp from the truth set. From the catalogued true variations, we create the truth signal $\vec{y}$ corresponding to the delly predictions. In this case, we use NA12891 as the training data and NA12878 as testing set.

6.5.4 Results

The proposed methods were able to significantly reduce the number of false positive classifications identified by the delly SV caller. In Table 6.5 we report a variety of metrics to evaluate performance of the methods on both datasets. It is clear from both Table 6.5 and the AUC

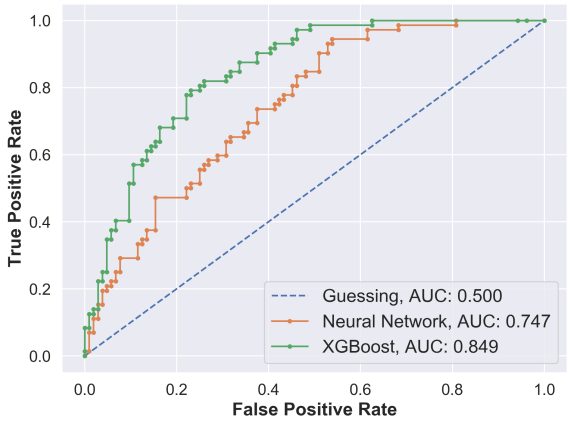


Figure 6.22: Receiver operating characteristic curve (ROC) for the simulated data offspring signal. We also report the area under the curve (AUC) for each method.

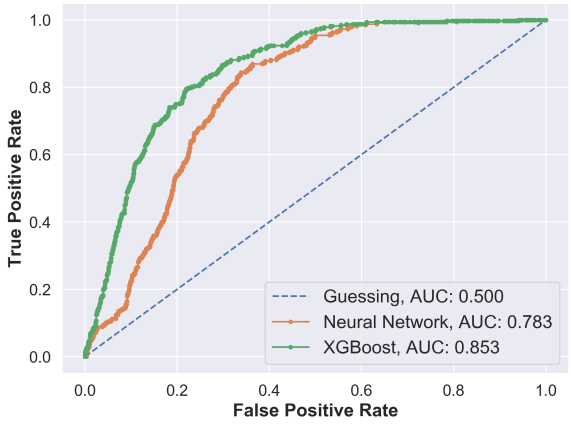


Figure 6.23: Receiver operating characteristic curve (ROC) for the *Platinum Genomes* NA12878 offspring signal. We also report the area under the curve (AUC) for each method.

curves in Figures 6.22 and 6.23 that XGBoost is clearly outperforming the Neural Network. Even though the ensemble method improves on the scores of the Deep Learning method, the authors feel that the results are promising and warrant further exploration. We are also encouraged by that fact that the AUC for both the simulated dataset and the real dataset behaved similarly under both methods.

For the simulated data, using an unrelated individual for the training data yields less predictive power for the offspring (results not shown). Although the simulated data reflects biologically-informed deletion sizes, the training and testing set resulted in balanced number of observations for each class. In contrast, for the *Platinum Genomes* data, we find less than one in five delly predictions to be true deletions. This imbalance may account for less improvement in precision and recall than in the simulated data tests. Including more related individuals across multiple generations may also improve the reduction of false positives in SV callers.

6.5.5 Conclusion

We present a supervised learning framework that incorporates relatedness information to reduce the number of false positives in SV-callers, like delly. Although we present our results in the context of deletions, our framework can be adapted for predicting other classes of structural variants. In the context of applying such methods, we also find that population-level supervised learning techniques may be more appropriate in refining variant predictions than an approach that does not consider differences in ancestry.

6.6 Summary of Contributions

In this chapter we presented deep learning as a method for a variety of applications in signal recovery. In Section 6.1 we implemented deep neural networks to address the photon-limited imaging problem originally presented in Chapter 3. We compared fully connected layers to convolutional layers and found that the performance of fully connected layers was robust to various levels of noise and compression. In Section 6.2 we proposed two architectures to perform image disambiguation. This signal retrieval problem, often referred to as blind source separation, is particularly ill-posed. Our network was able to disambiguate two super imposed images and in some cases viewed weaker images as noise, while extracting the other image. We then switched modalities to an application in remote sensing. We presented a neural network architecture which allowed us to capture the physics of the SAR sensing system. The same architecture can be used to recover an inverse, thus allowing us to solve the linear system associated with recovering the target of interest. We then extended this technique in Section 6.4 to the problem of target recognition. We showed that the quality of the reconstruction is of great importance when it comes to object detection. Furthermore, we were able to display that our method improves on a regime in which nothing is known about the imaging system. Finally in Section 6.5, we applied deep learning to the world of genomics. For this application we leveraged the output of current techniques in order to reduce the number of false signals when detecting structural variants in the human genome. While neural networks performed less than favorable when compared to an ensemble method, it shows promise when applied to the problem while exhibiting consistency from synthetic data to human genome data.

