

# Physics-based Learning for Large-scale Computational Imaging

*Michael Kellman*



Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2020-167

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-167.html>

August 14, 2020

Copyright © 2020, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Physics-based Learning for Large-scale Computational Imaging

by

Michael Robert Kellman

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Laura Waller, Co-chair  
Associate Professor Michael Lustig, Co-chair  
Assistant Professor Ren Ng  
Professor Bruno Olshausen

Summer 2020

Physics-based Learning for Large-scale Computational Imaging

Copyright 2020  
by  
Michael Robert Kellman

## Abstract

Physics-based Learning for Large-scale Computational Imaging

by

Michael Robert Kellman

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Associate Professor Laura Waller, Co-chair

Associate Professor Michael Lustig, Co-chair

In computational imaging systems (e.g. tomographic systems, computational optics, magnetic resonance imaging) the acquisition of data and reconstruction of images are co-designed to retrieve information which is not traditionally accessible. The performance of such systems is characterized by how information is encoded to (forward process) and decoded from (inverse problem) the measurements. Recently, critical aspects of these systems, such as their signal prior, have been optimized using deep neural networks formed from unrolling the iterations of a physics-based image reconstruction.

In this dissertation, I will detail my work, *physics-based learned design*, to optimize the performance of the entire computational imaging system by jointly learning aspects of its experimental design and computational reconstruction. As an application, I introduce how the LED-array microscope performs super-resolved quantitative phase imaging and demonstrate how physics-based learning can optimize a reduced set of measurements without sacrificing performance to enable the imaging of live fast moving biology.

In this dissertation's latter half, I will discuss how to overcome some of the computational challenges encountered in applying physics-based learning concepts to large-scale computational imaging systems. I will describe my work, *memory-efficient learning*, that makes physics-based learning for large-scale systems feasible on commercially-available graphics processing units. I demonstrate this method on two large-scale real-world systems: 3D multi-channel compressed sensing MRI and super-resolution optical microscopy.

*To my close friends and family, without all of you this would have not been possible.*

# Contents

<b>Contents</b>	ii
<b>List of Figures</b>	iv
<b>List of Tables</b>	x
<b>List of Algorithms</b>	xi
<b>1 Introduction</b>	1
<b>2 Computational Microscopy</b>	6
2.1 Optical Microscopy . . . . .	6
2.2 Quantitative Phase Imaging . . . . .	8
2.3 Computational LED Array Microscope . . . . .	10
2.4 Generalized Image Reconstruction . . . . .	11
<b>3 Physics-based Learned Design</b>	14
3.1 Introduction . . . . .	14
3.2 Quantitative Differential Phase Contrast . . . . .	15
3.3 Physics-Based Learned Design . . . . .	19
3.4 Results . . . . .	24
3.5 Remarks . . . . .	31
<b>4 Physics-based Learned Design for Fourier Ptychographic Microscopy</b>	33
4.1 Introduction . . . . .	33
4.2 Fourier Ptychographic Microscopy . . . . .	34
4.3 Physics-based Learned Design . . . . .	36
4.4 Results . . . . .	39
4.5 Remarks . . . . .	45
<b>5 Memory-efficient Learning for Large-scale Computational Imaging</b>	47
5.1 Introduction . . . . .	47
5.2 Related Works . . . . .	49

5.3 Methods	51
5.4 Hybrid Reverse Recalculation and Checkpointing	55
5.5 Results	55
5.6 Memory-Computation Analysis	63
5.7 Remarks	65
<b>6 Conclusion</b>	<b>68</b>
6.1 Outlook	68
6.2 Future work	68
<b>A Open Source: How to implement</b>	
<b>physics-based learning</b>	<b>72</b>
A.1 Introduction	72
A.2 Basic Tutorial	72
A.3 Advanced Implementation	74
<b>B Total Variation Regularization</b>	<b>78</b>
<b>Bibliography</b>	<b>80</b>



# List of Figures

2.1	Microscopy definitions of resolution: (a) Abbe resolution criteria, $k_{\text{Abbe}}$ , characterized by the transfer function's support in Fourier space. (b) Full-width half max or single point resolution, $d_{\text{FWHM}}$ , defined by the point spread function. (c) Rayleigh resolution criteria or two point resolution, $d_{\text{Ray}}$ , characterized by minimum resolvable distance between two points. On the left are two resolvable points as highlighted by the dip between the resultant image and on the right are two unresolvable points. Below (b) and (c) are cross sections through the system's response to a single point, two resolvable points, and two unresolvable points. . . . .	7
2.2	Bright-field sample contrasts: (a) absorption image and (b) quantitative phase image (computed using four measurements and quantitative differential phase contrast [9]) of 3T3 cells plated in a single layer. . . . .	8
2.3	Light-emitting diode microscope: (a) Schematic of experimental setup imaging the a USAF phase resolution target. (b) Each light-emitting diode (LED) modulates a different part of the sample's Fourier space into the passband of the microscope. The colored circles correspond to different LED's Fourier space support, each of which has a small bandwidth (low resolution). The outer dashed circle represents the upper limit of the Fourier support for the LED array microscope (determined by the position of the outer most LEDs). (c) Intensity images are captured in real space by the camera sensor under different LED illuminations (displayed). Each colored circle in (b) corresponds to a color-outlined measurement in (c). . . . .	9
2.4	Multiplexed LED illumination patterns: (a) the LED illumination patterns for bright field, dark field, differential phase contrast and (b) their resulting intensity measurements of a phase USAF resolution target. Insets highlight the variety of contrasts each illumination pattern produces. . . . .	11

3.1	LED illumination patterns: Absorption and phase weak object transfer functions (WOTF) and the resulting intensity measurements for (a) single-LED patterns and (b) multi-LED patterns. Asymmetric LED patterns provide a significant amounts of phase contrast, while symmetric pattern provide better absorption contrast. All weak object transfer functions are normalized for visualization purposes. . . . .	18
3.2	Physics-based networks (PbN) are formed by unrolling the iterations of an image reconstruction optimization. Each layer contains one iteration, made up of a data consistency update and signal prior update. The PbN input is the reconstruction's initialization, $\mathbf{x}^{(0)}$ , and the system's measurements, $\{\mathbf{y}_k\}_k$ , and the output is the reconstructed image from the $N^{\text{th}}$ layer, which is fed into the learning loss, $\mathcal{L}$ . . . . .	20
3.3	Coded-illumination designs and their corresponding phase weak object transfer functions (WOTFs) for: (a) Traditional qDPC and (b) learned designs for the case where 4, 3, or 2 measurements are allowed for each phase reconstruction. The illumination source patterns are in the upper left corners, with gray semi-circles denoting where the LEDs are constrained to be "off". . . . .	25
3.4	Phase reconstruction results using simulated measurements with different coded-illumination designs. I compare results from: traditional qDPC (half-circles), annular illumination, condition number optimization, A-optimal design, and the proposed physics-based learned designs. I show results for the cases of (a) four, (b) three, and (c) two measurements allowed for each phase reconstruction. Absolute error maps are shown below each reconstruction. . . . .	26
3.5	Coded-illumination designs and their phase weak object transfer functions for (a) traditional qDPC, (b) annular illumination, (c) condition number optimized, (d) A-optimal, and (e) physics-based learned designs. . . . .	27
3.6	USAF phase target reconstructions: Experimental comparison between phase results with (a) Fourier Ptychography (FP) using 69 images, (b) traditional qDPC and (c) learned designs, for the case of 4, 3, and 2 measurements. Error maps show the difference from the FP reconstruction. (d) Cross-sections show that phase from the learned designs (long-dashed red) is closer to that of FP (solid blue) than traditional qDPC (short-dashed green). . . . .	28
3.7	3T3 mouse fibroblast cells reconstructions: Experimental comparison between phase results with (a) Fourier Ptychography (FP) using 69 measurements, (b) traditional qDPC and (c) learned designs, for the case of 4, 3, and 2 measurements. Error maps show the difference from the FP reconstruction. (d) Cross-sections show that phase from the learned designs (long-dashed red) is closer to that of FP (solid blue) than traditional qDPC (short-dashed green). . . . .	29

4.1	Physics-based Learned Design: (a) We start with an input dataset of single-LED images. (b) For each measurement’s LED pattern, we learn the scalar weights, $c_i$ , that correspond to the brightness of each LED. (c) Each multiplexed measurement is then fed into (d) the Physics-based Network (PbN), where each layer represents a single gradient descent (GD) step of the image reconstruction. (e) The output super-resolved reconstruction is then compared to the target reconstruction via (f) a context-specific loss function, in order to optimize the brightness of each LED. . . . .	36
4.2	LED Designs: LED source patterns for (a) heuristic multiplexing designs with 15, 10 and 5 measurements, (b) the learned designs for super-resolved amplitude imaging with 15, 10 and 5 measurements, and (c) the learned designs for super-resolved quantitative phase imaging with 15, 10 and 5 measurements. The inner blue circle denotes LEDs in the bright-field region, while the outer blue shell denotes the dark-field region. An LED’s shade of green corresponds to that LED’s brightness. . . . .	39
4.3	Simulations of super-resolution amplitude imaging: (a) Ground truth amplitude. (b) Amplitude reconstructions from simulated measurements using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and their error maps. . . . .	40
4.4	Simulations of super-resolution quantitative phase imaging: (a) Ground truth phase. (b) Phase reconstructions from simulations using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and their error maps. . . . .	41
4.5	Experimental amplitude target results: (a) Single-LED amplitude reconstruction using 89 measurements serves as ground truth. (b) Amplitude reconstructions using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and difference with the ground truth. . . . .	43
4.6	Experimental phase target results: (a) Single-LED phase reconstruction using 89 measurements serves as ground truth. (b) Phase reconstructions using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and difference with the ground truth. . . . .	44
4.7	Bright-Field vs. dark-Field LED designs: (a) Asymmetric and symmetric <i>bright-field</i> LED patterns for amplitude and phase samples. (b) Asymmetric and symmetric <i>dark-field</i> LED patterns for amplitude and phase samples. . . . .	45

5.1	Overview of memory-efficient methods: Storage and computational complexity of standard backpropagation, forward recalculation, forward recalculation with checkpointing, reverse recalculation, and reverse recalculation with checkpointing for a physics-based network with $N$ layers and $K$ is the spacing between checkpoints. Methods' computational accuracy are qualitatively reported from exact (three green dots) to approximate (one green dot). . . . .	48
5.2	Memory-efficient learning for physics-based networks: (a) Physics-based networks (PbN) are formed by unrolling the iterations of an image reconstruction optimization. Each layer contains one iteration, made up of a data consistency update and signal prior update. The PbN input is the reconstruction's initialization, $\mathbf{x}^{(0)}$ , and the output is the reconstructed image from the $N^{\text{th}}$ layer, which is fed into the learning loss, $\mathcal{L}$ . (b) Memory-efficient learning procedure for a single layer: (1) recalculate the layer's input, $\mathbf{x}^{(n-1)}$ , from the output, $\mathbf{x}^{(n)}$ , by applying that layer's inverse operations. (2) Recompute the auto-differentiation graph for that single layer. (3) Backpropagate gradients, $\mathbf{q}^{(n)} = \partial\mathcal{L}/\partial\mathbf{x}^{(n)}$ , through the layer's auto-differentiation graph. . . . .	51
5.3	Learned measurements for 1D compressed sensing: (a) Mean testing loss for learning using standard and memory-efficient learning techniques. (b) Initial (Gaussian randomly distributed) and learned measurement matrices using standard and memory-efficient techniques. (c) Two testing examples of reconstructions with random and learned measurement schemes, demonstrating both improved signal recovery using the learned measurements in comparison to the random measurements and similarity between standard and memory-efficient learning (while requiring 4.1KB, $\sim 800\times$ less memory than standard backpropagation). . . . .	56
5.4	Learned priors for multi-channel 2D under-sampled MRI: (a) Mean testing loss is similar for both standard backpropagation and memory-efficient learning. (b) Ground truth reconstruction using fully sampled measurements, (c) linear parallel imaging reconstruction (no prior), (d) PbN reconstruction learned using standard backpropagation and (e) PbN reconstruction learned using memory-efficient learning ( $3.7\times$ reduced memory requirement, $1.2\times$ increase in compute time). Insets highlight fidelity of high-resolution features and noise reduction in both of the learned designs, as compared to the CG reconstruction. Reported memory and time required is for a single learning update with batch size one. . . . .	58
5.5	Learned priors for multi-channel under-sampled 3D MRI: (a) Mean training and testing loss for learning with the proposed memory-efficient technique. (b) One slice of ground truth 3D reconstruction using fully sampled measurements, (c) linear parallel imaging reconstruction (no prior), (d) PbN reconstruction using memory-efficient learning with $\sim 10\text{GB}$ of memory. Standard learning is not shown because it requires more memory than would fit on a commercial GPU. . . . .	60

5.6	Multi-channel MRI performance versus number of unrolls: Average testing PSNR for varying number of unrolled iterations. In this example we learn independent parameters for each layer. . . . .	61
5.7	Learned illumination design for Fourier Ptychographic Microscopy (FPM): (a) Mean testing loss is similar for both standard backpropagation and memory-efficient learning. (b) Example low-resolution measurement, (c) ground truth reconstruction using all 89 LED measurements to perform $3.1\times$ super resolution, (d) reconstruction from only 8 measurements learned using standard backpropagation and (e) memory-efficient learning ( $92\times$ reduced memory requirement, $1.7\times$ increase in compute time). Reported memory and time required is for a single learning update with batch size one. . . . .	61
5.8	Large-scale learned illumination design for FPM: (a) Training and testing loss for memory-efficient learned design. (b) Example low-resolution measurement, (c) ground truth reconstruction using all 293 LED measurements to perform $4.2\times$ super resolution, (d) reconstruction from only 16 measurements learned using memory-efficient learning with $\sim 3\text{GB}$ memory. Standard learning is not shown because it would require $\sim 500\text{GB}$ of memory, which is not available on commercial GPU. Insets highlight high-resolution features. . . . .	62
5.9	Fourier Ptychographic Microscopy performance versus number of unrolls: Average testing PSNR for varying number of unrolled iterations. . . . .	63
5.10	Fastest method for varying layer and checkpoint size: (a) Fastest method for three scenarios where the ratio between forward and inverse layer computation varies. The proposed method performs better with larger ratios because it calls each layer's inverse more and forward checkpointing calls each layer's forward more. (b) Fastest method for three scenarios with increasing number of unrolled iterations. The proposed method performs better as more iterations are unrolled. Sector color corresponds to the fastest method: purple, orange, yellow, and blue representing standard backpropagation, forward checkpointing, my method, and problems that do not fit on a single GPU (12 GB), respectively. . . . .	64
A.1	Implementation of a physics-based network for compressed sensing sparse recovery using automatic differentiation to compute gradients with respect to $\mathbf{x}$ . The network is setup to learn the measurement matrix, step size, and sparsity penalty. Soft thresholding is used as the proximal operator. . . . .	74
A.2	Memory-efficient Learning Framework: (a) physics-based layers encode the forward model process into the each layer's operations, (b) physics-based networks are formed from a list of physics-based layers to perform the image reconstruction, and (c) network managers handle how to compute gradients using backpropagation through the physics-based networks. . . . .	75

A.3	Physics-based Gradient Descent Layer: A template for a gradient descent based layer. The forward performs the layer's operations and reverse function performs the layer's inverse operations.	76
A.4	Physics-based Network: A template for a physics-based network formed from gradient descent. The main attribute of this class is a list, <code>torch.nn.ModuleList</code> , of physics-based layers.	76

# List of Tables

3.1	PSNR Results: Average and standard deviation PSNR (dB) of phase reconstructions from the simulated testing examples using different illumination schemes and different numbers of measurements. Factor format: Mean $\pm$ Std.	27
3.2	PSNR Results: Average and standard deviation PSNR (dB) of phase reconstructions from the simulated testing examples using learned design for different numbers of unrolled iterations. Factor format: Mean $\pm$ Std.	30
4.1	PSNR for simulated amplitude reconstructions: average testing LF-PSNR and HF-PSNR (dB) for different numbers of measurements.	42
4.2	PSNR for simulated phase reconstructions: average testing LF-PSNR and HF-PSNR (dB) for different numbers of measurements.	42

# List of Algorithms

2.1	Proximal Gradient Descent	13
2.2	Half Quadratic Splitting	13
3.1	Accelerated Proximal Gradient Descent (APGD) for Phase Recovery	19
3.2	Physics-based Learned Design Algorithm	22
3.3	Gradient Update for Single Training Example	24
5.1	Memory-efficient learning for physics-based networks	52
5.2	Inverse for gradient layer	53
A.1	Proximal Gradient Descent	73



## Acknowledgments

First and foremost, I would like to thank my advisors, Miki Lustig and Laura Waller. They have provided me the opportunity and resources to achieve all that I could dream over the past five and half years. Without their guidance and support I would not be the person I am today.

My parents have played a huge role in this stage of my life. Through times of success and of difficulty, they were there to help figure out how to move forward. Both of you provided me the inspiration to become an engineer. To my mom, for piquing my interest in cooking and making. To my dad, for our scientific conversations and many games of Scabble. To my brother, I am grateful that you live so close. I have enjoyed all our time together.

To Emrah Bostan, without your friendship and mentorship my path through life would look quick different. Thank you for our many lively and frank discussions. The ones technical sparked a collaboration that produced most of the work in this dissertation. The rest inspired me to be a more multifaceted individual. I want to thank you for introducing me to the world of climbing. Your invitation to go bouldering led to many more and has changed the way that I live my life.

To Jon Tamir, among many other things, you have taught me how to be a critical thinker. You introduced me to the fun of 3D printing and taught me so much about the *right* way to perform image reconstruction. Thank you for our lively conversations on just about every topic.

To Shirley Salanio, navigating the difficulties of graduate school would not have been possible without you. You have gone above and beyond for me. I will be forever thankful.

To the Wednesday boardgame crowd, thank you for the many nights spent cooking and hanging out. Thank you David, Jon, Eric, Nate, Sadie Mae, Brittany, and Karthik. Our time together has been invaluable to me. To those times and many more (I know there will be many more).

To many climbing adventures, thank you Keekee, Eric, Nate, Sadie Mae, Jonas, Will, Sophie, and Melissa. To Keekee, you have in large part inspired me to live a life outdoors. Our road trip down to El Potrero Chico was an epic. It is a miracle we got there, and I would do it again. Thank you for teaching me about climbing, systems, and friendship. All of this has made many future adventures possible.

To Eric, Nate, Sadie Mae, you are tremendous friends. To Eric, teaching me how to climb stronger, even when I wore socks at the beginning. Also for teaching me how to brew beer, inspiring me to embark on another culinary endeavor. To Nate, for our trailless backpacking trips and introduction to the world of fermentation. My apartment is now filled with many growing creations. To Sadie Mae, for endless cooking and baking inspiration, many hard cycling adventures, and an introduction to protesting. I have enjoyed our many rides and look forward to all the future ones.

To all my lab members current, past, and future, you are the people that make the experience of graduate school what it is. To all of our shared hardships and successes. I

will fondly remember our many loud conversations in and across the office.

To Emma Alexander, for sharing your many concerns with my safety and wellbeing. Thank you for all of the citrus and laughs.

To Nick Antipa, for many lively discussions on society and graduate school. I hope we can enact many these ideas. Our trip to Japan was marked by many memorably ridiculous experiences. From preparing talks and ramen to getting lost on the subway and almost getting hit by a bus, this was the awesome conference/vacation/birthday. I look forward to seeing what the future holds.

To my friends from Carnegie Mellon University, Brendan, Sophia, Franny, Matt, and Neha. You are some of the few that have seen me through several stages of my life. I fondly remember many sunny fourth of July barbeques and late-night parties. Thank you for not losing touch with me over the years. Your kindness and friendships have pushed me to be more like the friend I would like have.

To the 1:15'ers, always pushing my excitement for our sport. Swimming has been a lifelong pastime of mine, but over time my dedication to it has waxed and waned. The master swim team has reaffirmed this passion. To Lauren, for introducing me to open water swimming and letting me know, "the water isn't really that cold". To Ben, for being fiercely competitive and pushing me to my limits in the pool. To Anna, for making me feel welcome in the Berkeley swimming family. To all the coaches, Kendall, Tiffany, Gaku, and Tryn, for making it all possible. This team has been and will continue to be a community that I am very proud to be a part of.

This stage of my life has been a journey. From the onset, I believed that my main memories would be of the work and the research; was I wrong. I will remember these years for the people I have met along the way and the experiences we have shared. It is the people, it is the people, it is the people.

Michael Kellman  
August 14<sup>th</sup>, 2020

# Chapter 1

## Introduction

Over the past century optical imaging systems have seen a steady rise in the reliance on computation. Originally based solely on the theory of image formation and light propagation, optical imaging systems were designed to directly encode information about the world into a measurement device, *i.e.* in a one-to-one manner. As the desire to encode more information increased, *e.g.* color and angular information, imaging systems started to manipulate how that information was encoded onto the sensor, *e.g.* multiplexing or mosaicing. With these new rich measurements, the systems then relied on computational processing to retrieve that information, *i.e.* decoding. This idea of indirectly encoding information and computationally decoding it birthed the field of computational imaging. In this new paradigm, we consider the co-design of hardware and software to push the envelope for traditional imaging trade-offs, *e.g.* resolution, field-of-view, signal-to-noise ratio. This paradigm shift in optical microscopy, termed *computational microscopy*, has enabled imaging beyond the diffraction limit, *i.e.* super resolution [1, 2], imaging transparent samples, *i.e.* phase imaging [3], and the volumetric imaging of samples, *i.e.* diffraction tomography [4, 5]; all impractical with either hardware design or computation alone.

Since the discovery of phase contrast by Frits Zernike in 1930 [6], phase imaging has become the standard label and stain-free method to image transparent biological samples. The phase of a sample is proportional to the product between a sample's refractive index and thickness and thus provides an endogenous source of spatial contrast. Qualitative phase imaging methods such as Differential Intensity Contrast (DIC) [7] and phase contrast (PhC) offer the ability to visually discern a sample's edges from the background using a single measurement. Such systems are ubiquitous in biological laboratories, however, they do not extract the sample's actual phase information, required to measure dry mass and volumetric information about the sample. With the advent of *computational microscopy*, Quantitative phase imaging (QPI) has allowed for the consistent and robust processing of a sample's phase information that qualitative imaging does not. Such systems operate by indirectly mapping a sample's phase information into measurements, *e.g.* using interferometric, coded-illumination, or coded-pupil methods, and computationally decoding that information from the measurements. The performance of these systems is

fundamentally governed by how well the desired information is encoded to and decoded from the measurements.

Of great practical interest, *coded-illumination microscopy* has offered an accurate and inexpensive way to capture QPI. To realize coded-illumination, a commercial microscope's illumination unit can be replaced with a programmable light-emitting diode (LED) array [8]. Each LED illuminating the sample modulates unique spatial frequency information of the sample into the measurements. A set of measurements is captured under diverse illumination patterns and fed into a non-linear phase retrieval optimization to estimate the sample's phase information. Using the same hardware, quantitative phase information can be extracted with resolution twice that of the objective (using four measurements) [9], with resolution beyond twice that of the objective (using Fourier Ptychography and tens to hundreds of measurements) [1, 2], and in 3D (using diffraction tomography and hundreds of measurements) [4, 5]. While all enable label-free imaging of transparent samples, the multi-measurement nature of these methods inhibit their use to image live fast-moving biology due to reduced temporal resolution and the speed of the sample [10]. Reducing the number of measurements is possible [11], however, as fewer measurements are acquired, image quality degrades significantly. The natural questions to ask are: can we design a set of measurements without sacrificing image quality? How do we design that set of measurements to provide the best reconstruction quality for a class of images?

Traditionally, design questions similar to these are answered using optimal experiment design [12], where experimental parameters are optimized to maximize some metric of information, *e.g.* Fisher information. However, when either the encoding or decoding is non-linear, *e.g.* phase imaging or sparse priors, such methods become suboptimal and difficult to apply. Fortunately, machine learning offers an opportunity to model and optimize non-linear systems. Using the emerging tools from machine learning, the encoding and decoding process can be modeled as a neural network and optimized (referred to as *physics-free* or *model-free*) to map measurements to the desired reconstructed information. Using these approaches, the best measurement scheme (encoding) can also be optimized by including a system's experimental parameters as learnable variables [13-16]. However, when dealing with images or higher dimensional data, as is common in *computational imaging*, such *physics-free* methods typically use neural networks with millions of learnable parameters and thus require a large corpus of training data to properly optimize. In many fields, gathering large amounts of data for a specific application is expensive to capture or completely unavailable, making such blackbox methods impractical. In addition, such methods do not typically generalize well in experiment due to dataset distribution shift [17], varying experimental noise levels, and system imperfections.

In contrast, more traditional *physics-based* inverse problems are able to faithfully reconstruct a wide variety of samples/scenes without the need for ground truth data or training. There are two significant mechanisms behind this class of decoders: first, the inclusion of the *physics-based* forward model that informs the algorithm how measurements are made and second, the architecture of the decoder, *i.e.* optimization algorithm.

While there are no learnable parameters, such methods often include hyperparameters that are laboriously handtuned. The solution is to combine the two classes of decoders, termed *physics-based learning* [18-25]. The working principle of *physics-based learning* is to form a network from the operations of a *physics-based* image reconstruction. Specifically, the iterations of the *physics-based* reconstruction are *unrolled* to form the layers of a *physics-based* network. By combining these two methodologies, the burdens of *physics-based* and *physics-free* approaches can be reduced. The inclusion of known quantities, *e.g.* the forward model process and optimization architecture, reduces the magnitude of data required for learning and the data-driven optimization of certain variables alleviates the need for tuning parameters by hand.

## Contribution

This dissertation describes a novel design paradigm that harnesses data to learn a computational imaging system's encoding scheme to optimize the performance of the whole system. This work extends physics-based learning by including learnable experimental design parameters in the networks created from *unrolling* the iterations of traditional physics-based reconstruction. With the goal of improving the system's temporal resolution, processing time, and reconstructed image quality, I apply this new data-driven design methodology to optimize how measurements are acquired on a *computational microscope* to reconstruct QPI and to perform super-resolution microscopy (Fourier Ptychography). My approaches are validated in several experimental settings and compared against other traditional and heuristic design methodologies.

Further, I delve into the practical issues of implementing *physics-based learning* for large-scale real-world computational imaging systems. As our systems grow in scale, so does the memory required for training. In many real-world applications, *e.g.* higher dimensional magnetic resonance imaging and volumetric microscopy, the memory required inhibits computing gradients, via backpropagation, on commercially-available graphics processing units (GPU). In this dissertation, I present a novel memory-efficient technique to enable *physics-based learning* for large-scale computational imaging systems. I demonstrate our method's usefulness on several real-world applications: in *computational microscopy* to learn measurement scheme for super-resolution quantitative phase imaging and in *medical imaging* to learn image priors for 3D multi-channel magnetic resonance imaging.

Altogether, the methods in the thesis capture the themes of *Physics-based Learned Design* and *Memory-efficient Learned Design*.

## Outline

The remainder of this dissertation is organized as follows:

## **Chapter 2: Computational Microscopy**

This chapter provides an overview of the principles of optical microscopy and computational imaging in the context of quantitative phase imaging. I introduce quantitative phase imaging, using coded-illumination measurements and the LED array microscope, and how phase information can be decoded from these measurements using general image reconstruction techniques. In addition, I provide some basic intuition behind how different illumination patterns can encode a diverse set of contrasts and when to use which image reconstruction algorithm.

## **Chapter 3: Physics-based Learned Design**

This chapter introduces the techniques of physics-based learning to optimize the experimental design of a computational imaging system. I demonstrate these techniques by learning coded-illumination patterns to more efficiently perform quantitative phase imaging. I compare the designs physics-based learning produces to others produced using traditional optimal experimental design and heuristic design methods. Finally, I validate the methods experimentally on the LED array microscope using patterns trained in simulation.

## **Chapter 4: Physics-based Learned Design for Fourier Ptychographic Microscopy**

This chapter examines physics-based learned design for super-resolution microscopy (Fourier Ptychographic microscopy). Fourier Ptychography is typically slow due to the number of measurements required to perform phase imaging at a resolution above the microscope's diffraction limit. In this chapter, I utilize physics-based learning to improve the temporal resolution of the system by optimizing coded-illumination patterns for a reduced set of measurements. This is demonstrated in simulation and in experiment. Finally, the learned patterns are discussed to build new intuition for coded-illumination measurements.

## **Chapter 5: Memory-efficient Learning for Large-scale Computational Imaging**

For real-world large-scale inverse problems, computing gradients via backpropagation is infeasible due to the memory limitations of commercially available graphics processing units. In this chapter, I present a memory-efficient learning procedure that exploits the reversibility of the network's layers to enable physics-based learning for large-scale computational imaging systems. Here, I demonstrate our method on a small-scale compressed sensing example, as well as several large-scale real-world systems: 3D multi-channel magnetic resonance imaging and super-resolution optical microscopy.

## **Chapter 6: Conclusion**

This chapter discusses the themes presented throughout this dissertation and outlines several future directions.

## **Appendix A: Open Source: How to implement physics-based learning**

This appendix serves as a tutorial of how to implement physics-based learning. It starts at an introductory level for people who are familiar with python, computational imaging, or machine learning to build and optimize proof-of-concept computational imaging systems. It moves on to describe a more advanced framework that enables memory-efficient learning for people who are planning to use physics-based learning to design large-scale systems.

# Chapter 2

## Computational Microscopy

Technology behind computational microscopy combines numerous areas of expertises, from optics, to physics, and computation. In this chapter, I outline the basic concepts in microscopy, physics for phase imaging, and computational reconstruction required to understand the underpinnings of the work in this dissertation.

### 2.1 Optical Microscopy

#### Imaging and Resolution

Imaging is the technique of manipulating light to reform a scene at a desired plane. In the visible spectrum, this is often done using glass. Thanks to its refractive properties, glass lenses can be used to bend the rays of light to refocus them at another plane, thereby forming an image. Photography, microscopy, and astronomy all rely upon these principles to measure spatial information about the world by imaging scenes onto sensors.

In optical microscopy, these principles are used to relay an image of a sample to a camera sensor. The specific design of such a system is usually abstract to the end user, but can typically be summarized by two parameters, magnification and numerical aperture (NA) of the microscope's objective. Magnification determines how large the sample will appear at the image plane - thereby determining the field-of-view (FOV) of the microscope. NA describes the steepest angle of light that is allowed into the system and determines the system's diffraction-limited resolution. Specifically,  $NA_{\text{obj}} = n \cdot \sin(\theta)$ , where  $n$  is refractive index of the medium between the sample and the microscope's objective and  $\theta$  is the half angle of the aperture. And while a wide variety of objectives span the possible magnification-NA space, typically, lower magnification objectives have a lower NA and higher magnification objectives have a higher NA. This is due to practical issues related to manufacturing and optical aberrations. The implications of this is a trade off between the FOV and the resolution of the microscope. This relation is further affected by the fixed space-bandwidth product of sensor.



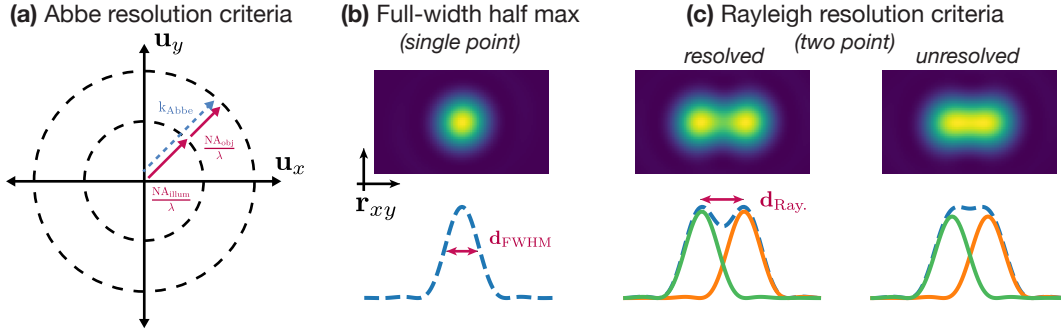


Figure 2.1: Microscopy definitions of resolution: (a) Abbe resolution criteria,  $k_{\text{Abbe}}$ , characterized by the transfer function's support in Fourier space. (b) Full-width half max or single point resolution,  $d_{\text{FWHM}}$ , defined by the point spread function. (c) Rayleigh resolution criteria or two point resolution,  $d_{\text{Ray}}$ , characterized by minimum resolvable distance between two points. On the left are two resolvable points as highlighted by the dip between the resultant image and on the right are two unresolvable points. Below (b) and (c) are cross sections through the system's response to a single point, two resolvable points, and two unresolvable points.

There are several main ways to define resolution in microscopy (*e.g.* Abbe criteria, full-width half max, Rayleigh criteria) each with their own interpretation. The Abbe criteria (Fig. 2.1a) defines an imaging system's maximum resolvable angular frequency,

$$k_{\text{Abbe}} = \frac{NA_{\text{obj}} + NA_{\text{illum}}}{\lambda}, \quad (2.1)$$

where  $NA_{\text{illum}}$  is the numerical aperture of the illumination's condenser objective and  $\lambda$  is the wavelength of light. When using spatially coherent illumination, the  $NA_{\text{illum}} = 0$  and the diffraction-limited resolution is  $k_{\text{Abbe}} = (NA_{\text{obj}}/\lambda)$ . When using spatially incoherent illumination, the best achievable resolution is when  $NA_{\text{obj}} = NA_{\text{illum}}$ , thus the incoherent diffraction-limited resolution is  $d = (2 \cdot NA/\lambda)$ .

