# Complex-valued Deep Learning with Applications to Magnetic Resonance Image Synthesis

*Pat Virtue*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-130
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-130.html

August 19, 2019

# Complex-valued Deep Learning with Applications to Magnetic Resonance Image Synthesis

By

Patrick M. Virtue

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:

Professor Michael Lustig, Co-chair
Professor Stella X. Yu, Co-chair
Professor Trevor Darrell
Professor Bruno Olshausen

Summer 2019

Complex-valued Deep Learning with
Applications to Magnetic Resonance Image Synthesis

Abstract

Complex-valued Deep Learning with
Applications to Magnetic Resonance Image Synthesis

by

Patrick M. Virtue

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Michael Lustig, Co-chair
Professor Stella X. Yu, Co-chair

Magnetic resonance imaging (MRI) has the ability to produce a series of images that each have different visual contrast between tissues, allowing clinicians to *qualitatively* assess pathologies that may be visible in one contrast-weighted image but not others. Unfortunately, these standard contrast-weighted images do not contain *quantitative* values, producing challenges for post-processing, assessment, and longitudinal studies. MR fingerprinting is a recent technique that produces quantitative tissue maps from a single pseudorandom acquisition, but it relies on computationally heavy nearest neighbor algorithms to solve the associated nonlinear inverse problem. In this dissertation, we present our deep learning methods to speed up quantitative MR fingerprinting and synthesize the standard contrast-weighted images directly from the same MR fingerprinting scan.

Adapting deep learning methodologies to MR image synthesis presents two specific challenges: 1) complex-valued data and 2) the presence of noise while undersampling.

MRI signals are inherently complex-valued, as they are measurements of rotating magnetization within the body. However, modern neural networks are not designed to support complex values. As an example, the pervasive ReLU activation function is undefined for complex numbers. This limitation curtails the impact of deep learning for complex data applications, such as MRI, radio frequency modulation identification, and target recognition in synthetic-aperture radar images. In this dissertation, we discuss the motivation for complex-valued networks, the changes that we have made to implement complex backpropagation, and our new complex cardioid activation function that made it possible to outperform real-valued networks for MR fingerprinting image synthesis.

In Fourier-based medical imaging, undersampling results in an underdetermined system, in which a linear reconstruction will exhibit artifacts. Another consequence is lower signal-to-noise ratio (SNR) because of fewer acquired measurements. The coupled effects of low SNR and underdetermined system during reconstruction makes it difficult to model the

signal and analyze image reconstruction algorithms. We demonstrate that neural networks trained only with a Gaussian noise model fail to process *in vivo* MR fingerprinting data, while our proposed empirical noise model allows neural networks to successfully synthesize quantitative images. Additionally, to better understand the impact of noise on undersampled imaging systems, we present an image quality prediction process that reconstructs fully sampled, fully determined data with noise added to simulate the SNR loss induced by a given undersampling pattern. The resulting prediction image empirically shows the effects of noise in undersampled image reconstruction without any effect from an underdetermined system, allowing MR pulse sequence and reconstruction developers to determine if low SNR, rather than the underdetermined system, is the limiting factor for a successful reconstruction.

For Caitlin and Molly

# Contents

# List of Figures

# List of Tables

# Acknowledgments

First, I would like to thank my advisors Miki Lustig and Stella Yu for making this dissertation possible. Miki and Stella provided the amazing opportunity to combine MRI with deep learning, and they always had a steady stream of technical innovations that I only wish I could keep up with. I would like to thank them for bringing me in as a founding member of their respective research groups and for riding this roller coaster with me to the end.

I would also like to thank my dissertation and qualifying committee members, including Steve Connolly, Trevor Darrell, and Bruno Olshausen, for their guidance and support throughout this process.

My experience working with MRI would have been extremely limited without my external collaborations with Shreyas Vasanawala from Stanford and Mariya Doneva from Philips to provide connections to the clinic and the product. I would also like to thank Peter Koken from Philips for his MR fingerprinting sequence work and Kevin Epperson from Stanford for his tireless effort to produce a beautiful, fully sampled MRI dataset [5].

I would like to thank members of MikGroup, especially Jon Tamir and Frank Ong for great technical discussions and moral support, Mark Murphy and Martin Uecker for building great reconstruction software, Wenwen Jiang for always walking in with a smile, Joe Corea for flying turkeys and finding my keys, and Anita Flynn for shepherding us all.

I have been extremely fortunate to work with members of ICSI Vision Group. Thank you to Jyh-Jing Hwang and Tsung-Wei Ke for deepening my knowledge of neural networks, to Baladitya Yellapragada for being the best office mate, and to Mattias Demant for bringing me into the world of solar cell fabrication.

I am especially grateful for the human side of machine learning. I always enjoyed deep discussions with my Berkeley colleagues about real-valued networks with the Caffe team of Evan Shelhamer, Jon Long, and Jeff Donahue, and about complex-valued networks with my ML@B group of Riley Edmunds, Renee Sweeney, and Vinay Ramashesh.

Ph.D. work can be filled with ups and downs, but the times spent working with my undergraduate research collaborators were the highest points of my graduate career. I am so grateful to have had the opportunity to work with Sana Vaziri, Joyce Toh, Harrison Wang, Asha Anoosheh, and Peter Wang.

As Clarence wrote to George at the end of *It's a Wonderful Life*, "No man is a failure who has friends". I would especially like to thank Justine Sherry, Maryam Vareth, Nicole

Rogers, Betsy Fordyce, and Brian Pawloski for being another set of footprints in the sand.

Finally, I would be nowhere without the loving support of my family, especially my parents for raising me to believe I could accomplish anything. I am most thankful for my wife Caitlin and my daughter Molly for always being there when I got home, always ready to hike, play sports, watch a show, and of course, eat.

# Chapter 1

# Introduction

Magnetic resonance imaging (MRI) has the ability to measure soft tissue contrast and subtle changes in anatomic function that cannot be detected by other imaging modalities. The capabilities of MRI lie far beyond what is used during a standard clinical exam. MRI has the ability to encode the three-dimensional velocity of blood flow, and MRI is able to quantitatively measure specific tissue parameters in the body, allowing for more accurate analysis and the ability to track a patient's progress over time. MRI produces many types of images that can be categorized into two basic groups: *qualitative* images, most commonly, images known as contrast-weighted images, and *quantitative* images, e.g., that contain numerical tissue properties. Unfortunately, generating all of these images requires many repeated acquisitions that consume significantly more time than is cost-effective for an already expensive clinical MRI exam. Hence, for the sake of time and money, clinicians limit themselves to a baseline set of images rather than taking advantage of all that MRI has to offer.

MR fingerprinting is a recent technique that aims to acquire the data required for a broad set of quantitative images with only a single scan [6]. MR fingerprinting then relies upon image synthesis methods to convert the raw data to quantitative images for health professionals to analyze. However, existing synthesis methods for reconstructing MR fingerprinting data are based on incomplete simulation models; furthermore, they are computationally expensive as their runtime grows exponentially with the number of desired quantitative images.

In this thesis, we propose deep learning as a new method for MR fingerprinting synthesis. Deep learning is able to learn a mapping from the acquired MRF data to *both* quantitative images and the standard contrast-weighted images without relying on oversimplified simulation models. Additionally, deep learning scales well as it is more computationally efficient in terms of the number of desired images.

To develop our deep-learning MRF synthesis methods, we confront two primary challenges. First, MRI produces complex-valued data but modern deep learning methodologies are designed for real-valued data. Second, MR fingerprinting data is undersampled, which results in an undetermined system and noisy artifacts in the data.

We make three contributions towards developing robust deep learning based models from undersampled MRI data in clinical settings: 1) evolving deep learning methods for complex-

valued applications, 2) leveraging deep learning to extend the capabilities of MR fingerprinting, and 3) developing tools to practically balance the degree of undersampling and the signal to noise ratio in a clinical acquisition.

## 1.1 MRI Overview

Certain atoms, such as hydrogen, have a magnetic moment that reacts to the magnetic field within a magnetic resonance imaging (MRI) scanner. MRI works by altering the magnetic field to excite these magnetic moments and then measuring the induced magnetic field as they relax back to their equilibrium state. Different tissues will react at different rates depending on their chemical composition and molecular structure.

One of the strengths of MRI is that it can adjust its scan parameters to maximize the visual contrast between different tissue types. For example, it can acquire an image with a strong contrast between white matter and gray matter to help measure the thickness of the cortex in the brain, a key parameter to diagnosing neurodegenerative disorders, such as Alzheimer's disease [7]. MRI's ability to customize these contrast-weighted images can also be a drawback as it often requires additional exam time to repeat scans with different settings to accentuate the contrast between various pairs of tissues, such as between fat and water or between benign and malignant tissue.

Advanced MRI techniques include the ability to measure the four-dimensional velocity of blood flow by encoding the velocity in the phase of the complex-valued MRI signal [8]. Additionally, MRI can quantitatively measure specific tissue parameters in the body, rather than the qualitative contrast-weighted image mentioned above. This quantitative parameter mapping allows for more accurate analysis and the ability to track a patient's progress over time [9, 10]. Unfortunately, these advanced techniques require many repeated acquisitions that consume significantly more time than is cost-effective for an already expensive clinical MRI exam.

MRI compressed sensing [11] and MR fingerprinting [6] are two recent developments that scan faster by pseudorandomly undersampling the acquisition and then rely on a model of the system to recover the full-fidelity image data. More specifically, compressed sensing is an image reconstruction technique that aims to reconstruct an image from undersampled data. MR fingerprinting, on the other hand, uses a single pseudorandom data acquisition to produce quantitative tissue maps, such as mapping the relaxation rates of the tissue at each pixel location.

Deep learning has been shown to be very well suited to learn a model to synthesize images from these undersampled acquisitions [12]. However, deep learning research for complex-valued clinical applications such as MR reconstruction and MR fingerprinting remains in its infancy.

## 1.2 Complex-valued Deep Learning

MRI data is acquired in the frequency domain of the image and is inherently complex-valued. This presents a challenge for MRI applications of deep learning methods, as neural network methods and software have been designed with only real values in mind. For example, the max operator in the popular ReLU activation function is undefined. Without viable complex-valued neural networks, applications will either drop the phase of the signal or split the real and imaginary values into two channels, ignoring the inherent relationship between the two components [13, 14, 15].

Pioneering work in the 1990s began to show that complex-valued neural networks could leverage mathematical properties of complex values, such as the complex identity theorem [16], to learn more efficient transform functions than with real-valued networks [1], see Figure 1.1.

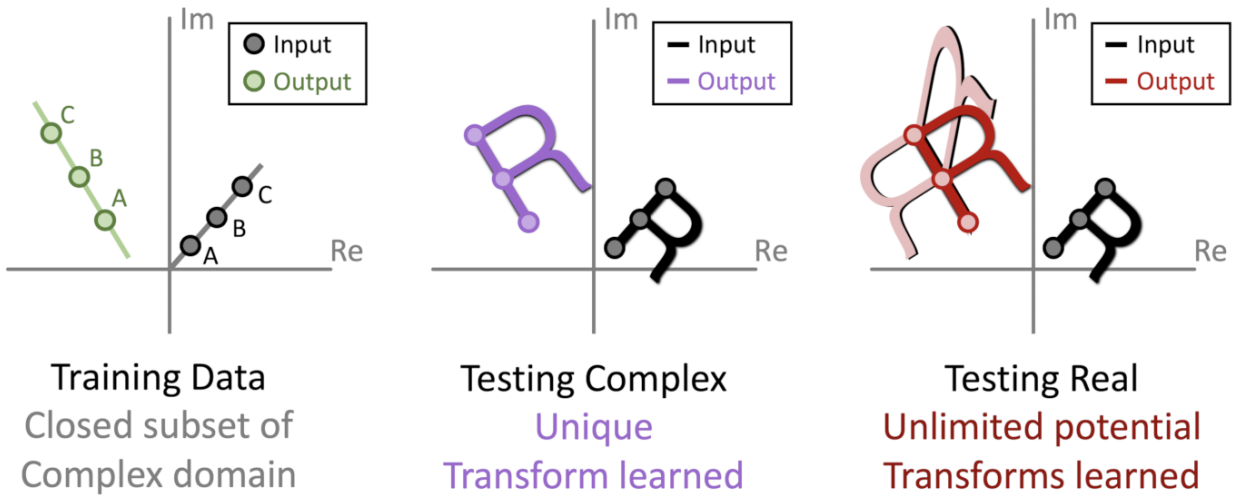


*Figure 1.1*: Illustration of how a complex-valued neural network could learn a unique transform function, despite an under-specified training dataset. Even though the training data lie on a single line, left, the complex identity theorem implies that this is sufficient to define a unique complex-valued holomorphic transform function, center [1]. However, there is no such theorem when considering real-valued transform functions, right.

These early works attempted to use various complex-valued activation functions [1, 17], but these attempts had limited success in applications to real-world settings.

More recent work in complex-valued deep learning aims to improve the convergence and stability of neural networks by building upon complex number theory, including unitary weight matrices [18] and wavelet transforms [19, 20]. Yet another set of researchers have begun to generalize complex deep learning by recognizing that complex-values could be extended to three dimensions as quaternions [21]. Most recently, complex-valued data are approached from a Riemannian manifold perspective, and new CNN layer functions are

developed in order to preserve equivariance and invariance properties that are important for complex numbers [4].