Bibliography

[1] *23AndMe*. `https://www.23andme.com/`. Accessed: 2020-02-29.

[2] L. Adhikari and R. F. Marcia. "Nonconvex relaxation for Poisson intensity reconstruction". *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2015, pp. 1483–1487.

[3] J. Alvarez, O. DeGuchy, and R. Marcia. "Image Classification in Synthetic Aperture Radar Using Reconstruction From Learned Inverse Scattering". Submitted, 2020.

[4] E. Alzaid and A. E. Allali. "PostSV: A Post–Processing Approach for Filtering Structural Variations". *Bioinformatics and Biology Insights* 14 (2020), p. 1177932219892957.

[5] S. Amari, A. Cichocki, and H. H. Yang. "A new learning algorithm for blind signal separation". *Advances in Neural Information Processing Systems*. 1996, pp. 757–763.

[6] *Ancestry*. `https://www.ancestry.com/`. Accessed: 2020-02-29.

[7] D. Antaki, W. M. Brandler, and J. Sebat. "SV2: accurate structural variation genotyping and de novo mutation detection from whole genomes". *Bioinformatics* 34.10 (2018), pp. 1774–1777.

[8] D. Aymeric. *TF Learn*. https://github.com/aymeric damien/TensorFlow-Examples/blob/master/examples/. GitHub Repository. 2017.

[9] I. Babajide M. and F. Saeed. "Bioactive molecule prediction using extreme gradient boosting". *Molecules* 21.8 (2016), p. 983.

[10] B. Baesens et al. "Using neural network rule extraction and decision tables for credit-risk evaluation". *Management science* 49.3 (2003), pp. 312–329.

[11] M. Banuelos et al. "Nonconvex regularization for sparse genomic variant signal detection". *2017 IEEE International Symposium on Medical Measurements and Applications*. IEEE. 2017, pp. 281–286.

[12] M. Banuelos et al. "Related inference: A supervised learning approach to detect signal variation in genome data". Submitted, 2020.

[13] M. Banuelos et al. "Sparse Signal Recovery Methods for Variant Detection in Next-Generation Sequencing Data". *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*. 2016.

[14] R. Baraniuk and P. Steeghs. “Compressive radar imaging”. *Radar Conference, 2007 IEEE*. IEEE. 2007, pp. 128–133.

[15] C. Bartenhagen and M. Dugas. “RSVSim: an R/Bioconductor package for the simulation of structural variations”. *Bioinformatics* 29.13 (2013), pp. 1679–1681.

[16] A. Belouchrani et al. “A blind source separation technique using second-order statistics”. *IEEE Transactions on Signal Processing* 45.2 (1997), pp. 434–444.

[17] H. S. Bhat and S. J. Goldman-Mellor. “Predicting adolescent suicide attempts with neural networks”. *arXiv preprint arXiv:1711.10057* (2017).

[18] P. Bofill and M. Zibulevsky. “Underdetermined blind source separation using sparse representations”. *Signal Processing* 81.11 (2001), pp. 2353–2362.

[19] L. Borcea, G. Papanicolaou, and C. Tsogka. “Coherent interferometric imaging in clutter”. *Geophysics* 71.4 (2006), SI165–SI175.

[20] L. Borcea et al. “Synthetic aperture imaging of directional and frequency dependent reflectivity”. *SIAM Journal on Imaging Sciences* 9.1 (2016), pp. 52–81.

[21] J.-F. Cardoso. “Blind signal separation: statistical principles”. *Proceedings of the IEEE* 86.10 (1998), pp. 2009–2025.

[22] C.-Y. Chen, T.-T. Yang, and W.-S. Sun. “Optics system design applying a micro-prism array of a single lens stereo image pair”. *Optics Express* 16.20 (2008), pp. 15495–15505.

[23] S. Chen and H. Wang. “SAR target recognition based on deep learning”. *2014 International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2014, pp. 541–547.