The full-width half max or single point resolution,  $d_{\text{FWHM}}$ , characterize the system's resolution by the spatial distance between the two points that intersect half of the point spread function's maximum value (Fig. 2.1b). And finally, the Rayleigh criteria (Fig. 2.1c) is the minimum resolvable distance between two points and is defined,

$$d_{\text{Ray}} = \frac{1.22 \cdot \lambda}{NA_{\text{obj}} + NA_{\text{illum}}}, \quad (2.2)$$

where 1.22 is the distance of the first dark circle of microscope's point spread function (Airy disk). More specifically, it is the first zero of the order-one Bessel function of the first kind divided by  $\pi$ .

While all three quantities are related, the Rayleigh criteria and full-width half max are defined by the shape and contrast of the imaging system's point spread function (or transfer function in Fourier space) and the Abbe criteria is defined merely by the support of the imaging system's transfer function in Fourier space. Throughout this dissertation I will discuss achievable resolution in terms of the Abbe resolution criteria and the maximum resolvable angular frequency because of its connection to the Nyquist sampling criteria.

Once an image is formed at the desired plane, it is captured and sampled on a camera using a 2D grid of discrete pixels. The pitch of pixels (*i.e.* size) to critically sample the image is governed by the Nyquist sampling criteria,  $p_{\text{Nyq}} = (1/2k_{\text{Abbe}})$ . That is, the image must be sampled at a minimum of twice the rate of the image's maximum angular frequency.

## 2.2 Quantitative Phase Imaging

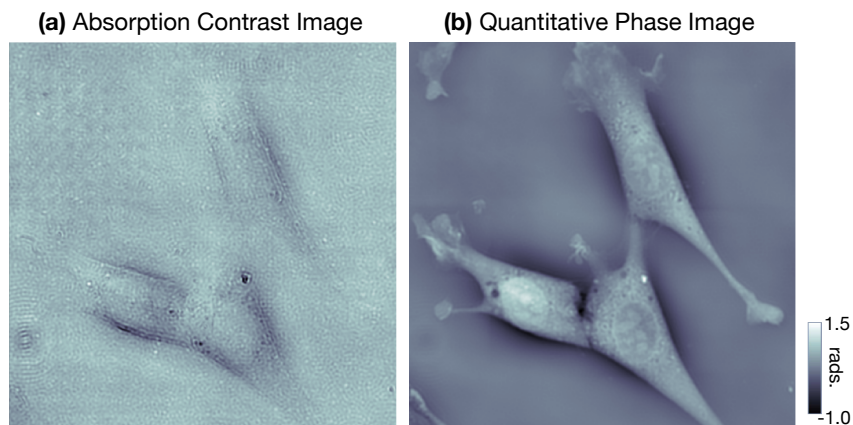


Figure 2.2: Bright-field sample contrasts: (a) absorption image and (b) quantitative phase image (computed using four measurements and quantitative differential phase contrast [9]) of 3T3 cells plated in a single layer.

Bright-field microscopy is a standard method for imaging biological samples *in vitro*. Unless stained or labeled, most biological samples are optically transparent (*i.e.* minimal absorption contrast) making them difficult to image directly (Fig. 2.2a). Quantitative Phase Imaging (QPI) allows for stain-free and label-free imaging with endogenous contrast proportional to the refractive index and thickness of the sample [3, 26] (Fig. 2.2b). Specifically, a thin sample can be characterized by its 2D complex transmittance function,

$$\mathbf{o}(\mathbf{r}) = e^{j\phi(\mathbf{r}) - \mu(\mathbf{r})}, \quad (2.3)$$

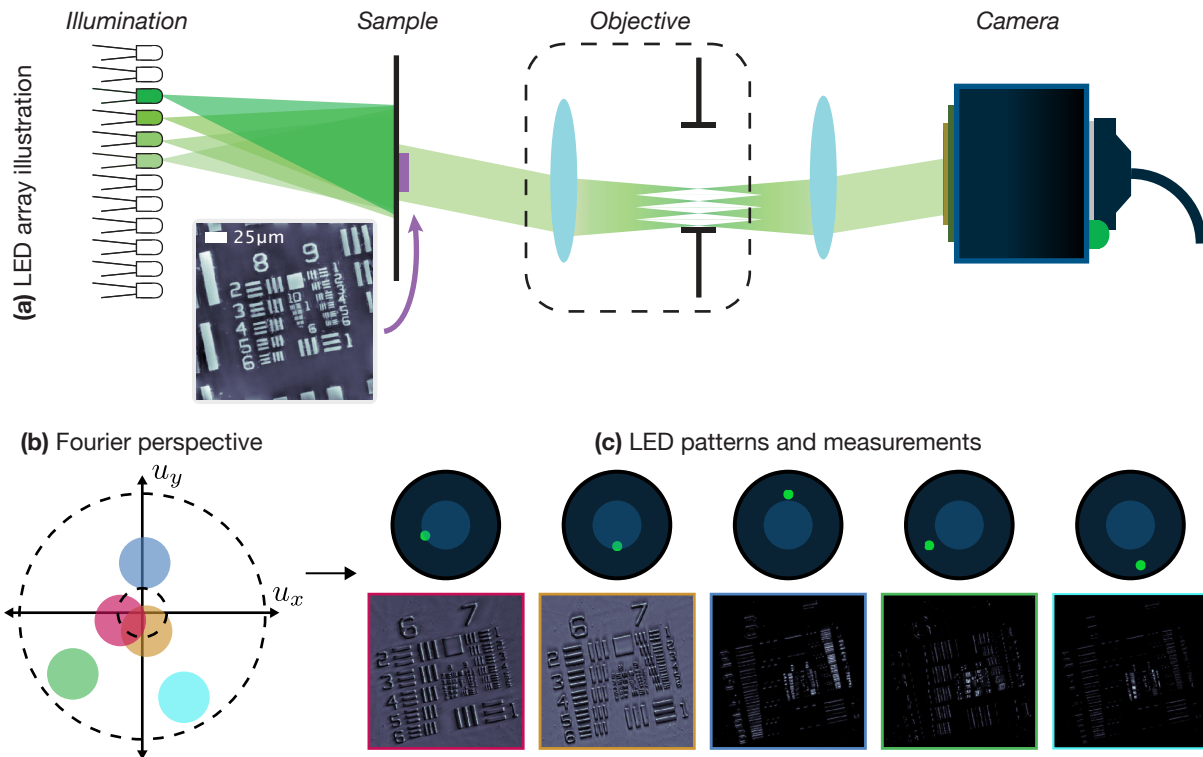


Figure 2.3: Light-emitting diode microscope: (a) Schematic of experimental setup imaging the a USAF phase resolution target. (b) Each light-emitting diode (LED) modulates a different part of the sample's Fourier space into the passband of the microscope. The colored circles correspond to different LED's Fourier space support, each of which has a small bandwidth (low resolution). The outer dashed circle represents the upper limit of the Fourier support for the LED array microscope (determined by the position of the outer most LEDs). (c) Intensity images are captured in real space by the camera sensor under different LED illuminations (displayed). Each colored circle in (b) corresponds to a color-outlined measurement in (c).

where  $\phi(\mathbf{r}) = \frac{2\pi}{\lambda}n(\mathbf{r})t(\mathbf{r})$  is the phase contrast,  $n(\mathbf{r})$  is the refractive index distribution,  $t(\mathbf{r})$  is the thickness,  $\mu(\mathbf{r})$  is the absorption contrast of the sample,  $\lambda$  is the wavelength of illumination,  $j = \sqrt{-1}$ , and  $\mathbf{r} \in \mathbb{R}^2$  is the spatial coordinate vector.

Phase contrast information cannot be directly observed in intensity measurements acquired on a camera. However, it is possible to indirectly map phase information into intensity. Common methods include: interference [27, 28], coded-illumination [1, 11], and coded-pupil (usually defocus) [29–31].

## 2.3 Computational LED Array Microscope

*Coded-illumination* microscopy can be efficiently and inexpensively implemented on a commercial microscope by replacing the standard illumination unit with an LED array (Fig. 2.3). The programmability of the illumination device allows it to reveal a variety of contrasts (*e.g.* bright field, dark field, differential phase contrast) (visualized in Fig. 2.4). By understanding how the different illumination patterns encode phase information about a sample, the model can be inverted to retrieve that information.

Each individual LED's illumination at the sample plane is modeled as a monochromatic plane wave (with wavelength  $\lambda$ ) propagating at an angle characterized by the LED's physical position with respect to the azimuth of the microscope [1]. When the  $l^{\text{th}}$  LED is "on", the complex-field after the sample can thus be written as  $x(\mathbf{r}) \exp(2\pi j \langle \boldsymbol{\xi}_l, \mathbf{r} \rangle)$ , where  $\boldsymbol{\xi}_l \in \mathbb{R}^2$  is the spatial frequency vector corresponding to the illumination's angle of propagation for the  $l^{\text{th}}$  LED. The illumination of the sample can be interpreted as a translation of the sample's spatial spectrum,

$$\mathcal{F}\{x(\mathbf{r})\}(\mathbf{u} - \boldsymbol{\xi}_l) = \mathcal{F}\{x(\mathbf{r}) \exp(2\pi j \langle \boldsymbol{\xi}_l, \mathbf{r} \rangle)\}, \quad (2.4)$$

where  $\mathbf{u}$  is the 2D spatial frequency vector.

The product of the illumination field and the sample's 2D transmittance function are propagated to the pupil plane of the microscope. This operation is modeled as a Fourier transform [32]. At the pupil plane, the sample's shifted Fourier space is low-pass filtered by the finite-aperture of the objective lens. Finally, an image is formed by propagating the field through the (2nd) (tube) lens of the microscope to the camera plane (performing another Fourier transform). At the camera, an intensity-only (*i.e.* phaseless) real-space image is captured of the complex field. Mathematically, the process relating the sample's transmission function to the measured low-resolution image  $y_l(\mathbf{r})$  is expressed as

$$y_l(\mathbf{r}) = \left| \mathcal{F}^{-1}\{P(\mathbf{u}) \mathcal{F}\{x(\mathbf{r})\}(\mathbf{u} - \boldsymbol{\xi}_l)\} \right|^2. \quad (2.5)$$

Here,  $\mathcal{F}$  and  $\mathcal{F}^{-1}$  denote Fourier transform and its inverse, respectively,  $P$  is the *pupil function* of the microscope objective, which has a hard frequency cutoff at the diffraction limit set by its numerical aperture ( $\|\mathbf{u}\|_2 < \frac{\text{NA}_{\text{obj}}}{\lambda}$ ) [32].

Often it is desirable to encode multiple LED's information into a single measurement, termed *multiplexing*, to increase signal-to-noise ratio (SNR) and to increase phase contrast (Fig. 2.4). When multiple LEDs are turned on simultaneously, the measured intensity is the weighted sum of the individual LED measurements, since LEDs are mutually incoherent with each other. The resultant measurement is,

$$y_m(\mathbf{r}) = \sum_{l=1}^L c_l y_l(\mathbf{r}), \quad (2.6)$$

where the non-negative scalar  $c_l$  represents the relative brightness of the  $l^{\text{th}}$  LED.

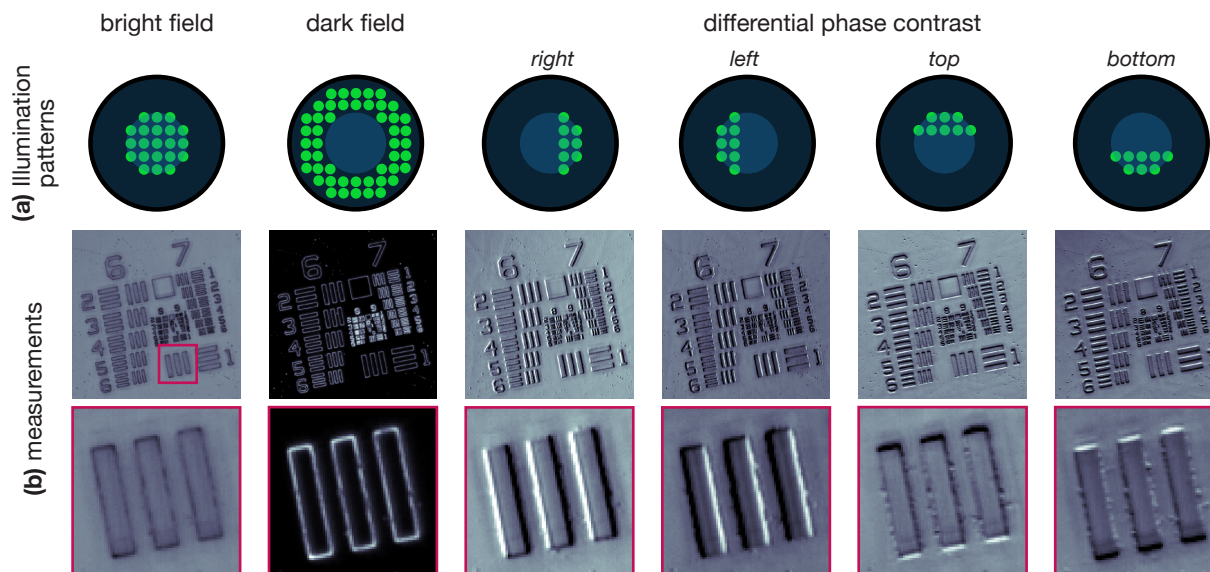


Figure 2.4: Multiplexed LED illumination patterns: (a) the LED illumination patterns for bright field, dark field, differential phase contrast and (b) their resulting intensity measurements of a phase USAF resolution target. Insets highlight the variety of contrasts each illumination pattern produces.

Depending on an LED's illumination angle, its resulting intensity image will either be a bright-field or dark-field image [33]. Bright-field images (see Fig. 2.3c) are produced by central LEDs (inner circle of LED patterns in Fig. 2.3c) whose spatial-frequency vectors are within the acceptance angle of the objective (where  $\|\xi\|_2 < \frac{NA_{\text{obj}}}{\lambda}$ ). These measurements encode a sample's lower spatial-frequency information, are generally very bright, with high signal-to-noise ratio (SNR), and have a strong background component. Dark-field images are produced by LEDs in the outer shell of patterns in Fig. 2.3c (where  $\|\mathbf{u}\|_2 > \frac{NA_{\text{obj}}}{\lambda}$ ). These measurements encode a sample's higher spatial-frequency information, are generally very dim, with low SNR, and have no background component. Because bright-field and dark-field images have orders-of-magnitude different brightness, they will be affected by Poisson noise differently [34]. Generally, the design of measurements consider bright-field and dark-field LEDs separately because of the high dynamic range between bright-field and dark-field measurements.

## 2.4 Generalized Image Reconstruction

Decoding, the latter half of a computational imaging system, is how one retrieves the encoded information from a set of measurements. Typically performed via an image reconstruction, the process requires a forward model representation of the measurement

process and information about the noise under which the measurements are taken. Decoding is then performed by inverting the model for the forward process for the given set of measurements.

The forward process for a typical computational imaging system describes how information about the image to be reconstructed,  $\mathbf{x}$ , is encoded into the measurements,  $\mathbf{y}$ . Specifically,

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{n}, \quad (2.7)$$

where  $\mathcal{A}$  is the forward model that characterizes the measurement system physics and  $\mathbf{n}$  is noise. The forward model is a continuous process, but is often represented by a discrete approximation. Similar to sampling measurements, bold face variables represent a discretized and vectorized version of their corresponding continuous signal (except for the noise vector  $\mathbf{n}$ , which is formed for each sensor pixel in a 2D grid).

The inverse problem (*i.e.* decoding) is commonly formulated as an optimization problem,

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \mathcal{D}(\mathbf{x}; \mathbf{y}) + \mathcal{P}(\mathbf{x}), \quad (2.8)$$

where  $\mathcal{D}(\cdot)$  is a data consistency penalty and  $\mathcal{P}(\cdot)$  is a signal prior penalty. The data consistency penalty is commonly chosen to be the negative-log likelihood of the noise's distribution (*e.g.*  $\|\mathcal{A}(\mathbf{x}) - \mathbf{y}\|_2$  for normally distributed noise). The signal prior penalty, also called regularization, models constraints on the signal  $\mathbf{x}$ , for example, non-negativity or image support, or other subspace constraints. Practically, it is used to reject noise, mitigate the amplification of noise, and stabilize the image reconstruction.

## Physics-based Iterative Reconstruction

Depending on the linearity and structure of the forward model,  $\mathcal{A}$ , and the prior,  $\mathcal{P}$ , there are several options to minimize Eq. 2.8. If  $\mathcal{A}$  is linear then optimizers such as half quadratic splitting (HQS) [35] and conjugate gradient (CG) can be very efficient. If  $\mathcal{A}$  is non-linear then methods such as proximal gradient descent (PGD) [36], or its accelerated variants, can be used. In both cases prior penalty,  $\mathcal{P}$  is typically minimized by iteratively applying proximal updates alternated between data consistency updates.

The PGD algorithm (Alg. A.1) is composed of alternating gradient and proximal update steps. In Alg. A.1,  $\alpha$  is the gradient step size,  $\nabla_{\mathbf{x}}$  is the gradient operator,  $\text{prox}_{\mathcal{P}}$  is a proximal function that enforces the prior [37], and  $\mathbf{x}^{(n)}$  and  $\mathbf{z}^{(n)}$  are intermediate variables for the  $n^{\text{th}}$  iteration.

While similar to PGD in alternating between data consistency and prior updates, HQS (Alg. 2.2) instead performs a full model inversion rather than a single gradient step for the data consistency update.

---

**Algorithm 2.1** Proximal Gradient Descent

---

**Inputs**  $\mathbf{x}^{(0)}$ -initialization,  $\alpha$ -step size,  $N$ -maximum number of iterations**Output**  $\mathbf{x}^{(N)}$ -final estimate of image

- 1:  $n \leftarrow 1$
  - 2: **for**  $n < N$  **do**
  - 3:      $\mathbf{z}^{(n)} \leftarrow \mathbf{x}^{(n-1)} - \alpha \nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}^{(n-1)}; \mathbf{y})$  ▷ Gradient update
  - 4:      $\mathbf{x}^{(n)} \leftarrow \arg \min_{\mathbf{x}} \mathcal{P}(\mathbf{x}) + \mu \|\mathbf{x} - \mathbf{z}^{(n)}\|_2^2$  ▷ Proximal update
  - 5:      $n \leftarrow n + 1$
  - 6: **end for**
- 

---

**Algorithm 2.2** Half Quadratic Splitting

---

**Inputs**  $\mathbf{x}^{(0)}$ -initialization,  $\mu$ -penalty parameter,  $N$ -maximum number of iterations**Output**  $\mathbf{x}^{(N)}$ -final estimate of image

- 1:  $n \leftarrow 1$
  - 2: **for**  $n < N$  **do**
  - 3:      $\mathbf{z}^{(n)} \leftarrow \arg \min_{\mathbf{z}} \mathcal{D}(\mathbf{z}; \mathbf{y}) + \mu \|\mathbf{z} - \mathbf{x}^{(n-1)}\|_2^2$  ▷ Regularized least squares update
  - 4:      $\mathbf{x}^{(n)} \leftarrow \arg \min_{\mathbf{x}} \mathcal{P}(\mathbf{x}) + \mu \|\mathbf{x} - \mathbf{z}^{(n)}\|_2^2$  ▷ Proximal update
  - 5:      $n \leftarrow n + 1$
  - 6: **end for**
- 

In Alg. 2.2,  $\mu$  is a penalty parameter that weights the data consistency and signal prior penalties. When the noise has a normal distribution and  $\mathcal{A}$  is linear,  $\mathcal{D}(\mathbf{x}; \mathbf{y}) = \|\mathcal{A}(\mathbf{x}) - \mathbf{y}\|_2$  and Alg. 2.2 line 3 can be efficiently solved via the CG method.

## Physics-free Learned Reconstruction

With the popularization of data-driven algorithms, many methods have been developed to learn the decoding process that maps the measurements back to encoded information. Unlike the physics-based methods outlined above, these methods rely on datasets of measurements and ground truth rather than the physics-based model and inverse problem formulation. The UNet architecture [38] is most commonly used for these methods [39] for its multi-level representative power.

# Chapter 3

## Physics-based Learned Design

### 3.1 Introduction

Quantitative Phase Imaging (QPI) enables stain-free and label-free microscopy of transparent biological samples *in vitro* [3, 26]. Compared with coherent methods [40, 41], QPI techniques that use partially coherent light achieve higher spatial resolution, more light throughput, and reduced speckle artifacts. Phase contrast can be generated by interference [27, 28] or defocus [29–31]. More recently, *coded-illumination microscopy* [2, 4, 11, 42–44] has been demonstrated as an accurate and inexpensive QPI scheme. To realize coded-illumination, I replace a commercial microscope’s illumination unit with a light-emitting diode (LED) domed array [8]. This flexible hardware platform has been used for various QPI applications including super-resolution [2, 11, 42], multi-contrast [33, 43], and 3D imaging [4, 44].

Coded-illumination microscopy uses intensity measurements with asymmetric source patterns [45] to retrieve 2D phase information. Quantitative Differential Phase Contrast [46–49] (qDPC) typically captures four measurements with rotated half-circle source patterns, from which the phase is computationally recovered using a partially coherent model. The performance of qDPC is predominantly determined by how the phase information is encoded in (via coded-illumination) and decoded from (via phase recovery) the intensity measurements.

The half-circle illumination designs of qDPC were derived analytically based on a Weak Object Approximation [47, 48, 50, 51] which linearizes the physics in order to make the inverse problem mathematically convenient. This linearized model enables one to derive a phase transfer function and analyze the spatial frequency coverage of any given source pattern [48, 49, 52, 53]. However, the non-linearities (camera intensity and phase information) of the actual system makes it impossible to predict an optimal source design without knowing the sample’s phase *a priori*. In addition, these types of analysis are inherently restricted to linear reconstruction algorithms and will not necessarily result in improved accuracy when the phase is retrieved via non-linear iterative methods.



Motivated by the success of deep learning [54] for image reconstruction problems [39, 55–59], data-driven approaches have been adopted for learning coded-illumination patterns. For instance, researchers have used machine learning to maximize the phase contrast of each coded-illumination measurement [60], to improve accuracy on classification tasks [14], and to reconstruct phase [61]. All of these techniques learn the input-output relationship with a deep convolutional neural network (CNN) using training data. It is not straightforward to include the well-characterized system physics; hence, the CNN is required to learn both the physical measurement formation and the phase reconstruction process. This task requires training of 10s to 100s of thousands of parameters and an immense number of training examples.

## Contribution

Here, I introduce a new data-driven approach to optimizing the source pattern design for coded-illumination phase retrieval by directly including both the system physics and the non-linear nature of the decoder in the learning process. The approach *unrolls* the iterations of a generic non-linear decoder to construct an *unrolled network* [18–20, 22, 62–64]. Similar to CNNs, *unrolled networks* consist of several layers (one for each iteration); however, in this case each layer consists of well-specified operations to incorporate measurement formation and sparse regularization, instead of standard operations such as generic convolutions. The key benefits of the approach are:

- incorporation of the system physics and reconstruction non-linearities in the illumination design process.
- efficient parameterization of the unrolled network.
- incorporation of practical constraints.
- reduced number of training examples required.

I deploy the data-driven approach to learn improved coded-illumination patterns for phase reconstruction. Each layer of the unrolled network is parameterized by only a few variables (LED brightness values), enabling an efficient use of training data (< 100 simulated training examples). I compare the QPI performance of the learned designs to previous work and demonstrate that the designs generalize well to the experimental setting with biological samples.

## 3.2 Quantitative Differential Phase Contrast

qDPC recovers a sample’s complex transmittance function from several coded-illumination measurements. The phase recovery optimization algorithm aims to minimize the Euclidean norm of the error between the measurements and the expected measurements

formed with the current phase estimate. Using a gradient-based procedure, the phase estimate is iteratively updated until convergence. For a partially coherent source, the phase can be recovered with resolution up to twice the coherent diffraction limit. In this section, I describe the measurement formation process and phase recovery optimization.

## System Modeling

As previously reviewed in Chap. 2, a thin sample's transmission function can be approximated as a 2D complex function,  $o(\mathbf{r}) = e^{j\phi(\mathbf{r}) - \mu(\mathbf{r})}$ , where  $\phi(\mathbf{r})$  and  $\mu(\mathbf{r})$  are the sample's phase and absorption, respectively. Intensity measurements,  $y(\mathbf{r})$ , of the sample are a non-linear function of  $o(\mathbf{r})$ , mathematically described by,

$$y(\mathbf{r}) = |p(\mathbf{r}) * (s(\mathbf{r}) \odot o(\mathbf{r}))|^2, \quad (3.1)$$

where  $*$  denotes 2D spatial convolution,  $\odot$  denotes elementwise multiplication,  $s(\mathbf{r})$  is the illumination's complex-field at the sample plane and  $p(\mathbf{r})$  is the point spread function (PSF) of the microscope. The illumination from each LED is approximated as a tilted plane wave,  $s(\mathbf{r}) = \exp(2\pi j \langle \boldsymbol{\xi}_l, \mathbf{r} \rangle)$ , with tilt angle,  $\boldsymbol{\xi}_l$ , determined by the physical position of the  $l^{\text{th}}$  LED relative the azimuth of the microscope [1].

Because the measured image in Eq. 3.1 is non-linear with respect to the sample's transmission function, recovering phase generally requires non-convex optimization. However, biological samples in closely index-matched fluid have a small *scatter-scatter* term. This means that a *weak object approximation* can be made; linearizing the measurement formation model such that phase recovery requires only a linear deconvolution of the measurements with their respective weak object transfer functions (WOTFs) [47-51]. Further, unstained biological samples are predominantly phase objects since they are only weakly absorbing (*i.e.*  $\mu(\mathbf{r})$  is small), however, throughout this section I will outline measurement formation a sample with both phase and absorption contrast. With these approximations, each intensity measurement can be modeled as a linear system with contributions from the background, phase contrast, and absorption contrast. Mathematically,

$$\mathbf{y}(\mathbf{r}) \approx B + h_\phi(\mathbf{r}) * \phi(\mathbf{r}) + h_\mu(\mathbf{r}) * \mu(\mathbf{r}), \quad (3.2)$$

where  $B$  is the measurement's background and  $h_\phi(\mathbf{r})$  and  $h_\mu(\mathbf{r})$  are the phase and absorption WOTF respectively. The WOTFs are a function of the source and the pupil distributions of the microscope [48]. For a single LED the WOTF for phase and absorption are,

$$h_\phi(\mathbf{u}) = j(p(\mathbf{u}) * (p(\mathbf{u}) \cdot s(\mathbf{u})) - (p(\mathbf{u}) \cdot s(\mathbf{u})) * p(\mathbf{u})) \quad (3.3)$$

$$h_\mu(\mathbf{u}) = p(\mathbf{u}) * (p(\mathbf{u}) \cdot s(\mathbf{u})) + (p(\mathbf{u}) \cdot s(\mathbf{u})) * p(\mathbf{u}), \quad (3.4)$$

where  $\mathbf{u}$  are 2D spatial frequency coordinates,  $p(\mathbf{u})$  and  $s(\mathbf{u})$  are the Fourier transform of the system's point spread function and illumination respectively,  $\star$  is the correlation operator, defined as  $(x_1 \star x_2)(\mathbf{r}) = \int x_1(\tilde{\mathbf{r}})x_2^*(\tilde{\mathbf{r}} - \mathbf{r})d\tilde{\mathbf{r}}$  for  $\mathbf{r}$  in the domain of  $p(\mathbf{u})$  and  $s(\mathbf{u})$ . An indepth derivation of these transfer functions can be found in App. ??.

Multiple LEDs can be turned on simultaneously to increase signal-to-noise (SNR) and improve phase contrast. As previously discussed, the fields generated by each LED's illumination are spatially incoherent with each other and the measurement from multiple LEDs will simply be the weighted sum of each LED's individual measurement, with weights corresponding to the LEDs' brightness values. It follows that, the phase WOTF for multi-LED illumination will also be the weighted sum of the single-LED phase WOTFs.

$$y_m(\mathbf{r}) = \sum_{l=0}^L c_l y_l(\mathbf{r}); \quad (3.5)$$

$$h_{\phi,m}(\mathbf{u}) = \sum_{l=0}^L c_l h_{\phi,l}(\mathbf{u}); \quad (3.6)$$

$$h_{\mu,m}(\mathbf{u}) = \sum_{l=0}^L c_l h_{\mu,l}(\mathbf{u}), \quad (3.7)$$

where  $c_l \geq 0$  is the  $l^{\text{th}}$  LED's brightness value.

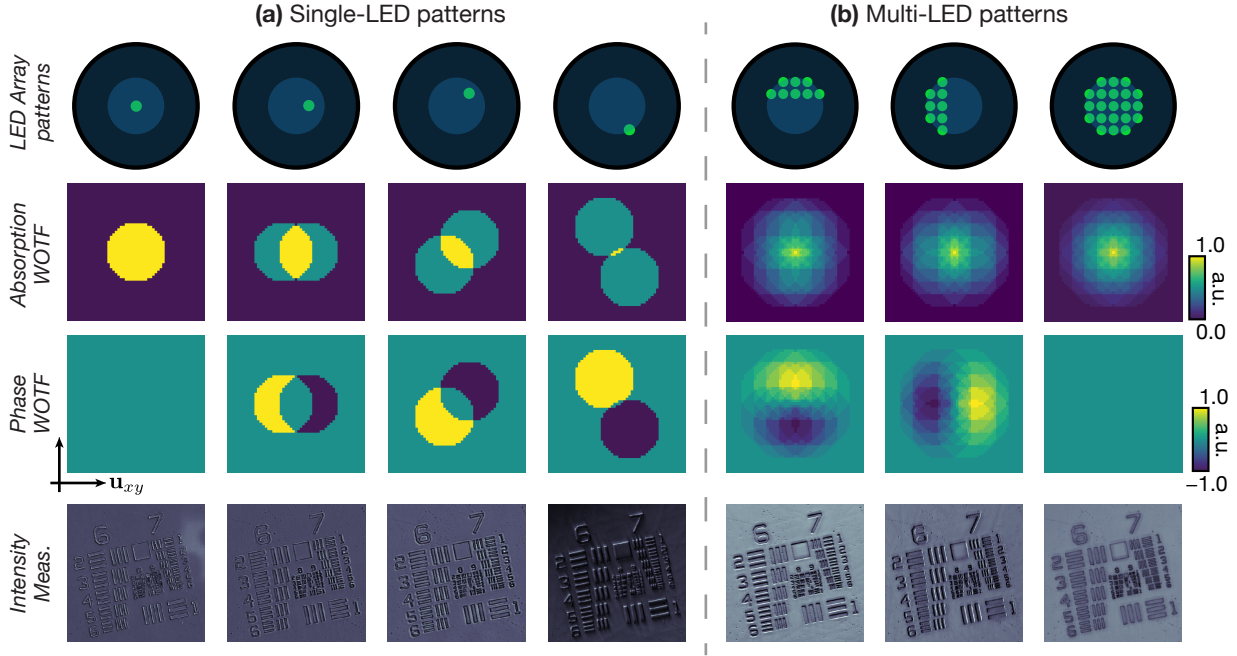


Figure 3.1: LED illumination patterns: Absorption and phase weak object transfer functions (WOTF) and the resulting intensity measurements for (a) single-LED patterns and (b) multi-LED patterns. Asymmetric LED patterns provide a significant amounts of phase contrast, while symmetric pattern provide better absorption contrast. All weak object transfer functions are normalized for visualization purposes.

Following common practice [65], I discretize the 2D spatial distributions and format them as vectors (bold lower case) (e.g.  $\mathbf{h}$  represents the transfer function's 2D spatial distribution,  $\phi$  represents the 2D spatial phase distribution,  $\mathbf{F}$  represents the 2D discrete Fourier transform). The measurements<sup>1</sup> are described as  $\mathbf{y} = \mathbf{F}^{-1} \mathbf{A} \mathbf{F} \phi$  with system function  $\mathbf{A} = \text{diag}(\mathbf{F} \mathbf{h}_\phi)$  (assuming the object is weakly absorbing).

Based on this model, I define  $\mathbf{Y} \in \mathbb{R}^{M \times S}$  as  $S$  single LED measurements,  $\mathbf{y}$ , stacked along the columns. Then,  $\mathbf{C} \in \mathbb{R}^{S \times K}$  is defined as the  $S$  single-LED weights for each of  $K$  measurements, and  $\mathbf{c}_k \in \mathbb{R}^S$  is the  $k^{\text{th}}$  column of  $\mathbf{C}$ . The product  $\mathbf{y}_k = \mathbf{Y} \mathbf{c}_k$  simulates the  $k^{\text{th}}$  multiple-LED measurement. Similarly, I define  $\mathbf{H} \in \mathbb{R}^{N \times S}$  as  $S$  single LED phase WOTFs,  $\mathbf{h}_\phi(\mathbf{u})$ , stacked along the columns, such that the product  $\mathbf{A}_k = \text{diag}(\mathbf{H} \mathbf{c}_k)$  gives the corresponding multiple-LED phase WOTF for the  $k^{\text{th}}$  measurement.