Our research takes a contrasting approach. Where prior work begins from mathematical principles and thence re-envisions deep learning, we take a systems-oriented approach. Starting from existing deep-learning architectures, we identify the limitations that prevent these approaches from incorporating complex-valued data; we then extend the existing systems with the missing mathematical foundations needed to support complex data.

We design and implement a complex-valued network and propose two new complex activation functions, including the complex cardioid function. The cardioid activation function preserves the input phase while gradually attenuating the input magnitude as the input phase rotates from the positive real axis to the negative real axis. Of particular note is that the cardioid activation reduces to a ReLU function when its domain is limited to the real axis. Where prior attempts at complex-valued activation functions failed to match the performance of their real-valued equivalents, cardioid not only meets but at times surpasses the accuracy of real-valued networks for MR fingerprinting.

## 1.3 Deep Learning for MR Fingerprinting

MR fingerprinting produces quantitative tissue maps from a single pseudorandom acquisition. Tissue maps include specific tissue parameters (T1, T2), as well as system parameters (B0, B1). However, MR fingerprinting relies on computationally heavy dictionary matching algorithms to solve the associated nonlinear inverse problem. Because the computational complexity of dictionary matching scales exponentially with the number of parameters, MR fingerprinting is limited to mapping only a small set of different parameters.

We replace the computationally heavy dictionary matching process entirely with a complex-valued neural network. Our approach scales linearly with the number of parameters, opening MR fingerprinting to a broader set of applications with more parameters being modeled.

Prior works to improve MR fingerprinting performance include more effective acquisition encoding [22, 23], improved reconstruction methods [24, 25], and dictionary compression via singular value decomposition [26]. Our proposed method complements these prior works and can be used in conjunction with them for future improvements.

Despite the quantitative information present in the parameter maps, clinicians still require contrast weighted images as part of an MRI exam protocol. We demonstrate that the MR fingerprinting signal is rich enough to synthesize contrast-weighted images as well, potentially reducing exam times by several minutes. By learning to directly convert the MRF signal into desired contrast-weighted images, a trained neural network bypasses the inadequate simulation steps required to synthesize contrast images via parameter maps [27], Figure 1.2.

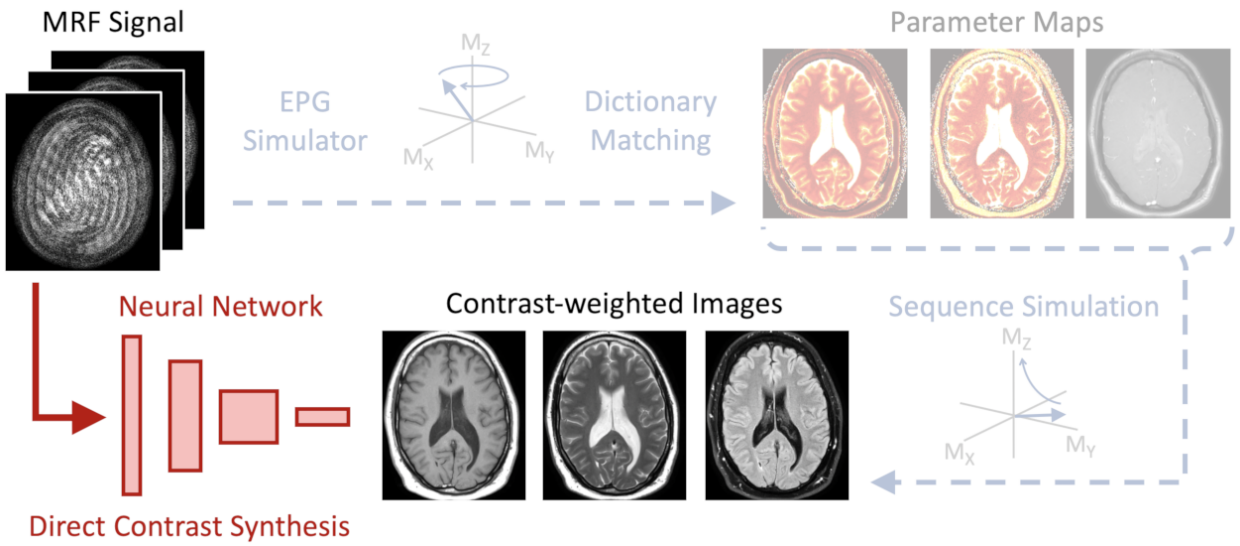


*Figure 1.2*: Our proposed direct contrast synthesis (DCS) uses a trained neural network to transform the MR fingerprinting signal directly into many different contrast-weighted images. DCS bypasses two simulation steps and avoids incomplete modeling assumptions and error propagation.

## 1.4 Noise in Undersampled Reconstruction

Our third contribution aids in the practical application of undersampled imaging systems. The challenge is that system noise is coupled with incoherent acquisition artifacts. When skipping acquisition measurements at regular intervals, the resulting data has aliasing ambiguities that cannot be resolved. Compressed sensing [11] and MR fingerprinting, on the other hand, acquire data using irregular sampling patterns, which cause incoherent aliasing that presents as noise. Unfortunately, this aliasing is coupled with the system noise during reconstruction, complicating an already challenging inverse problem.

Compressed sensing theory has provided us with an extensive analysis of the bounds of a successful recovery of undersampled data [28, 29, 30]. However, when testing a compressed sensing reconstruction algorithm on a new, undersampled clinical dataset and the image results are unacceptable, it is difficult to understand the cause of the failure from theoretical bounds.

We present an image quality prediction process to provide the tools to empirically analyze the effects of noise in undersampled reconstruction systems. This allows us to answer questions like, "Should I improve my SNR, or reduce my undersampling rate?" in a practical setting.

## 1.5 Organization and Previously Published Work

In Chapter 2, we propose novel complex-valued neural network activation functions and integrate complex-valued backpropagation and network layers into modern deep learning software frameworks. This chapter includes published work with Stella Yu and Michael Lustig [31, 32].

In Chapter 3, we apply deep learning to solve several problems in MR fingerprinting and introduce an empirical model to train networks to be robust to the noise and undersampling artifacts present in MRF acquisitions. This chapter includes published work with Jon Tamir, Mariya Doneva, Stella Yu, and Michael Lustig [31, 33, 34].

In Chapter 4, we propose a model to assess the effectiveness of undersampling reconstruction methods by simulating a system with undersampling noise that is no longer coupled to undersampling aliasing. This chapter includes published work with Michael Lustig [35].

# Chapter 2

# Complex-valued Deep Learning

During the last seven years, deep learning has fueled the current rejuvenation of artificial intelligence. The 2012 paper from Krizhevsky et al. [36] demonstrated that deep neural networks could be efficiently trained on GPUs with big data [37] to produce record-breaking results in the computer vision task of image classification. One of the primary reasons for the subsequent acceleration in deep learning research and development is the accessibility of neural network software frameworks. Shortly after the Krizhevsky paper, the neural network framework, Caffe, made it possible quickly reproduce results and build upon the latest deep learning research [38]. Caffe, a predecessor to deep learning frameworks TensorFlow (Google) and Caffe2 (Facebook), provided seamless GPU support, and being an open-source project, there was very little lag time between the publication of breakthrough techniques and when they were implemented as a layer/algorithm in Caffe.

While computer vision applications were quick to adopt deep learning, not all domains can leverage the benefits of existing neural network libraries. Domains with complex-valued data cannot directly integrate real-valued network software into their applications. Deep learning research is active in complex-valued applications, including radio frequency (RF) signal classification [39], synthetic aperture radar (SAR) target detection [40, 41], and MRI [12, 42], but the data passed to neural networks is either broken into independent real and imaginary channels or the phase of the complex-data is simply discarded. Without access to complex-valued implementations of deep learning software, alternative real-valued solutions are used and the benefits of retaining a complex representation remains an open question.

Initial work on complex-valued deep learning showed that complex networks could leverage mathematical properties of complex values, such as the complex identity theorem to learn more efficient transform functions than with real-valued networks [1]. These early works attempted to use various complex-valued activation functions [1, 17]. However, these attempts had limited success in applications to real-world settings.

More recent work in complex-valued deep learning aims to improve the convergence and stability of neural networks by building upon complex number theory, unitary weight matrices [18] and wavelet transforms [19, 20]. Yet another set of researchers have begun to generalize complex deep learning by recognizing that complex-values could be extended to

three dimensions as quaternions [21]. Most recently, complex-valued data are approached from a Riemannian manifold perspective, and new CNN layer functions are developed in order to preserve equivariance and invariance properties that are important for complex numbers [4].

In this chapter, our research takes a contrasting approach. Where prior work begins from mathematical principles and thence re-envisions deep learning, we take a systems-oriented approach. Starting from existing deep-learning architectures, we identify the limitations that prevent these approaches from incorporating complex-valued data; we then extend the existing systems with the missing mathematical foundations needed to support complex data. We present a specification for implementing modern complex-valued networks, including new complex activation functions and a reference to our associated implementation of complex-valued networks, built as an extension to the Caffe deep learning framework [32].

Later, in Chapter 3, we put complex networks to the test with the MR fingerprinting application, where we show that the proposed complex cardioid activation function enables complex networks to match and even exceed the performance of real-valued networks with separate input channels for real and imaginary components [31].

## 2.1 Complex numbers

Before getting into the motivation and background for complex-valued deep learning, we should first lay out the basics of complex numbers and the relationship to their various real-valued components.

A complex number, $z \in \mathbb{C}$, consists of a real component, $x \in \mathbb{R}$, and an imaginary component, $y \in \mathbb{R}$:

$$z = x + iy \tag{2.1}$$

where $i = \sqrt{-1}$. The same complex number may be written in exponential, or polar, form in terms of its magnitude, $m \in \mathbb{R}^+$, and its phase, $\theta \in \mathbb{R}$:

$$z = me^{i\theta} \tag{2.2}$$

The complex conjugate is a common operation used in complex arithmetic. The complex conjugate, $\bar{z}$, is a reflection of a complex point, $z$, over the real axis. It may be computed by negating the imaginary component or the phase of the input point:

$$\bar{z} = x - iy \tag{2.3}$$

$$\bar{z} = me^{-i\theta} \tag{2.4}$$

The real-valued components of a complex number may be extracted as follows:

$$\begin{array}{ccc} \text{Real} & x = \text{Re}(z) & x = \frac{1}{2}(z + \bar{z}) \\ \text{Imaginary} & y = \text{Im}(z) & y = \frac{1}{2i}(z - \bar{z}) \\ \text{Magnitude} & m = ||z|| & m = (z\bar{z})^{\frac{1}{2}} \\ \text{Phase} & \theta = \angle z & \theta = -i \log \frac{z}{||z||} \end{array} \tag{2.5}$$

## 2.2 Motivation

At first glance, complex numbers seem to be just an alternative way of representing a two-dimensional real vector. In that case, why would anyone bother with a complex representation when deep learning toolkits are already designed to handle vector data? We'll answer this question later in this motivation section, but first, let's discuss where complex values come into play in the context of neural network applications.

### 2.2.1 Where Do Complex Values Occur in Deep Learning Applications?

Opportunities to leverage complex-valued deep learning can arise in several different types of applications. Complex-values occur naturally in the data itself, in the complex spectral domain of real-valued applications, and in constructed complex representations. Additionally, complex representations have been used to enable more efficient learning in neural network models.

#### Naturally complex data

MRI measures the oscillating electrical current induced by changing magnetic fields within the body. The Fourier transform of this complex signal produces a complex-valued image containing the spatial location of different tissue types. While the magnitude of an MRI image pixel is often related to the quantity or characterization of different tissues, the phase cannot be disregarded as it can represent the velocity of blood flow, Figure 2.1, or contain valuable information aliased from other pixels during a fast, undersampled scan. Similarly, for the tasks of radio frequency signal classification [39, 4, 43], Figure 2.2, and synthetic-aperture radar (SAR) target detection [4], leveraging the relationship between real and imaginary components of the complex source signal can be critical.

#### Complex spectral domain

Even in applications with real-valued data, transformations to a complex domain can enable simpler or more effective data processing. Let us not forget that in classical image processing and computer vision, the complex-valued 2D Fourier transform converts image data to the spatial frequency domain where statistical algorithms, and in some cases, humans, are better able to process and interpret the data. Similarly, real-valued speech data is often transformed into complex spectrograms to visualize frequencies and harmonics. However, most methods discard the phase of this signal before processing. Recently, Fu et al. trained a neural network to denoise the spectrograms, but in this case, the real and imaginary signals were treated independently by the neural network [44].