[24] S. Chen et al. “Target classification using the deep convolutional networks for SAR images”. *IEEE Transactions on Geoscience and Remote Sensing* 54.8 (2016), pp. 4806–4817.

[25] T. Chen and C. Guestrin. “Xgboost: A scalable tree boosting system”. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[26] X. Chen et al. “EGBMMDA: extreme gradient boosting machine for MiRNA-disease association prediction”. *Cell death & disease* 9.1 (2018), pp. 1–16.

[27] M. Cheney. “A mathematical tutorial on synthetic aperture radar”. *SIAM Review* 43 (2001), pp. 301–312.

[28] M. Cheney and B. Borden. *Fundamentals of Radar Imaging*. SIAM, 2009.

[29] C. Chiang et al. “SpeedSeq: ultra-fast personal genome analysis and interpretation”. *Nature methods* 12.10 (2015), p. 966.

[30] A. Cichocki et al. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[31] R. Collobert, S. Bengio, and J. Mariéthoz. *Torch: a modular machine learning software library*. Tech. rep. Idiap Research Institute, 2002.

[32] P. Comon and C. Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic Press, 2010.

[33] I. H. G. S. Consortium et al. "Initial sequencing and analysis of the human genome". *Nature* 409 (2001), pp. 860–921.

[34] O. DeGuchy et al. *Forward and Inverse Scattering in Synthetic Aperture Radar Using Machine Learning*. 2020.

[35] O. DeGuchy, A. Ho, and R. F. Marcia. "Image disambiguation with deep neural networks". *Applications of Machine Learning*. Vol. 11139. International Society for Optics and Photonics. 2019, 111390A.

[36] O. DeGuchy et al. "Deep Neural Networks for Low-resolution Photon-limited Imaging". *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 3247–3251.

[37] D. L. Donoho. "Compressed sensing". *IEEE Transactions on information theory* 52.4 (2006), pp. 1289–1306.

[38] F.-X. Dupé, J. M. Fadili, and J.-L. Starck. "A proximal iteration for deconvolving Poisson noisy images using sparse representations". *IEEE Transactions on Image Processing* 18.2 (2009), pp. 310–321.

[39] M. A. Eberle et al. "A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree". *Genome research* 27.1 (2017), pp. 157–164.

[40] J. Fan et al. "Comparison of Support Vector Machine and Extreme Gradient Boosting for predicting daily global solar radiation using temperature and precipitation in humid subtropical climates: A case study in China". *Energy Conversion and Management* 164 (2018), pp. 102–111.

[41] A. C. Fannjiang, T. Strohmer, and P. Yan. "Compressed remote sensing of sparse objects". *SIAM Journal on Imaging Sciences* 3.3 (2010), pp. 595–618.

[42] J. A. Fessler and A. O. Hero. "Penalized maximum-likelihood image reconstruction using space-alternating generalized EM algorithms". *IEEE Trans. Image Processing* 4.10 (1995), pp. 1417–1429.

[43] M. A. T. Figueiredo and J. M. Bioucas-Dias. "Restoration of Poissonian images using alternating direction optimization". *IEEE transactions on Image Processing* 19.12 (2010), pp. 3133–3145.

[44] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Vol. 1. Springer series in statistics New York, 2001.

[45] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[46] R. Girshick et al. "Region-based convolutional networks for accurate object detection and semantic segmentation". *IEEE Trans. Pattern Anal. Mach. Intell.* 38.1 (2016), pp. 142–158.

[47] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press Cambridge, 2016.

[48] Z. T. Harmany, R. F. Marcia, and R. M. Willett. "Sparse Poisson intensity reconstruction algorithms". *Statistical Signal Processing, 2009. SSP'09. IEEE/SP 15th Workshop on*. IEEE. 2009, pp. 634–637.

[49] Z. T. Harmany, R. F. Marcia, and R. M. Willett. "This is SPIRAL-TAP: Sparse Poisson Intensity Reconstruction ALgorithms; Theory and Practice". *IEEE Trans. on Image Processing* 21.3 (2012), pp. 1084–1096.

[50] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

[51] S. S. Ho, A. E. Urban, and R. E. Mills. "Structural variation in the sequencing era". *Nature Reviews Genetics* (2019), pp. 1–19.

[52] N. Homer. "Dwgsim: whole genome simulator for next-generation sequencing". *GitHub repository* (2010).