## Phase Recovery

Phase recovery using the forward model described earlier in Sec. 3.2 can be formulated as a regularized linear inverse problem,

<sup>1</sup>In practice,  $\mathbf{y}$  typically refers to the so-called flattened image, where the background energy in (3.2) is removed via background subtraction.

$$\mathbf{x}^* = \mathcal{R}(\{\mathbf{y}_k\}_{k=1}^K, \{\mathbf{A}_k\}_{k=1}^K) \quad (3.8)$$

$$= \arg \min_{\mathbf{x}} \sum_{k=1}^K \|\mathbf{y}_k - \mathbf{F}^{-1} \mathbf{A}_k \mathbf{F} \mathbf{x}\|_2^2 + \mathcal{P}(\mathbf{x}), \quad (3.9)$$

where  $\mathbf{x}^*$  is the recovered phase,  $K$  is the number of measurements acquired,  $y_k$  is the Fourier transform of the  $k^{\text{th}}$  measurement and  $\mathcal{P}(\cdot)$  is a user-chosen regularizer. I solve this optimization problem efficiently using the accelerated proximal gradient descent (APGD) algorithm by iteratively applying an acceleration update, a gradient update and a proximal update [36, 66]. The algorithm is detailed in Alg. 3.1, where  $\alpha$  is the gradient step size,  $N$  is the number of iterations,  $\mathbf{s}$  and  $\mathbf{z}$  are intermediate variables,  $\mu^{(n)}$  is the acceleration parameter derived by the recursion,  $\mu^{(n)} = \frac{1 + \sqrt{1 + 4(\mu^{(n-1)})^2}}{2}$  [66], and  $\text{prox}_{\mathcal{P}}(\cdot)$  is the proximal operator corresponding to the user-chosen regularizer  $\mathcal{P}(\cdot)$  [36].

---

**Algorithm 3.1** Accelerated Proximal Gradient Descent (APGD) for Phase Recovery
 

---

**Inputs**  $\{\mathbf{y}_k\}_{k=1}^K$ -set of  $K$  measurements,  $\{\mathbf{A}_k\}_{k=1}^K$ -set of  $K$  transfer functions,  $N$ -number of iterations,  $\alpha$ -step size,  $\mathcal{P}(\cdot)$ -signal prior penalty

**Output**  $\mathbf{x}^{(N)}$ -final phase estimate after  $N^{\text{th}}$  iteration

- 1:  $\mathbf{x}^{(0)} \leftarrow \mathbf{0}, \mathbf{x}^{(-1)} \leftarrow \mathbf{0}, n \leftarrow 1$
  - 2: **for**  $n < N$  **do**
  - 3:    $\mathbf{s}^{(n)} \leftarrow \mu^{(n)} \mathbf{x}^{(n-1)} + (1 - \mu^{(n)}) \mathbf{x}^{(n-2)}$
  - 4:    $\mathbf{z}^{(n)} \leftarrow \mathbf{s}^{(n)} - \alpha \sum_{k=1}^K (-\mathbf{F}^H \mathbf{A}_k^H \mathbf{F})(\mathbf{y}_k - \mathbf{F}^H \mathbf{A}_k \mathbf{F} \mathbf{s}^{(n)})$
  - 5:    $\mathbf{x}^{(n)} \leftarrow \text{prox}_{\alpha \mathcal{P}}(\mathbf{z}^{(n)})$
  - 6:    $n \leftarrow n + 1$
  - 7: **end for**
- 

### 3.3 Physics-Based Learned Design

Given the phase recovery algorithm in Sec. 3.2, I now describe the main contribution of learning the coded-illumination designs for a given reconstruction algorithm and training set.

#### Unrolled Physics-based Network

Traditionally, DNNs contain many layers of weighted linear mixtures and non-linear activation functions [54] and are learned using large amounts of training data. Here, I consider structured linear functions which capture the system physics of measurement formation and non-linear activation functions which promote sparsity [18, 62]. Termed,

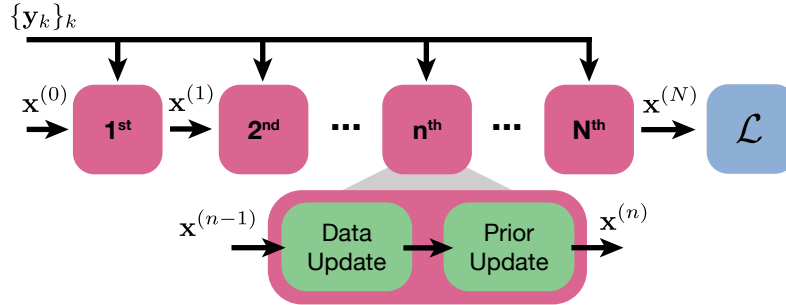


Figure 3.2: Physics-based networks (PbN) are formed by unrolling the iterations of an image reconstruction optimization. Each layer contains one iteration, made up of a data consistency update and signal prior update. The PbN input is the reconstruction's initialization,  $x^{(0)}$ , and the system's measurements,  $\{y_k\}_k$ , and the output is the reconstructed image from the  $N^{\text{th}}$  layer, which is fed into the learning loss,  $\mathcal{L}$ .

physics-based networks (PbN) are constructed from the operations of image reconstruction algorithms (*e.g.* proximal gradient descent or half quadratic splitting), where the iterations of the optimizer form the layers of the network. By including known structures and quantities, such as the forward model, data consistency, and signal prior, PbNs can be efficiently parameterized by only a limited number of learnable variables, thereby enabling an efficient use of training data [21] while still retaining the robustness and interpretability associated with conventional physics-based inverse problems.

Here, I form a PbN using APGD algorithm outlined in Alg. 3.1 used for phase retrieval under the WOA. Each layer of the PbN is formed by the operations of the acceleration, data consistency, and prior updates. During training time, the network's inputs comprise the single-LED measurements,  $\{y_l\}_{l=1}^L$ , and single-LED WOTFs,  $\{A_l\}_{l=1}^L$ , for  $L$  LEDs, and initialization,  $x^{(0)}$ , and the network's output is  $x^{(N)}$  (as outlined in Fig. 3.2). The network's learnable parameters are the brightness levels of the LED patterns for each measurement and are encapsulated within the input measurements and the system WOTFs. Specifically, the learnable parameters can be viewed as weights of linear layer with the single-LED measurements as input and multi-LED measurements as output (Eq. 3.5), similarly for the multi-LED WOTFs (Eq. 3.7 and Eq. 3.6).

## Learning Objective

The learning objective is to minimize the phase reconstruction error of the training data over the space of possible LED configurations, subject to constraints that enforce physical feasibility and reduce degenerate and trivial solutions:

$$\mathbf{C}^* = \arg \min_{\mathbf{C}} \mathcal{L}(\mathbf{C}) \quad (3.10)$$

$$\text{s.t. } \mathbf{c}_k \geq 0 \quad (\text{non-negativity}) \quad (3.11)$$

$$\|\mathbf{c}_k\|_1 = 1 \quad (\text{scale}) \quad (3.12)$$

$$\mathbf{m}_k \odot \mathbf{c}_k = \mathbf{0} \quad (\text{geometric}) \quad (3.13)$$

$$\forall k \in \{1 \dots K\},$$

where,

$$\mathcal{L}(C) = \frac{1}{W} \sum_{w=1}^W \mathcal{L}_w(\mathbf{C}) \quad (3.14)$$

$$= \frac{1}{W} \sum_{w=1}^W \|\mathcal{R}(\{\mathbf{Y}_w \mathbf{c}_k\}_{k=1}^K, \{\mathbf{H}\mathbf{c}_k\}_{k=1}^K) - \mathbf{x}'_w\|_2^2. \quad (3.15)$$

Here,  $(\mathbf{Y}_w, \mathbf{x}'_w)_{w=1}^W$  are  $W$  training pairs for which  $\mathbf{Y}_w$  is a matrix of single-LED measurements for the  $w^{\text{th}}$  sample with ground truth optical phase,  $\mathbf{x}'_w$ .  $\odot$  is the elementwise product operator,  $\mathbf{m}_k$  is a geometric constraint mask for the  $k^{\text{th}}$  measurement, and  $\mathbf{0}$  is the null vector.

The non-negativity constraint (Eq. 4.9) prevents non-physical solutions by enforcing the brightness of each LED to be greater than or equal to zero. This is enforced by projecting the parameters onto the set of non-negative real numbers. The scale constraint (Eq. 3.12) enforces that each coded-illumination design must have weights with sum equal to 1, in order to eliminate arbitrary scalings of the same design. This is enforced by scaling the parameters for each measurement such that their sum is one. The geometric constraint (Eq. 4.8) enforces that the coded-illumination designs do not use conjugate-symmetric LED pairs to illuminate the sample within the same measurement, since these would also result in degenerate solutions (*e.g.* two symmetric LEDs produce opposite phase contrast measurements that would cancel each other out). To prevent this, I force the source patterns for each measurement to reside within only one of the major semi-circle sets (*e.g.* top, bottom, left, right). This constraint is enforced by setting the LED brightnesses outside the allowed semi-circle to zero.

I solve Eq. 3.10 iteratively via accelerated projected gradient descent (Alg. 3.2). At each iteration, the coded-illumination design for each measurement is updated with the analytical gradient, projected onto the constraints (denoted by  $\mathcal{B}(\cdot)$ ) and updated again with a contribution from the previous iteration (weighted by  $\beta^{(t)}$ ).  $\mathcal{B}(\cdot)$  enforces the constraints in the following order: non-negativity, geometric, and scale.

**Algorithm 3.2** Physics-based Learned Design Algorithm

**Inputs**  $\{\mathbf{Y}_w, \mathbf{x}'_w\}_{w=1}^W$ -set of  $W$  training examples,  $\mathbf{C}$ -illumination design initialization,  $\gamma$ -learning rate,  $T$ -number of learning iterations

**Output**  $\mathbf{C}^{(T)}$ -learned illumination design

---

```

1: for  $t < T$  do                                     ▷ Learning loop
2:   for  $w < W$  do                                     ▷ Training data loop
3:      $r_w \leftarrow \mathcal{R}(\{\mathbf{Y}_w \mathbf{c}_k\}_{k=1}^K, \{\mathbf{H} \mathbf{c}_k\}_{k=1}^K) - \mathbf{x}'_w$ 
4:      $\mathbf{G}_w \leftarrow \text{BackPropagation}(r_w)$ 
5:      $w \leftarrow w + 1$ 
6:   end for
7:    $\mathbf{C}^{(t+1)} \leftarrow \mathcal{B}(\mathbf{C}^{(t)} - \frac{\gamma}{W} \sum_{w=1}^W \mathbf{G}_w)$            ▷ Projected gradient step
8:    $\mathbf{C}^{(t+1)} \leftarrow \beta^{(t)} \mathbf{C}^{(t+1)} + (1 - \beta^{(t)}) \mathbf{C}^{(t)}$    ▷ Acceleration step
9:    $t \leftarrow t + 1$ 
10: end for

```

---

## Gradient Update

The gradient of the loss function (Eq. 3.10) with respect to the design parameters has contributions at every layer of the unrolled network through both the measurement terms,  $\mathbf{y}_k$ , and the phase WOTF terms,  $\mathbf{A}_k$ , for each measurement  $k \in \{1 \dots K\}$ . Here, I outline the algorithm for updating the coded-illumination design weights via a two-step procedure: backpropagating the error from layer-to-layer and computing each layer's gradient contribution. For simplicity, I outline the gradient update for only a single training example,  $w$ , as the gradient for all the training examples is the sum of their individual gradients.

Unlike pure gradient descent, where each iteration's estimate only depends on the previous', accelerated methods like Alg. 3.1 linearly combine the previous two iteration's estimates to improve convergence. As a consequence, backpropagating error from layer-to-layer requires contributions from two successive layers. Specifically, I compute the error at all  $N$  layers with the recursive relation,

$$\begin{aligned}
\frac{\partial \mathcal{L}_w}{\partial \mathbf{x}^{(n-2)}} &= \frac{\partial \mathbf{s}^{(n)}}{\partial \mathbf{x}^{(n-2)}} \frac{\partial \mathbf{z}^{(n)}}{\partial \mathbf{s}^{(n)}} \frac{\partial \mathbf{x}^{(n)}}{\partial \mathbf{z}^{(n)}} \frac{\partial \mathcal{L}_w}{\partial \mathbf{x}^{(n)}} \\
&+ \frac{\partial \mathbf{s}^{(n-1)}}{\partial \mathbf{x}^{(n-2)}} \frac{\partial \mathbf{z}^{(n-1)}}{\partial \mathbf{s}^{(n-1)}} \frac{\partial \mathbf{x}^{(n-1)}}{\partial \mathbf{z}^{(n-1)}} \frac{\partial \mathcal{L}_w}{\partial \mathbf{x}^{(n-1)}}, \tag{3.16}
\end{aligned}$$

where each partial gradient constitutes a single step in Alg. 3.1 and are defined,



$$\frac{\partial \mathbf{x}^{(n)}}{\partial \mathbf{z}^{(n)}} = \frac{\partial \text{prox}_{\alpha \mathcal{P}}(\mathbf{z}^{(n)})}{\partial \mathbf{z}^{(n)}}; \quad \frac{\partial \mathbf{z}^{(n)}}{\partial \mathbf{s}^{(n)}} = \left( I - \alpha \sum_{k=1}^K \mathbf{F}^H \mathbf{A}_k^H \mathbf{A}_k \mathbf{F} \right); \quad (3.17)$$

$$\frac{\partial \mathbf{s}^{(n)}}{\partial \mathbf{x}^{(n-2)}} = (1 - \mu^{(n)}); \quad \frac{\partial \mathbf{s}^{(n-1)}}{\partial \mathbf{x}^{(n-2)}} = \mu^{(n-1)}. \quad (3.18)$$

With the backpropagated error at each layer, I compute the gradient of the loss function with respect to  $\mathbf{C}$  as,

$$\nabla_{\mathbf{C}} \mathcal{L}_w(\mathbf{C}) = \sum_{n=0}^N \mathbf{Q}^{(n)}, \quad (3.19)$$

for which,

$$\mathbf{Q}^{(n)} = \alpha \sum_{k=1}^K \left( \frac{\partial \mathbf{F}^H \mathbf{A}_k^H \mathbf{F} \mathbf{y}_k}{\partial \mathbf{C}} - \frac{\partial \mathbf{F}^H \mathbf{A}_k^H \mathbf{A}_k \mathbf{F}}{\partial \mathbf{C}} \mathbf{s}^{(n-1)} \right) \frac{\partial \mathbf{x}^{(n)}}{\partial \mathbf{z}^{(n)}} \frac{\partial \mathcal{L}_w}{\partial \mathbf{x}^{(n)}}. \quad (3.20)$$

Here,  $(\partial \mathbf{x}^{(n)} / \partial \mathbf{z}^{(n)})$  backpropagates the error through the proximal operator and other partials with respect to  $\mathbf{C}$  relate the backpropagated error at each layer to the changes in  $\mathbf{C}$ . In what follows, I establish the update step for a single measurement—as the gradient updates are computed independently for each measurement—by using the product rule for computing derivatives:

$$\frac{\partial \mathbf{F}^H \mathbf{A}_k^H \mathbf{F} \mathbf{y}_k}{\partial \mathbf{c}_k} = \frac{\partial \mathbf{F}^H \text{diag}(\mathbf{H} \mathbf{c}_k)^H \mathbf{F} (\mathbf{Y} \mathbf{c}_k)}{\partial \mathbf{c}_k} \quad (3.21)$$

$$= \mathbf{F}^H \frac{\partial \text{diag}(\mathbf{H} \mathbf{c}_k)^H}{\partial \mathbf{c}_k} \mathbf{F} \mathbf{Y} \mathbf{c}_k + \mathbf{F}^H \text{diag}(\mathbf{H} \mathbf{c}_k)^H \mathbf{F} \frac{\partial \mathbf{Y} \mathbf{c}_k}{\partial \mathbf{c}_k};$$

$$\frac{\partial \mathbf{F}^H \mathbf{A}_k^H \mathbf{A}_k \mathbf{F}}{\partial \mathbf{c}_k} = \mathbf{F}^H \frac{\partial \text{diag}(\mathbf{H} \mathbf{c}_k)^H \text{diag}(\mathbf{H} \mathbf{c}_k)}{\partial \mathbf{c}_k} \mathbf{F} \quad (3.22)$$

$$= 2 \mathbf{F}^H \frac{\partial \text{diag}(\mathbf{H} \mathbf{c}_k)}{\partial \mathbf{c}_k} \text{diag}(\mathbf{H} \mathbf{c}_k) \mathbf{F}$$

In Alg. [3.3](#), I unite these two steps to form a recursive algorithm which efficiently computes the analytic gradient for a single training example. Alternatively, general purpose auto-differentiation included in learning libraries (*e.g.* PyTorch, TensorFlow) can be used to perform the gradient updates.

**Algorithm 3.3** Gradient Update for Single Training Example

**Inputs**  $\mathbf{r}^{(N)}$ -residual,  $\{\mathbf{y}_k\}_{k=1}^K$ -set of  $K$  measurements,  $\{\mathbf{A}_k\}_{k=1}^K$ -set of  $K$  transfer functions  
**Output**  $\nabla_{\mathbf{C}}\mathcal{L}_w(\mathbf{C})$ -gradient of loss function wrt. illumination designs

---

```

1:  $n \leftarrow N$ 
2: for  $n \geq 0$  do
3:    $\mathbf{b}^{(n)} \leftarrow \frac{\partial \mathbf{x}}{\partial \mathbf{z}} \mathbf{r}^{(n)}$ 
4:    $\mathbf{v}^{(n)} \leftarrow (I - \alpha \sum_{k=1}^K \mathbf{F}^H \mathbf{A}_k^H \mathbf{A}_k \mathbf{F}) \mathbf{b}^{(n)}$ 
5:    $\mathbf{r}^{(n-1)} \leftarrow \mu^{(n)} \mathbf{v}^{(n)} + (1 - \mu^{(n+1)}) \mathbf{v}^{(n+1)}$ 
6:    $\mathbf{Q}^{(n)} \leftarrow \alpha \sum_{k=1}^K (\frac{\partial \mathbf{F}^H \mathbf{A}_k^H \mathbf{F} \mathbf{x}_k}{\partial \mathbf{C}} - \frac{\partial \mathbf{F}^H \mathbf{A}_k^H \mathbf{A}_k \mathbf{F}}{\partial \mathbf{C}} s^{(n-1)}) \mathbf{b}^{(n)}$ 
7:    $n \leftarrow n - 1$ 
8: end for
9:  $\nabla_{\mathbf{C}}\mathcal{L}_w(\mathbf{C}) \leftarrow \sum_{n=0}^N \mathbf{Q}^{(n)}$ 

```

---

## 3.4 Results

the proposed method learns the coded-illumination design for a given reconstruction and training set (Fig. 3.3b), yet up to this point I have not detailed specific parameters of the phase reconstruction. In the results, I set the parameters of the reconstruction algorithm (Alg. 3.1) to have a fixed CPU time by fixing the number of iterations at  $N = 40$  and the step size to  $\alpha = 0.2$  (see supplement for parameter analysis). In addition, the regularization term,  $\mathcal{P}(\phi)$ , has been defined generally (e.g.  $\ell_1$  penalty, total variation (TV) penalty [67], BM3D [68]). Here, I choose to enforce TV-based sparsity:

$$\mathcal{P}(\mathbf{x}) = \tau \sum_i \|D_i \mathbf{x}\|_1, \quad (3.23)$$

where  $\tau = 1e^{-3}$  is set to trade off the TV cost with the data consistency cost and  $D_i$  is the first-order difference operator along the  $i^{th}$  image dimension. I efficiently implement the proximal operator of Eq. 3.23 in closed form via parallel proximal method [64, 69, 70] (details in App. B).

## Learning

To train the coded-illumination design parameters using Alg. 3.2, I generate a dataset of 100 examples (90 for training, 10 for testing). Each example contains ground truth phase from a small region ( $95 \times 95$  pixels) of a larger image and 69 simulated single LED measurements (using Eq. 3.1). The LEDs are uniformly spaced within a circle such that each single-LED intensity measurement is a brightfield measurement. The physical system parameters used to generate the phase WOTFs and simulate the training data measurements

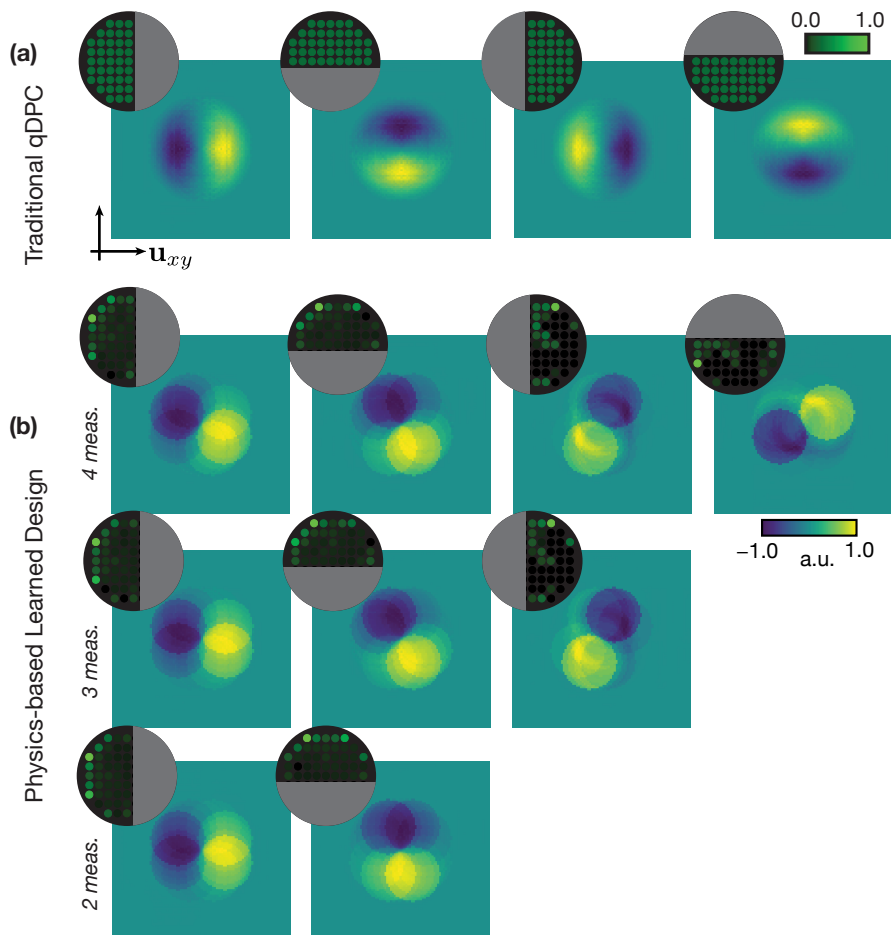


Figure 3.3: Coded-illumination designs and their corresponding phase weak object transfer functions (WOTFs) for: (a) Traditional qDPC and (b) learned designs for the case where 4, 3, or 2 measurements are allowed for each phase reconstruction. The illumination source patterns are in the upper left corners, with gray semi-circles denoting where the LEDs are constrained to be “off”.

are  $\lambda = 0.532\mu m$ , pixel pitch =  $6.5\mu m$ , magnification =  $20\times$ , and  $NA_{obj} = 0.25$ . To train, I use  $\ell_2$  cost between reconstructed phase and ground truth phase as the loss function and approximate the full gradient of Eq. 3.10 with a batch gradient from random batches of 10% of the training pairs at each iteration. I use a learning rate of  $\gamma = 1e^{-2}$  (training and testing convergence curves are provided in the supplement). The training is performed on a multi-core CPU (Dual-socket Intel Xeon® E5 Processor @ 2.1GHz with 64 cores and 504GB of RAM) and batch updates are computed in parallel with each training example on a single core. Each batch update takes  $\sim 6$  seconds. 200 updates are performed, resulting in a total training time of 20 minutes.

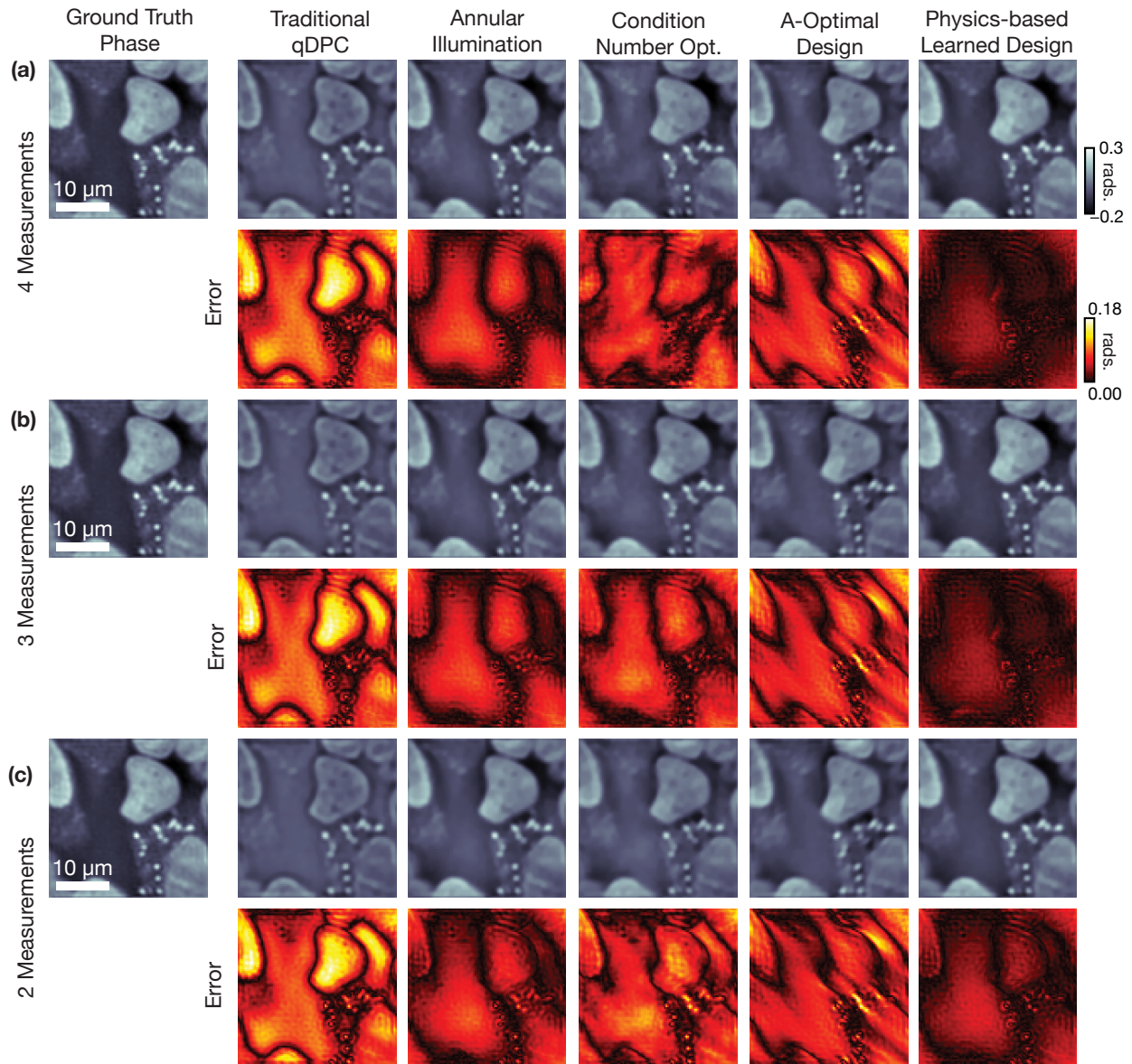


Figure 3.4: Phase reconstruction results using simulated measurements with different coded-illumination designs. I compare results from: traditional qDPC (half-circles), annular illumination, condition number optimization, A-optimal design, and the proposed physics-based learned designs. I show results for the cases of (a) four, (b) three, and (c) two measurements allowed for each phase reconstruction. Absolute error maps are shown below each reconstruction.

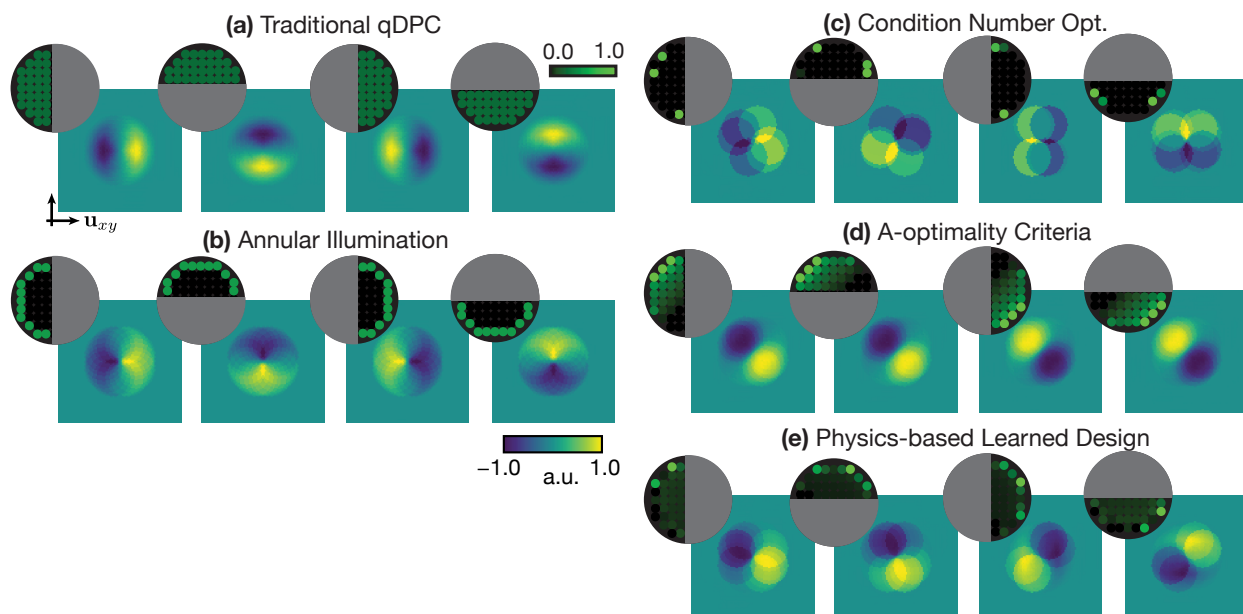


Figure 3.5: Coded-illumination designs and their phase weak object transfer functions for (a) traditional qDPC, (b) annular illumination, (c) condition number optimized, (d) A-optimal, and (e) physics-based learned designs.

Table 3.1: PSNR Results: Average and standard deviation PSNR (dB) of phase reconstructions from the simulated testing examples using different illumination schemes and different numbers of measurements. Factor format: Mean  $\pm$  Std.

# Meas.	Random Illumination	Traditional qDPC	Annular Illumination	Cond. Number Optimization	A-optimal Design	Physics-based Learned Design
4	12.30 $\pm$ 2.12	15.67 $\pm$ 2.19	20.40 $\pm$ 2.09	20.37 $\pm$ 2.41	17.94 $\pm$ 2.54	<b>28.46</b> $\pm$ 2.50
3	12.33 $\pm$ 2.12	15.28 $\pm$ 2.18	20.44 $\pm$ 2.26	19.33 $\pm$ 2.03	18.05 $\pm$ 2.59	<b>28.04</b> $\pm$ 2.59
2	12.25 $\pm$ 2.12	14.87 $\pm$ 2.23	20.21 $\pm$ 2.24	17.19 $\pm$ 2.28	18.08 $\pm$ 2.64	<b>23.73</b> $\pm$ 2.18

## Analysis

Traditional qDPC uses 4 measurements to adequately cover frequency space. the learned designs are more efficient and may require fewer measurements; hence, I show learned designs for the cases of 4, 3 and 2 measurements. The designs and their corresponding phase WOTFs are shown in Fig. 3.3.