Recent work on neural networks for graph-structured data [45, 46], has opened the door to deep learning solutions in a wide range of applications, including computer graphics [47],

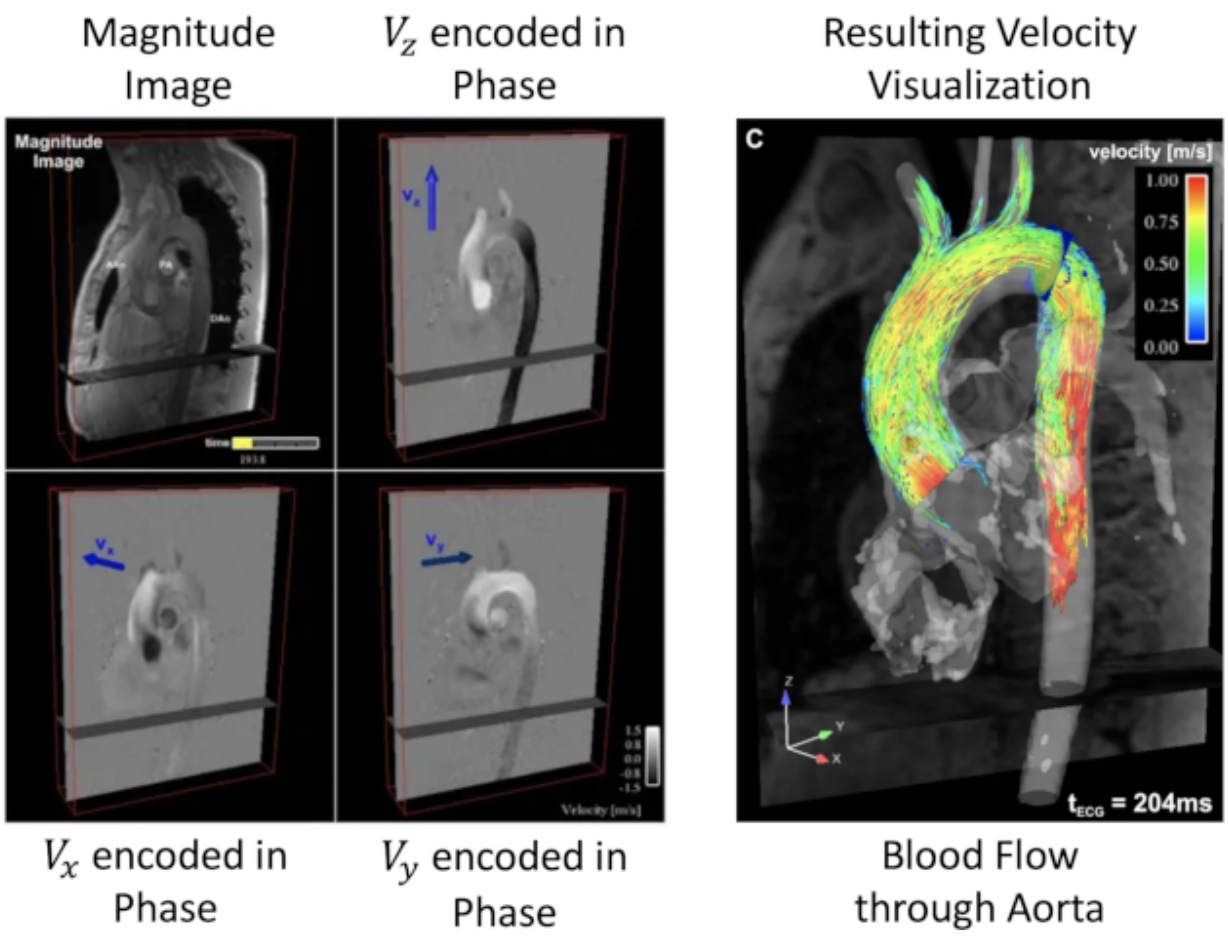


*Figure 2.1*: Visualization of blood flow, right, computed from velocity encoded in the phase of the MRI signal, left. Repeated scans are performed to encode in the phase the x-, y-, and z-components of the velocity vector. Images from [2].

quantum chemistry [48], and molecular biology [49]. A key aspect of doing convolutions on graphs is to use the graph Laplacian [46]. For undirected graphs, this spectral transformation produces all real-valued eigenvalues, while directed graphs lead to complex-valued eigenvalues [50].

### Constructed complex representations

Complex numbers can provide a natural representation for values that have a positive scalar intensity and a directional or cyclical relationship. The work from Marie et al. on AffinityCNN image segmentation uses a complex representation to simultaneously embed both the prediction confidence and the relative depth between neighboring pixels [51]. This complex embedding occurs at the end of their segmentation pipeline but could potentially

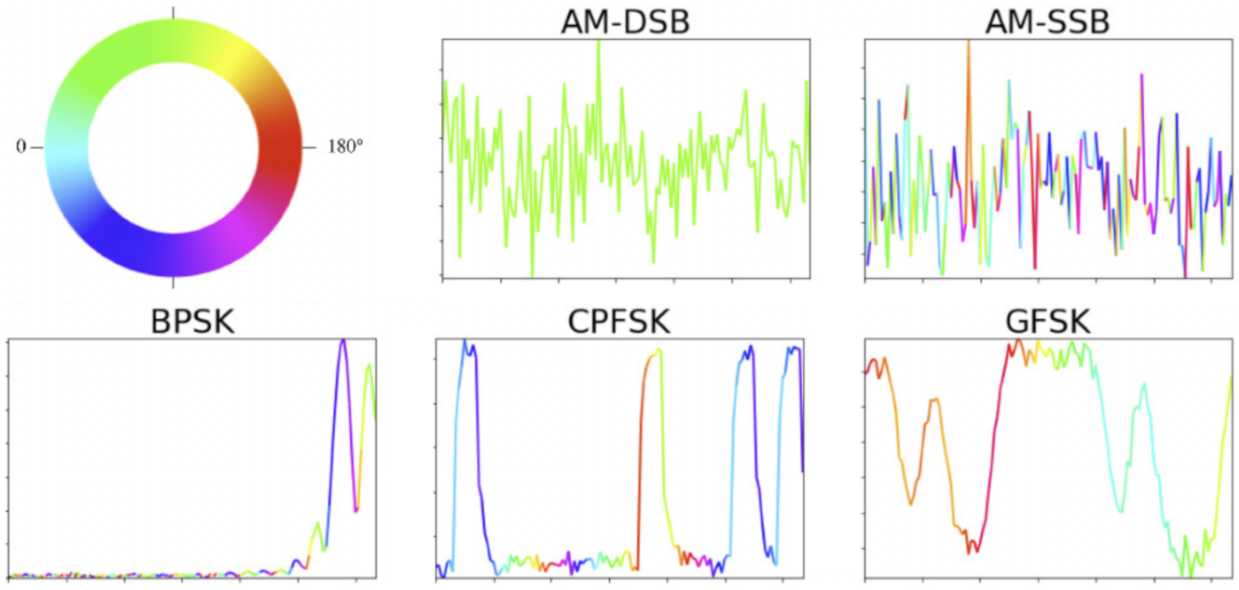


*Figure 2.2*: Example of time series of complex-valued radio frequency signals from the RadioML dataset [3]. This figure from [4] shows the magnitude (vertical axis) and color-coded phase of example radio frequency signals over time (horizontal axis).

benefit from an end-to-end network that optimizes over complex representations in the hidden layers as well. Amin and Murase employ an even simpler complex embedding, mapping a finite range of real values to the complex unit circle [52].

Rather than explicitly mapping real values to a complex representation, Cadieu and Oshausen train a complex-valued sparse coding model to convert real-valued videos into two features that capture edge structure and motion structure, where the latter is captured via the phase of their complex-valued sparse coding coefficients [53].

### 2.2.2 Why Not Use a Real Network with Two Channels?

Complex numbers are isomorphic to a two-dimensional vector consisting of real and imaginary components. Just like an image with three-dimensional RGB color vectors, the real and imaginary components could be fed into a real-valued neural network with multiple input channels. As in standard color image networks, the two input channels would be linearly combined as part of the first fully connected (or convolution) layer. Unfortunately, this real-valued inner-product layer operation dismisses the mathematical correlation between the real and imaginary channels.

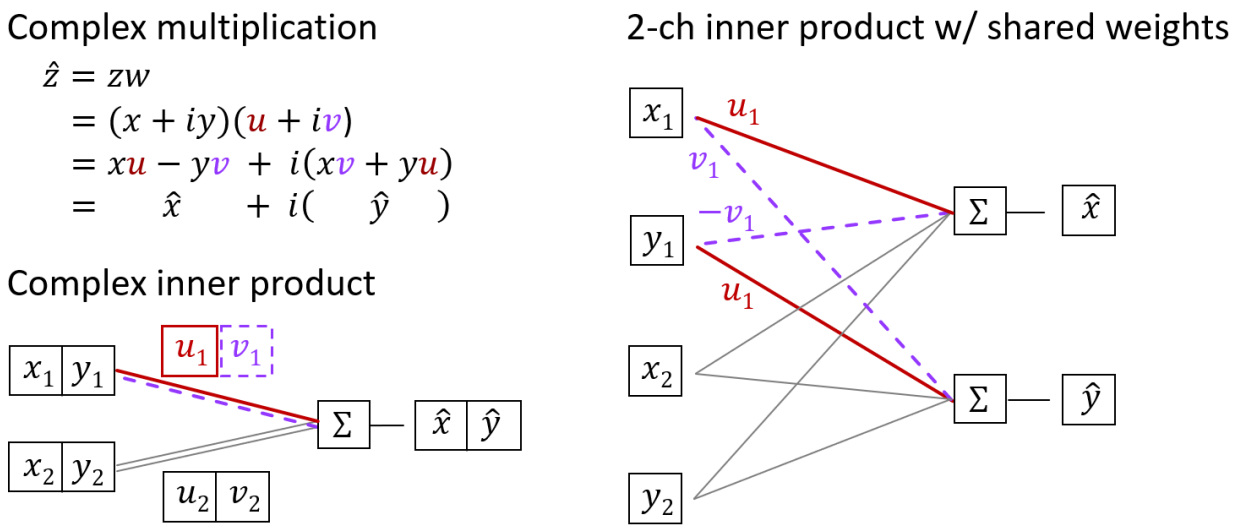


*Figure 2.3*: Illustration of the implicit shared weights in a complex inner product layer. In complex multiplication, the real and imaginary components of the factors affect both effect the real and the imaginary components of the output (left, top). In order to replicate the behavior of a complex-valued fully connected layer (left, bottom) in a real-valued network (right), the components of the weight must be replicated, or shared, in the 2-ch fully connected layer.

### Linear Operations

Complex multiplication defines a specific operation for combining the real and imaginary components to produce the real and imaginary components of the output, Figure 2.3, top left. It is possible to connect real-valued neural network components to exactly mimic the complex multiplication in the network's linear fully connected, or "inner product", layers, but as shown in Figure 2.3, right, the real and imaginary components of the weights must be shared, as they each affect both the real and the imaginary components of the output.

When complex numbers are written in exponential form in terms of their magnitude, $m$, and phase, $\theta$, it is apparent that multiplying one complex number, $z_1$, by another, $z_2$, has two effects: the magnitude is scaled by $m_2$ and the phase is rotated by $\theta_2$.

$$z_1 \coloneqq m_1 e^{i\theta_1} \tag{2.6}$$

$$z_2 \coloneqq m_2 e^{i\theta_2} \tag{2.7}$$

$$z_1 z_2 = m_1 e^{i\theta_1} m_1 e^{i\theta_1} \tag{2.8}$$

$$= m_1 m_2 e^{i(\theta_1 + \theta_2)} \tag{2.9}$$

When we contrast complex multiplication with scalar vector multiplication, we can see that complex multiplication naturally has two degrees of freedom, scaling and rotation, rather than the four degrees of freedom from multiplying by a 2x2 weight matrix, Figure 2.4. For

a more detailed discussion of complex representations, please see [16].

In applications such as MRI and SAR, the relative change in phase between neighboring pixels may be of importance and the absolute phase value irrelevant. Two-channel real networks that treat real and imaginary independently can be problematic if an application needs to be invariant to this arbitrary global complex scaling across an image [4].

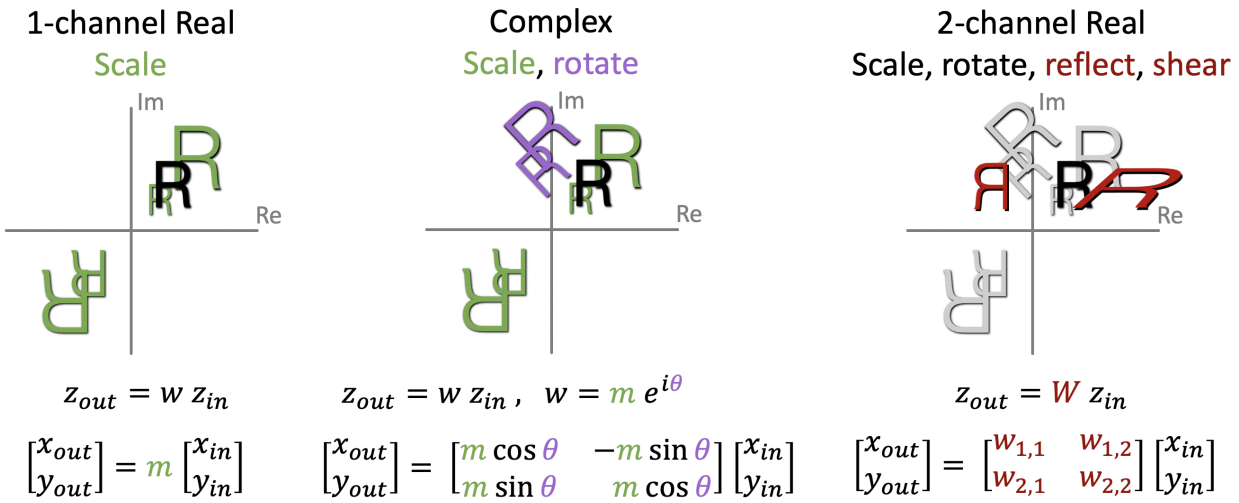


*Figure 2.4*: Illustration of the two degrees of freedom, scaling and rotation, in complex multiplication (center) compared to real-valued implementations (left and right). The set of points on the complex plane that make up the black letter 'R', $z_{in}$, may be changed in different ways depending on the network implementation of multiplication. With single-channel real weights (left), the effect on the input 'R' is only scaling. If implemented with a real-valued fully connected layer with two inputs and two outputs (right), there are four weight values to be learned, which is a more complicated model than complex multiplication (center).