[53] R. Horisaki and J. Tanida. "Multi-channel data acquisition using multiplexed imaging with spatial encoding". *Optics Express* 18.22 (2010), pp. 23041–23053.

[54] A. J. Iafrate et al. "Detection of large-scale variation in the human genome". *Nature genetics* 36.9 (2004), pp. 949–951.

[55] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". *arXiv preprint arXiv:1502.03167* (2015).

[56] D. C. Jeffares et al. "Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast". *Nature communications* 8.1 (2017), pp. 1–11.

[57] A. Khemphila and V. Boonjing. "Heart disease classification using neural network and feature selection". *2011 21st International Conference on Systems Engineering*. IEEE. 2011, pp. 406–409.

[58] Y. Khoo and L. Ying. "SwitchNet: a neural network model for forward and inverse scattering problems". *arXiv preprint arXiv:1810.09675* (2018).

[59] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". *arXiv preprint arXiv:1312.6114* (2013).

[60] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[61] W. P. Kloosterman et al. “Characteristics of de novo structural changes in the human genome”. *Genome research* 25.6 (2015), pp. 792–801.

[62] A. Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009.

[63] A. Krizhevsky, I. Sutskever, and G. Hinton. “Imagenet classification with deep convolutional neural networks”. *Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 1097–1115.

[64] K. Kulkarni et al. “Reconnet: Non-iterative reconstruction of images from compressively sensed measurements”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 449–458.

[65] Y. LeCun et al. “Gradient-based learning applied to document recognition”. *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[66] D. J. Lingenfelter, J. A. Fessler, and Z. He. “Sparsity regularization for image reconstruction with Poisson data”. *Computational Imaging VII*. Vol. 7246. International Society for Optics and Photonics. 2009, 72460F.

[67] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. *IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[68] K. G. Lore, A. Akintayo, and S. Sarkar. “LLNet: A deep autoencoder approach to natural low-light image enhancement”. *Pattern Recognition* 61 (2017), pp. 650–662.

[69] G. H. Lubke et al. “Gradient boosting as a SNP filter: An evaluation using simulated and hair morphology data”. *Journal of data mining in genomics & proteomics* 4 (2013).

[70] R. F. Marcia et al. “Superimposed video disambiguation for increased field of view”. *Optics Express* 16.21 (2008), pp. 16352–16363.

[71] J. Masci et al. “Stacked convolutional auto-encoders for hierarchical feature extraction”. *International Conference on Artificial Neural Networks*. Springer. 2011, pp. 52–59.

[72] A. Moreira et al. “A tutorial on synthetic aperture radar”. *IEEE Geoscience and remote sensing magazine* 1.1 (2013), pp. 6–43.

[73] D. Morgan. “Deep convolutional neural networks for ATR from SAR imagery”. *Algorithms for Synthetic Aperture Radar Imagery XXII*. Vol. 9475. International Society for Optics and Photonics. 2015, 94750F.

[74] A. Mousavi and R. G. Baraniuk. “Learning to invert: Signal recovery via deep convolutional networks”. *arXiv preprint arXiv:1701.03891* (2017).

[75] A. Mousavi, A. Patel, and R. G. Baraniuk. “A deep learning approach to structured signal recovery”. *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE. 2015, pp. 1336–1343.

[76] H. Park et al. “Detection of chromosome structural variation by targeted next-generation sequencing and a deep learning application”. *Scientific reports* 9.1 (2019), pp. 1–9.

[77] A. Paszke et al. “Automatic differentiation in PyTorch”. *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*. Long Beach, CA, US, 2017.

[78] B. S. Pedersen and A. R. Quinlan. “cyvcf2: fast, flexible variant analysis with Python”. *Bioinformatics* (2017).

[79] L. C. Potter et al. “Sparsity and compressed sensing in radar imaging”. *Proceedings of the IEEE* 98.6 (2010), pp. 1006–1020.

[80] T. Rausch et al. “DELLY: structural variant discovery by integrated paired-end and split-read analysis”. *Bioinformatics* 28.18 (2012), pp. i333–i339.

[81] T. Remez et al. “Deep Convolutional Denoising of Low-Light Images”. *arXiv preprint arXiv:1701.01687* (2017).

[82] M. Riedmiller and H. Braun. “A direct adaptive method for faster backpropagation learning: The RPROP algorithm”. *Neural Networks, 1993., IEEE International Conference on*. IEEE. 1993, pp. 586–591.

[83] S. Ruder. “An overview of gradient descent optimization algorithms”. *arXiv preprint arXiv:1609.04747* (2016).