Comparing the learned designs with previous work, Fig. 3.4 shows the phase reconstruction for a single simulated test example using 4, 3 and 2 measurements. The ground truth phase is compared with the phase reconstructed using traditional qDPC designs [48], annular illumination designs [48], condition number optimized designs [71], A-optimal designs [72], and the physics-based learned designs. Figure 3.5 visualizes the LED patterns and WOTFs for these heuristic and optimized designs. Table 3.1 reports the

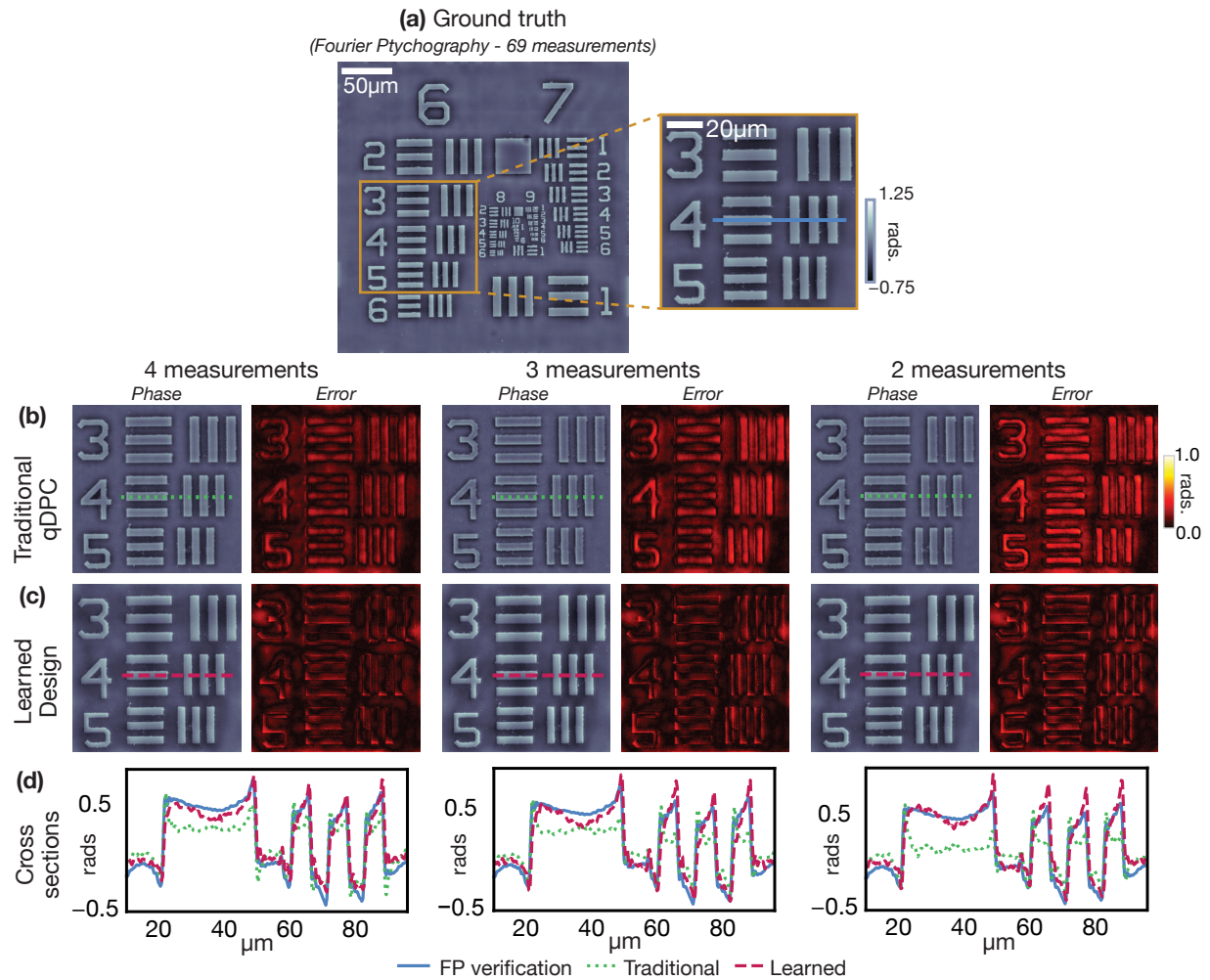


Figure 3.6: USAF phase target reconstructions: Experimental comparison between phase results with (a) Fourier Ptychography (FP) using 69 images, (b) traditional qDPC and (c) learned designs, for the case of 4, 3, and 2 measurements. Error maps show the difference from the FP reconstruction. (d) Cross-sections show that phase from the learned designs (long-dashed red) is closer to that of FP (solid blue) than traditional qDPC (short-dashed green).

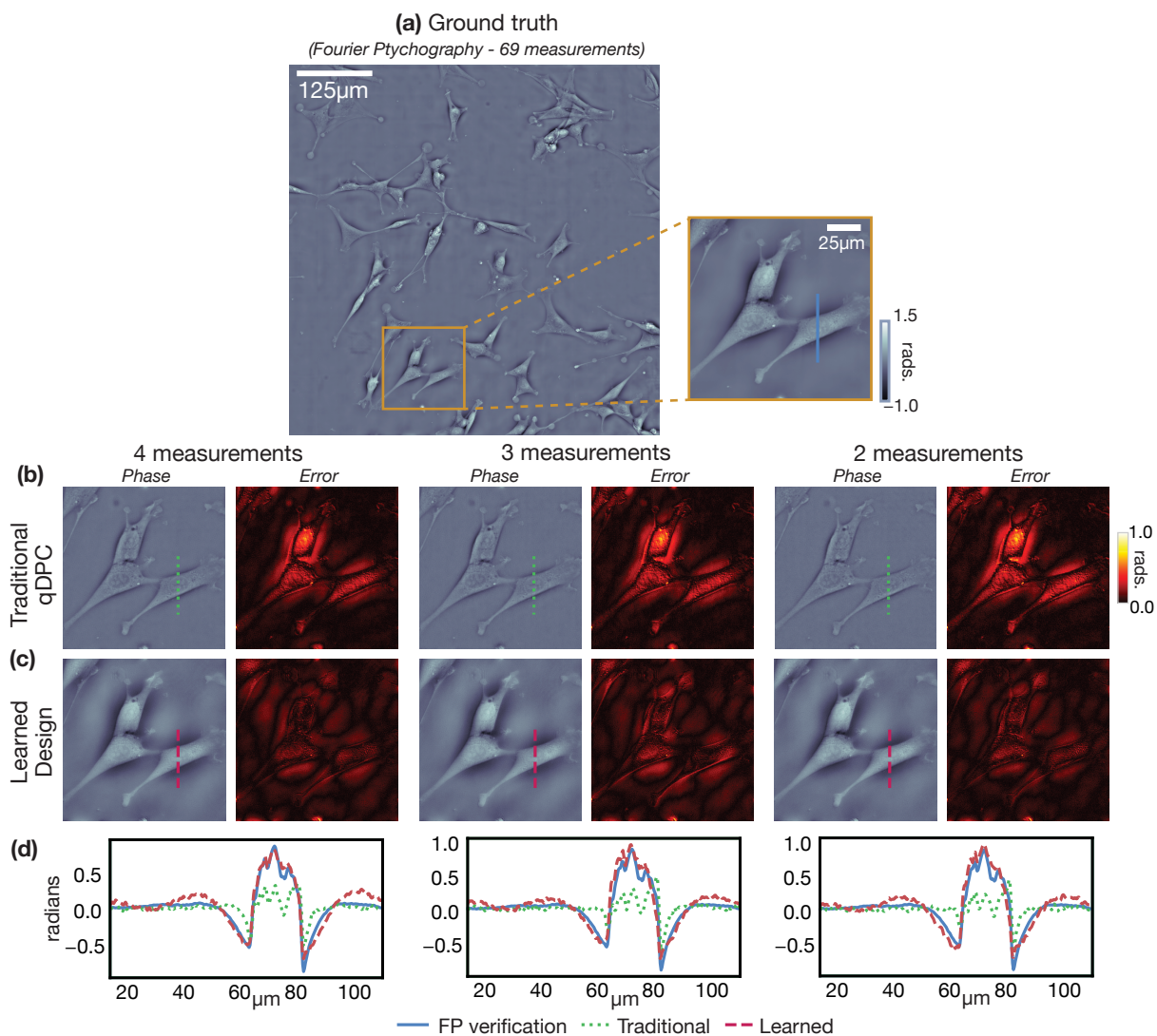


Figure 3.7: 3T3 mouse fibroblast cells reconstructions: Experimental comparison between phase results with (a) Fourier Ptychography (FP) using 69 measurements, (b) traditional qDPC and (c) learned designs, for the case of 4, 3, and 2 measurements. Error maps show the difference from the FP reconstruction. (d) Cross-sections show that phase from the learned designs (long-dashed red) is closer to that of FP (solid blue) than traditional qDPC (short-dashed green).

Table 3.2: PSNR Results: Average and standard deviation PSNR (dB) of phase reconstructions from the simulated testing examples using learned design for different numbers of unrolled iterations. Factor format: Mean  $\pm$  Std.

# of unrolled iterations	10	40	100
Mean $\pm$ Std. (dB)	22.53 $\pm$ 2.29	<b>28.48</b> $\pm$ 2.50	27.39 $\pm$ 1.92

peak SNR (PSNR) statistics (mean and standard deviation) for the phase reconstructions from  $\mathcal{R}$  evaluated on the set of testing examples. The learned designs give significant improvement over other designs, recovering both the high and low frequencies more accurately. The reduction in performance for learned design with 2 measurements (as compared to 3 and 4 measurements) is due to reduced sensitivity to low frequencies.

Comparing varying depth networks, in Table 3.2 I report the PSNR statistics for the phase reconstructions from  $\mathcal{R}$  evaluated on the set of testing examples using learned designs for networks with 10, 40, and 100 unrolled iterations with fixed step size,  $\alpha = 0.2$ , and regularization parameter,  $\tau = 1e^{-3}$ . If too few iterations are unrolled, the network cannot fully reconstruct the phase, resulting in lower mean PSNR. As more iterations are unrolled, the mean PSNR is reduced due to over regularization and over smoothing.

## Experimental Validation

To demonstrate that the learned designs generalize well in the experimental setting, I implement the method on an LED array microscope. A commercial Nikon TE300 microscope is equipped with a custom quasi-Dome [8] illumination system (581 programmable RGB LEDs:  $\lambda_R = 625$  nm,  $\lambda_G = 532$  nm,  $\lambda_B = 450$  nm) and a PCO.edge 5.5 monochrome camera ( $2560 \times 2160$ ,  $6.5\mu\text{m}$  pixel pitch, 16 bit). I image two samples: a USAF phase target (Benchmark Technologies) and fixed 3T3 mouse fibroblast cells (prepared as detailed in the supplement). In order to validate the method, I compare results against phase experimentally estimated via pupil-corrected Fourier Ptychography (FP) [1, 11, 73] with equivalent resolution. FP is expected to have good accuracy, since it uses significantly more measurements (69 single-LED measurements) and a non-linear reconstruction process.

Using the USAF target, I compare phase reconstructions from FP with traditional qDPC and the learned design measurements (Fig. 3.6). Traditional qDPC reconstructions consistently under-estimate the phase values. However, phase reconstructions using the learned design measurements are similar to phase estimated with FP. As the number of measurements is reduced, the performance quality of the reconstruction using traditional qDPC degrades, while the reconstruction using the learned design remains accurate.

To demonstrate the method with live biological samples, I repeated the experiments with 3T3 mouse fibroblast cells. Figure 3.7 shows that phase reconstructions from traditional qDPC again consistently under-estimate phase values, while phase reconstructions



using learned design measurements match the phase estimated with FP well.

## 3.5 Remarks

### Discussion

The proposed experimental design method efficiently learns the coded-illumination designs by incorporating both the system physics and the non-linear nature of iterative phase recovery. Learned designs with only 2 measurements can efficiently reconstruct phase with quality similar to Fourier Ptychography (69 measurements) and better than qDPC (4 measurements), giving an improvement in temporal resolution by a factor of  $2\times$  over traditional qDPC and far fewer than FP. Additionally, I demonstrate (Table 3.1) that the performance of the designs on a set of testing examples is superior to previously-proposed coded-illumination designs. Visually, the learned design reconstructions closely resemble the ground truth phase, with both low-frequency and high-frequency information accurately recovered.

By parameterizing the learning problem with only a few weights per measurement, the method can efficiently learn an experimental design with a small simulated dataset. This enables fast training and reduces computing requirements significantly. Obtaining large experimental datasets for training may be difficult in microscopy, so it is important that the method can be trained on simulated data only. Experimental results in Sec. 3.4 show similar quality to simulated results, with both using the designs learned from simulated data only.

Phase recovery with the learned designs' measurements are trained with a given number of reconstruction iterations (*e.g.* determined by a CPU budget). This makes the method particularly well-suited for real-time processing. qDPC can also be implemented in real-time, but limiting the compute time for the inverse problem (by restricting the number of iterations) limits convergence and causes low-frequency artifacts. The learned designs incorporate the number of iterations (and hence processing time) into the design process, producing high-quality phase reconstructions within a reasonable compute time. While one can reduce the number of iterations, if too few iterations are unrolled the accuracy of the model inversion decreases (Table 3.2).

Finally, rather than using a fixed step size and regularization parameter (as outlined in Sec. 4.4), these parameters can be jointly learned with the illumination patterns to optimize the whole system. For given noise statistics the regularization parameter could be learned; however, it would perform sub optimally for different noise statistics and would require retraining. Future systems should learn regularization parameters that can be adapted post training to account for variable noise levels similar to in [74].

## Outlook

The method is general to the problem of experimental design. Similar to QPI, many fields (*e.g.* Magnetic resonance imaging (MRI), fluorescence microscopy) use physics-based non-linear iterative reconstruction techniques to achieve state-of-the-art performance. With the correct model parameterization and physically-relevant constraints, the method could be applied to learn optimal design for these applications (*e.g.* undersampling patterns for compressed sensing MRI [75], PSFs for fluorescence microscopy [76]).

Requirements for applying the method are simple: the reconstruction algorithm's updates must be differentiable (*e.g.* gradient update and proximal update) so that analytic gradients of the learning loss can be computed with respect to the design parameters. Of practical importance, the proximal operator of the regularizer should be chosen so that it has a closed form. While this is not a strict requirement, if the operator itself requires an additional iterative optimization, error will have to be backpropagated through an excessive number of iterations. Here, I choose to penalize anisotropic TV, whose proximal operator can be approximated in closed form [70]. Further, including an acceleration update improves the convergence of gradient-based reconstructions. As a result, the *unrolled network* can be constructed using fewer layers than its unaccelerated counterpart. This will reduce both computation time and training requirements.

## Conclusion

I have presented a general framework for incorporating the non-linearities of regularized reconstruction and known system physics to learn optimal experimental design. Here, I have applied this method to learn coded-illumination source designs for quantitative phase recovery. the coded-illumination designs can improve the temporal resolution of the acquisition and enable real-time processing, while maintaining high accuracy. I demonstrated here that the learned designs achieve high-quality reconstructions experimentally without the need for retraining.

# Chapter 4

## Physics-based Learned Design for Fourier Ptychographic Microscopy

### 4.1 Introduction

Fourier Ptychographic Microscopy (FPM) [1] is a computational imaging method that achieves both large field-of-view (FOV) and high resolution for both amplitude and quantitative phase imaging (QPI), enabling high-throughput imaging for applications in pathology [1] and live cell imaging [77]. It can be conveniently implemented using an inexpensive hardware modification to a conventional microscope simply by replacing the illumination source with a programmable LED array [1, 8]. By capturing many low-resolution images, each using a different LED to encode a distinct part of the sample's Fourier space, the high-resolution complex transmittance function of the sample can be recovered via non-linear phase retrieval optimization.

FPM increases the space-bandwidth product of the microscope, but comes at the cost of significantly reduced temporal resolution. In the originally proposed FPM system, a single measurement is acquired per LED [1] (single-LED design). To produce a high-quality reconstruction, measurements must redundantly encode information such that neighboring LEDs result in at least 60% overlapping coverage of the sample's Fourier space [78, 79]. This results in a poor data input:output ratio - approximately  $10\times$  more pixels are acquired than the number of pixels reconstructed [77]. In practice, dozens of measurements are used and the slow speed of capture prevents imaging of live cell dynamics. Several works have improved upon the temporal resolution of the single-LED design, by turning on multiple LEDs simultaneously [11, 77], spatially multiplexing the measurements across the sensor [80], color-multiplexing [81], only acquiring the most important measurements [82], or motion correction [83, 84].

*Multiplexed* FPM [11, 77] improves upon the measurement requirement for single-LED design to achieve a data input:output ratio of approximately  $1\times$  without sacrificing FOV or resolution. This is accomplished by acquiring fewer measurements, each taken under

multiple-LED illumination, thereby encoding multiple parts of the sample’s Fourier space into each measurement. However, as fewer measurements are acquired and more of the sample’s Fourier space is encoded per measurement, the reconstructions become blurry. Practically, the reconstruction’s performance is governed by the system’s experimental design: which LEDs are turned on in each measurement and how many measurements are acquired.

Recent work has shown that supervised learning can be used to find the experimental design that optimizes the performance of a computational imaging system. Several physics-free methods [85, 86] consider learning design parameters in addition to a black-box Neural Network (NN) that learns an image reconstruction. These methods require a huge number of training examples to properly learn millions of parameters that model the reconstruction process, and they often do not transfer well to the experimental setting. In comparison, physics-based methods [24, 87] are able to efficiently learn the experimental design with very few training examples and have been shown to learn designs that do transfer well to the experimental setting. This is achieved by *unrolling* the image reconstruction process [19, 22, 88, 89] to form a Physics-based Network (PbN) [24] and learning the experimental design parameters to maximize system’s performance.

## Contribution

In this work, I use the physics-based learned design framework [24] (outlined in Chap. 3) to find LED pattern designs that maximize the reconstruction performance of the FPM system. Using the learned designs, I demonstrate the ability to reduce the data input:output ratio below 1 without compromising reconstruction performance, thereby improving temporal resolution of the system, without sacrificing FOV or resolution. Next, by incorporating a context-specific loss function, I am able to tailor the learned LED patterns to different applications (*e.g.* amplitude contrast imaging vs. QPI). In experiment, I demonstrate that the LED patterns learned in simulation work well for both amplitude contrast imaging and QPI applications. Finally, I discuss interpretations of the learned designs and confirm that they match conventional intuition, as well as practical implementation details required to build PbNs for large-scale systems.

## 4.2 Fourier Ptychographic Microscopy

As mentioned in Section A.1, FPM achieves super-resolution, in that it resolves features beyond the diffraction limit of the microscope’s objective. To do so, a set of low-resolution measurements recorded under varying illumination configurations are combined using a computational reconstruction. In this section, we revisit the principles of FPM and mathematically describe how a complex-valued reconstruction is obtained.

## Forward Model

As reviewed in Chap. 2, an optically-thin sample can be modeled by its 2D transmittance function,  $x(\mathbf{r}) = e^{j\phi(\mathbf{r}) - \mu(\mathbf{r})}$ , where  $\phi(\mathbf{r})$  and  $\mu(\mathbf{r})$  represent the phase and absorption distributions of the sample, respectively, and  $\mathbf{r} \in \mathbb{R}^2$  is the spatial coordinate vector. Measurements of this sampling using the LED array microscope are formed following,

$$y(\mathbf{r}) = \left| \mathcal{F}^{-1} \{P(\mathbf{u}) \mathcal{F} \{x(\mathbf{r})\} (\mathbf{u} - \boldsymbol{\xi}_l)\} \right|^2. \quad (4.1)$$

Here,  $\mathcal{F}$  and  $\mathcal{F}^{-1}$  denote Fourier transform and its inverse, respectively,  $\boldsymbol{\xi}$  is the illumination's angle determined by the LED's position, and  $P$  is the *pupil function* of the microscope objective, which has a hard frequency cutoff at the diffraction limit set by its numerical aperture ( $\|\mathbf{u}\|_2 < \frac{\text{NA}_{\text{obj}}}{\lambda}$ ) [32]. In *multiplexed* FPM (where  $L$  LEDs illuminate the sample simultaneously), the measured intensity is the weighted sum of the individual LED measurements, since LEDs are mutually incoherent with each other [11, 77]:

$$y_m(\mathbf{r}) = \sum_{l=1}^L c_l y_l(\mathbf{r}), \quad (4.2)$$

where the non-negative scalar  $c_l$  represents the relative brightness of the  $l$ th LED. The implication of Eq. (4.2) is that multiplexing mixes images, each of which non-linearly encodes information from a distinct region in the sample's Fourier space, thereby reducing the number of required measurements and improving the temporal resolution over the single-LED design.

However, there is a limit: as more LEDs are used per measurement, it becomes harder to retrieve the high-resolution complex image. It has been shown that the data input:output ratio can be reduced to approximately 1, without a significant reduction in reconstruction quality [11, 77]. Beyond a data input:output ratio of 1, image quality is reduced and appears blurred.

## Inverse Problem

Given a forward model described above, the sample's complex-field is reconstructed by solving a non-linear inverse problem. Following convention, we discretize the 2D transmission function and multiplexed measurements as  $\mathbf{x} \in \mathbb{C}^q$  and  $\mathbf{y}_m \in \mathbb{R}^p$ , respectively. Note that  $p < q$  since the reconstructed transmission function has a higher space-bandwidth product than that of the measurements [1].

Reconstruction of the super-resolved complex transmittance function is then cast as an optimization problem:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \sum_{k=1}^K \left\| \mathbf{y}_{m_k} - \sum_{l \in \Omega_k} c_l |\mathbf{A}_l \mathbf{x}|^2 \right\|_2^2 \quad (4.3)$$

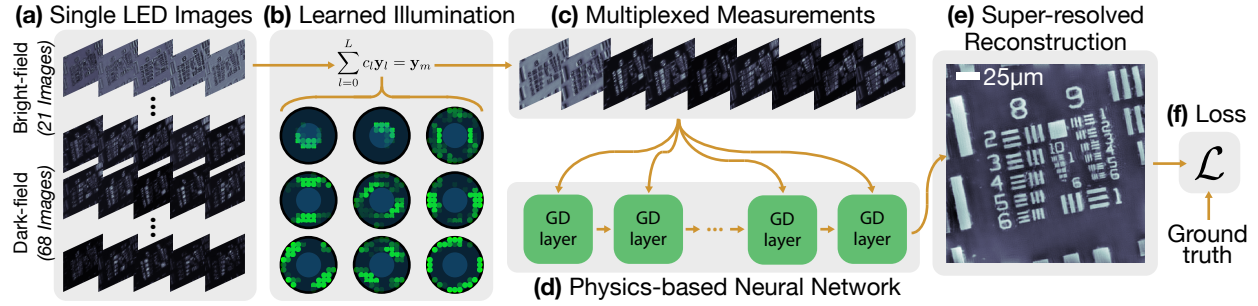


Figure 4.1: Physics-based Learned Design: (a) We start with an input dataset of single-LED images. (b) For each measurement’s LED pattern, we learn the scalar weights,  $c_l$ , that correspond to the brightness of each LED. (c) Each multiplexed measurement is then fed into (d) the Physics-based Network (PbN), where each layer represents a single gradient descent (GD) step of the image reconstruction. (e) The output super-resolved reconstruction is then compared to the target reconstruction via (f) a context-specific loss function, in order to optimize the brightness of each LED.

where  $\Omega_k$  is the index set for LEDs that are “on” during the  $k^{\text{th}}$  multiplexed measurement  $y_{m_k}$ . In Eq. (4.3), the discretized system model is given by

$$\mathbf{A}_l = \mathbf{F}^H \mathbf{P}_l \mathbf{F}, \quad (4.4)$$

where  $\mathbf{P}_l \in \mathbb{C}^{p \times q}$  is the matrix representation of the pupil function shifted by  $\xi_l$ , and  $\mathbf{F}$  and  $\mathbf{F}^H$  represent the discrete (normalized) Fourier transform matrix and its inverse, respectively. In this work we solve the optimization problem via gradient descent for which the convergence of the iterates to a stationary point is established [34, 90].

### 4.3 Physics-based Learned Design

Physics-based Learned Design [24] formulates the experimental design as a supervised learning problem that optimizes the source patterns to maximize the performance of the system. The method takes advantage of the PbN, a rethinking of iterative optimization procedures as NNs that incorporate known quantities such as the system model and reconstruction non-linearities. The inclusion of these quantities provide the robustness and generality associated with physics-based reconstruction and reduces the number of learnable parameters, thereby significantly reducing the number of training examples required. In this section, we show how to utilize this framework for FPM, analyze tailoring the experimental design for a specific application, and discuss memory limitations.

## Physics-based Network

The PbN is defined by the gradient-based optimizer used to solve the inverse problem in Eq. 4.3 (Fig. 4.1d). Rather than using a network made up of millions of learnable parameters (*e.g.* convolutional filters), which would require a large number of training examples, the network is constructed from well-defined functions (*i.e.* gradient operations and the system model). Each layer of the network corresponds to a single iteration of the gradient-based optimizer. The network,

$$\mathbf{x}^* = R_{\mathbf{C}}(\{\mathbf{y}_l\}_{l=1}^L), \quad (4.5)$$

takes as input a set of single LED measurements  $\{\mathbf{y}_l\}_l$  (Fig. 4.1a), linearly combines them according to the scalar weights in  $\mathbf{C} \in \mathbb{R}^{K \times L}$  (Fig. 4.1b) (which represent the relative brightness for each of  $L$  LEDs across  $K$  measurements), and outputs the reconstructed super-resolved complex image,  $\mathbf{x}^*$  (Fig. 4.1e). The only learnable parameters in the PbN are the scalar weights that set the relative brightness of each LED, thereby enabling learning with just a few training examples.

## Context-Specific Learning Objective

For specific applications, we can learn a custom experimental design by tailoring the loss function and training dataset. For example, with stained pathology slides, only amplitude contrast imaging is desired and phase is not used. In this case, the loss function penalizes only the amplitude error of the reconstruction. For live cell imaging applications, samples are nearly transparent and essentially pure phase objects. Hence, the loss function penalizes just the phase error of the reconstruction. For the general case, we can design for a spectrum of applications using this non-convex loss function,

$$\mathcal{L}(\mathbf{C}) = \sum_{w=1}^W \underbrace{\gamma \|\|\mathbf{x}_w^*(\mathbf{C})\| - \|\tilde{\mathbf{x}}_w\|_2\|_2^2}_{\text{amplitude loss}} + (1 - \gamma) \underbrace{\|\angle \mathbf{x}_w^*(\mathbf{C}) - \angle \tilde{\mathbf{x}}_w\|_2^2}_{\text{phase loss}}, \quad (4.6)$$

where  $\tilde{\mathbf{x}}_w$  is the ground truth complex transmittance function for the  $w^{\text{th}}$  training example,  $W$  is the total number of training examples,  $\gamma \in [0, 1]$  weights the loss between the phase ( $\gamma = 0$ ) and amplitude ( $\gamma = 1$ ) loss functions, and  $\|\cdot\|$  and  $\angle$  return the amplitude and phase of the complex image, respectively.

In addition to tailoring the loss function, we also tailor the training examples to be predominately amplitude or phase samples, simulated from stock images of cells. For pathology applications, the dominant structural contrast is simulated in the absorption component of the sample's transmittance function. For QPI, the dominant structural contrast is in the phase component. Thanks to the efficient parameterization of the network by only the design parameters, we do not require a large amount of training data (90 image patches).

The loss function is then minimized via stochastic gradient descent (SGD) subject to several practical constraints,

$$\mathbf{C}^* = \arg \min_{\mathbf{C}} \mathcal{L}(\mathbf{C}) \quad \text{s.t.} \quad (4.7)$$

$$\mathbf{b}_k \odot \mathbf{c}_k = \mathbf{0} \quad (\text{geometric}) \quad (4.8)$$

$$c_{kl} \geq 0 \quad \forall l \in \{1 \dots L\}, \quad (\text{non-negativity}) \quad (4.9)$$

$$\|\mathbf{c}_k\|_1 = 1 \quad (\text{scaling}) \quad (4.10)$$

$$\forall k \in \{1 \dots K\},$$

where  $\odot$  is element-wise multiplication. The geometric constraint (Eq. 4.8) allows us to insert prior knowledge about the design by constraining certain LEDs to be “on” or “off” in each measurement. This is enforced using a mask,  $\mathbf{b}_k$ , for the  $k^{\text{th}}$  measurement. In practice, we use the geometric constraint to separately design source patterns for the bright-field and dark-field regions. The non-negativity constraint (Eq. 4.9) on the LED brightnesses enables the designs to be feasibly implemented in hardware. Finally, the scaling constraint (Eq. 4.10) removes degenerate solutions by eliminating arbitrary scalings of the same solution. After training, the overall brightness of the LED patterns can be scaled to utilize the full dynamic range of the camera or to a match desired image noise level.

## Computational Limitations

The feed-forward process of the PbN has identical speed and memory complexity to that of its iterative reconstruction. However, when using graphics processing units (GPU) to accelerate the training process, the memory required for backpropagation to compute gradients to learn is limiting. The memory required is proportional to the product of the number of unrolled iterations, the number of measurements, and the size of the reconstructed image. To ensure convergence, we must use a sufficient number of unrolled iterations, which limits us to tuning only the latter two factors. To stay within the GPU’s memory limit (12GB on an NVIDIA TITAN X Pascal GPU), we learn the design for reconstructing small patches ( $35\text{px} \times 35\text{px}$ ) using up to 15 measurements. Thanks to the Fourier relationship between FOV and sampling in Fourier space, the designs generalize well to larger FOVs. Broadly, this challenge is seen beyond just this particular application and has led to interesting line of research. In the next chapter (Chap. 5), I propose a new methodology to enable physics-based learning for large-scale computational imaging systems.



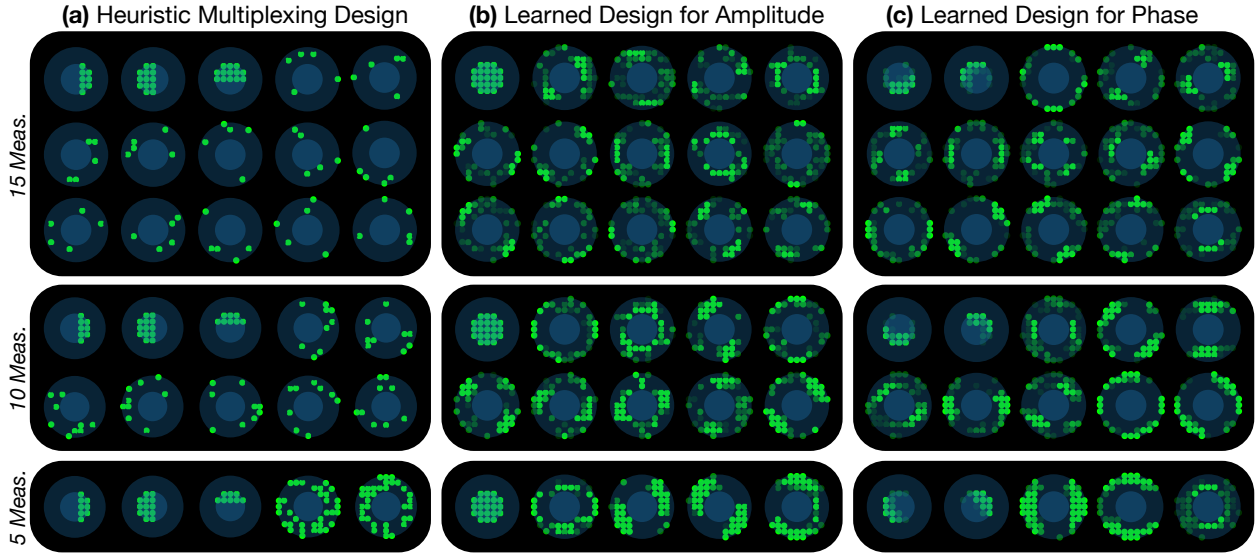


Figure 4.2: LED Designs: LED source patterns for (a) heuristic multiplexing designs with 15, 10 and 5 measurements, (b) the learned designs for super-resolved amplitude imaging with 15, 10 and 5 measurements, and (c) the learned designs for super-resolved quantitative phase imaging with 15, 10 and 5 measurements. The inner blue circle denotes LEDs in the bright-field region, while the outer blue shell denotes the dark-field region. An LED’s shade of green corresponds to that LED’s brightness.

## 4.4 Results

I validate the learned designs for FPM with both simulated and experimental results. To learn the designs, we create a PbN by unrolling 100 iterations of gradient descent (as described in Sec. 4.1) with the step size of 0.5. Training is conducted in Pytorch using auto-differentiation [91] to compute the gradients with respect to the learnable parameters. Data examples (100 patches each of  $35\text{px} \times 35\text{px}$ ) are generated in simulation for two context-specific applications (amplitude contrast imaging and QPI). Each set of examples is shuffled and split into groups of 90 and 10 for training and testing, respectively. The training process is initialized using LED brightnesses drawn from a uniform random distribution.

Simulations are set up to match the experimental system parameters ( $8\times$  objective,  $\text{NA}_{\text{obj}} = 0.2$ , with camera pixel size  $\text{ps}_{\text{camera}} = 6.5\mu\text{m}$  and LEDs of wavelength  $\lambda = 0.514\mu\text{m}$ ). I consider 89 total LEDs on a Cartesian grid separated by a distance of  $4\text{mm}$  in  $x$  and  $y$  with 21 in the bright-field region and 68 in the dark-field region, up to  $\text{NA}_{\text{illum.}} = 0.42$ . This allows us to reconstruct features up to  $\text{NA}_{\text{recon.}} = \text{NA}_{\text{obj}} + \text{NA}_{\text{illum.}} = 0.62$ . To mimic experimental noise, each measurement is simulated with a fixed exposure time such that the shot noise for bright-field measurements has a mean rate of 10,000.