### Non-linear Operations

While we can use shared weights to replicate complex multiplication in real-valued networks, Figure 2.3, non-linear activation functions for complex values are not as straightforward. As we discuss more below, standard neural network non-linearities are not trivially translated to complex values. When passing complex values into a 2-channel real-valued network, the non-linear activation functions in the network will operate on the different channels independently, which again dismisses the natural correlation between real and imaginary components in a complex number. For example, when applying the standard ReLU non-linearity independently to the real and imaginary components of a complex number, some complex values with large magnitude will be set to zero, which may not be the intent, Figure 2.5. Figure 2.5 also shows that the phase of the complex numbers is altered after applying

ReLU operations that were designed to either pass or block a signal, but not rotate it.

### 2.2.3 Why Not Directly Extend Real-valued Layers to Complex?

*Table 2.1*: Summary of issues regarding complex versions of common neural network layers.

| Layer Type | | Math | | Complex Conversion Concerns |
|---|---|---|---|---|
| Multiplication | Inner product | $\sum_i w_i z_i$ | ✓ | Well-defined, but 6x floating point operations: $(a+ib)(u+iv) = au-bv+i(av+bu)$ |
| | Convolution | $\sum_i w_i z_{r-i}$ | ✓ | |
| Activation | Sigmoid (also softmax, tanh) | $\frac{1}{1+e^{-z}}$ | ✗ | Unbounded (infinity at $i\pi(2N-1)$). Probabilistic interpretation lost. |
| | ReLU | $\max(0, z)$ | ✗ | Max function undefined in complex domain. |
| Pooling | Max | $\max_i z_i$ | ✗ | Max function undefined in complex domain. |
| | Average | $\frac{1}{N}\sum_i z_i$ | ✓ | Well-defined, but subject to poor performance as seen in real networks. |
| Normalization | Batch Norm. | $\hat{\mathbf{z}} = \frac{\mathbf{z}-\mathbb{E}[\mathbf{z}]}{\sqrt{\mathbb{V}[\mathbf{z}]+\epsilon}}$ | ✓ | Well-defined, including complex affine post processing, $\mathbf{y} = \gamma\hat{\mathbf{z}} + \beta$, which may be critical depending on the activation function. |
| Loss | Euclidean | $\|\|Y - Z\|\|_2^2$ | ✓ | Well-defined. |
| | Cross Entropy | $\sum_i y_i \log z_i$ | ✗ | Cannot optimize over complex output. |

Modern deep learning libraries have generic data structures that are agnostic to the specific numerical type. These data structures can hold complex data types in addition to real-valued floating point types. The structures are used to hold input and output data, learned parameters, and intermediate features/activations. The software interface for the network layers and optimization algorithms are designed to process these generic data structures, so perhaps we could just push our complex data through standard deep learning architecture implementations. Unfortunately, even assuming the underlying software implementation supports complex data types for each layer (which is not always the case), the mathematical operations within many of the network layers are undefined or unstable for

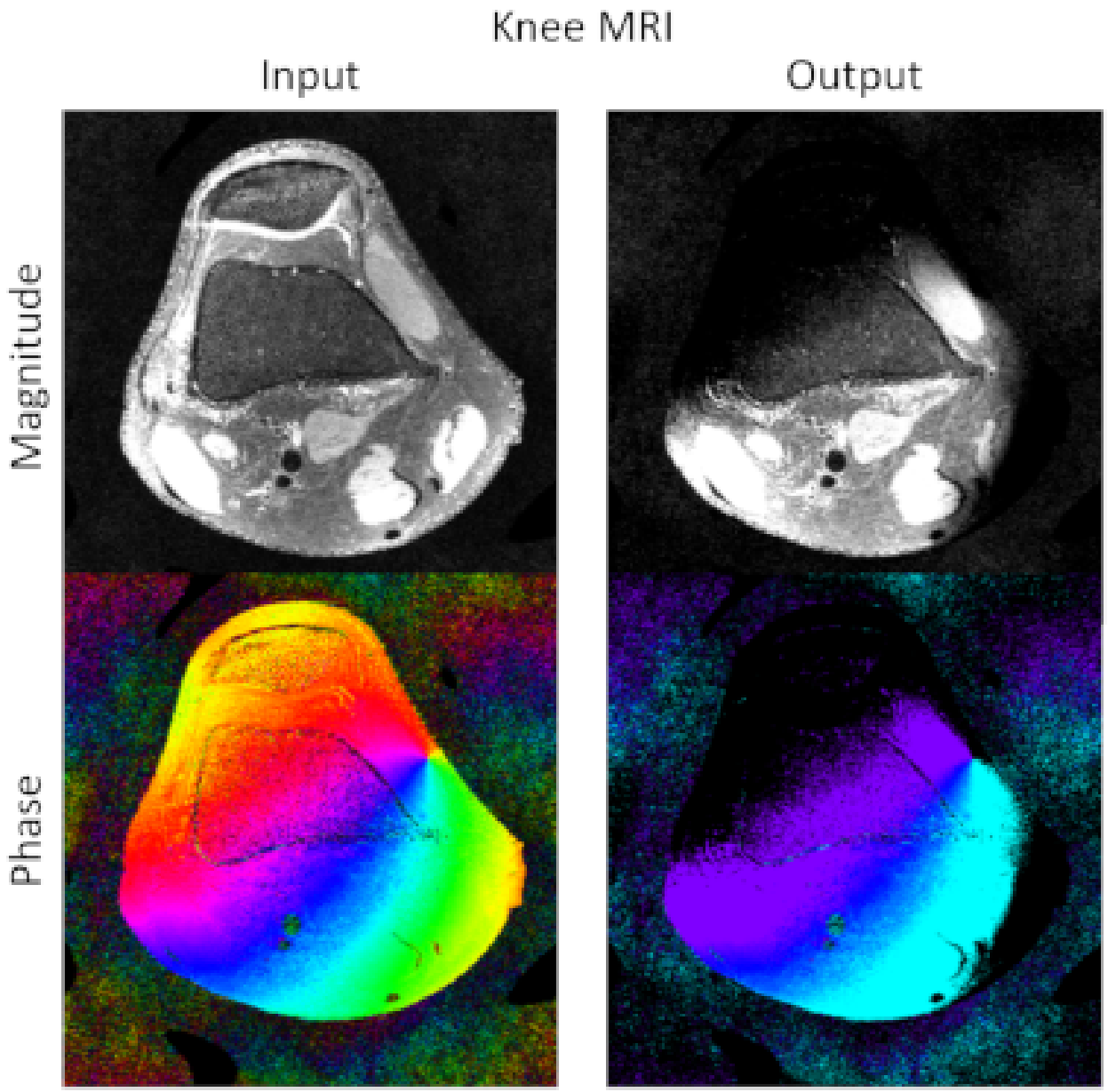


*Figure 2.5*: Effect of ReLU applied independently to real and imaginary channels of an axial slice of an MRI image. The phase of the image is rendered with a circular rainbow color map that is darkened towards black for complex values with very small magnitude. Note that in addition to attenuating pixels with high magnitude, the ReLU applied independently alters the phase of the input image, as seen by the larger purple and cyan regions in the output phase image. MRI knee data is from [5].

complex numbers. Table 2.1 provides a quick summary of which common neural network layers may be problematic.

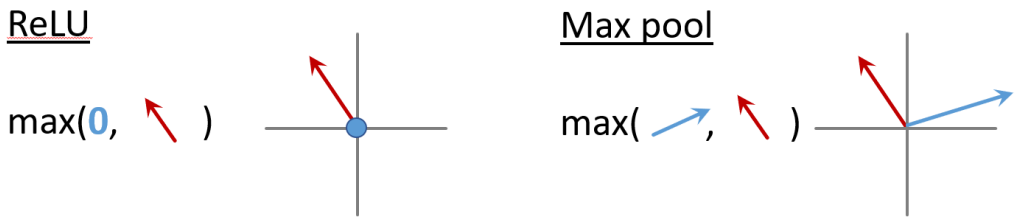


*Figure 2.6*: The maximum operator, found in ReLU and max pooling, is undefined for complex numbers.

#### Max undefined for complex values

ReLU and max pooling have become key non-linear functions in modern neural networks, providing numerical stability during optimization and an effective means to reduce the dimensionality of the data. Figure 2.6 display complex numbers as a two-dimensional vector to illustrate the problem with the max operation in these two functions. As with comparing 2D vectors, it is unclear how one would determine the max of two complex numbers. In order to use ReLU or max pooling layers, the max operation would need to be redefined, e.g. to take the max of the magnitude or the max of just the real component.

#### Unstable activations

The undefined ReLU activation could be replaced by the traditional activation functions sigmoid or tanh, but both of these functions have singularities along the imaginary axis of the complex plane, Figure 2.7. The softmax operation used for classification networks will also have to be redefined, constrained, or replace, as it is similarly unstable on the complex plane.

#### Lost probabilistic interpretation.

With complex input values and resulting complex output, the sigmoid function does not map values to the range [0, 1] required by any probability distribution. In other words, it no longer represents the inverse of the log odds models used in logistic regression. Likewise, the softmax function with complex input values does not represent a probability distribution over a set of different outcomes.

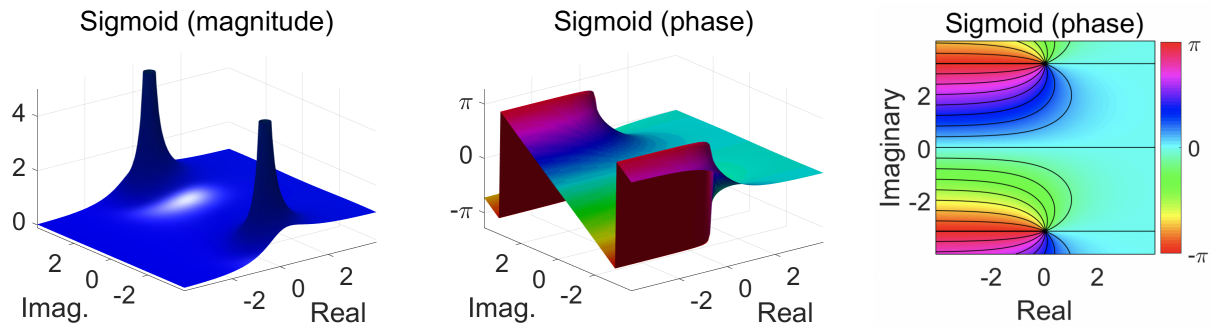


*Figure 2.7*: Sigmoid activation function defined over the complex plane. Output magnitude (left) goes to ±infinity at $i\pi(2N-1)$. These singularities may also be seen in the output phase (surface center; contour right).

**Optimization of complex-valued output.**

The loss function of a neural network cannot produce a complex value, because, as we discussed with max functions above, the minimization of complex values is undefined. If optimization of the loss function is undefined for complex-valued output, it is clear that we cannot simply reuse deep learning architectures designed for real values without first understanding the complex mathematics in neural networks forward and backward.

## 2.3 Complex Calculus Background

### 2.3.1 Complex-differentiable

The derivative of a complex-valued function, $f(z)$, at a point, $z_0 \in \mathbb{C}$ is defined by:

$$\left.\frac{df}{dz}\right|_{z_0} = \lim_{h\to 0} \frac{f(z_0+h)-f(z_0)}{(z_0+h)-z_0} \tag{2.10}$$

This is the same definition as real-valued functions, except that for a function to be complex-differentiable, this derivative definition must produce the same result regardless of the direction from which $h$ approaches zero. A function is called *holomorphic* if, for all points in its domain, it is complex-differentiable in a neighborhood around that point.

Unfortunately, even some seemingly simple functions are not holomorphic. Take, for example, the absolute square function:

$$f(z) = |z|^2 = \overline{z}z \tag{2.11}$$

where $\overline{z}$ is the complex conjugate of $z$. This norm-squared function, so often used in optimization, is not holomorphic. For instance, at $z_0 = 1$, when $h$ approaches zero from the

positive real axis, e.g. when $h = \epsilon = 1\text{e}{-10}$, the derivative is equal to two:

$$\frac{f(1+\epsilon) - f(1)}{(1+\epsilon) - 1} = \frac{\overline{(1+\epsilon)}(1+\epsilon) - 1}{\epsilon} = \frac{1 + 2\epsilon + \epsilon^2 - 1}{\epsilon} = 2 + \epsilon \to 2 \tag{2.12}$$

However, when $h$ approaches zero from the positive imaginary axis, $h = i\epsilon$, the derivative is equal to zero:

$$\frac{f(1+i\epsilon) - f(1)}{(1+i\epsilon) - 1} = \frac{(1-i\epsilon)(1+i\epsilon) - 1}{i\epsilon} = \frac{1 - i\epsilon + i\epsilon + \epsilon^2 - 1}{i\epsilon} = \frac{\epsilon}{i} \to 0 \tag{2.13}$$

In fact, according to Liouville's theorem, the only functions that are *bounded* (do not approach ±infinity) and *entire* (complex-differentiable in a neighborhood of every point in the finite complex plane) are constant functions. Flat, constant functions are not particularly useful as activation or loss functions in a neural network. That being said, just as subgradients enable us to optimize functions that are not differentiable across their domain, e.g. ReLU, we can leverage Wirtinger calculus to optimize functions that are not holomorphic but are differentiable with respect to their real and imaginary components.