[84] R. H. Shepard et al. “Design architectures for optically multiplexed imaging”. *Optics Express* 23.24 (2015), pp. 31419–31435.

[85] P. Y. Simard, D. Steinkraus, J. C. Platt, et al. “Best practices for convolutional neural networks applied to visual document analysis”. *ICDAR*. Vol. 3. 2003, pp. 958–962.

[86] S. S. Sindi et al. “An integrative probabilistic model for identification of structural variation in sequencing data”. *Genome biology* 13.3 (2012), R22.

[87] D. L. Snyder and M. I. Miller. *Random Point Processes in Time and Space*. Springer, 1991.

[88] M. Spence et al. “Detecting Novel Structural Variants In Genomes By Leveraging Parent-Child Relatedness”. *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. Dec. 2018, pp. 943–950. DOI: 10.1109/BIBM.2018.8621488.

[89] V. Treeaporn, A. Ashok, and M. A. Neifeld. “Increased field of view through optical multiplexing”. *Optics Express* 18.21 (2010), pp. 22432–22445.

[90] M. Trencseni. *Solving CIFAR-10 with Pytorch and SKL*. http://bytepawn.com/solving-cifar-10-with-pytorch-and-skl.html. 2019.

[91] S. Uttam et al. “Optically multiplexed imaging with superposition space tracking”. *Optics Express* 17.3 (2009), pp. 1691–1713.

[92] P. Vincent et al. “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion”. *Journal of Machine Learning Research* 11.Dec (2010), pp. 3371–3408.

[93] P. Vincent et al. “Extracting and composing robust features with denoising autoencoders”. *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103.

[94] N. Wang and D.-Y. Yeung. “Learning a deep compact image representation for visual tracking”. *Advances in Neural Information Processing Systems*. 2013, pp. 809–817.

[95] J. Weischenfeldt, F. Symmons O.and Spitz, and J. Korbel. “Phenotypic impact of genomic structural variation: insights from and for human disease”. *Nature Reviews Genetics* 14.2 (2013), pp. 125–138.

[96] J. Xie, L. Xu, and E. Chen. “Image denoising and inpainting with deep neural networks”. *Advances in Neural Information Processing Systems*. 2012, pp. 341–349.

[97] J. Xu et al. “Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images”. *IEEE Transactions on Medical Imaging* 35.1 (2016), pp. 119–130.

[98] B. Yonel, E. Mason, and B. Yazıcı. “Deep Learning for Passive Synthetic Aperture Radar”. *IEEE Journal of Selected Topics in Signal Processing* 12.1 (Feb. 2018), pp. 90–103. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2017.2784181.

[99] X. X. Zhu et al. “Deep learning in remote sensing”. *IEEE Geoscience and remote sensing magazine* 5.4 (2017), pp. 8–36.

[100] M. Zibulevsky and B. A. Pearlmutter. “Blind source separation by sparse decomposition in a signal dictionary”. *Neural Computation* 13.4 (2001), pp. 863–882.

Chapter 7

Conclusion

7.1 Summary

Signals appear in a diverse number of forms including images, sounds, time series, etc. These signals are constantly being captured by the increasingly complex technology available in our lives for a variety of applications. In many cases we are required to recover these signals from corrupt or low quality data, leaving much of the work to the algorithms used for reconstruction. In most compressive sensing algorithms we rely on optimization techniques to minimize an objective function providing us with some semblance of a reconstruction. Often, this requires large scale optimization techniques to handle the increased dimensions of the signals. In this work, we explored optimization techniques for large scale sparse signal recovery. We also focused on the photon-limited regime, which increases the difficulty of the problem in that the noise is no longer additive as in a Gaussian model. The problem becomes highly non-linear and non-convex. While numerical optimization has been the standard approach for signal recovery. This works attempts to draw on experience in optimization and in signal recovery in order to apply deep learning techniques as an alternative to standard methods.

The following summarizes my contributions of the dissertation towards data driven signal processing algorithms:

- We proposed a novel approach for solving the sparse ℓ_2-ℓ_1 problem in a trust-region setting. By using an approximation of second-derivative information known as the L-BFGS update we improve the quality of the optimization routine and provide improved reconstructions from noisy observations.

- We extended the formulation of the compact representation typically used to compute the L-BFGS to include the Full Broyden class of quasi-Newton updates allowing the quasi-Newton method to update at each iteration. We demonstrate the ability to accurately and efficiently use the formulation to solve large scale linear systems often involved in solving large scale optimization problems.