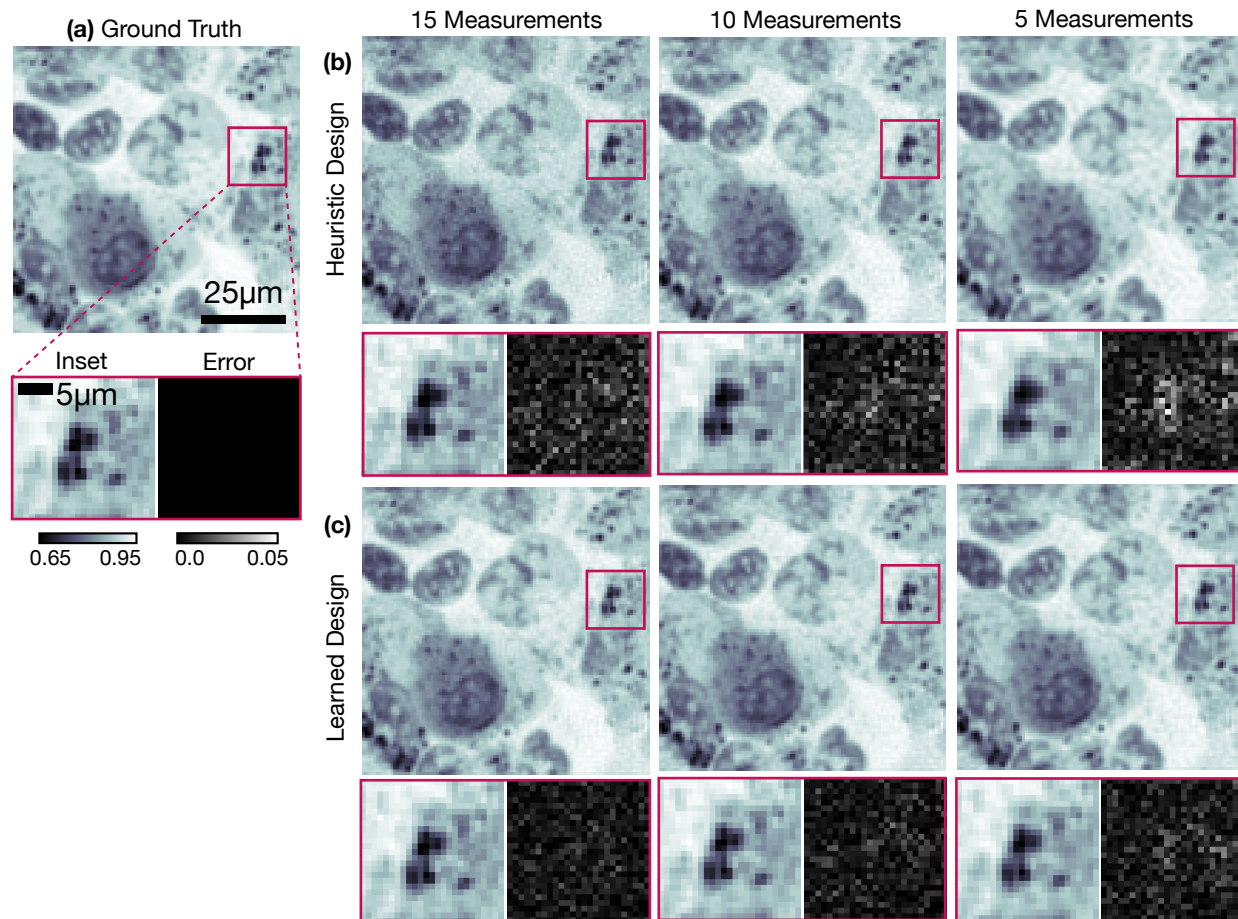


Figure 4.3: Simulations of super-resolution amplitude imaging: (a) Ground truth amplitude. (b) Amplitude reconstructions from simulated measurements using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and their error maps.

## Simulation Results

In Fig. 4.2, we display LED patterns for heuristic multiplexing designs for 15, 10, and 5 measurements [11, 77], as compared to the proposed learned designs for 15, 10, and 5 measurements for both amplitude contrast imaging ( $\gamma = 1$ ) and QPI ( $\gamma = 0$ ) applications. The heuristic multiplexing design for  $K$  measurements consists of 3 half-circle bright-field measurements and  $K - 3$  dark-field measurements. The LEDs in the dark-field measurements are randomly distributed such that each measurement has an equal number of LEDs “on” with equal brightness and each LED is “on” exactly once.

For the learned designs, the geometric constraint (Eq. 4.8) enforces separate LED patterns for bright-field and dark-field LEDs. Specifically, we learn a single bright-field measurement for amplitude contrast imaging and two bright-field measurements for QPI.

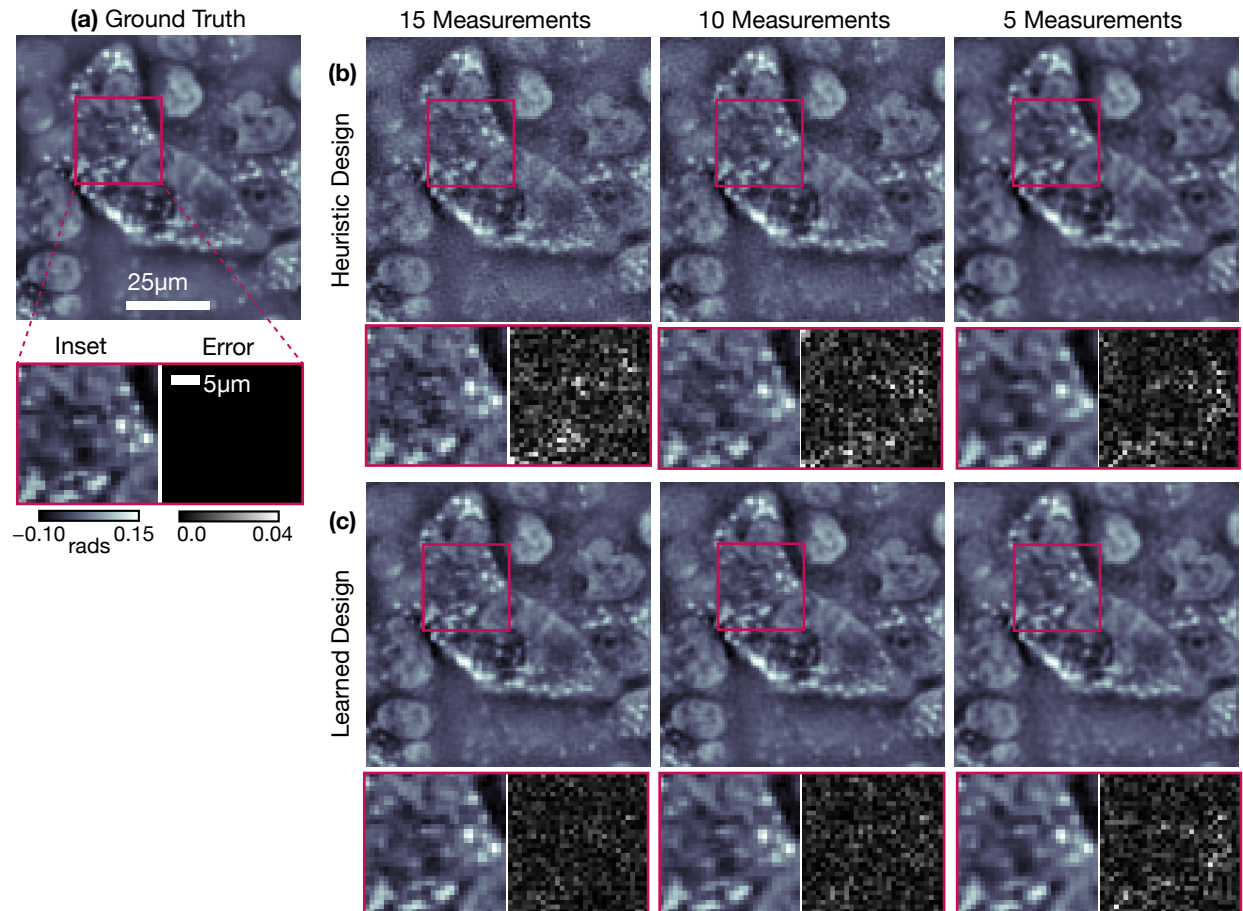


Figure 4.4: Simulations of super-resolution quantitative phase imaging: (a) Ground truth phase. (b) Phase reconstructions from simulations using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and their error maps.

Generally, the amplitude contrast learned designs are symmetric in the bright-field, while the QPI learned designs are asymmetric in the bright-field (Fig. 4.2b,c). This makes sense because phase is anti-symmetrically encoded in Fourier space, whereas amplitude is symmetrically encoded. Unlike in heuristic multiplexing designs, the learned dark-field patterns are not random, but systematic. First, the learned designs are predominantly symmetric and LEDs are turned on in clusters, thereby encoding similar information of the sample's Fourier space in each measurement. Second, the LED clusters are fairly disjoint between measurements, thereby encoding different parts of the sample's Fourier space. Finally, as the number of measurements increases, the LED clusters decrease in size. These themes are further discussed in Section 4.5.

In Fig. 4.3 and Fig. 4.4, we show super-resolved amplitude and QPI reconstructions from simulated measurements. We compare results for heuristic multiplexing designs [11],

Table 4.1: PSNR for simulated amplitude reconstructions: average testing LF-PSNR and HF-PSNR (dB) for different numbers of measurements.

# Meas.	Heuristic	Physics-based
	Multiplexing (LF-PSNR/HF-PSNR)	Learned Design (LF-PSNR/HF-PSNR)
15	47.84 / 16.15	<b>50.30 / 19.86</b>
10	46.28 / 17.37	<b>49.39 / 19.83</b>
5	43.83 / 18.40	<b>47.03 / 19.43</b>

Table 4.2: PSNR for simulated phase reconstructions: average testing LF-PSNR and HF-PSNR (dB) for different numbers of measurements.

# Meas.	Heuristic	Physics-based
	Multiplexing (LF-PSNR/HF-PSNR)	Learned Design (LF-PSNR/HF-PSNR)
15	31.01 / 14.97	<b>32.80 / 19.76</b>
10	30.47 / 16.25	<b>31.85 / 19.88</b>
5	<b>29.94 / 19.42</b>	28.93 / <b>20.00</b>

[77](#) and the learned designs to ground truth for 15, 10, and 5 measurements (corresponding to data input:output ratios of 1, 0.66, and 0.33, respectively). Insets highlight a small region and their differences with ground truth. Using heuristic multiplexing designs, as the number of measurements increases, finer detail features are resolved, but reconstructions become corrupted by artifacts. As the number of measurements decreases, reconstruction results become less noisy, but blurrier. In comparison, reconstructions using the learned designs remain sharp as the number of measurements decreases.

To further quantify this trend, we report average testing peak signal-to-noise ratio (PSNR) for different regions of the reconstruction’s Fourier space. Specifically, we calculate the PSNR for the low spatial-frequency region (LF-PSNR) up to the incoherent diffraction limit ( $NA = 0.4$ ) and the PSNR for the high spatial-frequency region (HF-PSNR) ( $NA = 0.4$  to  $NA = 0.62$ ) to quantify high-frequency noise and the quality of super-resolved features. In [Table 4.1](#) and [Table 4.2](#) we report average testing LF-PSNR and HF-PSNR for super-resolved amplitude contrast imaging and super-resolved QPI applications, respectively. For heuristic multiplexing design, blurring causes the HF-PSNR for 5 measurements to be relatively higher and noise causes the HF-PSNR for 15 measurements to be relatively lower. In comparison, HF-PSNR for the learned designs are able to achieve approximately consistent performance as the number of measurements decreases.

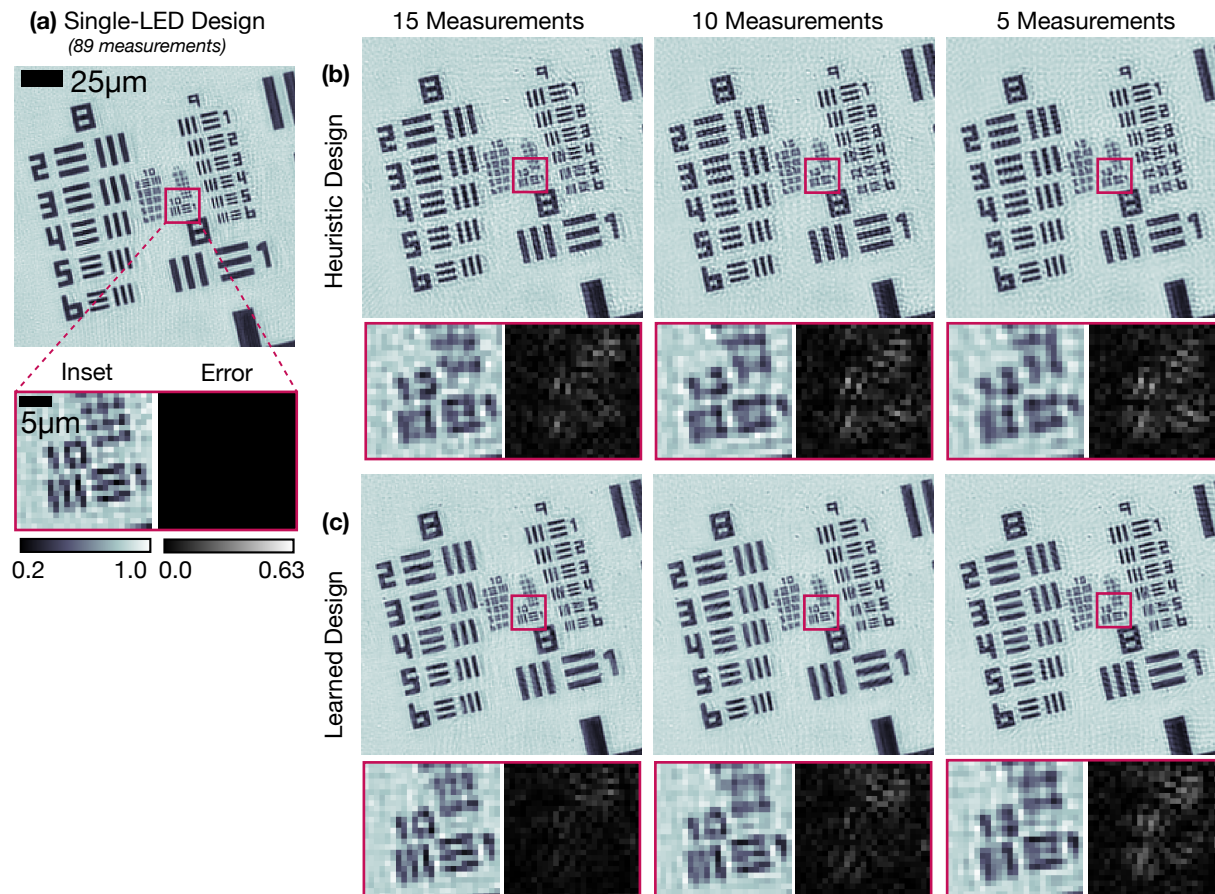


Figure 4.5: Experimental amplitude target results: (a) Single-LED amplitude reconstruction using 89 measurements serves as ground truth. (b) Amplitude reconstructions using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and difference with the ground truth.

## Experimental Results

To validate the learned designs experimentally and show that training on simulated data is sufficient, the method is implemented on an LED array microscope. The setup is a commercial Nikon TE300 microscope equipped with a custom LED array illumination system and a PCO.edge 5.5 monochrome camera ( $2560 \times 2160$ ,  $6.5\mu\text{m}$  pixel pitch, 16 bit). We image two samples: a USAF amplitude target and a USAF phase target (Benchmark Technologies).

The experimental reconstructions of amplitude (Fig. 4.5) and phase (Fig. 4.6) targets compare the proposed learned designs to heuristic multiplexing designs [11, 77] and FPM single-LED design (89 measurements), which will serve as “ground truth” for validation. To make a fair comparison between different methods, we synthesize measurements for different designs by digitally combining a fixed set of single LED measurements. For am-

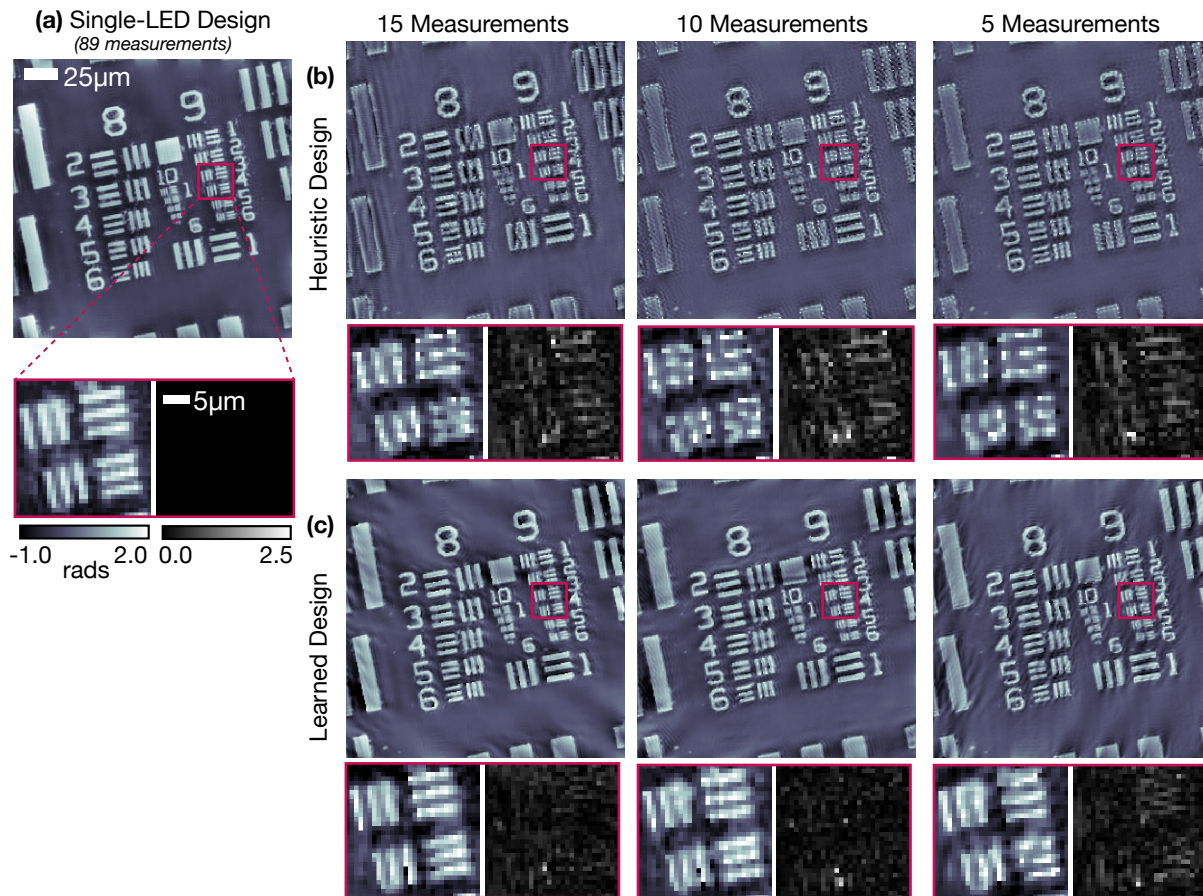


Figure 4.6: Experimental phase target results: (a) Single-LED phase reconstruction using 89 measurements serves as ground truth. (b) Phase reconstructions using heuristic multiplexing designs and (c) the proposed learned designs, with 15, 10 and 5 measurements. Insets highlight detailed features and difference with the ground truth.

plitude contrast imaging (Fig. 4.5), we show insets and their error maps, demonstrating resolution of features with a pitch of  $0.97\mu\text{m}$ , even as the number of measurements is reduced. Heuristic multiplexing designs, on the other hand, lose quality and resolution as the number of measurements decreases. For QPI (Fig. 4.6), we show insets and their error maps, demonstrating resolution of features with a pitch of  $1.38\mu\text{m}$ , while reconstructions with heuristic multiplexing designs degrade with fewer measurements.

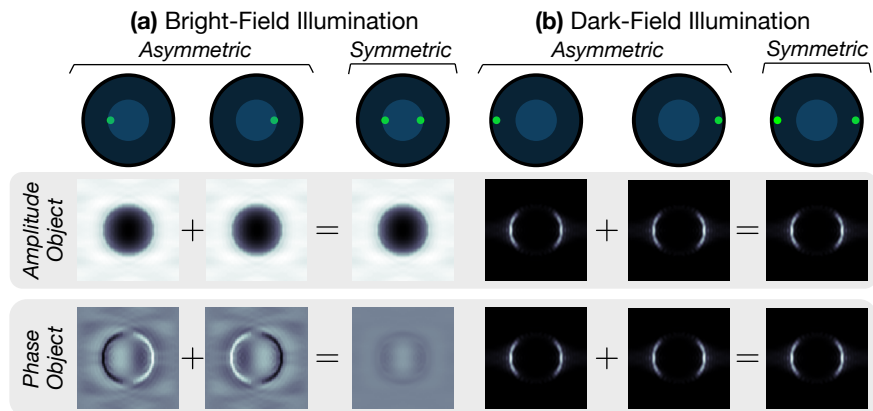


Figure 4.7: Bright-Field vs. dark-Field LED designs: (a) Asymmetric and symmetric *bright-field* LED patterns for amplitude and phase samples. (b) Asymmetric and symmetric *dark-field* LED patterns for amplitude and phase samples.

## 4.5 Remarks

### Discussion

Physics-based learned design for FPM breaks the trade-off between temporal resolution and reconstruction quality. While heuristic multiplexing designs have significantly improved temporal resolution, as compared to single-LED design, reconstruction quality degrades as fewer measurements are acquired (data input:output ratio less than 1). The learned designs produce systematic LED patterns that more efficiently encode the sample's Fourier space, thereby allowing for high-quality reconstructions using even fewer measurements.

For LED patterns in the bright-field region, the learned designs (Fig. 4.2) can be explained intuitively. Asymmetric illumination yields strong phase contrast and symmetric illumination yields strong amplitude contrast. In Fig. 4.7a, we illustrate this by simulating measurements of a circular object with asymmetric and symmetric LED illumination. For an amplitude object, a pair of asymmetric LEDs produce identical contrast intensity measurements, such that when the sample is symmetrically illuminated the image contrasts add constructively. For a phase object, a pair of asymmetric LEDs produce opposite contrast intensity measurements, such that when the sample is illuminated the two image contrasts add destructively, reducing image contrast. Hence, phase samples will provide better contrast with asymmetric bright-field LED patterns, whereas amplitude samples favor symmetric patterns.

In contrast, the learned designs for dark-field LEDs always form symmetric clusters (Fig. 4.2). In Fig. 4.7b, we simulate measurements of a circular object with asymmetric and symmetric dark-field LED patterns to understand why. For both amplitude and phase

samples, pairs of asymmetric LEDs produce similar image contrasts such that when either sample is illuminated symmetrically the two image contrasts add constructively. We speculate that by turning on dark-field LEDs that produce similar contrast within the same measurement, it is less ambiguous where the encoded information is from in the sample's Fourier space and thus the reconstruction is able to retrieve the sample's high-resolution Fourier space more faithfully. Practically, symmetrical illumination increases the total illumination brightness, thereby allowing a reduction in the total acquisition time.

How to decide the number of iterations to unroll (*i.e.* the *depth* of the physics-based network) is still unclear. Practically, one can set the number of unrolled iterations to fill the allowed computation budget, however, as we have seen in Chap. 3, the image reconstruction quality does not monotonically increase with the number of unrolled iterations, thus as the computational budget is increased improvement in performance is not always observed. This effect can be caused by model error or over regularization. The solution, referred to as early stopping, can improve performance by mitigating these effects (*e.g.* over smoothing) by terminating the optimization after a fixed number of iterations rather than running the inverse problem to convergence. Ideally, this fixed number of iterations should be optimized, similar to the rest of the learnable design (*e.g.* experimental design and signal prior). This could be achieved by making the number of unrolled iterations adaptive. Rather than specifying a number of iterations, a stopping criteria could be specified (*e.g.* mean square error with ground truth). In addition, the maximum number of unrolled iteration can be limited by the memory required for physics-based learning (discussed in Sec. 4.3 and Chap. 5).

## Conclusion

In this work, we have introduced a physics-based learned design framework to create interpretable context-specific LED source patterns for Fourier Ptychographic Microscopy, a highly non-linear computational imaging system. With these learned designs, we can achieve a more favorable trade off between temporal resolution and reconstruction quality, and tailor the learned source designs for context-specific applications such as amplitude imaging and quantitative phase imaging. Finally, we demonstrate that designs learned in simulation generalize well in the experimental setting.



## Chapter 5

# Memory-efficient Learning for Large-scale Computational Imaging

### 5.1 Introduction

Computational imaging systems (*e.g.* tomographic systems, computational optics, magnetic resonance imaging (MRI)) jointly design software and hardware to retrieve information which is not traditionally accessible. Generally, such systems are characterized by how the information is encoded (forward process) and decoded (inverse problem) from the measurements. Recently, physics-based learning [88] has demonstrated the ability to directly optimize a computational imaging system's performance [19–21, 23–25, 87, 92–97]. Physics-based learning takes advantage of both the known physics of the system's forward model process and the architecture of the decoder's iterative optimizer to build a differentiable neural network that is efficiently parameterized by only a limited number of learnable variables, thereby enabling training using less data [21, 24, 25], while still retaining the robustness and interpretability associated with conventional physics-based inverse problems. Specifically, physics-based networks (PbN) are constructed by unrolling the iterations of an image reconstruction algorithm (*e.g.* proximal gradient descent [37] or half quadratic splitting [35]), where the iterations of the optimizer form the layers of the network. Hence, the physical forward model is built into the architecture of the network. Commonly, standard signal prior models (*e.g.* total variation) or a function that enforces consistency (*i.e.* proximal operators) have been replaced by a learnable convolutional neural network [19, 21, 23, 94, 96, 98] to model the image priors. One can also learn the data capture scheme (*i.e.* experimental design) by making the system parameters that form the measurements learnable [24, 25, 87]).

Many computational imaging systems present a unique challenge for PbN implementation, due to the large size and dimensionality of variables that are decoded from the measurements. Training such a PbN relies on gradient-based updates computed using backpropagation (an implementation of reverse-mode differentiation [99]) for learn-

ing. As the quantity of decoded information grows, the memory required to perform backpropagation (via automatic differentiation) may exceed the memory capacity of the graphics processing unit (GPU).









Complexity	Standard Backpropagation	Forward Recalculation	Forward Recalculation (Checkpointing)	Reverse Recalculation	Reverse Recalculation (Checkpointing)	
Storage	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(N/K)$	$\mathcal{O}(1)$	$\mathcal{O}(N/K)$	<b>Accuracy Key</b>  exact   approximate
Computational	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(NK)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	
Accuracy						

Figure 5.1: Overview of memory-efficient methods: Storage and computational complexity of standard backpropagation, forward recalculation, forward recalculation with checkpointing, reverse recalculation, and reverse recalculation with checkpointing for a physics-based network with  $N$  layers and  $K$  is the spacing between checkpoints. Methods’ computational accuracy are qualitatively reported from exact (three greens dots) to approximate (one green dot).

Methods to save memory during backpropagation (forward recalculation, forward checkpointing, and reverse recalculation) trade off storage and computational complexity (*i.e.* the amount of memory and time required for each unrolled layer) [99]. Rather than storing the whole computational graph required for auto-differentiation in memory, these methods reform the graph on an on-demand basis. For a PbN with  $N$  layers, standard backpropagation stores the whole graph, achieving  $\mathcal{O}(N)$  computational and storage complexity. Forward recalculation instead reforms unstored parts of the graph by reevaluating the operations of the network forward from the beginning. This achieves  $\mathcal{O}(1)$  storage complexity, but has  $\mathcal{O}(N^2)$  computational complexity because layers of the graph are recomputed from the beginning of the network, while backpropagation requires access to the layers in reverse order. Forward checkpointing saves variables every  $K$  layers and forward-recalculates unstored layers of the graph from the closest stored variables (checkpoints), thus directly trading off computational,  $\mathcal{O}(NK)$ , and storage,  $\mathcal{O}(N/K)$ , complexity. Reverse recalculation provides a practical solution to beat the trade-off between storage vs. computational complexities by reforming unstored layers of the graph in reverse order from the output of the network (in the same order as required for backpropagation), yielding  $\mathcal{O}(N)$  computational and  $\mathcal{O}(1)$  storage complexities.

## Contribution

Here, we propose a memory-efficient learning procedure based on the concept of reverse recalculation and invertibility that will enable physics-based learning for general large-scale computational imaging systems. The main contributions of this work are:

1. **General memory-efficient learning procedure for physics-based networks.** I describe how to compute gradients for physics-based learning for networks formed

from gradient and proximal update layers without any significant modifications to the architecture of the layers. These layers are seen in a large swath of image reconstruction algorithms, making physics-based learning more feasible for many large-scale physics-based networks. In this work, we focus on the commonly-used proximal gradient descent [37] and half quadratic splitting [35] algorithms.

2. **A hybrid reverse-recalculation and checkpointing scheme to ensure accuracy.** There are several common sources of error: accumulation due to numerical precision and the convergence of PbNs constructed from convex programs. Our hybrid scheme mitigates both issues with a small number of checkpoints.
3. **Demonstration of results for general computational imaging systems.** I learn the design for several large-scale computational imaging systems: 3D multi-channel compressed sensing MRI and super-resolution optical microscopy. In each of these applications, the method are able to learn the computational imaging system’s design using PbNs previously proposed in literature at a scale larger than was previously possible. These specific architectures were selected from already published works to demonstrate the broad class of physics-based network to which the method applies. I observe similar quality results to those works, but do not claim that they are the best performing. To the best of my knowledge, this is the first demonstration of memory-efficient learning in the area of computational microscopy.
4. **An open source implementation.** The implementation of the work contained here in this chapter is available open source [1] and is further discussed in Appendix A.

## 5.2 Related Works

This work considers memory-efficient learning for a general class of physics-based networks to enable learning for large-scale computational imaging systems. Previous work can be classified under the following categories:

**Learning for Computational Imaging:** Methods can be categorized into *physics-based* and *physics-free* approaches. *Physics-based* [19–21, 23–25, 87, 88, 92–97] methods consider the inclusion of the forward model process and the structure of inverse problem optimization in the physics-based network. *Physics-free* approaches use a black box architecture (e.g. UNets [38]) to learn the decoder relationship without prior knowledge of the forward model. The former require fewer learnable parameters than the later, allowing them to be trained using less data. In addition, *physics-based* methods are more robust in experimental settings and inherit the interpretability associated with classic inverse problems. The recent work by Ongie et al. [97] has a comprehensive review of these methods.

**Invertible Learning:** Recently, invertible networks have been popularized to perform reverse-mode differentiation to save memory [98, 100–103] and model high-dimensional

---

<sup>1</sup><https://github.com/kellman/MELD>

densities [104–110]. All based on the concept of reverse recalculation [99], these methods form networks from a sequence of invertible operations, thereby alleviating the need to store intermediate variables in memory for computing gradients using backpropagation. Our method relies on the same concept of reverse recalculation to perform memory-efficient learning [99, 101] for networks composed of gradient and proximal update layers, making physics-based learning feasible for a wider variety of large-scale physics-based networks.

The work by Putzky and Welling (2019) [98] is most similar to our work. It demonstrates memory-efficient learning using a modified recurrent inference machine [95] architecture that relies on the forward model and an invertible layer with orthogonal  $1 \times 1$  convolutions for applications in MRI. With a similar in goal in mind, our work presents a more general method that does not require any significant modification to the physics-based network for invertibility and includes many options for layers (*i.e.* gradient, proximal, least-squares update layers). This will make physics-based learning more feasible for users wanting to design large-scale computational imaging systems. The storage and computation required for our method and the work in [98] are further contrasted in Sec. 5.6.

**Implicit function theorem:** Memory-efficient differentiation can be performed using implicit function theorem (IFT) when the network minimizes an objective function. Specifically, IFT can compute gradients of a network that achieves a fixed point or an optimum by differentiating its optimality equations at that point with respect to the learnable parameters. It’s usefulness has been demonstrated to differentiation through optimization problems via the Karush–Kuhn–Tucker conditions [111], for meta-learning [112, 113], to learn fixed-point methods [114], and to backpropagate through recurrent networks [115]. Physics-based networks are formed by optimization problems and thus could potentially benefit from these concepts. However, in many cases physics-based networks are stopped early prior to convergence to a fixed point due to limited computation or as a method of regularization. Using IFT also does not allow independent parameters to be learned for different layers of the network and variables associated with the optimizer itself (*e.g.* step size and acceleration rate), while computing gradients via backpropagation does.

### 5.3 Methods

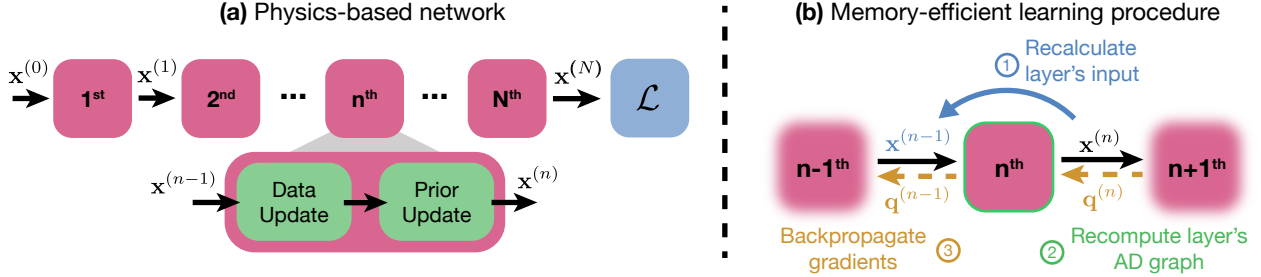


Figure 5.2: Memory-efficient learning for physics-based networks: (a) Physics-based networks (PbN) are formed by unrolling the iterations of an image reconstruction optimization. Each layer contains one iteration, made up of a data consistency update and signal prior update. The PbN input is the reconstruction’s initialization,  $\mathbf{x}^{(0)}$ , and the output is the reconstructed image from the  $N^{\text{th}}$  layer, which is fed into the learning loss,  $\mathcal{L}$ . (b) Memory-efficient learning procedure for a single layer: (1) recalculate the layer’s input,  $\mathbf{x}^{(n-1)}$ , from the output,  $\mathbf{x}^{(n)}$ , by applying that layer’s inverse operations. (2) Recompute the auto-differentiation graph for that single layer. (3) Backpropagate gradients,  $\mathbf{q}^{(n)} = \partial\mathcal{L}/\partial\mathbf{x}^{(n)}$ , through the layer’s auto-differentiation graph.