### 2.3.2 Wirtinger/$\mathbb{CR}$ Calculus

As stated in Table 2.1, standard activation functions, such as sigmoid, are problematic because they are either undefined or unbounded when applied to the complex plane. As discussed in the previous section, requiring neural network functions to have complex-differentiable properties can be quite limiting. Fortunately, we can leverage Wirtinger calculus [54], or $\mathbb{CR}$ calculus [55], to do gradient descent on functions that are not holomorphic as long as they are differentiable with respect to their real and imaginary components. The first of the two Wirtinger calculus derivatives is the $\mathbb{R}$-derivative (or real derivative) which computes $\partial f/\partial z$ by treating $z$ as a real variable and holding instances of $\overline{z}$ constant. Likewise, the second derivative is the conjugate $\mathbb{R}$-derivative, $\partial f/\partial \overline{z}$, where $\overline{z}$ acts as a real variable and $z$ is held constant. Note: we will often use blue and purple color-coding when we want to highlight the different uses of the $\mathbb{R}$-derivative and the conjugate $\mathbb{R}$-derivative, respectively. To illustrate with a specific example, the magnitude function, $m$, of a complex variable, $z$, has the following Wirtinger derivatives:

$$m(z) := |z| = (\overline{z}z)^{\frac{1}{2}} \qquad \frac{\partial m}{\partial z} = \frac{1}{2}\frac{\overline{z}}{(\overline{z}z)^{\frac{1}{2}}} \qquad \frac{\partial m}{\partial \overline{z}} = \frac{1}{2}\frac{z}{(\overline{z}z)^{\frac{1}{2}}}$$

($\overline{z}$ treated as constant) ($z$ treated as constant)

Wirtinger derivatives are a convenient means to derive the calculus of complex functions without expanding the input variable into its real and imaginary components, $z = x + iy$, and then differentiating $f$ with respect to $x$ and $y$. For example, deriving the derivative for $f(z) = \overline{z}^2 z$ with respect to $\overline{z}$ (with $z$ held as a constant):

$$\frac{\partial f}{\partial \overline{z}} = 2\overline{z}z \tag{2.14}$$

is much more straight forward than the derivation with respect to $x$:

$$f(z) = f(x, y) = (x - iy)^2(x + iy) \tag{2.15}$$

$$\frac{\partial f}{\partial x} = 2(x - iy)(x + iy) + (x - iy)^2 \tag{2.16}$$

$$= 2x^2 + 2y^2 + x^2 - 2ixy - y^2 \tag{2.17}$$

$$= 3x^2 - 2ixy + y^2 \tag{2.18}$$

The $\mathbb{R}$-derivative and conjugate $\mathbb{R}$-derivatives of a real- or complex-valued function, $f$, are related to the derivatives with respect to the real and imaginary components, $x$ and $y$. From [55] Equation 9 and proved in [56]:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial x} - i\frac{\partial f}{\partial y}\right) \qquad \text{and} \qquad \frac{\partial f}{\partial \overline{z}} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}\right) \tag{2.19}$$

When working with composite complex functions, the chain rule for the $\mathbb{R}$-derivative and the conjugate $\mathbb{R}$-derivative is the same as the standard real-valued chain rule when you treat the complex variables and their conjugates as a pair of two separate values, Figure 2.3.2.

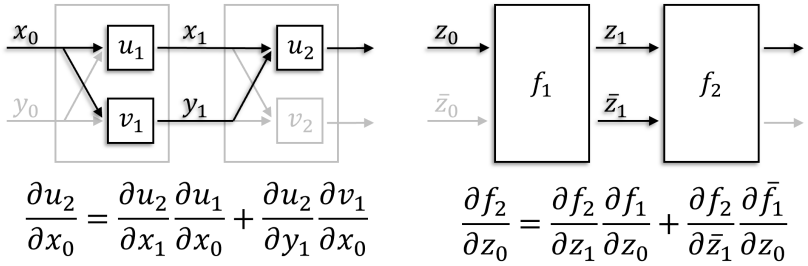


*Figure 2.8*: Illustration of complex chain rule when treating the input and output as real/imaginary pairs (left) or as variable/conjugate pairs (right). For the former, the derivative $\partial u_2/\partial x_0$ contains two components related to intermediate variables $x_1$ and $y_1$. Likewise, for the $\mathbb{R}$-derivative, the chain rule for $\partial f_2/\partial z_0$ has a term for the intermediate variable $z_1$ and another for its conjugate $\overline{z}_1$.

From [55] Equations 12-13, the complex derivative identities include:

$$\frac{\partial \overline{f}}{\partial \overline{z}} = \overline{\left(\frac{\partial f}{\partial z}\right)} \qquad\qquad \frac{\partial \overline{f}}{\partial z} = \overline{\left(\frac{\partial f}{\partial \overline{z}}\right)} \tag{2.20}$$

With these identities, the chain rule for the $\mathbb{CR}$ derivatives of the composite function $f \circ g$, where $g$ is a complex-valued function of $z$, is as follows:

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial z} + \frac{\partial f}{\partial \overline{g}}\overline{\left(\frac{\partial g}{\partial \overline{z}}\right)} \qquad\qquad \frac{\partial f}{\partial \overline{z}} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial \overline{z}} + \frac{\partial f}{\partial \overline{g}}\overline{\left(\frac{\partial g}{\partial z}\right)} \tag{2.21}$$

When a function has a real-valued output, $f(z) \in \mathbb{R}$, there is an additional pair of identities:

$$\frac{\partial f}{\partial z} = \overline{\left(\frac{\partial f}{\partial \overline{z}}\right)} \tag{2.22}$$

$$\frac{\partial f}{\partial \overline{z}} = \overline{\left(\frac{\partial f}{\partial z}\right)} \tag{2.23}$$

from [55] Equation 17. As we discuss more in Section 2.4.1, this identity greatly simplifies the computations in complex backpropagation, as the final loss function of any network is always real-valued.

### 2.3.3 Gradient Descent

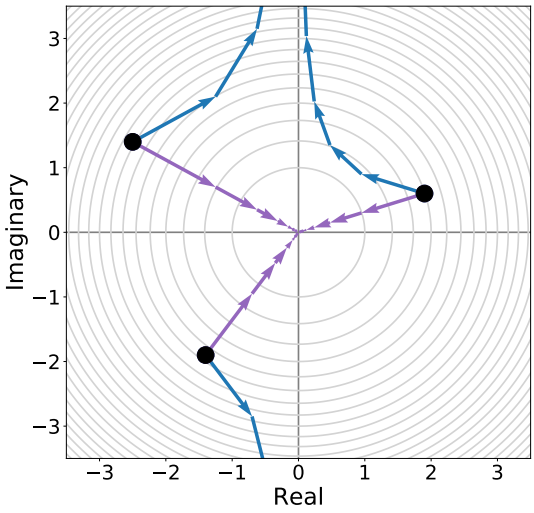


*Figure 2.9*: One-dimensional example of gradient descent minimizing the real-valued function $f(z) = \overline{z}z$: correctly, using the complex cogradient, $\nabla_{\overline{z}} f = z$, purple; and incorrectly, using the cogradient, $\nabla_z f = \overline{z}$, blue. The complex cogradient steps towards the minimizing point at the origin, while the cogradient (the conjugate of the complex gradient) moves in the wrong direction.

At the end of a neural network, there is a real-valued loss function that we are trying to optimize. We can update complex weight vectors, $\boldsymbol{w}$ by doing gradient descent with complex-valued gradients:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla_{\overline{\boldsymbol{w}}} f, \text{ where } \nabla_{\overline{\boldsymbol{w}}} f = \left[\partial f / \partial \overline{w}_1, \quad \ldots, \quad \partial f / \partial \overline{w}_n\right]^T \tag{2.24}$$

This is the same as gradient descent with real-valued gradients with careful attention paid to the gradient operator. As shown in [56] and illustrated in Figure 2.9, the direction of steepest descent is the complex cogradient, $\nabla_{\overline{w}} f$. Gradient descent extensions such as momentum and weight decay may be applied on top of this basic update equation.

## 2.4 Complex-valued Neural Networks

Neural network software libraries contain implementations of multidimensional data structures, network layers, and optimization algorithms. From Wirtinger calculus backpropagation to specific software implementation challenges, this section describes the details of complex-valued neural network components and how they relate to existing real-valued network implementations. We show that real-valued layers can be accurately used after complex layer functions. We describe prior works, as well as our proposed complex layers, including iGaussian and cardioid activation functions.

The network design presented in this section is based on our reference implementation, which is an extension of the Caffe deep learning library [38, 32], but it should be a valuable resource when considering complex-value support for other neural net frameworks.

We group complex network layers into four categories: loss functions, complex-to-real functions, activation functions, and other, well-defined complex functions. Table 2.2 summarizes the complex network functions that we have implemented in our extension to Caffe [32]. As shown in Section 2.4.2, all of these layers are designed to interact with existing real-valued layers commonly implemented in deep learning libraries.

### Notation

In general, we denote a complex scalar $z \in \mathbb{C}$ as $z = x + iy$, where $x \in \mathbb{R}$ and $y \in \mathbb{R}$ are its scalar real and imaginary components, respectively. When referring to vector values, we will use bold font, such as $\mathbf{z} = \mathbf{x} + i\mathbf{y}$, $\mathbf{z} \in \mathbb{C}^N$, $\mathbf{x} \in \mathbb{R}^N$, and $\mathbf{y} \in \mathbb{R}^N$. To simplify notation, we will refer to the partial derivative of a single input and output of a multidimensional function rather than defining each element of the gradient or Jacobian. In these cases, we often drop the notation for the index of the vector or matrix, e.g. $\partial f / \partial z$ rather than $\partial f_i / \partial z_j$.

We denote the function of the $\ell$-th layer of a neural network as:

$$\mathbf{z}_\ell = \boldsymbol{f}_\ell(\mathbf{z}_{\ell-1}, \mathbf{w}_\ell) \tag{2.25}$$

where $\mathbf{z}_{\ell-1}$ is the output of the previous layer and $\mathbf{w}_\ell \in \mathbb{C}^M$ is the vector of any learnable weight parameters in the $\ell$-th layer. To again simplify notation, when there is no loss of generality we will refer to the scalar version of the layer function, dropping the bold type face.

We define a neural network as a series of composite functions that input data, $\mathbf{z}_0$ and ultimately output the loss value:

$$f_L \circ f_{L-1} \cdots f_2 \circ f_1 \tag{2.26}$$

*Table 2.2*: Summary of complex network functions.

| **Category** | **Layer** | **Function** | **Complex Network Source/ Section** |
|---|---|---|---|
| Real-valued Loss | Euclidean | $\|\|Y - Z\|\|_2^2$ | [57, 58], 2.4.1 |
| Complex to Real | Magnitude | $\|z\| = (\overline{z}z)^{1/2}$ | 2.4.2 |
| | Phase | $-i \log e^{i\angle z}$ | 2.4.2 |
| | Real component | $\frac{1}{2}(\overline{z} + z)$ | 2.4.2 |
| | Imaginary component | $\frac{1}{2i}(z - \overline{z})$ | 2.4.2 |
| Activation | Separable sigmoid | $g\big(\mathrm{Re}(z)\big) + ig\big(\mathrm{Im}(z)\big)$, $g(x) = \frac{1}{1+e^{-x}}$ | [1], 2.4.3 |
| | Siglog | $\frac{z}{c + \frac{1}{r}\|z\|}$ | [17], 2.4.3 |
| | iGaussian | $\left(1 - e^{-\overline{z}z/2\sigma^2}\right)\frac{z}{\|z\|}$ | 2.4.3 |
| | Cardioid | $\frac{1}{2}(1 + \cos(\angle z))z$ | 2.4.3 |
| Well-defined | Fully connected/ | $\sum_i w_i z_i + b$ | [57, 58], 2.4.4 |
| | Convolution/ Deconvolution | $\sum_i w_i z_{r-i} + b$ | 2.4.4 |
| | Max pooling (mag) | $z_n$, $n = \mathrm{argmax}_k\{\|z_k\|\}$ | 2.4.4 |
| | Average pooling | $\frac{1}{N}\sum_n\{z_n\}$ | 2.4.4 |
| | Batch normalization | $\hat{\mathbf{z}} = \frac{\mathbf{z} - \mathbb{E}[\mathbf{z}]}{\sqrt{\mathbb{V}[\mathbf{z}] + \epsilon}}$ | 2.4.4 |
| | Complex normalization | $\frac{z}{\|z\|} = e^{i\angle z}$ | 2.4.4 |
| | Dropout | $\frac{1}{p}z_n$, if $n \in$ mask | 2.4.4 |

We will refer to the final loss function in the network as $f_L$ and $L$, interchangeably.