- We proposed an approach to fluorescence lifetime imaging where in we use aprori knowledge about the sparsity and intensity of the intended signal in order to speed up and improve signal estimation. We were required to model the imaging process and formulate the bounded optimization problem for signal recovery.

- We proposed a non-convex Shannon entropy regularization method in order to promote sparsity in the context of photon limited imaging. We introduced the weighted Shannon Entropy function as an alternative to ℓ_1 based approaches.
- We proposed an asynchronous parallel pattern search method in order to tune the hyperparamters involved in the photon-limited recovery problem. The objective of the pattern search was to find the optimal parameter p for the p-norm being used as well as the intensity of the sparsity controlled by τ. These parameters have a significant impact on the quality of the reconstruction and had previously been chosen through trial and error. The method achieves favorable results without any *a priori* information about the signal.
- We proposed a Quasi-Newton based method as an alternative to the gradient descent based methods typically used to train neural networks. We approximate second derivative information using the gradient within a trust-region setting. The trust region method improves our approximation of the objective function and promotes generality of the network to data outside of the training set.
- We proposed a method which incorporates second derivative information without explicitly storing or computing the hessian. This was also presented in a trust-region setting where the associated subproblem was solved using a conjugate based method. The benefit of the method is that the quality of the optimization is improved and requires less iterations and less data to train the neural network.
- We implemented deep neural networks to address the photon-limited imaging problem originally addressed as a constrained optimization problem. By using a deep neural network, the method is completely data driven. Furthermore we no longer have to make any assumptions on the nature of the signal. We compared fully connected layers to convolutional layers and found that the performance of fully connected layers was robust to various levels of noise and compression.
- We proposed neural networks to perform image disambiguation. The signal retrieval problem, often referred to as blind source separation, is particularly ill-posed. Our network was able to disambiguate two super imposed images and in some cases viewed weaker images as noise, while extracting the other image.
- We proposed the use of a simplified neural network in the application of remote sensing. The neural network architecture allowed us to capture the physics of the SAR sensing system. The same architecture can be used to recover an inverse, thus allowing us to solve the linear system associated with recovering the target of interest.
- We extended the SAR recovery technique to the problem of target recognition. We showed that the quality of the reconstruction is of great importance when it comes to object detection. Furthermore, we were able to display that our method improves on a regime in which nothing is known about the imaging system.
- Recently we applied deep learning to the world of genomics. For this application we leveraged the output of current techniques in order to reduce the number of false signals when detecting structural variants in the human genome. While neural networks performed less than favorable when compared to an ensemble method, it shows promise when applied to the problem while exhibiting consistency from synthetic data to human genome data.

7.2 Future Work

We believe that the use of deep learning techniques for digital signal recovery is still in its early stages. In particular, when it comes to denoising applications there is still room for compelling advancements. Therefore, to further extend some of the applications we would like to propose the following future work:

- More advanced architectures such as generative adversarial networks (GANs) [1] or variational autoencoders [2] have shown a tremendous amount of promise in applications of inpainting and super resolution. While these methods have been explored for the standard Gaussian noise model, the photon-limited regime would benefit from these types of techniques.
- We have been focused on applications in the photon-limited regime and thus our work has been determined by the Poisson model. We would like to explore different types of noise, particularly what is known as speckle [3]. Our interest in speckle is motivated in the fact that it is often found in synthetic aperture radar data.
- Recurrent neural networks (RNNs) have been shown to be powerful tools when it comes to time-dependent or sequential data such as speech recognition [4]. We would like to formulate the signal recovery process as a time-dependent process in order to leverage the RNN architecture.

As we look towards the future we hope to continue to leverage our experience in optimization to improve the algorithms used to train neural networks. We also look forward to incorporating problem specific information into architectures which could improve the quality of the signal reconstruction.

Bibliography

[1] I. Goodfellow et al. “Generative adversarial nets”. *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[2] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. *arXiv preprint arXiv:1312.6114* (2013).

[3] J.-S. Lee. “Speckle analysis and smoothing of synthetic aperture radar images”. *Computer graphics and image processing* 17.1 (1981), pp. 24–32.

[4] M. Sundermeyer, R. Schlüter, and H. Ney. “LSTM neural networks for language modeling”. *Thirteenth annual conference of the international speech communication association*. 2012.