The main contribution this method is to improve the storage and computational complexity of computing gradients via backpropagation for PbNs. First, we treat the single large graph for auto-differentiation as a series of smaller graphs. Then, we rely on each physics-based layer’s invertibility to reform each smaller graph from the network’s output in reverse order. By only requiring a single layer to be stored in memory at a time, we save a factor of  $N$  in memory. By computing the smaller graphs in reverse order, we save on computation compared to other methods such as forward checkpointing.

Consider a PbN,  $\mathcal{F}$ , composed of a sequence of layers,

$$\mathbf{x}^{(n)} = \mathcal{F}^{(n)}(\mathbf{x}^{(n-1)}; \theta^{(n)}), \quad (5.1)$$

where  $\mathbf{x}^{(n-1)}$  and  $\mathbf{x}^{(n)}$  are the  $n^{\text{th}}$  layer input and output, respectively, and  $\theta^{(n)}$  are its learnable parameters. When performing reverse-mode differentiation, the method treats a PbN of  $N$  layers as  $N$  separate smaller graphs, generated on demand, processed and stored one at a time, rather than as a single large graph, thereby saving a factor  $N$  in memory. As outlined in Alg. 5.1 and Fig. 5.2, we first recalculate the current layer’s input,  $\mathbf{x}^{(n-1)}$ , from its output,  $\mathbf{x}^{(n)}$ , using  $\mathcal{F}_{\text{inverse}}^{(n)}$  (Alg. 5.1 line 3), and then form one of the smaller graphs by recomputing the output of the layer,  $\mathbf{v}^{(n)}$ , from the recalculated input (Alg. 5.1 line 4). To compute gradients, we then rely on auto-differentiation of each layer’s smaller graph to compute the gradient of the loss,  $\mathcal{L}$ , with respect to  $\mathbf{x}^{(n)}$  (denoted  $\mathbf{q}^{(n)}$ ) (Alg. 5.1 line 5) and  $\nabla_{\theta^{(n)}}\mathcal{L}$  (Alg. 5.1 line 6). The procedure is repeated for all  $N$  layers in reverse order.

---

**Algorithm 5.1** Memory-efficient learning for physics-based networks
 

---

**Inputs**  $\mathbf{x}^{(N)}$ -output of physics-based network,  $\mathbf{q}^{(N)}$ -gradient of loss with respect to state at output

**Output**  $\{\nabla_{\theta^{(n)}} \mathcal{L}\}_{n=1}^N$ -gradients with respect to learnable parameters at each layer

```

1:  $n \leftarrow N$ 
2: for  $n > 0$  do
3:    $\mathbf{x}^{(n-1)} \leftarrow \mathcal{F}_{\text{inverse}}^{(n)}(\mathbf{x}^{(n)}; \theta^{(n-1)})$ 
4:    $\mathbf{v}^{(n)} \leftarrow \mathcal{F}^{(n)}(\mathbf{x}^{(n-1)}; \theta^{(n-1)})$ 
5:    $\mathbf{q}^{(n-1)} \leftarrow \frac{\partial \mathbf{v}^{(n)}}{\partial \mathbf{x}^{(n-1)}} \mathbf{q}^{(n)}$ 
6:    $\nabla_{\theta^{(n)}} \mathcal{L} \leftarrow \frac{\partial \mathbf{v}^{(n)}}{\partial \theta^{(n)}} \mathbf{q}^{(n)}$ 
7:    $n \leftarrow n - 1$ 
8: end for
    
```

---

In order to perform the reverse-mode differentiation efficiently, our method must be able to compute each layer's inverse operation,  $\mathcal{F}_{\text{inverse}}^{(n)}$ . The remainder of this section overviews the procedures to invert gradient and proximal update layers. In addition, special treatment is given to the proximal operation performed in the least-squares update layer to highlight several computational details.

### Inverse of gradient update layer

A common interpretation of gradient descent is as a forward Euler discretization of the continuous-time ordinary differential process [37] gradient flow,

$$\frac{d\mathbf{x}^{(t)}}{dt} = -\nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}^{(t)}; \mathbf{y}) \quad \Rightarrow \quad \frac{\mathbf{x}^{(t+\alpha)} - \mathbf{x}^{(t)}}{\alpha} = -\nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}^{(t)}; \mathbf{y}). \quad (5.2)$$

With uniform steps of size  $\alpha$  the  $n^{\text{th}}$  gradient descent step is,

$$\mathbf{x}^{(n)} \leftarrow \mathbf{x}^{(n-1)} - \alpha \nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}^{(n-1)}; \mathbf{y}). \quad (5.3)$$

As a consequence, the inverse of the gradient update layer (Eq. 5.3) can be viewed as a backward Euler step,

$$\mathbf{x}^{(n-1)} = \mathbf{x}^{(n)} + \alpha \nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}^{(n-1)}; \mathbf{y}). \quad (5.4)$$

This implicit equation can be solved iteratively via the backward Euler method using a fixed point algorithm (Alg. 5.2) [37]. Convergence is guaranteed if

$$\text{Lip}(\alpha \nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}; \mathbf{y})) < 1, \quad (5.5)$$

where  $\text{Lip}(\cdot)$  computes the Lipschitz constant of its argument [116]. In the setting when  $\mathcal{D}(\mathbf{x}; \mathbf{y}) = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$  and the forward model,  $\mathbf{A}$ , is linear, this can be ensured if  $\alpha < \frac{1}{\sigma_{\max}(\mathbf{A}^H \mathbf{A})}$ , where  $\sigma_{\max}(\cdot)$  computes the largest singular value of its argument. Finally, as given by Banach Fixed Point Theorem, the fixed point algorithm (Alg. 5.2) will have an exponential rate of convergence [116].

---

**Algorithm 5.2** Inverse for gradient layer

---

**Inputs**  $\mathbf{z}$ -output of gradient descent layer,  $L$ -number of iterations

**Output**  $\mathbf{x}^{(L)}$ -estimate of gradient descent layer's input

- 1:  $l \leftarrow 0$
  - 2:  $\mathbf{x}^{(l)} \leftarrow \mathbf{z}$
  - 3: **for**  $l < L$  **do**
  - 4:      $\mathbf{x}^{(l+1)} \leftarrow \mathbf{z} + \alpha \nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}^{(l)}; \mathbf{y})$
  - 5:      $l \leftarrow l + 1$
  - 6: **end for**
- 

## Inverse of proximal update layer

The proximal update is defined as the optimization problem [37],

$$\mathbf{x}^{(n)} \leftarrow \text{prox}_{\mathcal{P}}(\mathbf{x}^{(n-1)}) \quad (5.6)$$

$$\leftarrow \arg \min_{\mathbf{v}} \frac{1}{2} \|\mathbf{v} - \mathbf{x}^{(n-1)}\|_2^2 + \mathcal{P}(\mathbf{v}). \quad (5.7)$$

For differentiable  $\mathcal{P}(\cdot)$ , the solution to Eq. 5.7 gives,

$$\mathbf{x}^{(n)} = \mathbf{x}^{(n-1)} - \nabla_{\mathbf{x}} \mathcal{P}(\mathbf{x}^{(n)}). \quad (5.8)$$

In contrast to the gradient update layer, the proximal update layer can be thought of as a backward Euler step of a continuous-time ordinary differential process [37],

$$\frac{d\mathbf{x}^{(t)}}{dt} = -\nabla_{\mathbf{x}} \mathcal{P}(\mathbf{x}^{(t)}) \quad \Rightarrow \quad \frac{\mathbf{x}^{(t+\alpha)} - \mathbf{x}^{(t)}}{\alpha} = -\nabla_{\mathbf{x}} \mathcal{P}(\mathbf{x}^{(t+\alpha)}), \quad (5.9)$$

with step size of  $\alpha = 1$ . This allows its inverse to be expressed as a forward Euler step,

$$\mathbf{x}^{(n-1)} = \mathbf{x}^{(n)} + \nabla_{\mathbf{x}} \mathcal{P}(\mathbf{x}^{(n)}), \quad (5.10)$$

when the proximal function is bijective (e.g.  $\text{prox}_{\ell_2}$ ). If the proximal function is not bijective (e.g.  $\text{prox}_{\ell_1}$ ), the inversion is not straightforward; however, in many cases we can substitute it with a bijective function with similar behavior. For example, soft thresholding, the proximal operator of  $\ell_1$  norm, is not bijective, but can be made so by adding a small slope.

### Inverse of least-squares update layer

The least-squares update is used in optimizers such as HQS and alternating direction method of multipliers (ADMM) and is more efficient than PGD in the number of unrolled layers required as it performs the complete minimization of the data consistency penalty at each iteration. When examined, this update is technically a proximal operation and thanks to its differentiability it has an exact inverse as outlined in the previous section (Sec. 5.3). We treat it separately because of its importance in many algorithms and because of efficient solution using conjugate gradient method (CG).

This update minimizes the data consistency penalty regularized by the previous estimate,  $\mathbf{x}^{(n-1)}$ ,

$$\mathbf{x}^{(n)} \leftarrow \arg \min_{\mathbf{v}} \|\mathcal{A}(\mathbf{v}) - \mathbf{y}\|_2^2 + \mu \|\mathbf{v} - \mathbf{x}^{(n-1)}\|_2^2, \quad (5.11)$$

where  $\mu$  varies the amount of regularization. Giving its name, this optimization can be solved in closed form using least-squares when the forward model,  $\mathcal{A}(\cdot)$ , is linear,

$$\mathbf{x}^{(n)} \leftarrow (\mathbf{A}^H \mathbf{A} + \mu \mathbf{I})^{-1} (\mathbf{A}^H \mathbf{y} + \frac{\mu}{2} \mathbf{x}^{(n-1)}), \quad (5.12)$$

where  $\mathbf{A}$  denotes the linear forward model. When  $\mathbf{A}$  models a linear translation invariant system, it is a circular convolution, the Fourier transform diagonalizes the model, and Eq. 5.12 can be computed in closed form by dividing by the power spectrum of the system's convolution kernel. However, computational imaging systems often form  $\mathbf{A}$  not as an explicit matrix, but as a series of operators. In this case, the inversion can be efficiently computed using the CG method.

The inverse operation of this layer (Eq. 5.12) can be expressed in closed form as

$$\mathbf{x}^{(n-1)} = \frac{1}{\mu} ((\mathbf{A}^H \mathbf{A} + \mu \mathbf{I}) \mathbf{x}^{(n)} - \mathbf{A}^H \mathbf{y}). \quad (5.13)$$

When using a CG method to perform the forward model inversion (Eq. 5.12), Eq. 5.13 is accurate only if CG performs the forward model inversion accurately. This source of numerical error is further discussed in Sec. 5.4.



## 5.4 Hybrid Reverse Recalculation and Checkpointing

Reverse recalculation of the unstored variables is non-exact, as the operations to calculate the variables are not identical to forward calculation. The result is numerical error between the original forward and reverse calculated variables. As more iterations are unrolled, numerical errors can accumulate.

To mitigate these effects, we can use checkpointing. Some of the intermediate variables can be stored from forward calculation and used in substitution for the recalculated variables, that could incur accumulated numerical errors. Memory permitting, as many checkpoints as possible should be stored to ensure accuracy while performing reverse recalculation. Due to the size of the intermediate variables, large-scale PbNs cannot afford to store all variables required for reverse-mode differentiation, but it is often possible to store a few as checkpoints.

Further, when enough iterations of the reconstruction optimization (Eq. 2.8) are unrolled, convergence of the intermediate variables can often be observed. When this occurs, inversion of each layer’s operations (Alg. 5.1 line 3) becomes ill-posed. For example, when PGD converges the gradient of the reconstruction loss will be zero, thus Alg. 5.2 will return its input and the inversion will fail.

Checkpointing can again be used to reduce these effects. If convergence behavior is observed, then checkpoints can be stored during later layers to correct inversion error. Economically, checkpoints should be placed closer together for later layers and less frequently for earlier layers (this further discussed in Sec. 4.5).

## 5.5 Results

I first demonstrate memory-efficient learning method with a compressed sensing system as an example, then with two real-world large-scale applications. In the compressed sensing example, I learn the measurement matrix to improve reconstruction performance and empirically test the method’s storage and computational complexities. In the first of the large-scale applications, I improve the image quality for 3D multi-channel compressed sensing MRI by learning better signal priors to regularize the reconstruction. In the second, I improve the temporal resolution of super-resolution microscopy (Fourier Ptychography) by learning the system’s experimental design.

### Learned measurements for compressed sensing

Compressed sensing combines random measurements and regularized optimization to reduce the sampling requirements of a signal below the Nyquist rate [117]. It has seen practical success in many fields (*e.g.* MRI [118], holography [119], optical imaging [120]). A natural question to ask is, which measurements provide the best signal recovery for a class of signals? Specifically, I recover arbitrary one-sparse signals from linear measure-

ments and learn the linear measurement matrix with a PbN. I learn a set of 7 coded 1D masks; each scalar measurement is the dot product of a mask with the signal. I optimize recovery of the signal in terms of mean square error, for a problem with relatively small dimensions and scale. This small-scale problem lets us rapidly demonstrate the accuracy of our method and compare against other methods. The PbN is constructed by unrolling PGD for the reconstruction loss,

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{x}\|_1. \quad (5.14)$$

where  $\mathbf{A} \in \mathbb{R}^{7 \times 10}$ ,  $\mathbf{x} \in \mathbb{R}^{10}$  is a one-sparse signal,  $\mathbf{y} \in \mathbb{R}^7$  is a measurement signal, and  $\lambda$  trades off the data consistency and sparsity prior penalties. The PbN is formed from 800 unrolled iterations of PGD with a step size of 0.05 and  $\lambda = 0.06$ . For the method, a modified soft thresholding function is used as the proximal operator, where a small slope (on the order of  $1e^{-6}$ ) is added to the zeroed region to make it an invertible function (as discussed in Sec. 5.3). Training was conducted for 20 epochs with 20 training data points, batch size of 4, and learning rate of  $1e^{-2}$  using ADAM [121]. 50 checkpoints are used to mitigate error due to numerical precision (as discussed in Sec. 5.4).

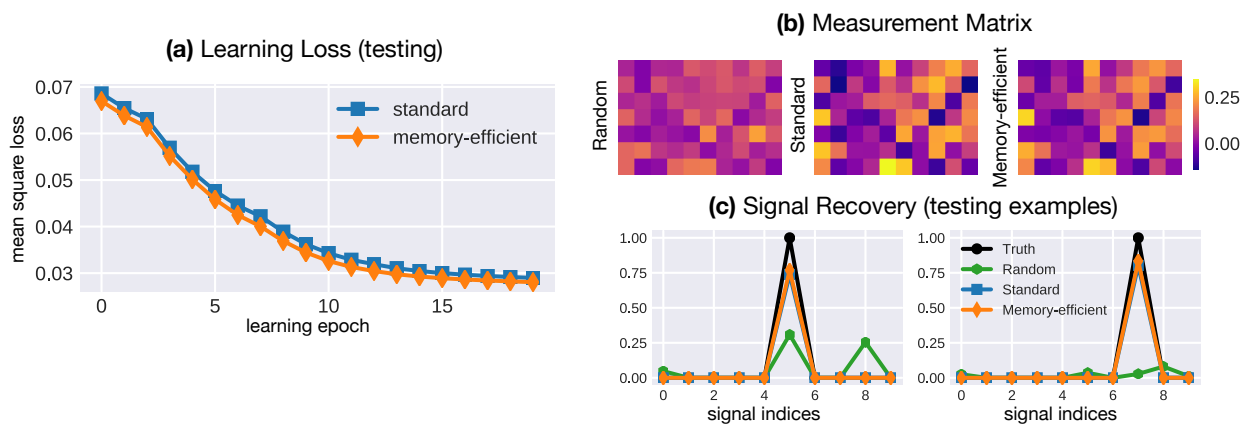


Figure 5.3: Learned measurements for 1D compressed sensing: (a) Mean testing loss for learning using standard and memory-efficient learning techniques. (b) Initial (Gaussian randomly distributed) and learned measurement matrices using standard and memory-efficient techniques. (c) Two testing examples of reconstructions with random and learned measurement schemes, demonstrating both improved signal recovery using the learned measurements in comparison to the random measurements and similarity between standard and memory-efficient learning (while requiring 4.1KB,  $\sim 800\times$  less memory than standard backpropagation).

Figure 5.3 shows a comparison between the testing loss for standard and memory-efficient learning techniques, initial random and optimized measurement matrices, and several testing data points for the ground truth with the signal recovered using the learned

and initial matrix (random Gaussian variable). As seen in Fig. 5.3c, the learned measurement matrices have better signal recovery than the random matrix. The learned measurements and signal recovery using the method and standard learning are similar, but  $\sim 800\times$  less memory is used. Where as the method uses a modified soft thresholding function, standard learning uses the ordinary version of the function. Learning results are comparable (Fig. 5.3) between the uses of the two functions, suggesting the affect of the modification is negligible; further discussion is included in Sec. 4.5.

## Learned priors for multi-channel MRI

As our first large-scale example of real-world applications, we look at MRI, a powerful medical imaging modality that non-invasively captures rich biophysical information without ionizing radiation. Since MRI acquisition time is often directly proportional to the number of acquired measurements, reducing measurements leads to immediate impact on scan time, patient throughput, and enables capturing fast-changing physiological dynamics. Multi-channel MRI is the standard-of-care in clinical systems and uses multiple receive coils distributed around the body to acquire measurements in parallel. This parallel imaging technique reduces the total number of required acquisition frames for decoding [122]. Further, scan time and noise amplification can be additionally reduced by relying on signal prior knowledge, allowing undersampling of the acquisition frames (*i.e.* with compressed sensing [118]). Recently, PbNs have been developed to learn the signal priors, achieving state-of-the-art performance for multi-channel accelerated MRI [20, 21]. However, the PbNs are limited in network size and number of unrolled iterations due to the amount of memory required for training. This is an especially prominent problem when moving to high-dimensional problems (*e.g.* 3D anatomical imaging, temporal dynamics, etc.). The proposed memory-efficient learning reduces memory footprint at training time, thereby enabling learning for larger problems.

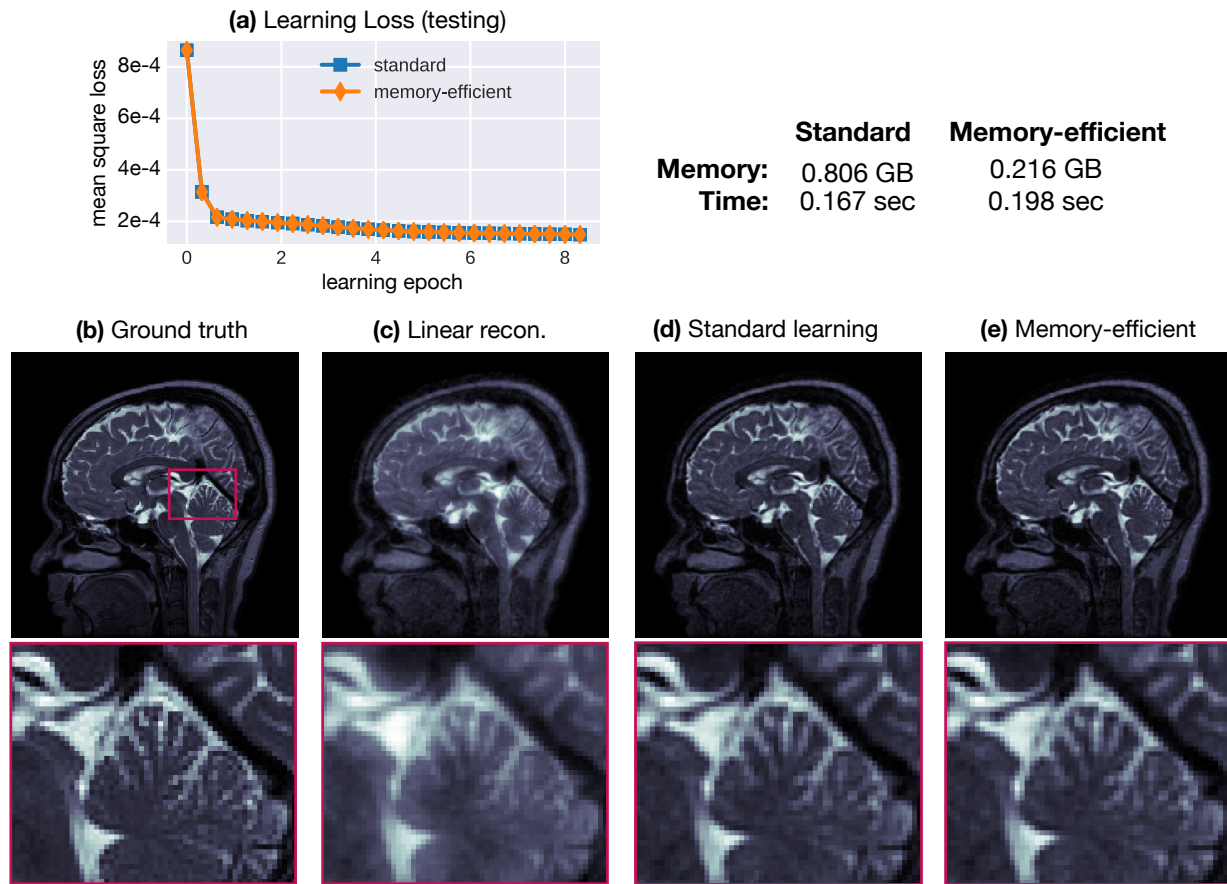


Figure 5.4: Learned priors for multi-channel 2D under-sampled MRI: (a) Mean testing loss is similar for both standard backpropagation and memory-efficient learning. (b) Ground truth reconstruction using fully sampled measurements, (c) linear parallel imaging reconstruction (no prior), (d) PbN reconstruction learned using standard backpropagation and (e) PbN reconstruction learned using memory-efficient learning ( $3.7\times$  reduced memory requirement,  $1.2\times$  increase in compute time). Insets highlight fidelity of high-resolution features and noise reduction in both of the learned designs, as compared to the CG reconstruction. Reported memory and time required is for a single learning update with batch size one.

To validate the method, we first show results for the 2D problem in [21], which has small enough memory requirements for the standard backpropagation to fit on my GPUs. The PbN is formed from 4 unrolled iterations of the HQS method and uses a learnable Resnet as the image prior [21, 23]. Specifically, the objective the PbN is minimizes

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \|\mathbf{PFSx} - \mathbf{y}\|_2^2 + \mu \|\mathbf{x} - \mathcal{R}(\mathbf{x})\|_2^2, \quad (5.15)$$

where  $\mathbf{S}$  are the multi-channel coil sensitivities,  $\mathbf{F}$  denotes Fourier transform, and  $\mathbf{P}$  is the undersampling mask used for compressed sensing. The image prior is learned using a

network,  $\mathcal{R}(\mathbf{x})$ , with the invertible residual convolutional neural network (RCNN) [101, 110, 123] architecture composed of a 5-layer CNN where each layer has 64 channels and filters of  $3 \times 3$ . The RCNN’s learnable parameters are shared between each PbN layer.

I learn to reconstruct  $256 \times 320$  slices with measurements from 8 channels and variable density Poisson Disc Fourier undersampling at a rate of  $4\times$ . Data used for training and testing is from [21], where ground truth brain images are used and data is synthetically generated given the sensitivity and undersampling masks. Training was conducted for 10 epochs with 350 training data points and 10 testing data points, a batch size of 4, and a learning rate of  $1e^{-5}$  using ADAM [121]. In Fig. 5.4 we compare image reconstructions using the priors learned by standard and memory-efficient learning. As shown in Fig. 5.4a, testing losses and image reconstruction quality are similar for both methods (Fig. 5.4d,e), but our method uses  $4.82\times$  less memory, while only requiring a  $1.09\times$  increase in time.

Finally, I demonstrate the proposed method’s ability to learn priors for a 3D volume reconstruction from under-sampled multi-channel measurements - a problem that does not typically fit within standard GPU memory limits. Specifically, I reconstruct volumes of  $50 \times 256 \times 320$  with measurements from 8 channels and variable density Poisson Disc undersampling at a rate of  $4\times$ . Data used is from [21] and is augmented to create more training examples by cropping down larger volumes to  $50 \times 256 \times 320$ . We use a similar PbN architecture as before for the reconstruction and training parameters, but now with a RCNN with 3D filters ( $3 \times 3 \times 3$ ) and 32 channels. This model would ordinarily require  $\sim 40$ GB of memory using standard backpropagation ( $\sim 10$ GB per unrolled iteration), but only requires  $\sim 10$ GB of memory using our method. In Fig. 5.5 we show results of the learning loss and a single slice of the reconstructed volumes from the ground truth (fully sampled), conjugate gradient (no learning or signal prior), and after learning priors with our memory-efficient learning scheme.

To further motivate the need for large-scale models, I perform an analysis of performance (PSNR) versus number of unrolled iterations (Fig. 5.6). While the return diminishes as the number of unrolled iterations increases, performance continues to increase past 20 iterations. In this setting, when even single-layer and checkpointing require approximately 12GB of memory, the method will be more computationally efficient than other memory-friendly methods such as forward checkpointing (further discussed in Sec. 5.6).

## Learned experimental design for Fourier Ptychographic Microscopy

In this section, I revisit the problem of learning illumination patterns for Fourier ptychographic microscopy (FPM) (Chap. 4). As highlighted in Chap. 4 Sec. 4.3, 100s gigabytes to terabytes of memory is required to perform physics-based learning for FPM at higher factors of super resolution. Here, I show that the proposed memory-efficient learning framework reduces the necessary memory only a few gigabytes, thereby enabling full-scale learning on a consumer-grade GPU.

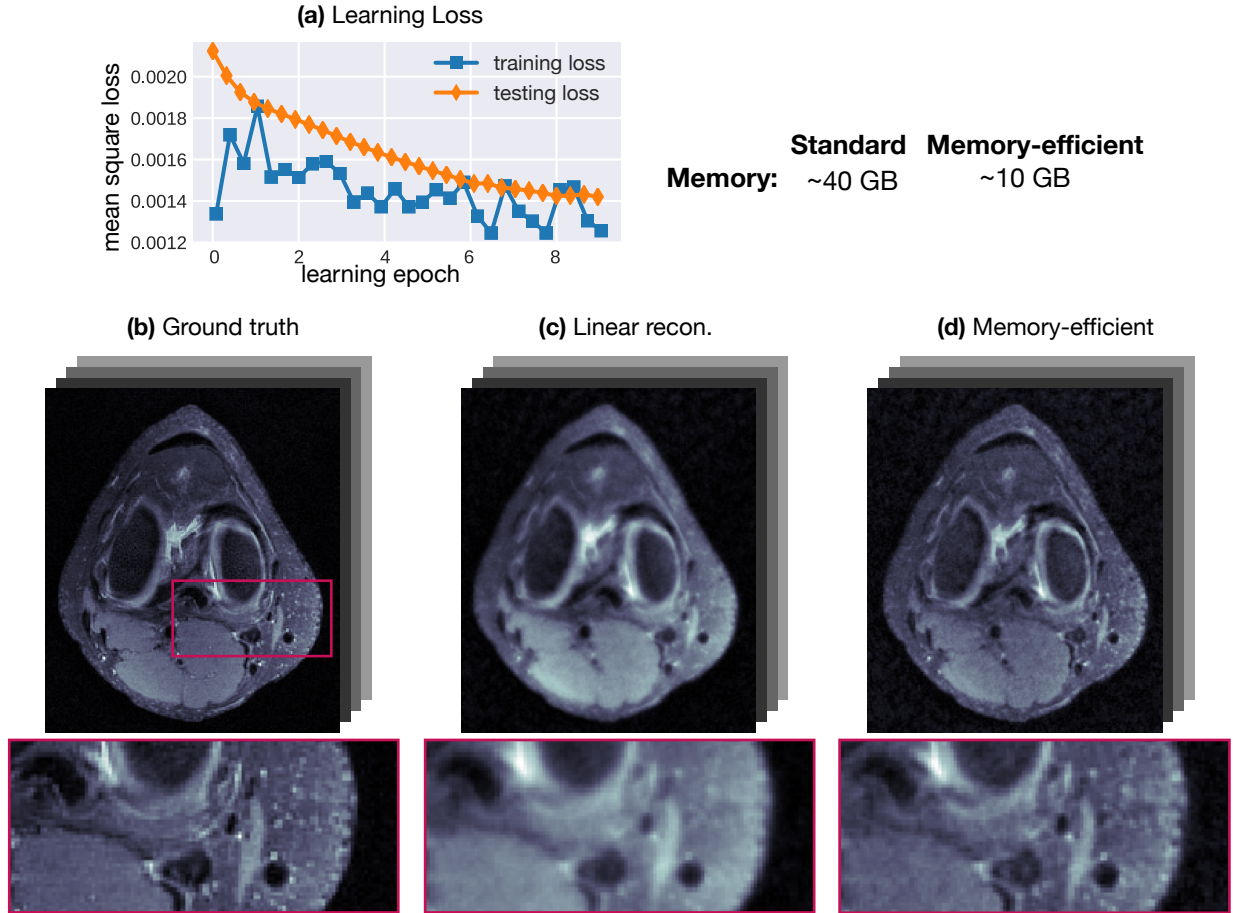


Figure 5.5: Learned priors for multi-channel under-sampled 3D MRI: (a) Mean training and testing loss for learning with the proposed memory-efficient technique. (b) One slice of ground truth 3D reconstruction using fully sampled measurements, (c) linear parallel imaging reconstruction (no prior), (d) PbN reconstruction using memory-efficient learning with  $\sim 10$ GB of memory. Standard learning is not shown because it requires more memory than would fit on a commercial GPU.

As detailed in Chap. 4, the PbN for learning FPM LED source patterns is formed from the following phase retrieval optimization:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \sum_{k=1}^K \left\| \mathbf{y}_{m_k} - \sum_{l=1}^L c_{kl} |\mathbf{A}_l \mathbf{x}|^2 \right\|_2^2, \quad (5.16)$$

where  $\mathbf{y}_{m_k}$  is the  $k^{\text{th}}$  multi-LED measurement,  $\mathbf{A}_l = \mathbf{F}^H \mathbf{P}_l \mathbf{F}$  is the forward model for the  $l^{\text{th}}$  LED [1, 25],  $\mathbf{P}_l$  is the microscope's pupil function for the  $l$  LED,  $\mathbf{F}$  denotes 2D Fourier transform, and  $c_{kl}$  is the learnable brightness for the  $l^{\text{th}}$  LED in the  $k^{\text{th}}$  measurement. A PbN is formed from  $N$  unrolled iterations of gradient descent. I then minimize the loss

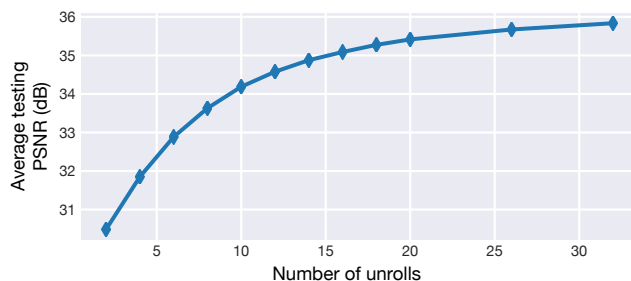


Figure 5.6: Multi-channel MRI performance versus number of unrolls: Average testing PSNR for varying number of unrolled iterations. In this example we learn independent parameters for each layer.

between the output of the PbN and the ground truth to learn LED brightnesses over the dataset.