### 2.4.1 Backpropagation with Real-valued Loss

During backpropagation, each network layer must leverage the gradient computation of later layers to compute the gradient with respect to its input as well as with respect to any of its learned parameters. In real-valued networks, this involves applying the chain rule multiply the upstream derivative times the local derivative for that layer function, Table 2.3, left. With complex numbers and Wirtinger/$\mathbb{CR}$ calculus, there now two derivatives to consider instead of one, Table 2.3, center. To complicate matters, the chain rule to compute both of these derivatives has two terms rather than one, causing more than four times as many calculations during the backward pass.

Fortunately, we can significantly reduce both memory and computation time during complex backpropagation by assuming that our final network loss function produces a real-valued output. This is a valid assumption because the loss is the value that we are trying to minimize and, as we discussed in Section 2.2.3, minimizing a complex value rather than real is an undefined operation. With the real-valued loss assumption and Equation 2.23, backpropagation only needs to pass one of the two $\mathbb{CR}$ derivatives back to the earlier layers 2.3, right.

Note that although only one of two derivatives is passed backward, both the $\mathbb{R}$-derivative and the conjugate $\mathbb{R}$-derivative of the layer function are still needed to compute the derivative of the final layer for the layer input or layer weight. Figure 2.10 illustrates the activations, weights, and derivatives that flow into and out of each layer during the forward and backward passes.

From the gradient descent step in Equation 2.24, it is ultimately the conjugate $\mathbb{R}$-derivative that is needed, so the $\mathbb{R}$-derivative can be dropped. If the software implementation of gradient descent generically supports complex types, the weight update function will not need to be changed as long as the conjugate $\mathbb{R}$-derivative is stored in the weight gradient variables:

$$w_n \leftarrow w_n - \alpha \partial f / \partial \overline{w}_n \tag{2.27}$$

Even if the software framework generally treats complex-values arrays 2-channel floating point arrays, as in our complex Caffe implementation, we can still reuse the real-valued solver to update the real and imaginary components of the weights, assuming the weight gradients are also stored in 2-channel floating point arrays:

$$w_{n,real} \leftarrow \mathrm{Re}(w_n) - \alpha\, \mathrm{Re}(\partial f / \partial \overline{w}_n) \tag{2.28}$$

$$w_{n,imag} \leftarrow \mathrm{Im}(w_n) - \alpha\, \mathrm{Im}(\partial f / \partial \overline{w}_n) \tag{2.29}$$

#### Euclidean Loss

A real-valued implementation of the Euclidean loss layer is one of the few layers that could be used as-is in the complex-valued network. The real-valued forward and backward

*Table 2.3*: Comparison of backpropagation calculus.

| **Standard Real calculus** | **Complex calculus** | **Complex calculus, Assuming real-valued loss** |
|---|---|---|
| Input from layer $\ell+1$: | | |
| $\frac{\partial f_L}{\partial x_\ell}$ | $\frac{\partial f_L}{\partial z_\ell}$ and $\frac{\partial f_L}{\partial \overline{z}_\ell}$ | $\frac{\partial f_L}{\partial \overline{z}_\ell}$ |
| Output from layer $\ell$: | | |
| $\frac{\partial f_L}{\partial x_{\ell-1}} = \frac{\partial f_L}{\partial x_\ell}\frac{\partial f_\ell}{\partial x_{\ell-1}}$ | $\frac{\partial f_L}{\partial z_{\ell-1}} = \frac{\partial f_L}{\partial z_\ell}\frac{\partial f_\ell}{\partial z_{\ell-1}} + \frac{\partial f_L}{\partial \overline{z}_\ell}\overline{\left(\frac{\partial f_\ell}{\partial \overline{z}_{\ell-1}}\right)}$ | |
| | $\frac{\partial f_L}{\partial \overline{z}_{\ell-1}} = \frac{\partial f_L}{\partial z_\ell}\frac{\partial f_\ell}{\partial \overline{z}_{\ell-1}} + \frac{\partial f_L}{\partial \overline{z}_\ell}\overline{\left(\frac{\partial f_\ell}{\partial z_{\ell-1}}\right)}$ | $\frac{\partial f_L}{\partial \overline{z}_{\ell-1}} = \overline{\left(\frac{\partial f_L}{\partial \overline{z}_\ell}\right)}\frac{\partial f_\ell}{\partial \overline{z}_{\ell-1}} + \frac{\partial f_L}{\partial \overline{z}_\ell}\overline{\left(\frac{\partial f_\ell}{\partial z_{\ell-1}}\right)}$ |

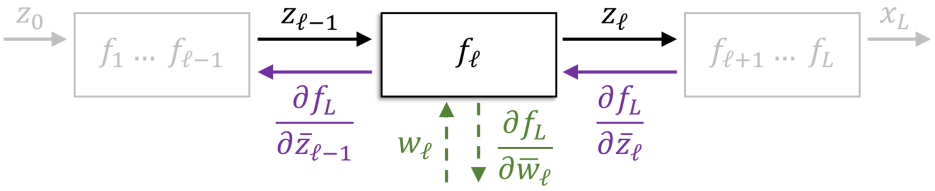


*Figure 2.10*: Layer $\ell$ input and output as well as derivatives required during backpropagation.

calculations are equivalent to a complex version of the layer, assuming that the real-valued layer interprets the complex input, as well as the output derivative, as just a 2x length vector of real components concatenated to the imaginary components.

The complex Euclidean loss function of the network prediction vector $\mathbf{z} = \mathbf{x} + i\mathbf{y}$, given the fixed ground truth vector, $\mathbf{z}_{gt} = \mathbf{x}_{gt} + i\mathbf{y}_{gt}$ is:

$$L = L(\mathbf{z}; \mathbf{z}_{gt}) = \frac{1}{2}||\mathbf{z} - \mathbf{z}_{gt}||_2^2 \tag{2.30}$$

$$= \frac{1}{2}||[\mathbf{x}^T, \mathbf{y}^T]^T - [\mathbf{x}_{gt}^T, \mathbf{y}_{gt}^T]^T||_2^2 \tag{2.31}$$

$$= L_{real}([\mathbf{x}^T, \mathbf{y}^T]^T; [\mathbf{x}_{gt}^T, \mathbf{y}_{gt}^T]^T) \tag{2.32}$$

See the appendix, Section A.8 for the expanded derivation.

The backward pass of a real-valued Euclidean loss layer with input $[\text{Re}(\mathbf{z})^T, \text{Im}(\mathbf{z})^T]^T$ produces the real and imaginary components of the complex Euclidean conjugate $\mathbb{R}$-derivative,

but with a 2x scaling factor:

$$\frac{\partial L}{\partial \overline{\mathbf{z}}} = \frac{1}{2}(\mathbf{z} - \mathbf{z}_{gt}) \tag{2.33}$$

$$\frac{\partial L_{real}}{\partial \mathbf{x}} = \mathbf{x} - \mathbf{x}_{gt} \tag{2.34}$$

$$\frac{\partial L_{real}}{\partial [\mathbf{x}^T, \mathbf{y}^T]^T} = [\mathbf{x}^T - \mathbf{x}_{gt}^T, \mathbf{y}^T - \mathbf{y}_{gt}^T]^T \tag{2.35}$$

$$= [\mathrm{Re}(\mathbf{z} - \mathbf{z}_{gt})^T, \mathrm{Im}(\mathbf{z} - \mathbf{z}_{gt})^T]^T \tag{2.36}$$

$$= \left[ \mathrm{Re}\left(2\frac{\partial L}{\partial \overline{\mathbf{z}}}\right)^T, \mathrm{Im}\left(2\frac{\partial L}{\partial \overline{\mathbf{z}}}\right)^T \right]^T \tag{2.37}$$

There is an extra factor of two in the derivative when using the real-valued implementation, but this can be accounted for during optimization with a gradient descent learning rate of half the size.

### 2.4.2 Hybrid Real/Complex Networks

The output of the loss function of complex-valued neural networks must be a real-valued scalar to allow optimization algorithms to minimize that loss value. Thus, at some point in a complex-valued neural network, the complex values must be converted to real values. In some network architectures, this complex to real conversion may happen at the final layer in the loss function, for example with Euclidean loss functions. However, if the complex values are converted to real prior to the loss function, the network will be a hybrid of complex-valued layers and real-valued layers. This begs the question, do we have to re-implement the real-valued layers to return the conjugate $\mathbb{R}$-derivatives during backpropagation? In other words, *can we use real-valued implementations for any network layers after the complex activations have been converted to real values?* As we show below, the answer is, fortunately, yes.

To use real-valued layer implementations after complex layers, we just need to account for this transition in any layers that convert complex values to real values.

**Theorem 2.1.** *Given a network layer function, $f_\ell(z_{\ell-1}) \to x_\ell$, which maps complex-valued input, $z_{\ell-1}$, to real-valued output, $x_\ell$, the conjugate $\mathbb{R}$-derivative of the final network layer function, $f_L$, with respect to $z_{\ell-1}$ is:*

$$\frac{\partial f_L}{\partial \overline{z_{\ell-1}}} = \frac{\partial f_L}{\partial x_\ell} \frac{\partial f_\ell}{\partial \overline{z_{\ell-1}}} \tag{2.38}$$

*Proof.* From [55] Equation 9, for an $\mathbb{R}$-differentiable function $f_L$ with input $z_\ell \coloneqq x_\ell + iy_\ell$,

$$\frac{\partial f_L}{\partial z_\ell} = \frac{1}{2}\left(\frac{\partial f_L}{\partial x_\ell} - i\frac{\partial f_L}{\partial y_\ell}\right) \text{ and} \tag{2.39}$$

$$\frac{\partial f_L}{\partial \overline{z_\ell}} = \frac{1}{2}\left(\frac{\partial f_L}{\partial x_\ell} + i\frac{\partial f_L}{\partial y_\ell}\right) \tag{2.40}$$

When $f_L : \mathbb{R} \to \mathbb{R}$, its derivative with respect to a nonexistent imaginary component is zero, $\partial f_L / \partial y_\ell = 0$, so:

$$\frac{\partial f_L}{\partial z_\ell} = \frac{1}{2}\frac{\partial f_L}{\partial x_\ell} \text{ and} \tag{2.41}$$

$$\frac{\partial f_L}{\partial \overline{z_\ell}} = \frac{1}{2}\frac{\partial f_L}{\partial x_\ell} \tag{2.42}$$

Leveraging Equations 2.41 and 2.42 along with Equation 2.23 for the real-valued $f_\ell$, we can start with the $\mathbb{CR}$ calculus chain rule, [55] Equation 16, and modify the conjugate $\mathbb{R}$-derivative of the final network layer function $f_L$ with respect to $z_{\ell-1}$ as follows:

$$\frac{\partial f_L}{\partial \overline{z_{\ell-1}}} = \frac{\partial f_L}{\partial z_\ell}\frac{\partial f_\ell}{\partial \overline{z_{\ell-1}}} + \frac{\partial f_L}{\partial \overline{z_\ell}}\overline{\left(\frac{\partial f_\ell}{\partial z_{\ell-1}}\right)} \tag{2.43}$$

$$= \frac{1}{2}\frac{\partial f_L}{\partial x_\ell}\frac{\partial f_\ell}{\partial \overline{z_{\ell-1}}} + \frac{1}{2}\frac{\partial f_L}{\partial x_\ell}\overline{\left(\frac{\partial f_\ell}{\partial z_{\ell-1}}\right)} \tag{2.44}$$

$$= \frac{1}{2}\frac{\partial f_L}{\partial x_\ell}\frac{\partial f_\ell}{\partial \overline{z_{\ell-1}}} + \frac{1}{2}\frac{\partial f_L}{\partial x_\ell}\frac{\partial f_\ell}{\partial \overline{z_{\ell-1}}} \tag{2.45}$$

$$= \frac{\partial f_L}{\partial x_\ell}\frac{\partial f_\ell}{\partial \overline{z_{\ell-1}}} \tag{2.46}$$

■

Though this theorem may be the intuitive way to combine complex and real deep learning components, it is important to show that we are indeed using the derivatives of existing real-valued infrastructure correctly with any complex-valued layers.

In addition to the Euclidean loss mentioned above, functions to extract the magnitude, phase, and real and imaginary components of a complex number are examples of operations that convert complex-valued input to real-valued output. We have implemented these four core operations as layers in our complex network reference implementation. We specify the forward operation for these functions below and use Theorem 2.1 to derive the backward pass gradients.

### Magnitude Layer

The forward function for the complex magnitude layer with complex scalar input $z$ and real scalar output $\hat{z} := \hat{x} + i\hat{y}$ is as follows:

$$\hat{z} = m(z) = (\overline{z}z)^{1/2} \tag{2.47}$$

Note that because $m(z) \in \mathbb{R}^+$, the imaginary component of $\hat{z}$ is zero, thus $\hat{z} = \hat{x}$.