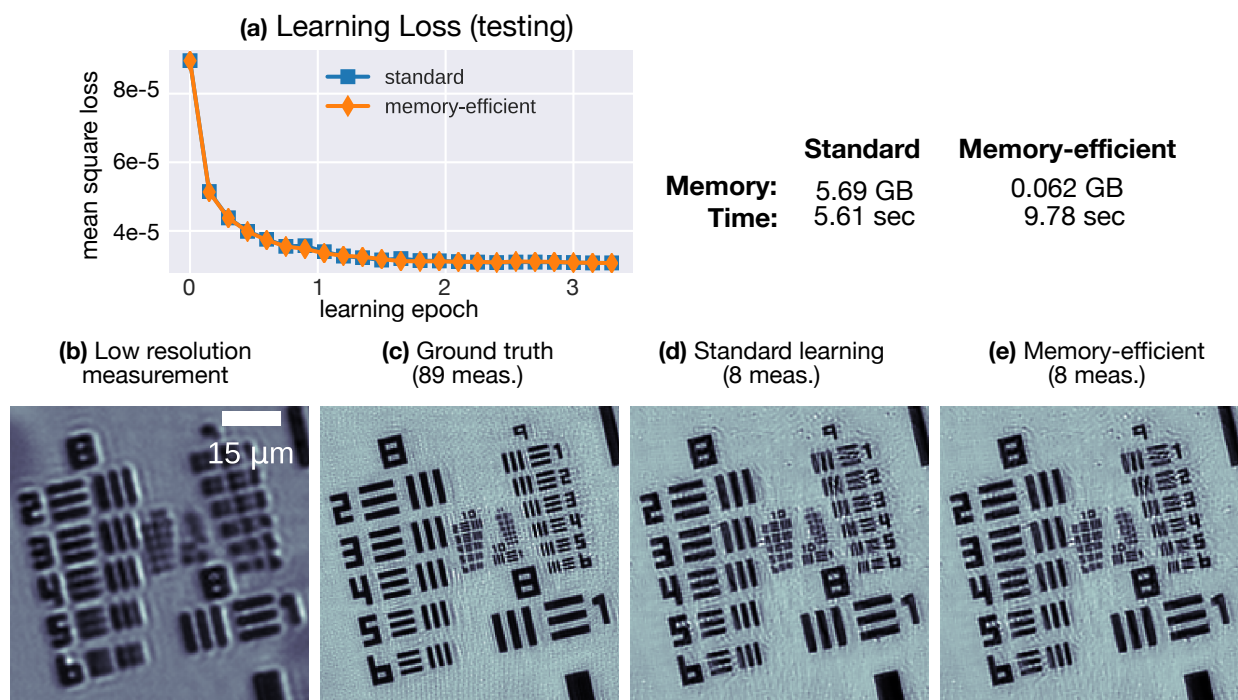


Figure 5.7: Learned illumination design for Fourier Ptychographic Microscopy (FPM): (a) Mean testing loss is similar for both standard backpropagation and memory-efficient learning. (b) Example low-resolution measurement, (c) ground truth reconstruction using all 89 LED measurements to perform  $3.1\times$  super resolution, (d) reconstruction from only 8 measurements learned using standard backpropagation and (e) memory-efficient learning ( $92\times$  reduced memory requirement,  $1.7\times$  increase in compute time). Reported memory and time required is for a single learning update with batch size one.

I again start by validating the method’s accuracy on a small-scale problem that fits in GPU memory using standard learning. I reproduce results in Chap. 4, learning illumination patterns for eight measurements, which gives  $3.1\times$  resolution improvement and  $10\times$  faster data capture. I set  $L = 4$ , the number of fixed point iterations to invert gradient layers, and checkpoints every 10 unrolled iterations. The testing loss between the proposed method and standard learning are similar (Fig. 5.7a), and the SR reconstructions with learned designs using standard (Fig. 5.7d) and memory-efficient (Fig. 5.7e) methods are both similar to the ‘ground truth’ reconstruction using 89 measurements (Fig. 5.7c). The memory-efficient learning approach, however, reduces memory required from 5.69GB to 0.062GB, with compute time increasing by less than a factor of  $2\times$ . Hence, the proposed method produces comparable quality results as the standard learning, but with significantly reduced (more than  $91\times$ ) memory requirements.

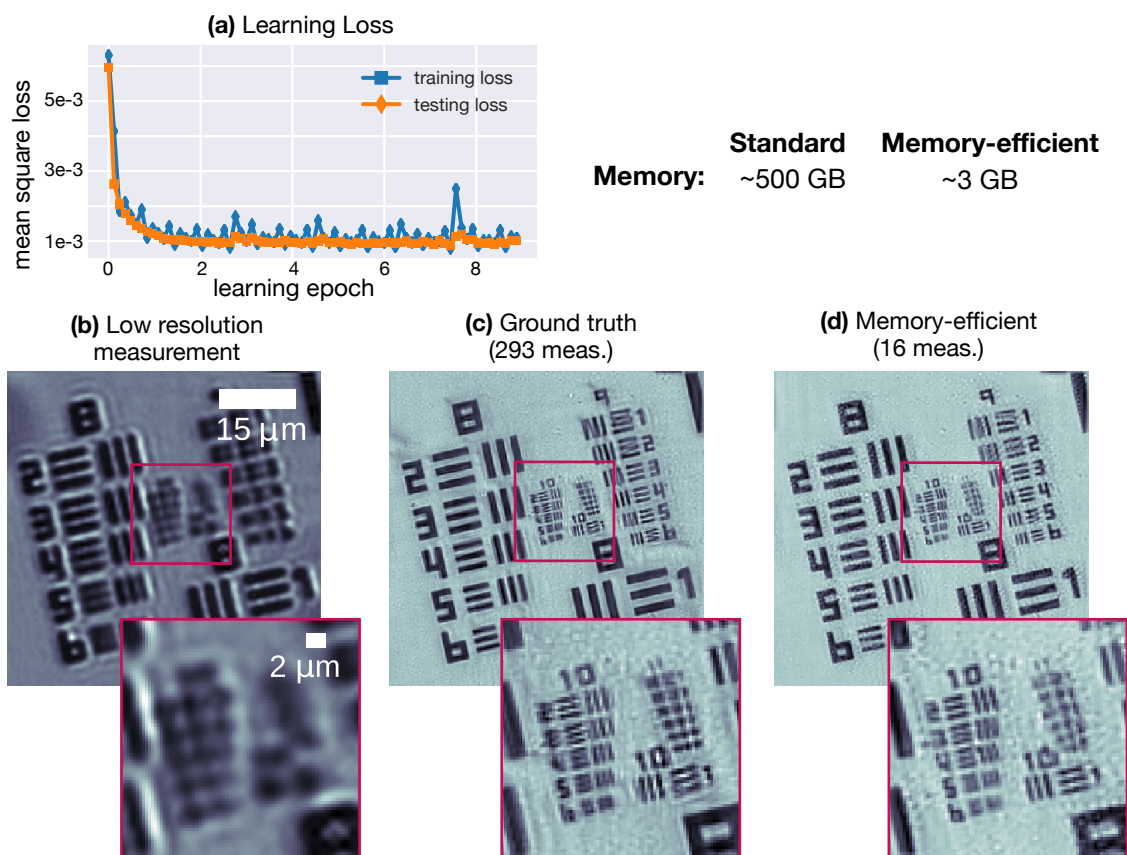


Figure 5.8: Large-scale learned illumination design for FPM: (a) Training and testing loss for memory-efficient learned design. (b) Example low-resolution measurement, (c) ground truth reconstruction using all 293 LED measurements to perform  $4.2\times$  super resolution, (d) reconstruction from only 16 measurements learned using memory-efficient learning with  $\sim 3\text{GB}$  memory. Standard learning is not shown because it would require  $\sim 500\text{GB}$  of memory, which is not available on commercial GPU. Insets highlight high-resolution features.



Next, I use the proposed memory-efficient learning scheme to solve a larger-scale problem than was previously possible. For FPM, that means using all 293 LEDs to achieve a higher factor of super resolution ( $4.2\times$ ). 200 iterations are unrolled to create the PbN, I set  $L = 4$ , and checkpoints every 13 unrolled iterations. For this problem, standard backpropagation would require  $\sim 500\text{GB}$  of memory, while the proposed method only requires  $\sim 3\text{GB}$  (using 15 checkpoints). In Fig 5.8, I demonstrate the learned design’s ability to reduce the number of measurements required from 293 to 16, demonstrating  $20\times$  faster data capture with comparable image quality to ground truth.

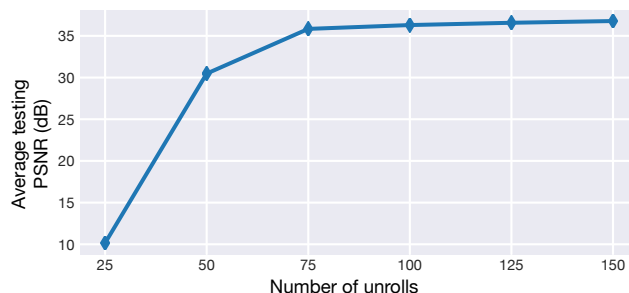


Figure 5.9: Fourier Ptychographic Microscopy performance versus number of unrolls: Average testing PSNR for varying number of unrolled iterations.

In this example, the number of unrolled iterations is determined by the conditioning of the problem as aspects of the reconstruction are not learned (as they are in Sec. 5.5). I perform an analysis of performance (PSNR) versus number of unrolled iterations (Fig. 5.9) and show that for this problem at least 100 iterations must be unrolled before diminishing returns in performance. This analysis is in the context of this level of super resolution and the number of measurements learned. As the degree of super resolution grows or the number of measurements is decreased, the problem will become worst conditioned and more unrolled iterations will be required.

## 5.6 Memory-Computation Analysis

In Sec. 4.4 I demonstrated several example uses of the proposed method, each representing a single point in the storage-computation trade-off space. Here, I provide analysis to determine when our method provides advantage over standard backpropagation and forward checkpointing. I visualize the complete storage-computation space and calculate when each method is best (the fastest method that accommodates the memory required). Specifically, I visualize the relationship between storage-computation for varying input sizes (referred to as checkpoint sizes), varying physics-based layer sizes, varying numbers of unrolls, and varying amounts of computation required.

First, I calculate the memory and computation time requirements for all three methods. For standard backpropagation the memory required is  $N \times A + B$ , where  $N$  is the

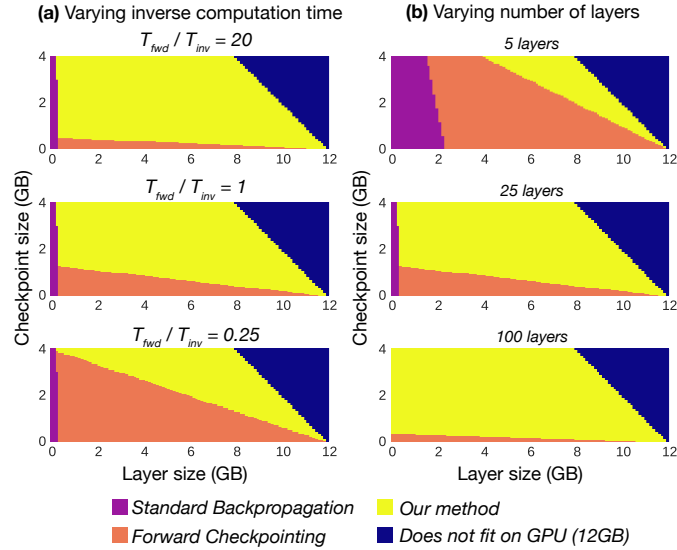


Figure 5.10: Fastest method for varying layer and checkpoint size: (a) Fastest method for three scenarios where the ratio between forward and inverse layer computation varies. The proposed method performs better with larger ratios because it calls each layer’s inverse more and forward checkpointing calls each layer’s forward more. (b) Fastest method for three scenarios with increasing number of unrolled iterations. The proposed method performs better as more iterations are unrolled. Sector color corresponds to the fastest method: purple, orange, yellow, and blue representing standard backpropagation, forward checkpointing, my method, and problems that do not fit on a single GPU (12 GB), respectively.

number of layers,  $A$  is the memory for a single physics-based layer, and  $B$  is the memory for a single data input. The time required is  $N \times (T_{fwd} + T_{bck})$ , where  $T_{fwd}$  and  $T_{bck}$  are the computation times for running a layer forward and backwards, respectively. Forward checkpointing stores as many checkpoints as memory affords. Once there are more layers than available memory for checkpoints, layers are recomputed from the previous closest checkpoint. Because backpropagation is performed in the reverse order of the forward pass, it is computationally expensive to recompute layers when the number of layers far exceeds the number of checkpoints. When checkpoints are stored every  $K$  iterations the computational time required is  $N \times (T_{fwd} + T_{bck}) + \frac{N(K+1)}{2} \times T_{fwd}$ . Finally, the proposed method only requires  $A + B$  in memory and  $N \times (2T_{fwd} + T_{bck} + T_{inv})$ , where  $T_{inv}$  is the time to invert each layer’s operations.

Both forward checkpointing and my method are memory-friendly, but the computation time they require varies drastically based on how many checkpoints can be stored and the computational cost of performing a layer’s forward versus its inverse operations. Once forward checkpointing cannot store a single checkpoint per layer, its computation time grows quadratically, while my method remains linear. Further, forward checkpointing relies on the evaluation of layers’ forward operations, while my method relies on the

evaluation of layers' inverse operations.

Figure 5.10 illustrates when each method is fastest for varying size physics-based layers and checkpoint sizes (determining how many checkpoints can fit in memory). This analysis is done using a GPU with 12GB of memory. Figure 5.10a shows the fastest choice of method for three different cases: when inverse layer computation is faster, the same speed, and slower than forward layer computation. The proposed method (yellow) spans a larger area of the trade-off space than forward checkpointing (orange) and standard backpropagation (purple) when inverse layer evaluation is faster. This is the case when evaluating proximal and least-squares update layers, often implemented using iterative methods such as conjugate gradient, but whose inverse can be expressed in closed form. For this experiment the number of unrolls is held constant with a value of 25. Figure 5.10b shows which method is fastest for varying size problems as the number of unrolled iterations varies. As more iterations are unrolled, forward checkpointing can store relatively fewer checkpoints and thus slows relative to the proposed method. For this experiment  $T_{fwd} = T_{inv}$ .

The work of Putzky and Welling [98] ensures invertibility by passing two intermediate variables into each layer requiring double a checkpoint's memory size in storage. The proposed method only requires a single variable to be stored to ensure invertibility, thereby enabling larger physics-based networks. The computation time required in [98] to perform inversion for each layer is equal to the forward evaluation time,  $T_{fwd}/T_{inv} = 1$ . For the proposed method, the amount of computation required for a layer's inversion varies depending on its architecture. For gradient layers, the typical amount of computation is  $4\times$  slower than its forward evaluation,  $T_{fwd}/T_{inv} < 1$ . For proximal layers, iterative in nature, the inverse can often be expressed in closed form, *i.e.*  $T_{fwd}/T_{inv} > 1$ .

## 5.7 Remarks

### Discussion

The proposed memory-efficient learning opens the door to using unrolled physics-based networks for learning the design of large-scale computational imaging systems that are not otherwise possible due to GPU memory constraints, without a significant increase in training time. Within this work I detail how physics-based networks composed of gradient and proximal update layers can be reversible to allow for memory-efficient gradient computation. While I have demonstrated our procedure for PbNs formed from PGD and HQS methods, the update layers I describe form the fundamental building blocks of many larger PbNs (*e.g.* unrolling the updates of alternating minimization).

For the proposed method, the physics-based network must be invertible. To achieve this at the layer level, sufficient conditions for invertibility must be met. For gradient update layers, this comes in the form of a Lipschitz constant constraint (Eq. 5.5). At the network level, the convergent behavior of physics-based networks and reconstruction

optimization (Eq. 2.8) makes accurate reverse recalculation ill-posed and can cause numerical error accumulation (as outlined in Sec. 5.4). This is not an issue for many PbNs as they truncate the number of unrolled iterations prior to the optimizer’s convergence as an additional form of regularization (*i.e.* early stopping) or to save computation. In the case when convergent behavior is observed, checkpoints should be used. A possible option is to measure the difference between successive intermediate variables on the forward pass of the network. If that quantity falls below a threshold, then the optimization is approaching convergence and checkpoints should be placed to mitigate the accumulation of error on the reverse pass.

In some situations the relationship between storage and computational complexity can be traded off with accuracy. For gradient descent layers, the fixed-point method outlined in Alg. 5.2 is used to invert and, if not run to convergence, the inversion will be less accurate. When the Lipschitz constant of the gradient operator is large, more iterations (a larger value of  $L$ ) will be required to accurately invert the layer. Unfortunately, the ideal Lipschitz constant for a gradient descent layer is larger. Practically, we find that only a few (*e.g.* 4 to 8) iterations are required. For proximal layers, the inversion is accurate up to numerical precision, but requires the iterative forward process of the layer to be computed accurately for the proposed method to be accurate.

In Sec. 5.5 a modified soft thresholding function is used in place of the proximal operator for the  $\ell_1$ . While results in Fig. 5.3 suggest the effect of this change is negligible, the performance of the reconstruction could be reduced to allow for the invertibility of the operation (Sec. 5.3) and use of the proposed method. Depending on the slope added the soft thresholding function, the performance and invertibility are traded off. When the slope is very small (on the order of machine epsilon), the performance of the reconstruction will behave similar to the ordinary function, however, it will be less invertible due to floating point quantization. When the slope is larger, the reconstruction performance could be reduced because the operator does not well model the original proximal function, but will be more linear, thus be less affected by quantization and more invertible.

Acceleration layers to improve convergence of image reconstruction are commonly used in variants of PGD (termed FISTA [124]) and can be incorporated into the proposed framework. Typically, such layers linearly combine the output of the current and previous layers, so inherently, the acceleration layer cannot be inverted from only the current layer’s output. However, with the storage of additional information (this layer’s output and the previous layer’s output) it is possible to invert an acceleration layer by computing the inverse of a  $2 \times 2$  matrix.

A limitation of our method is when each smaller auto-differentiation graph (discussed in Sec. 5.3) is still too large to fit in memory. In this situation more context-specific solutions (*e.g.* coil compression for multi-channel MRI, using a smaller field-of-view) or more efficient implementation of the system’s fundamental operations is required.

## **Conclusion**

Memory-efficient learning with physics-based networks is a practical tool for large-scale computational imaging problems. Using the concept of reversibility, I implement reverse-mode differentiation with favorable storage and computational complexities. I demonstrated the proposed method on several representative large-scale applications: 3D multi-channel compressed sensing MRI and super-resolution optical microscopy, and expect many other computational imaging systems to fall within my framework.

# Chapter 6

## Conclusion

### 6.1 Outlook

Physics-based learned design allows us to rely on data to learn aspects of a computational imaging system that we do not understand and to hard code parts that we do understand. The benefits of this are several fold. The design learned is optimized to directly improve the performance of the system. Less training data is required because the physics-based network has fewer learnable parameters than its physics-free counterpart. The results are more interpretable because the network's architecture is constructed from the structure of an inverse problem's optimization problem.

The demand for the methods presented in this dissertation (*physics-based learned design* and *memory-efficient learning*) will continue to increase as future computational imaging systems grow in size and dimension. Even now, examples of larger-scale systems exist: extreme MRI [125] and XD-GRASP [126] in the area of medical imaging, 3D fluorescence microscopy [127, 128] and optical diffraction tomography [129] in the area of biological imaging, and hyper-spectral imaging in the area of remote sensing. As these systems scale in size, it will become ever more difficult to capture large datasets or complete ground truth to supervise physics-based learning methodologies. To continue, new unsupervised methods will be required to perform learning in the field of computational imaging.

### 6.2 Future work

Beyond the topics I have detailed in the previous chapters, the next few sections outline several future ideas and their challenges.

## Task-based computational imaging

Computational imaging seeks to extract information that is ordinarily inaccessible. Throughout this manuscript my outlook, and I also believe most other image researchers' outlook, is to produce an image that contains that information which a human will then "read". This will suffice when we design a general purpose imaging system, but when we design a task-specific imaging system, I believe we can achieve better. Typically, computational imaging systems reconstruct information as pixels and are designed to improve each pixel's SNR and resolution, but what if we could design an imaging system to perform that task directly. How will this effect the design of the imaging system? Could that imaging system be made to be more efficient?

Prior to learning in the field of computational imaging, data-driven methods have seen huge success in performing higher-level tasks such as classification, detection, and segmentation. These systems do not ordinarily consider how or where the images are captured or come from and thus require immense training sets to learn these relations. With physics-based learning, we now understand how to include critical aspects of the imaging system into the learning pipeline and using the same learning mechanisms and the details of this manuscript we can learn how best to design an imaging system for a particular task. This is currently becoming a popular idea in the area of photography [130] and microscopy [131, 132].

The main challenge I envision regarding this trajectory of research is the over optimization of the system. Today, imaging systems are built to be general and as a consequence are robust in many scenarios. The idea of task-based imaging goes the other way, designing the system to image a specific class of samples or specimens. If in reality that particular class of sample or specimen is altered, then the performance of such a system could be significantly reduced. One potential remedy for this situation is to still require the imaging system to produce reasonable images as an intermediate step.

## Implicit differentiation for memory-efficient auto-differentiation

Memory-efficient learning (introduced in Chap. 5) has enabled the practical data-driven design of large-scale computational imaging systems using commercially-available hardware. The method relies on the invertibility of the physics-based network. As discussed in Chap. 5, if the iterates of the optimizer that form the physics-based network converge, the layer's lose their invertibility properties. As a solution, we propose the use of checkpoints to mitigate these errors, but there is no theory as to where to place them. Fortunately, backpropagation is not the only way to compute gradients for learning.

The backpropagation procedure relies on the mechanics of chain rule to compute gradients for learning. This is accomplished by performing a series of Jacobian vector products (JVP). For a physics-based network,

$$\nabla_{\theta} \mathcal{L}(\theta) = \frac{\partial \mathbf{x}_{\theta}^{(N)}}{\partial \theta} \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{x}_{\theta}^{(N)}}, \quad (6.1)$$

where  $\mathcal{L}$  is the learning loss and  $\theta$  are the learnable parameters, is the first application of chain rule. When the outlined problem occurs, it is the red highlighted partial derivative that is difficult to compute. Instead of computing it, my proposed solution relies on  $\mathbf{x}^{(N)} = \mathbf{x}^*$  and differentiation of this physics-based network's optimality equations to compute this derivative (termed *implicit differentiation*).

To simplify the inverse problem formulation from Chap. 2,

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}, \mathbf{y}), \quad (6.2)$$

let  $\mathcal{R}$  encapsulates both the data consistency and signal prior penalties and  $\theta$  are the user-specified learnable parameters. Once solved, the optimum (or referred to as a fixed point),  $\mathbf{x}^*$ , must satisfy primal optimality,

$$\nabla_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}^*, \mathbf{y}) = \mathbf{0}, \quad (6.3)$$

where  $\mathbf{0}$  is the zero vector. Differentiating the above equation with respect to  $\theta$ ,

$$\frac{\partial \nabla_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}^*, \mathbf{y})}{\partial \theta} = \mathbf{0} \quad (6.4)$$

$$\frac{\partial \nabla_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}^*, \mathbf{y})}{\partial \mathbf{x}_{\theta}^*} \frac{\partial \mathbf{x}_{\theta}^*}{\partial \theta} + \frac{\partial \nabla_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}^*, \mathbf{y})}{\partial \theta} = \mathbf{0} \quad (6.5)$$

$$\frac{\partial \nabla_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}^*, \mathbf{y})}{\partial \mathbf{x}_{\theta}^*} \frac{\partial \mathbf{x}_{\theta}^*}{\partial \theta} = - \frac{\partial \nabla_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}^*, \mathbf{y})}{\partial \theta}, \quad (6.6)$$

reveals a new equation with access to  $(\partial \mathbf{x}_{\theta}^* / \partial \theta)$ . Solving for the highlighted term amounts of inverting  $(\partial \nabla_{\mathbf{x}} \mathcal{R}_{\theta}(\mathbf{x}^*, \mathbf{y}) / \partial \mathbf{x}_{\theta}^*)$  and can accomplished in many cases using the conjugate gradient method. Once computed, Eq. [6.1](#) can be used to form the desired gradients and learning can commence.

If the physics-based network has converged, implicit differentiation offers the ability to compute derivatives without needing to store any intermediate variables. The main challenge of this approach will be implementation and integration with standard learning libraries (e.g. PyTorch and Tensorflow).



## Unsupervised physics-based learning

In many scenarios training data that contains the ground truth decoded information is unavailable. This will inhibit the use of most supervised learning techniques, however, large amounts of measurements (sometimes incomplete) exist and this opens the door to unsupervised learning approaches. Throughout this dissertation the presented methods have been paired with supervised learning techniques, but I am optimistic of their usefulness when combined with unsupervised techniques.

Intrinsically, image reconstruction is not supervised by ground truth data, but rather guided by the knowledge of the forward model process and supervised by measurements. Image reconstruction uses the forward model process to estimate information by penalizing its inconsistency with the measurements (*i.e.* data consistency). This concept should be reused. Rather than penalize an estimate of some information with ground truth (supervised learning), penalize that estimated quantity in the measurement domain. The learning loss,

$$\mathcal{L}(\theta) = \|\mathcal{A}(\mathbf{x}_\theta^{(N)}) - \mathbf{y}\|_2^2, \quad (6.7)$$

does not rely on having ground truth,  $\mathbf{x}'$ , but uses the same measurements,  $\mathbf{y}$ , used as input to the network to supervise learning. This concept has been recently demonstrated in the field of MRI [133-135], as well as for the problem of image denoising [136-138].

The fundamental difficulty with this concept occurs in two places. First, when the measurements are noisy. This injects noise into the learning process. Second, when the inverse problem is underdetermined (*e.g.* in compressed sensing) and the forward model process has a non-trivial null space. Physics-based learning can be viewed as reducing the size of the forward model's null space (*i.e.* experimental design) or finding the best solution of many plausible (*i.e.* signal prior). When learning is penalized through the forward model's transform two estimates could result in equal learning loss even though one is much better than the other. In both cases, this will be create difficulty reducing the learning loss to match that of supervised learning methods.

# Appendix A

## Open Source: How to implement physics-based learning

### A.1 Introduction

This chapter overviews two implementations of physics-based learning: one a basic tutorial that prioritizes rapid prototyping and another more advanced implementation that incorporates memory-efficient learning. The tutorial demonstrates how to use several Pytorch mechanisms to construct physics-based networks and to perform physics-based learning, while only require an implementation of the forward model process. The advanced implementation is for the more expert user that knows their system already works and is ready to learn for larger-scale versions that would require memory-efficient learning or use more specific physics-based networks (*e.g.* half quadratic splitting).

### A.2 Basic Tutorial

The goal of this tutorial is to explain step-by-step how to implement physics-based learning for the rapid prototyping of a computational imaging system. Specifically, I advocate exploiting the auto-differentiation functionality [99] twice, once to build a physics-based network and again to perform physics-based learning. Thus, the user need only implement the forward model process for their system, speeding up prototyping time. I provide an open-source Pytorch [91] implementation [1] of a physics-based network and training procedure.

For this demonstration, I implement physics-based learning on a compressed sensing problem – under-determined sparse recovery from linear Gaussian random measurements. The sparse signals have normally-distributed amplitude values. The learnable parameters in the physics-based network will be the measurement matrix, the sparsity

---

<sup>1</sup>[https://github.com/kellman/physics\\_based\\_learning](https://github.com/kellman/physics_based_learning)

prior penalty, and the step size of the optimizer. Specifically, I solve the  $\ell_1$  relaxation of the sparse recovery problem,

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \underbrace{\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2}_{\mathcal{D}(\mathbf{x};\mathbf{y})} + \lambda \|\mathbf{x}\|_1, \quad (\text{A.1})$$

where  $\lambda$  is the sparsity prior penalty that trades off the penalty of the prior with the data consistency term, and  $\mathbf{A} \in \mathbb{R}^{J \times K}$  is the linear measurement matrix. I use the proximal gradient descent algorithm (Alg. A.1) to solve the optimization problem (Eq. A.1) and to form the architecture of the physics-based network.

---

**Algorithm A.1** Proximal Gradient Descent

---

**Inputs**  $\mathbf{x}^{(0)}$ -initialization,  $\alpha$ -step size,  $N$ -number of iterations,  $\mathbf{y}$ -measurements

**Output**  $\mathbf{x}^{(N)}$ -final estimate of image

- 1:  $n \leftarrow 1$
  - 2: **for**  $n < N$  **do**
  - 3:      $\mathbf{z}^{(n)} \leftarrow \mathbf{x}^{(n-1)} - \alpha \nabla_{\mathbf{x}} \mathcal{D}(\mathbf{x}^{(n-1)}; \mathbf{y})$
  - 4:      $\mathbf{x}^{(n)} \leftarrow \text{soft\_thr}_{\alpha\lambda}(\mathbf{z}^{(n)})$
  - 5:      $n \leftarrow n + 1$
  - 6: **end for**
- 

## Physics-based network

With some extra Pytorch specific flags and functionalities, the physics-based network (Fig. A.1) mirrors the basic structure of Alg. A.1. In implementation, it requires all learnable parameters as input (measurement matrix, step size, and sparsity penalty), initialization, ground truth, and number of iterations (or depth of network). As output, the network returns the final estimate of the sparse recovery optimization.

The specific Pytorch flags and functionalities that enable us to properly use the automatic differentiator for sparse recovery and for physics-based learning are contained in lines 14, 15, and 26. Lines 14 and 15 set up the initialization to the network, first in line 14 detaching it from any previous automatic differentiation graphs and second in line 15 setting its `requires_grad` field to be `True`. This will allow the automatic differentiation to track operations related to  $\mathbf{x}$  for taking derivatives of  $\mathcal{D}(\mathbf{x}; \mathbf{y})$  with respect to  $\mathbf{x}$ . Line 26 sets the `create_graph` flag of the automatic differentiator to `True`. This tells Pytorch to store these operations so that they can be traced through by a second automatic differentiator for computing derivatives with respect to learnable parameters at training time.

While some physics-based networks take in measurements as input, this is often not possible when learning a system’s experimental design. Here, in line 17, the measurements are synthesized using the learnable measurement matrix and ground truth sparse vector.

```
1 def pbnet(A, alpha, lamb, x0, xt, K, testFlag=True):
2     # pbnet - Physics-based Network
3     # args in -
4     # A - measurement matrix
5     # alpha - step size
6     # lamb - sparsity penalty
7     # x0 - initialization
8     # xt - ground truth sparse vector
9     # K - number of layers (i.e. iterations)
10    # testFlag - disables training (default True)
11    # args out -
12    # x - output of network (final estimate)
13
14    x = x0.detach().clone()
15    x.requires_grad = True
16
17    y_meas = Aop(A, xt)
18    if testFlag: y_meas = y_meas.detach()
19
20    for kk in range(K):
21        y_est = torch.matmul(A, x)
22        res = y_est - y_meas
23        loss_dc = torch.sum(res**2)
24        g = torch.autograd.grad(loss_dc,
25                                x,
26                                create_graph = not testFlag)
27
28        x = x - alpha*g[0] # gradient update
29        x = softthr(x, lamb*alpha) # proximal update
30    return x
```

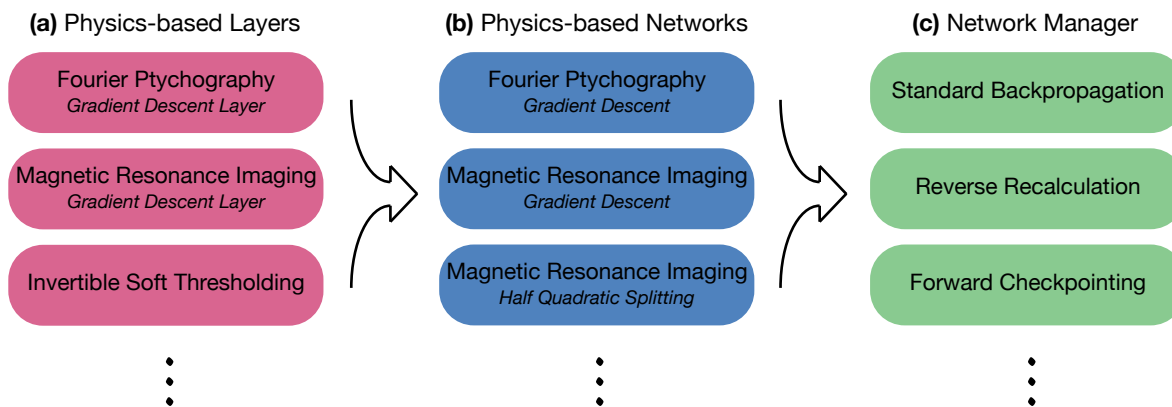
Figure A.1: Implementation of a physics-based network for compressed sensing sparse recovery using automatic differentiation to compute gradients with respect to  $x$ . The network is setup to learn the measurement matrix, step size, and sparsity penalty. Soft thresholding is used as the proximal operator.

## Physics-based Learning

The mechanics of training a physics-based network are similar to that of training any neural net or convolutional neural network in that it relies on a dataset, an optimizer, and automatic differentiation to compute gradients. In our particular application, our dataset consists of sparse vectors with amplitudes that are normally distributed. As for the optimizer, we use the adaptive moment estimation (Adam) [121] algorithm to perform the gradient updates. For researchers that are interested in learning for large-scale computational imaging systems, automatic differentiation will require more memory than available on commercial graphical processing units. To enable learning at these scales, vanilla automatic differentiation can be replaced with memory-efficient techniques [139] as outlined in Chapter 5 and the following section.

## A.3 Advanced Implementation

This section is for more expert users that want to use physics-based learning to design large-scale systems. It overviews how to use the memory-efficient learning framework



figures-03.png

Figure A.2: Memory-efficient Learning Framework: (a) physics-based layers encode the forward model process into the each layer’s operations, (b) physics-based networks are formed from a list of physics-based layers to perform the image reconstruction, and (c) network managers handle how to compute gradients using backpropagation through the physics-based networks.

(concepts introduced in Chapter 5). The main difference that sets this implementation apart from the tutorial one is that it no longer relies on using the auto-differentiator to compute image reconstruction gradients. The implementation of memory-efficient learning hinges on turning off the auto-differentiation for the forward computation of the network and it becomes difficult to use it for individual layers’ operations. The code for this memory-efficient learning framework is available open source [\[1\]](#)

## Framework Overview

The framework is comprised of three layers of abstraction: physics-based layers, physics-based networks, and network managers. Physics-based layers (Fig. A.2a) capture the system forward model process, architecture of the image reconstruction optimization, and contain the network’s learnable parameters. Physics-based networks (Fig. A.2b) contain a list (`torch.nn.ModuleList`) of physics-based layers, as well as basic initialization procedures. Finally, the network manager handles how to compute gradients with respect to the system’s learnable parameters.

### Physics-based Layers

The physics-based layer inherits from the Pytorch `torch.nn.Module` class, requiring a forward function. In this framework, we additionally require it to have a reverse function that performs the forward function’s inverse operations. Fig. A.3 highlights the structure

<sup>2</sup><https://github.com/kellman/MELD>

```

1 class physics_based_layer(torch.nn.Module):
2     def __init__(self, step_size, L, ...):
3         super(physics_based_layer, self).__init__()
4         self.step_size = step_size           # step size of gradient descent step
5         self.L = L                           # number of iterations to perform reverse
6
7     def forward(self, x, device='cpu'):
8         return x + self.step(x, device=device)
9
10    def reverse(self, x, device='cpu'):
11        with torch.no_grad():
12            z = x
13            for _ in range(self.L):
14                x = z - self.step(x, device=device)
15            return x
16
17    def step(self, x, device='cpu'):
18        g = self.grad(x, device=device)
19        return -1 * self.step_size * g
20
21    def grad(self, x, device='cpu'):
22        ### GRADIENT OF IMAGE RECONSTRUCTION MODEL ###

```

Figure A.3: Physics-based Gradient Descent Layer: A template for a gradient descent based layer. The forward performs the layer’s operations and reverse function performs the layer’s inverse operations.

```

1 class physics_based_network():
2     def __init__(self, Nlayers, device='cpu'):
3         self.Nlayers = Nlayers # number of layers in network
4
5         ### Setup Network ###
6         self._make_model(device=device)
7
8     def _make_model(self, device='cpu'):
9         self.gradient = physics_based_layer(...)
10        self.projection()
11
12        ### Generate Network ###
13        layer = torch.nn.ModuleList([self.grad])
14        self.network = torch.nn.ModuleList([layer for _ in range(self.Nlayers)])
15
16    def _initialize_model(self, device='cpu'):
17        ### Initialize learnable parameters ###
18
19    def projection(self,):
20        ### Projects learnable parameters on their feasible limits and constraints ###
21
22    def initialize(self, input_data, device='cpu'):
23        ### Returns the network's input for some input data ###

```

Figure A.4: Physics-based Network: A template for a physics-based network formed from gradient descent. The main attribute of this class is a list, `torch.nn.ModuleList`, of physics-based layers.

of a gradient descent based update and can serve as a template to implement your own physics-based layer.

## Physics-based Networks

The physics-based network contains basic functionality to setup and initialize the network, as well as, several functions useful for training and testing. The main attribute of the class is the network, which is a list of lists, `torch.nn.ModuleList`, of physics-based layers that comprise the network. Each sublist contains the operations for a single layer (e.g. gradient and proximal update). The projection function is useful for constraining learnable parameters to be feasible. The initialize function is useful for computing the input to the network given a new training or testing data point. Fig. [A.4](#) highlights the basic

structure of a gradient descent based network and can serve as a template to implement your own.

### **Network Manager**

This class performs gradient computation to update the learnable parameters of the network. At setup it evaluates the memory required for backpropagation and with the memory limit of current hardware determines what style of gradient computation is most efficient (*i.e.* fastest). It chooses from standard backpropagation, checkpointing with forward recalculation, and our memory efficient learning procedure (reverse recalculation). Chapter 5 contains an indepth discussion on the speed of each method and when each method should be used.

## Appendix B

# Total Variation Regularization

One of the most popular approaches for regularized image reconstruction is to use total variation (TV) [67]. For our learning framework, we consider the parallel proximal algorithm implementation of TV regularization. At its core, the method uses a simple wavelet-domain soft-thresholding operation to compute the proximal operator associated with TV in closed form. This eliminates the usual need for an additional iterative solver to compute the TV proximal operator.

### Parallel Proximal Method

We now provide the basic concepts of the parallel proximal method [70]. Let us first note that the (anisotropic) TV functional is defined as

$$\mathcal{P}_{\text{TV}}(\mathbf{x}) = \tau \sum_i \|\mathbf{D}_i \mathbf{x}\|_1, \quad (\text{B.1})$$

where  $\mathbf{x}$  represents the image,  $\mathbf{D}_i$  is the discrete gradient operator along the  $i^{\text{th}}$  dimension, and  $\tau$  is a positive scalar that determines the strength of regularization.

Using a union of orthogonal transforms  $\{\mathbf{W}_k\}_k$ , composed of first-level shifted Haar wavelet functions, one can rewrite the TV regularizer in (B.1) as

$$\mathcal{P}_{\text{TV}}(\mathbf{x}) = \tau \sqrt{2} \sum_{k=1} \sum_{n \in P_k} |[\mathbf{W}_k \mathbf{x}]_n|, \quad (\text{B.2})$$

where  $P_k$  is the set of indices that correspond to the detail coefficients for the  $k^{\text{th}}$  transform. As a consequence of this reinterpretation, the corresponding proximal operator admits a closed form since each  $\mathbf{W}_k$  is an orthogonal transform. Given a vector  $\mathbf{x}$ , the proximal step reads



$$\text{prox}_{\text{TV}}(\mathbf{x}) = \sum_k \mathbf{W}_k^H \mathcal{S}_{\sqrt{2}\tau}(\mathbf{W}_k \mathbf{x}), \quad (\text{B.3})$$

where  $\mathcal{S}_\xi(\cdot) = \max(|\cdot| - \xi, 0)$  is the soft thresholding function operating only on the detail coefficients with  $\xi > 0$  being the threshold parameter.

### Proximal Backpropagation

As referenced earlier, we must be able to backpropagate error through our proximal operator in order to compute the analytic gradient with respect to our design parameters. For our particular choice of proximal operator the gradient with respect its input is expressed as,

$$\frac{\partial \text{prox}_{\text{TV}}(\mathbf{x})}{\partial \mathbf{z}} = \sum_k \mathbf{W}_k^H \mathcal{S}'_{\sqrt{2}\tau}(\mathbf{W}_k \mathbf{x}) \mathbf{W}_k, \quad (\text{B.4})$$

where  $\mathcal{S}'_\xi(\cdot)$  is the derivative of  $\mathcal{S}_\xi(\cdot)$  with respect to its input argument.

## Bibliography

- [1] G. Zheng, R. Horstmeyer, and C. Yang. "Wide-field, high-resolution Fourier ptychographic microscopy". In: *Nature Photonics* 7.9 (July 2013), pp. 739–745.
- [2] L. Tian, Z. Liu, L.-H. Yeh, M. Chen, J. Zhong, and L. Waller. "Computational illumination for high-speed in vitro Fourier ptychographic microscopy". In: *Optica* 2.10 (2015), pp. 904–911.
- [3] G. Popescu. *Quantitative Phase Imaging of Cells and Tissues*. McGraw-Hill biophotonics. McGraw-Hill Education, 2011.
- [4] L. Tian and L. Waller. "3D intensity and phase imaging from light field measurements in an LED array microscope". In: *Optica* 2.2 (Feb. 2015), pp. 104–111.
- [5] M. Chen, L. Tian, and L. Waller. "3D differential phase contrast microscopy". In: (Sept. 2016), pp. 1–11.
- [6] F. Zernike. "How I Discovered Phase Contrast". In: *Science* 121.3141 (1955), pp. 345–349.
- [7] W. Lang. *Nomarski differential interference-contrast microscopy*. Carl Zeiss, 1982.
- [8] Z. F. Phillips, R. Eckert, and L. Waller. "Quasi-Dome: A Self-Calibrated high-NA LED Illuminator for Fourier Ptychography". In: *Imaging and Applied Optics 2017*. Optical Society of America, June 2017.
- [9] L. Tian and L. Waller. "Quantitative differential phase contrast imaging in an LED array microscope". In: *Optics Express* 23.9 (May 2015), pp. 11394–11403.
- [10] R. Milo and R. Phillips. *Cell Biology by the numbers*. Garland Science: Taylor & Francis Group, 2015.
- [11] L. Tian, X. Li, K. Ramchandran, and L. Waller. "Multiplexed coded illumination for Fourier Ptychography with an LED array microscope". In: *Biomedical Optics Express* 5.7 (June 2014), pp. 1–14.
- [12] F. Pukelsheim. *Optimal design of experiments*. SIAM, 2006.
- [13] A. Chakrabarti. "Learned Sensor Multiplexing Design through Back-propagation". In: (Nov. 2016), pp. 1–9.

- [14] R. Horstmeyer, R. Chen, B. Kappes, and B. Judkewitz. "Convolutional Neural Networks That Teach Microscopes How to Image". In: (Sept. 2017), pp. 1–14.
- [15] C. D. Bahadir, A. V. Dalca, and M. R. Sabuncu. "Adaptive Compressed Sensing MRI with Unsupervised Learning". In: *arXiv preprint arXiv:1907.11374* (2019).
- [16] H. Sun, A. V. Dalca, and K. L. Bouman. "Learning a Probabilistic Strategy for Computational Imaging Sensor Selection". In: *arXiv preprint arXiv:2003.10424* (2020).
- [17] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. "Do imagenet classifiers generalize to imagenet?" In: *arXiv preprint arXiv:1902.10811* (2019).
- [18] K. Gregor and Y. LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Jun. 2010, pp. 399–406.
- [19] J. Sun, H. Li, Z. Xu, et al. "Deep ADMM-Net for compressive sensing MRI". In: *Advances in Neural Information Processing Systems*. 2016, pp. 10–18.
- [20] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll. "Learning a Variational Network for Reconstruction of Accelerated MRI Data". In: *Magnetic Resonance in Medicine* 79.6 (Nov. 2017), pp. 3055–3071.
- [21] H. K. Aggarwal, M. P. Mani, and M. Jacob. "Modl: Model-based deep learning architecture for inverse problems". In: *IEEE transactions on medical imaging* 38.2 (2018), pp. 394–405.
- [22] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein. "Unrolled Optimization with Deep Priors". In: *arXiv:1705.08041 [cs.CV]* (May 2017), pp. 1–11.
- [23] K. Zhang, W. Zuo, S. Gu, and L. Zhang. "Learning deep CNN denoiser prior for image restoration". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3929–3938.
- [24] M. Kellman, E. Bostan, N. Repina, and L. Waller. "Physics-based learned design: Optimized coded-illumination for quantitative phase imaging". In: *IEEE Transactions on Computational Imaging* 5.3 (2019), pp. 344–353.
- [25] M. Kellman, E. Bostan, M. Chen, and L. Waller. "Data-Driven Design for Fourier Ptychographic Microscopy". In: *Proceedings of the International Conference on Computational Photography*. 2019.
- [26] M. Mir, B. Bhaduri, R. Wang, R. Zhu, and G. Popescu. *Quantitative phase imaging*. Vol. 57. Elsevier Amsterdam, The Netherlands, 2012, pp. 133–217.
- [27] B. Bhaduri, H. Pham, M. Mir, and G. Popescu. "Diffraction phase microscopy with white light". In: *Opt. Lett.* 37.6 (Mar. 2012), pp. 1094–1096. DOI: [10.1364/OL.37.001094](https://doi.org/10.1364/OL.37.001094).
- [28] Z. Wang, L. Millet, M. Mir, H. Ding, S. Unarunotai, J. Rogers, M. U. Gillette, and G. Popescu. "Spatial light interference microscopy (SLIM)". In: *Optics Express* 19.2 (Jan. 2011), pp. 1016–1026.

- [29] T. E. Gureyev, A. Roberts, and K. A. Nugent. "Partially coherent fields, the transport-of-intensity equation, and phase uniqueness". In: *Journal of the Optical Society of America A* 12.9 (Sep. 1995), pp. 1942–1946.
- [30] N. Streibl. "Phase imaging by the transport equation of intensity". In: *Optics communications* 49.1 (1984), pp. 6–10.
- [31] L. Waller, L. Tian, and G. Barbastathis. "Transport of Intensity phase-amplitude imaging with higher order intensity derivatives". In: *Optics Express* 18.12 (Jun. 2010), pp. 12552–12561.
- [32] J. Goodman. *Introduction to Fourier Optics*. McGraw-Hill, 2008.
- [33] Z. Liu, L. Tian, S. Liu, and L. Waller. "Real-time brightfield, darkfield, and phase contrast imaging in a light-emitting diode array microscope". In: *Journal of Biomedical Optics* 19.10 (Oct. 2014), p. 106002.
- [34] L.-H. Yeh, J. Dong, J. Zhong, L. Tian, M. Chen, G. Tang, M. Soltanolkotabi, and L. Waller. "Experimental robustness of Fourier ptychography phase retrieval algorithms". In: *Optics Express* 23.26 (Dec. 2015), pp. 33214–33227.
- [35] D. Geman and C. Yang. "Nonlinear image recovery with half-quadratic regularization". In: *IEEE transactions on Image Processing* 4.7 (1995), pp. 932–946.
- [36] N. Parikh and S. Boyd. "Proximal Algorithms". In: *Foundations and Trends® in Optimization* 1.3 (Aug. 2014), pp. 127–239.
- [37] N. Parikh and S. Boyd. "Proximal algorithms". In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.
- [38] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [39] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. "Deep Convolutional Neural Network for Inverse Problems in Imaging". In: *IEEE Transactions on Image Processing* 9 (Jun. 2017), pp. 4509–4522.
- [40] B. Rappaz, B. Breton, E. Shaffer, and G. Turcatti. "Digital holographic microscopy: A quantitative label-free microscopy technique for phenotypic screening". In: *Combinatorial Chemistry & High Throughput Screening* 17.1 (Jan. 2014), pp. 80–88.
- [41] E. CuChe, P. Marquet, and C. Depeursinge. "Simultaneous amplitude-contrast and quantitative phase-contrast microscopy by numerical reconstruction of Fresnel off-axis holograms". In: *Appl. Opt.* 38.34 (Dec. 1999), pp. 6994–7001. DOI: [10.1364/AO.38.006994](https://doi.org/10.1364/AO.38.006994).
- [42] G. Zheng, R. Horstmeyer, and C. Yang. "Wide-field, high-resolution Fourier ptychographic microscopy". In: *Nature Photonics* 7.9 (July 2013), pp. 739–745.
- [43] G. Zheng, C. Kolner, and C. Yang. "Microscopy refocusing and dark-field imaging by using a simple LED array". In: *Optics Letters* 36.20 (Oct. 2011), pp. 3987–3989.

- [44] R. Ling, W. Tahir, H.-Y. Lin, H. Lee, and L. Tian. "High-throughput intensity diffraction tomography with a computational microscope". In: *Biomedical Optics Express* 9.5 (Jan. 2018), pp. 2130–2141.
- [45] B. Kachar. "Asymmetric illumination contrast: a method of image formation for video light microscopy". In: *Science* 227.4688 (Feb. 1985), pp. 766–768.
- [46] D. Hamilton and C. Sheppard. "Differential phase contrast in scanning optical microscopy". In: *Journal of microscopy* 133.1 (1984), pp. 27–39.
- [47] S. B. Mehta and C. J. Sheppard. "Quantitative phase-gradient imaging at high resolution with asymmetric illumination-based differential phase contrast". In: *Optics letters* 34.13 (2009), pp. 1924–1926.
- [48] L. Tian and L. Waller. "Quantitative differential phase contrast imaging in an LED array microscope". In: *Optics express* 23.9 (2015), pp. 11394–11403.
- [49] R. A. Claus, P. P. Naulleau, A. R. Neureuther, and L. Waller. "Quantitative phase retrieval with arbitrary pupil and illumination". In: *Optics Express* 23.20 (Oct. 2015), pp. 26672–26682.
- [50] D. K. Hamilton, C. J. R. Sheppard, and T. Wilson. "Improved imaging of phase gradients in scanning optical microscopy". In: *Journal of Microscopy* 135.3 (Sep. 1984), pp. 275–286.
- [51] N. Streibl. "Three-dimensional imaging by a microscope". In: *Journal of the Optical Society of America A* 2.2 (Feb. 1985), pp. 121–127.
- [52] J. Li, Q. Chen, J. Zhang, Y. Zhang, L. Lu, and C. Zuo. "Efficient quantitative phase microscopy using programmable annular LED illumination". In: *Biomedical Optics Express* 8.10 (Oct. 2017), pp. 4687–4705.
- [53] Y.-Z. Lin, K.-Y. Huang, and Y. Luo. "Quantitative differential phase contrast imaging at high resolution with radially asymmetric illumination". In: *Optics Letters* 43.12 (2018), pp. 2973–4.
- [54] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444.
- [55] S. Wang, Z. Su, L. Ying, X. Peng, S. Zhu, F. Liang, D. Feng, and D. Liang. "Accelerating magnetic resonance imaging via deep learning". In: *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*. Apr. 2016, pp. 514–517.
- [56] Y. Rivenson, Y. Zhang, H. Günaydin, D. Teng, and A. Ozcan. "Phase recovery and holographic image reconstruction using deep learning in neural networks". In: *Light: Science & Applications* 7.2 (Feb. 2018), p. 17141.
- [57] A. Sinha, J. Lee, S. Li, and G. Barbastathis. "Lensless computational imaging through deep learning". In: *Optica* 4.9 (Sep. 2017), pp. 1117–1125.
- [58] Y. Rivenson, Z. Göröcs, H. Günaydin, Y. Zhang, H. Wang, and A. Ozcan. "Deep learning microscopy". In: *Optica* 4.11 (Nov. 2017), pp. 1437–1443.

- [59] T. Nguyen, Y. Xue, Y. Li, L. Tian, and G. Nehmetallah. "Deep learning approach for Fourier ptychography microscopy". In: *arXiv preprint arXiv:1805.00334* (Sept. 2018), pp. 1–15.
- [60] B. Diederich, R. Wartmann, H. Schadwinkel, and R. Heintzmann. "Using machine-learning to optimize phase contrast in a low-cost cellphone microscope". In: *PLoS ONE* 13.3 (Mar. 2018), e0192937–20.
- [61] A. Robey and V. Ganapati. "Optimal physical preprocessing for example-based super-resolution". In: *Optics Express* 26.24 (Nov. 2018), pp. 31333–31350.
- [62] B. Xin, Y. Wang, W. Gao, D. Wipf, and B. Wang. "Maximal sparsity with deep networks?" In: *Advances in Neural Information Processing Systems*. 2016, pp. 4340–4348.
- [63] U. Kamilov and H. Mansour. "Learning optimal nonlinearities for iterative thresholding algorithms". In: *arXiv preprint arXiv:1512.04754s* (Dec. 2015), pp. 1–9.
- [64] E. Bostan, U. S. Kamilov, and L. Waller. "Learning-Based Image Reconstruction via Parallel Proximal Algorithm". In: *IEEE Signal Processing Letters* 25.7 (May 2018), pp. 989–993.
- [65] E. Bostan, U. S. Kamilov, M. Nilchian, and M. Unser. "Sparse Stochastic Processes and Discretization of Linear Inverse Problems". In: *IEEE Transactions on Image Processing* 22.7 (Jul. 2013), pp. 2699–2710.
- [66] A. Beck and M. Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM Journal on Imaging Sciences* 2.1 (Jan. 2009), pp. 183–202.
- [67] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. "An iterative regularization method for total variation-based image restoration". In: *Multiscale Modeling & Simulation* 4.2 (2005), pp. 460–489.
- [68] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on Image Processing* 16.8 (Aug. 2007), pp. 2080–2095.
- [69] P. L. Combettes and J.-C. Pesquet. "Proximal splitting methods in signal processing". In: *Fixed-point algorithms for inverse problems in science and engineering*. 2011, pp. 185–212.
- [70] U. S. Kamilov. "A Parallel Proximal Algorithm for Anisotropic Total Variation Minimization". In: *IEEE Transactions on Image Processing* 26.2 (Dec. 2016), pp. 539–548.
- [71] P. Marechal and J. Ye. "Optimizing Condition Numbers". In: *SIAM Journal on Optimization* 20.2 (2009), pp. 935–947.
- [72] C. S. Wong and J. C. Masaro. "A-optimal design matrices". In: *Discrete Mathematics* 50 (1984), pp. 295–318.

- [73] X. Ou, G. Zheng, and C. Yang. "Embedded pupil function recovery for Fourier ptychographic microscopy". In: *Optics Express* 22.5 (Mar. 2014), pp. 4960–4972.
- [74] H. Q. Nguyen, E. Bostan, and M. Unser. "Learning convex regularizers for optimal Bayesian denoising". In: *IEEE Transactions on Signal Processing* 66.4 (2018), pp. 1093–1105.
- [75] M. Lustig, D. Donoho, and J. M. Pauly. "Sparse MRI: The application of compressed sensing for rapid MR imaging". In: *Magnetic Resonance in Medicine* 58.6 (Dec. 2007), pp. 1182–1195.
- [76] S. R. P. Pavani, M. A. Thompson, J. S. Biteen, S. J. Lord, N. Liu, R. J. Twieg, R. Piestun, and W. E. Moerner. "Three-dimensional, single-molecule fluorescence imaging beyond the diffraction limit by using a double-helix point spread function". In: *Proceedings of the National Academy of Sciences* 106.9 (Mar. 2009), pp. 2995–2999.
- [77] L. Tian, Z. Liu, L.-H. Yeh, M. Chen, J. Zhong, and L. Waller. "Computational illumination for high-speed in vitro Fourier ptychographic microscopy". In: *Optica* 2.10 (Oct. 2015), pp. 904–908.
- [78] J. Sun, Q. Chen, Y. Zhang, and C. Zuo. "Sampling criteria for Fourier ptychographic microscopy in object space and frequency space". In: *Optics Express* 24.14 (July 2016), pp. 15765–15781.
- [79] O. Bunk, M. Dierolf, S. Kynde, I. Johnson, O. Marti, and F. Pfeiffer. "Influence of the overlap parameter on the convergence of the ptychographical iterative engine". In: *Ultramicroscopy* 108.5 (Apr. 2008), pp. 481–487.
- [80] P. Sidorenko and O. Cohen. "Single-shot ptychography". In: *Optica* 3.1 (Jan. 2016), pp. 9–14.
- [81] S. Dong, R. Shiradkar, P. Nanda, and G. Zheng. "Spectral multiplexing and coherent-state decomposition in Fourier ptychographic imaging". In: *Biomedical Optics Express* 5.6 (June 2014), pp. 1757–1767.
- [82] L. Bian, J. Suo, G. Situ, G. Zheng, F. Chen, and Q. Dai. "Content adaptive illumination for Fourier ptychography". In: *Optics Letters* 39.23 (Dec. 2014), pp. 6648–6651.
- [83] M. Kellman, M. Chen, Z. F. Phillips, M. Lustig, and L. Waller. "Motion-resolved quantitative phase imaging". In: *Biomedical Optics Express* 9.11 (Nov. 2018), pp. 5456–5466.
- [84] L. Bian, G. Zheng, K. Guo, J. Suo, C. Yang, F. Chen, and Q. Dai. "Motion-corrected Fourier ptychography". In: *Biomedical Optics Express* 7.11 (Nov. 2016), pp. 4543–4553.
- [85] A. Robey and V. Ganapati. "Optimal Physical Preprocessing for Example-Based Super-Resolution". In: *Optics Express* 26.24 (Nov. 2018), pp. 31333–31350.

- [86] H. Haim, S. Elmaleh, R. Giryes, A. M. Bronstein, and E. Marom. "Depth Estimation From a Single Image Using Deep Learned Phase Coded Mask". In: *IEEE Transactions on Computational Imaging* 4.3 (Sept. 2018), pp. 298–310. DOI: [10.1109/TCI.2018.2849326](https://doi.org/10.1109/TCI.2018.2849326).
- [87] V. Sitzmann, S. Diamond, Y. Peng, X. Dun, S. Boyd, W. Heidrich, F. Heide, and G. Wetzstein. "End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging". In: *ACM Transactions on Graphics* 37.4 (July 2018), pp. 1–13.
- [88] K. Gregor and Y. LeCun. "Learning Fast Approximations of Sparse Coding". In: *Proceedings of the 27th International Conference on Machine Learning*. 2010, pp. 399–406.
- [89] H. Jiang, Q. Tian, J. Farrell, and B. A. Wandell. "Learning the Image Processing Pipeline". In: *IEEE Transactions on Image Processing* 26.10 (Oct. 2017), pp. 5032–5042. DOI: [10.1109/TIP.2017.2713942](https://doi.org/10.1109/TIP.2017.2713942).
- [90] E. Bostan, M. Soltanolkotabi, D. Ren, and L. Waller. "Accelerated Wirtinger Flow for Multiplexed Fourier Ptychographic Microscopy". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. Oct. 2018, pp. 3823–3827. DOI: [10.1109/ICIP.2018.8451437](https://doi.org/10.1109/ICIP.2018.8451437).
- [91] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch". In: (2017).
- [92] Y. Chen and T. Pock. "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration". In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1256–1272.
- [93] J. Zhang and B. Ghanem. "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1828–1837.
- [94] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein. "Unrolled optimization with deep priors". In: *arXiv preprint arXiv:1705.08041* (2017).
- [95] P. Putzky and M. Welling. "Recurrent inference machines for solving inverse problems". In: *arXiv preprint arXiv:1706.04008* (2017).
- [96] M. Mardani, Q. Sun, D. Donoho, V. Pappyan, H. Monajemi, S. Vasanaawala, and J. Pauly. "Neural proximal gradient descent for compressive imaging". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9573–9583.
- [97] G. Ongie, A. Jalal, C. A. M. R. G. Baraniuk, A. G. Dimakis, and R. Willett. "Deep learning techniques for inverse problems in imaging". In: *IEEE Journal on Selected Areas in Information Theory* (2020).
- [98] P. Putzky and M. Welling. "Invert to learn to invert". In: *Advances in Neural Information Processing Systems*. 2019, pp. 444–454.



- [99] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.
- [100] D. Maclaurin, D. Duvenaud, and R. Adams. “Gradient-based hyperparameter optimization through reversible learning”. In: *International Conference on Machine Learning*. 2015, pp. 2113–2122.
- [101] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. “The reversible residual network: Backpropagation without storing activations”. In: *Advances in neural information processing systems*. 2017, pp. 2214–2224.
- [102] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. “Reversible architectures for arbitrarily deep residual neural networks”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [103] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. “Analyzing inverse problems with invertible neural networks”. In: *arXiv preprint arXiv:1808.04730* (2018).
- [104] L. Dinh, D. Krueger, and Y. Bengio. “Nice: Non-linear independent components estimation”. In: *arXiv preprint arXiv:1410.8516* (2014).
- [105] L. Dinh, J. Sohl-Dickstein, and S. Bengio. “Density estimation using real nvp”. In: *arXiv preprint arXiv:1605.08803* (2016).
- [106] D. P. Kingma and P. Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”. In: *Advances in neural information processing systems*. 2018, pp. 10215–10224.
- [107] J.-H. Jacobsen, A. Smeulders, and E. Oyallon. “i-revnet: Deep invertible networks”. In: *arXiv preprint arXiv:1802.07088* (2018).
- [108] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. “Neural ordinary differential equations”. In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.
- [109] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. “Fjord: Free-form continuous dynamics for scalable reversible generative models”. In: *arXiv preprint arXiv:1810.01367* (2018).
- [110] J. Behrmann, D. Duvenaud, and J.-H. Jacobsen. “Invertible residual networks”. In: *arXiv preprint arXiv:1811.00995* (2018).
- [111] B. Amos and J. Z. Kolter. “Optnet: Differentiable optimization as a layer in neural networks”. In: *arXiv preprint arXiv:1703.00443* (2017).
- [112] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. “Bilevel programming for hyperparameter optimization and meta-learning”. In: *arXiv preprint arXiv:1806.04910* (2018).

- [113] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. "Meta-learning with implicit gradients". In: *Advances in Neural Information Processing Systems*. 2019, pp. 113–124.
- [114] S. Bai, J. Z. Kolter, and V. Koltun. "Deep equilibrium models". In: *Advances in Neural Information Processing Systems*. 2019, pp. 690–701.
- [115] R. Liao, Y. Xiong, E. Fetaya, L. Zhang, K. Yoon, X. Pitkow, R. Urtasun, and R. Zemel. "Reviving and improving recurrent back-propagation". In: *arXiv preprint arXiv:1803.06396* (2018).
- [116] S. Banach. "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales". In: *Fund. math* 3.1 (1922).
- [117] D. L. Donoho. "Compressed sensing". In: *IEEE Transactions on Information Theory* 52.4 (2006), pp. 1289–1306. DOI: [10.1109/TIT.2006.871582](https://doi.org/10.1109/TIT.2006.871582).
- [118] M. Lustig, D. Donoho, and J. M. Pauly. "Sparse MRI: The application of compressed sensing for rapid MR imaging". In: *Magn. Reson. Med.* 58.6 (Dec. 2007), pp. 1182–1195. DOI: [10.1002/mrm.21391](https://doi.org/10.1002/mrm.21391).
- [119] D. J. Brady, K. Choi, D. L. Marks, R. Horisaki, and S. Lim. "Compressive holography". In: *Optics express* 17.15 (2009), pp. 13040–13049.
- [120] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk. "Single-pixel imaging via compressive sampling". In: *IEEE signal processing magazine* 25.2 (2008), pp. 83–91.
- [121] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).
- [122] K. P. Pruessmann, M. Weiger, M. B. Scheidegger, and P. Boesiger. "SENSE: sensitivity encoding for fast MRI". In: *Magn. Reson. Med.* 42.5 (Nov. 1999), pp. 952–962.
- [123] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [124] A. Beck and M. Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.
- [125] F. Ong, X. Zhu, J. Y. Cheng, K. M. Johnson, P. E. Larson, S. S. Vasanawala, and M. Lustig. "Extreme MRI: Large-scale volumetric dynamic imaging from continuous non-gated acquisitions". In: *Magnetic Resonance in Medicine* (2020).
- [126] L. Feng, L. Axel, H. Chandarana, K. T. Block, D. K. Sodickson, and R. Otazo. "XD-GRASP: golden-angle radial MRI with reconstruction of extra motion-state dimensions using compressed sensing". In: *Magnetic resonance in medicine* 75.2 (2016), pp. 775–788.

- [127] B.-C. Chen, W. R. Legant, K. Wang, L. Shao, D. E. Milkie, M. W. Davidson, C. Jane-topoulos, X. S. Wu, J. A. Hammer, Z. Liu, et al. "Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution". In: *Science* 346.6208 (2014).
- [128] F. L. Liu, G. Kuo, N. Antipa, K. Yanny, and L. Waller. "Fourier DiffuserScope: Single-shot 3D Fourier light field microscopy with a diffuser". In: *arXiv preprint arXiv:2006.16343* (2020).
- [129] S. Chowdhury, M. Chen, R. Eckert, D. Ren, F. Wu, N. A. Repina, and L. Waller. "High-resolution 3D refractive index microscopy of multiple-scattering samples from intensity images". In: *Optica* 6.9 (Sept. 16, 2019), pp. 1211–1219. DOI: [10.1364/OPTICA.6.001211](https://doi.org/10.1364/OPTICA.6.001211), published.
- [130] E. Tseng, F. Yu, Y. Yang, F. Mannan, K. S. Arnaud, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide. "Hyperparameter optimization in black-box image processing using differentiable proxies." In: *ACM Trans. Graph.* 38.4 (2019), pp. 27–1.
- [131] R. Horstmeyer, R. Y. Chen, B. Kappes, and B. Judkewitz. "Convolutional neural networks that teach microscopes how to image". In: *arXiv preprint arXiv:1709.07223* (2017).
- [132] C. L. Cooke, F. Kong, A. Chaware, K. C. Zhou, K. Kim, R. Xu, D. M. Ando, S. J. Yang, P. C. Konda, and R. Horstmeyer. "Physics-enhanced machine learning for virtual fluorescence microscopy". In: *arXiv preprint arXiv:2004.04306* (2020).
- [133] J. I. Tamir, X. Y. Stella, and M. Lustig. "Unsupervised deep basis pursuit: Learning reconstruction without ground-truth data". In: *Proc. Intl. Soc. Mag. Reson. Med.* Vol. 27. 2019, p. 0660.
- [134] J. Liu, Y. Sun, C. Eldeniz, W. Gan, H. An, and U. S. Kamilov. "RARE: Image Reconstruction using Deep Priors Learned without Ground Truth". In: *IEEE Journal of Selected Topics in Signal Processing* (2020).
- [135] B. Yaman, S. A. H. Hosseini, S. Moeller, J. Ellermann, K. Uğurbil, and M. Akçakaya. "Self-supervised learning of physics-guided reconstruction neural networks without fully sampled reference data". In: *Magnetic Resonance in Medicine* (2020).
- [136] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. "Noise2noise: Learning image restoration without clean data". In: *arXiv preprint arXiv:1803.04189* (2018).
- [137] J. Batson and L. Royer. "Noise2self: Blind denoising by self-supervision". In: *arXiv preprint arXiv:1901.11365* (2019).
- [138] A. Krull, T.-O. Buchholz, and F. Jug. "Noise2void-learning denoising from single noisy images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2129–2137.

- [139] M. Kellman, J. Tamir, E. Boston, M. Lustig, and L. Waller. “Memory-efficient Learning for Large-scale Computational Imaging”. In: *arXiv preprint arXiv:1912.05098* (2019).