Local $\mathbb{CR}$ derivatives of $m$:

$$\frac{\partial m}{\partial z} = \frac{1}{2}\frac{\overline{z}}{(\overline{z}z)^{\frac{1}{2}}} = \frac{1}{2}\frac{\overline{z}}{m(z)} \qquad \frac{\partial m}{\partial \overline{z}} = \frac{1}{2}\frac{z}{(\overline{z}z)^{\frac{1}{2}}} = \frac{1}{2}\frac{z}{m(z)} \tag{2.48}$$

As expected from Equation 2.23,

$$\frac{\partial m}{\partial z} = \overline{\left(\frac{\partial m}{\partial \overline{z}}\right)} \tag{2.49}$$

Because $m(z) : \mathbb{C} \to \mathbb{R}$, the conjugate $\mathbb{R}$-derivative of final layer with respect to the conjugates of the input follows Theorem 2.1:

$$\frac{\partial L}{\partial \overline{z}} = \frac{\partial L}{\partial \hat{x}}\frac{\partial m}{\partial \overline{z}} \tag{2.50}$$

$$= \frac{1}{2}\frac{\partial L}{\partial \overline{\hat{x}}}\frac{z}{m(z)} \tag{2.51}$$

### Phase Layer

To compute the phase of complex scalar $z$, we can normalize $z$ to have unit magnitude and then take the log and divide by $i$:

$$\hat{z} = f(z) = \frac{1}{i}\log e^{i\angle z} = -i\log\frac{z}{|z|} \tag{2.52}$$

Local $\mathbb{CR}$ derivatives of $f(z)$:

$$\frac{\partial f}{\partial z} = \frac{-i}{2z} \qquad \frac{\partial f}{\partial \overline{z}} = \frac{i}{2\overline{z}} \tag{2.53}$$

See the appendix, Section A.2 for the expanded derivation.

Because $f(z) : \mathbb{C} \to \mathbb{R}$, the conjugate $\mathbb{R}$-derivative of final layer with respect to the conjugates of the input follows Theorem 2.1:

$$\frac{\partial L}{\partial \overline{z}} = \frac{\partial L}{\partial \hat{x}}\frac{\partial f}{\partial \overline{z}} \tag{2.54}$$

$$= \frac{\partial L}{\partial \overline{\hat{x}}}\frac{i}{2\overline{z}} \tag{2.55}$$

**Real Component**

The function returning the real component of the complex scalar input $z$ is:

$$\hat{z} = \mathrm{Re}(z) = \frac{1}{2}(z + \overline{z}) \tag{2.56}$$

Local $\mathbb{CR}$ derivatives of $\mathrm{Re}(z)$:

$$\frac{\partial \,\mathrm{Re}}{\partial z} = \frac{1}{2} \qquad\qquad \frac{\partial \,\mathrm{Re}}{\partial \overline{z}} = \frac{1}{2} \tag{2.57}$$

Because $\mathrm{Re}(z) : \mathbb{C} \to \mathbb{R}$, the conjugate $\mathbb{R}$-derivative of final layer with respect to the conjugates of the input follows Theorem 2.1:

$$\frac{\partial L}{\partial \overline{z}} = \frac{\partial L}{\partial \hat{x}} \frac{\partial \,\mathrm{Re}}{\partial \overline{z}} \tag{2.58}$$

$$= \frac{1}{2} \frac{\partial L}{\partial \overline{\hat{x}}} \tag{2.59}$$

**Imaginary Component**

The function returning the imaginary component of the complex scalar input $z$ is:

$$\hat{z} = \mathrm{Im}(z) = \frac{1}{2i}(z - \overline{z}) \tag{2.60}$$

Local $\mathbb{CR}$ derivatives of $\mathrm{Im}(z)$:

$$\frac{\partial \,\mathrm{Im}}{\partial z} = \frac{1}{2i} \qquad\qquad \frac{\partial \,\mathrm{Im}}{\partial \overline{z}} = \frac{i}{2} \tag{2.61}$$

Because $\mathrm{Im}(z) : \mathbb{C} \to \mathbb{R}$, the conjugate $\mathbb{R}$-derivative of final layer with respect to the conjugates of the input follows Theorem 2.1:

$$\frac{\partial L}{\partial \overline{z}} = \frac{\partial L}{\partial \hat{x}} \frac{\partial \,\mathrm{Im}}{\partial \overline{z}} \tag{2.62}$$

$$= \frac{i}{2} \frac{\partial L}{\partial \overline{\hat{x}}} \tag{2.63}$$

### 2.4.3 Complex Activation Functions

A significant facet of complex network research since the early 1990s has been to overcome the issue that standard real-valued non-linear layers do not transfer well to complex-valued networks. Standard non-linear layer functions are either unbounded (e.g. $sigmoid(i\pi)$,

Figure 2.7) or undefined (e.g. the max operator in max pooling and ReLU, Figure 2.6). Additionally, with complex outputs, we have lost the probabilistic interpretations that functions like sigmoid and softmax provide. Without an obvious solution to this issue, past research has provided a range of potential solutions, for example, limiting the domain of the activation input to avoid unbounded regions [57], or applying non-linearities to real and imaginary components separately [1, 52]. In this section, we step through several complex activation functions found in other work and then propose additional complex non-linear functions.

### Prior Works

#### Complex Sigmoid

In 1990 and 1991, respectively, Kim & Guest and Leung & Haykin introduced the generalization of the neural network backpropagation calculus for complex values [58, 57]. In these works, they leveraged the popular sigmoid activation function in their complex-valued neural network, simply passing complex values through this nonlinearity rather than only real numbers:

$$\hat{z} = g(z) = \frac{1}{1 + e^{-z}} \tag{2.64}$$

Leung & Haykin did recognize that the sigmoid function goes to plus/minus infinity periodically on the imaginary axis of the complex plane, Figure 2.7, left. They accounted for this instability by limiting the domain of the values passed to the activation function; though it is not clear what mechanism they used to enforce this constraint.

The $\mathbb{R}$-derivative and conjugate $\mathbb{R}$-derivative are as follows:

$$\frac{\partial g}{\partial z} = \frac{e^{-z}}{(1 + e^{-z})^2} = g(z)(1 - g(z)) \qquad \frac{\partial g}{\partial \overline{z}} = 0 \tag{2.65}$$

The $\mathbb{R}$-derivative is the same as the standard real derivative, while, as Leung & Haykin noted, the conjugate $\mathbb{R}$-derivative is zero. This is because the sigmoid function is defined only in terms of $z$ and not $\overline{z}$.

#### Separable Sigmoid

Rather than constraining the complex domain to avoid unstable points in the sigmoid activation function, Nitta applied a real-valued sigmoid independently to the real and imaginary components, placing the respective results into the corresponding channels of the activation output [1]:

$$f(z) = g\big(\operatorname{Re}(z)\big) + ig\big(\operatorname{Im}(z)\big), \text{ where } g(x) = \frac{1}{1 + e^{-x}} \tag{2.66}$$

The $\mathbb{R}$-derivative and conjugate $\mathbb{R}$-derivative are as follows:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\Big(h\big(\operatorname{Re}(z)\big) + h\big(\operatorname{Im}(z)\big)\Big) \tag{2.67}$$

$$\frac{\partial f}{\partial \overline{z}} = \frac{1}{2}\Big(h\big(\mathrm{Re}(z)\big) - h\big(\mathrm{Im}(z)\big)\Big) \tag{2.68}$$

where $h(z) = g(z)(1-g(z))$ is the $\mathbb{R}$-derivative of the sigmoid function. Derivations of these derivatives are included in the appendix, Section A.3.

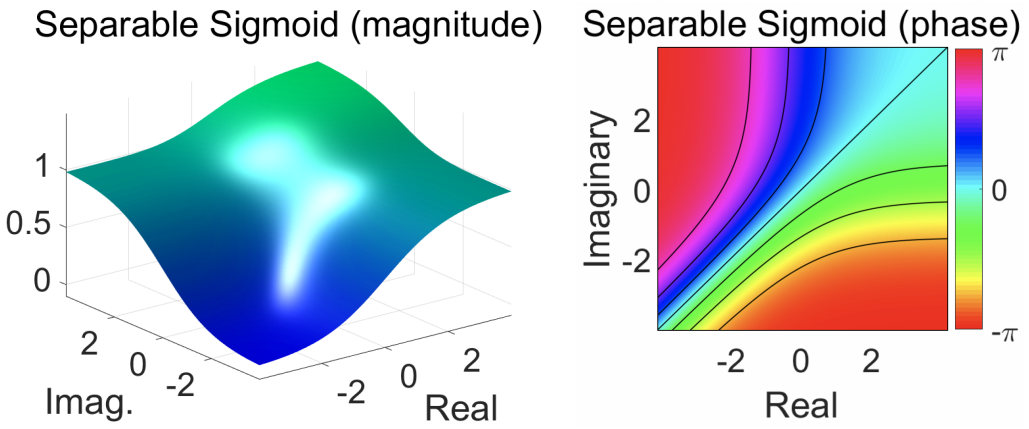


*Figure 2.11*: Magnitude (left) and phase (right) output of the separable sigmoid activation function applied to the complex plane.

Note that this separable sigmoid changes the phase of the signal to be in the range $[0, 2\pi]$ because the function output always has positive real and imaginary components, Figure 2.11, right and Figure 2.12, center.

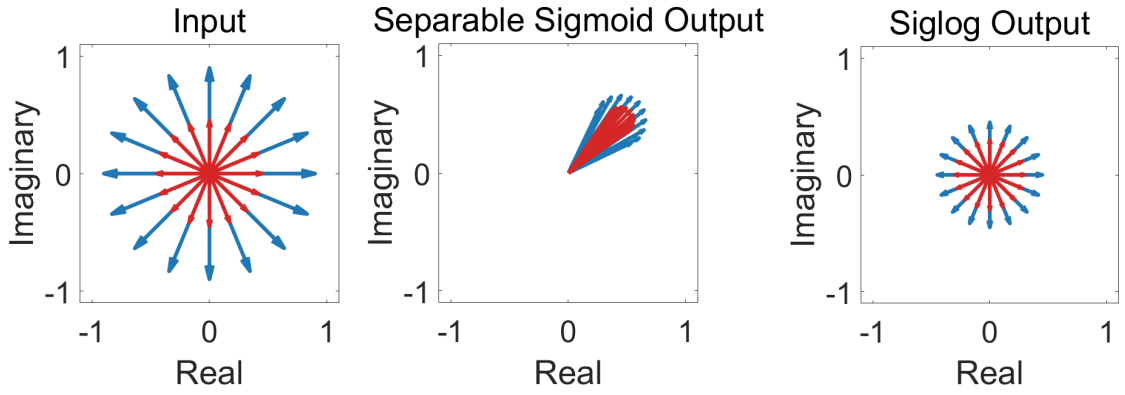


*Figure 2.12*: A set of 16 complex values, eight with magnitude 0.5 and eight with magnitude 1.0, left, are passed to separable sigmoid (center) and siglog (right) activation functions. Note that the separable sigmoid function maps the input values all to the first quadrant, while siglog preserves the input phase.

In 2009, Amin and Murase proposed applying the sigmoid separately to the real and imaginary components but combining the results to produce a real-valued output [52]. Specifically,

they propose the following two combinations of sigmoid on the real and imaginary:

$$f_1(z) = |g\big(\operatorname{Re}(z)\big) + ig\big(\operatorname{Im}(z)\big)| \tag{2.69}$$

$$f_2(z) = \Big(g\big(\operatorname{Re}(z)\big) - g\big(\operatorname{Im}(z)\big)\Big)^2 \tag{2.70}$$

They use these activation functions to solve real-valued classification problems with complex-valued neural networks, where they embed the real-valued input on to the complex unit circle before sending it through the network.

### Siglog

In their 1992 paper [17], Georgiou and Koutsougeras presented an activation that attenuates the magnitude of the signal while preserving the phase, Figure 2.12, right and Figure 2.4.3. We refer to this activation as *siglog* because it is equivalent to applying the sigmoid operator to the log of the input magnitude and then restoring the phase:

$$\operatorname{siglog}(z) = g(\log(|z|))e^{-i\angle z}, \text{ where } g(z) = \frac{1}{1+e^{-z}} \tag{2.71}$$

$$= \frac{z}{1+|z|} \tag{2.72}$$

Unlike the sigmoid and separable sigmoid activation functions, input values with large magnitudes are mapped to output values with magnitudes near one. Likewise, as the input magnitude approaches zero, so does the output magnitude.

Georgiou and Koutsougeras [17] included two positive constants to adjust the scale, $r$, and steepness, $c$, of the activation function:

$$\operatorname{siglog}(z; r, c) = f(z; r, c) = \frac{z}{c + \frac{1}{r}|z|} \tag{2.73}$$

The $\mathbb{CR}$ derivatives of this function are:

$$\frac{\partial f}{\partial z} = \frac{c + \frac{1}{r}\frac{1}{2}m(z)}{\big(c + \frac{1}{r}m(z)\big)^2} \tag{2.74}$$

$$\frac{\partial f}{\partial \overline{z}} = -\frac{1}{2r}\frac{z^2}{m(z)}\frac{1}{\big(c + \frac{1}{r}m(z)\big)^2} \tag{2.75}$$

Derivations of these derivatives are included in the appendix, Section A.4.

### Proposed Activations

#### iGaussian

The siglog activation function described above both preserves the phase of input and maps input magnitude values from $[0, \infty)$ to $[0, 1)$. However, with the sharp drop towards

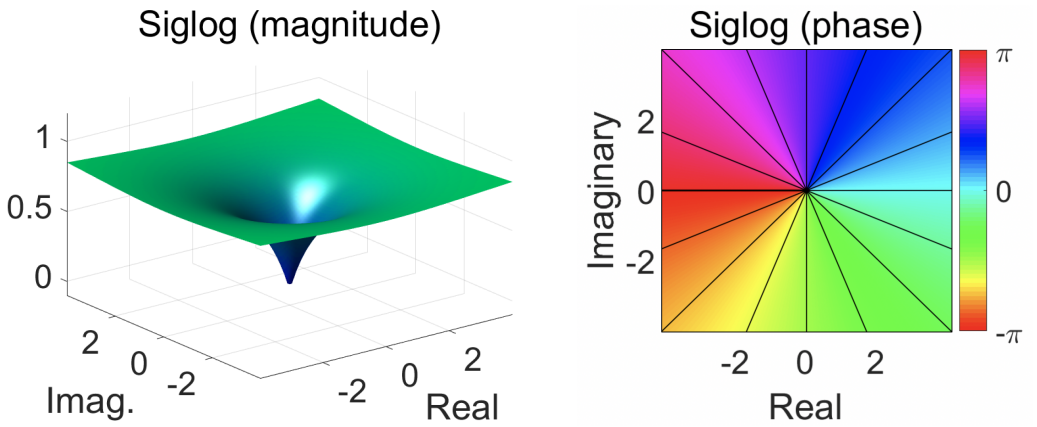


*Figure 2.13*: Magnitude and phase of the siglog activation function output. Complex input values with large magnitude are scaled to have a magnitude near one. At first glance, the surface plot of the phase (right) may look complicated, but on the contrary, the helical shape shows that the phase remains unchanged from input to output, i.e. the output phase increases from $-\pi$ to $\pi$ as the input value rotates counter-clockwise starting from the negative real axis.

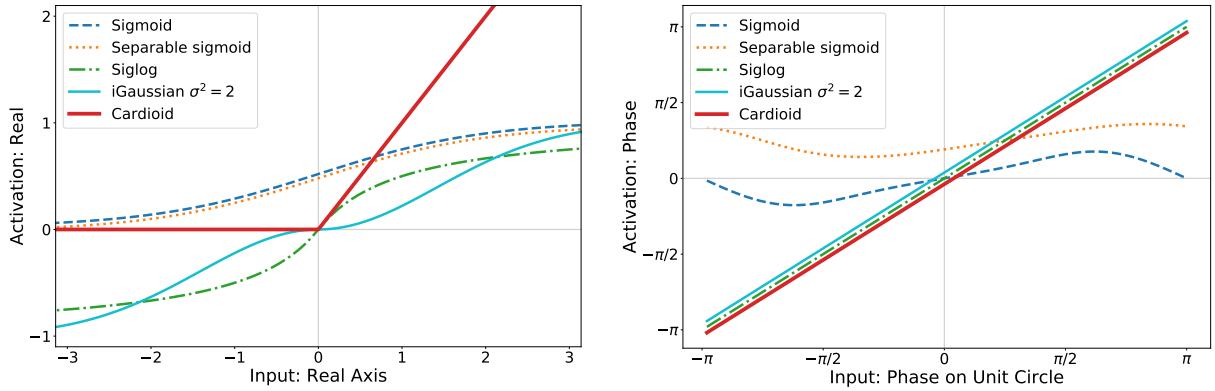


*Figure 2.14*: Comparison of activation functions. The left shows the real component of the activations applied to input values on the real axis. The right shows how the input phase maps to output phase.

zero from all directions, the siglog function has a nonzero gradient in the neighborhood of the origin of the complex plane, which can lead to gradient descent optimization algorithms to continuously stepping past the origin rather than approaching zero and staying there, Figure 2.4.3, left.

We propose the iGaussian activation function that like siglog preserves the phase and attenuates the magnitude, but it also has a gradient that approaches zero as the input approaches zero, allowing for more stable network optimization, Figure 2.4.3.

iGaussian is an inverted Gaussian, which is one minus a scaled Gaussian function (and then the phase restored). The forward function warps the magnitude of $z$ by the inverted Gaussian function, $g(z)$, and then restores the phase by multiplying by the normalized version of $z$, $n(z)$:

$$f(z;\sigma^2) = g(z;\sigma^2)n(z) \tag{2.76}$$

$$g(z;\sigma^2) = 1 - e^{-\overline{z}z/2\sigma^2} \text{ (inverted Gaussian)} \tag{2.77}$$

$$n(z) = e^{i\angle z} = \frac{z}{(\overline{z}z)^{\frac{1}{2}}} \text{ (see complex normalization Section A.1)} \tag{2.78}$$

where $\sigma$ is the standard deviation parameter that scales the width of the Gaussian.

Derivations of the $\mathbb{CR}$ derivatives of the iGaussian function are included in the appendix, Section A.5.

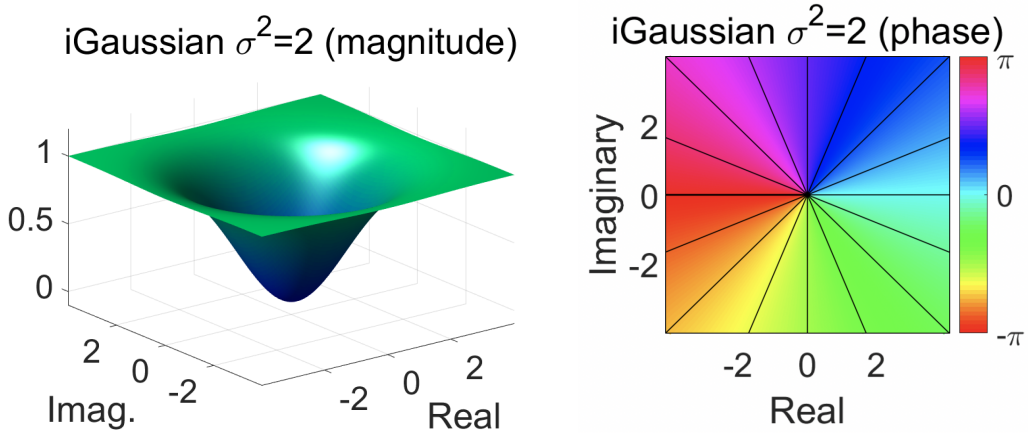


*Figure 2.15*: Magnitude and phase of the iGaussian activation function applied on the complex plane. Input magnitude from zero to infinity is mapped smoothly from zero to one (left). The input phase is preserved (right).

### Complex Cardioid

While the iGaussian activation function has nice properties that allow network optimization to smoothly guide complex features towards smaller or larger magnitudes, it still contains

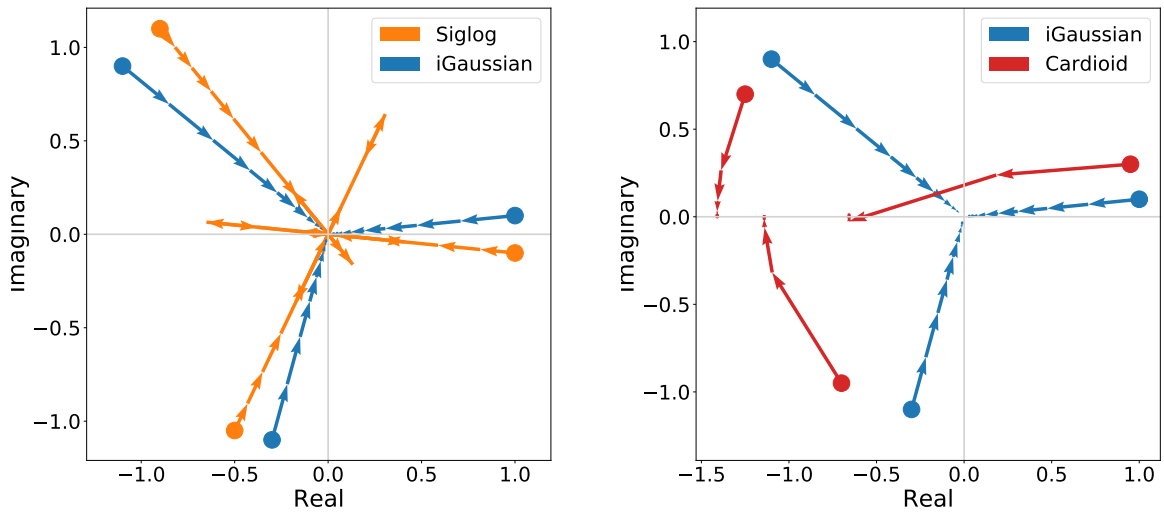


*Figure 2.16*: Example of gradient descent minimizing $L(z) = |f(z)|$, the magnitude of activation function, $f$. Tested with learning rate 1.5 on three initial points for $f$=siglog, (left, orange), $f$=iGaussian (blue), and $f$=cardioid (right, red). siglog overshoots the origin, while iGaussian gradually approaches zero from each direction. Cardioid takes a very few large steps to efficiently jump toward the negative real axis, where the magnitude of the cardioid output is zero.

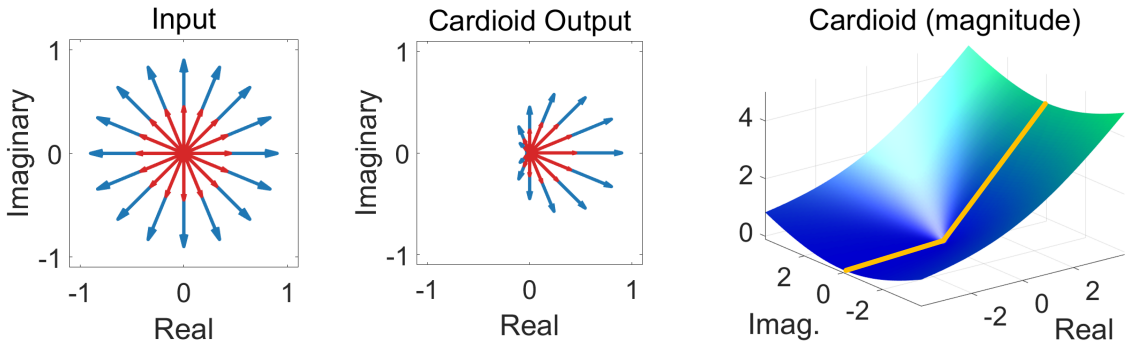


*Figure 2.17*: Our new cardioid activation function is a phase-sensitive complex extension of ReLU. Left / Center: Each arrow indicates a sample input/output of our cardioid function on the complex plane. Right: The magnitude transformation of the cardioid function shows that it is reduced to ReLU on the real axis (orange line).

the vanishing gradients for large values that have historically hindered the performance of sigmoid-like activation functions.

We propose a second new complex activation function, **complex cardioid**, which acts as an extension of the highly effective ReLU function to the complex plane. Like ReLU, the cardioid activation scales by one input values that lie on the positive real axis, and it scales by zero the input values on the negative real axis. Input values with nonzero imaginary components are gradually scaled from one to zero as the complex number rotates in phase from the positive real axis towards the negative real axis, Figure, 2.4.3. The cardioid function is sensitive to the input phase rather than the input magnitude. The output magnitude is attenuated based on the input phase using a cardioid function. Meanwhile, the output phase remains equal to the input phase. The complex cardioid is defined as:

$$f(z) = \frac{1}{2}(1 + \cos(\angle z))z \tag{2.79}$$

When the input values are restricted to real values, the complex cardioid function is simply the ReLU activation function, Figure 2.4.3, left.

The local $\mathbb{CR}$ [55] derivatives are as follows:

$$\frac{\partial f}{\partial z} = \frac{1}{2} + \frac{1}{2}\cos\left(\angle z\right) + \frac{i}{4}\sin\left(\angle z\right) \tag{2.80}$$

$$\frac{\partial f}{\partial \overline{z}} = \frac{-i}{4}\sin\left(\angle z\right)\frac{z}{\overline{z}} \tag{2.81}$$

Derivations of these derivatives are included in the appendix, Section A.6.

As demonstrated in the MR fingerprinting experiments in Chapter 3, the proposed cardioid activation function enables the complex-valued neural networks to converge to an effective learned model when other complex activation struggled to produce adequate results. The inclusion of the complex cardioid allowed the complex-valued neural networks to meet and even surpass the performance of 2-channel real networks for the MR fingerprinting reconstruction task. See the next chapter, Section 3.2 for detailed experimental setup and results.

### 2.4.4 Well-defined Layers

**Fully Connected, Convolution, Deconvolution.**

The fully connected layer with length $K$ complex-valued input vector, $\mathbf{z}$, with length $K$ complex-valued weight vector, $\mathbf{w}$, and complex scalar bias term, $\beta$ is:

$$\hat{z} = f(\mathbf{z}, \mathbf{w}) = \sum_k z_k w_k + \beta \tag{2.82}$$

$$= \mathbf{z}^T\mathbf{w} + \beta \tag{2.83}$$

The local $\mathbb{CR}$ derivatives of $f$ are:

$$\frac{\partial f}{\partial \mathbf{z}} = \mathbf{w} \qquad \frac{\partial f}{\partial \overline{\mathbf{z}}} = \mathbf{0} \tag{2.84}$$

$$\frac{\partial f}{\partial \mathbf{w}} = \mathbf{z} \qquad \frac{\partial f}{\partial \overline{\mathbf{w}}} = \mathbf{0} \tag{2.85}$$

$$\frac{\partial f}{\partial \beta} = 1 \qquad \frac{\partial f}{\partial \overline{\beta}} = 0 \tag{2.86}$$

Because the local conjugate $\mathbb{R}$-derivatives above are zero, we can simplify the conjugate $\mathbb{R}$-derivative of the final network layer, $L$, with respect to the conjugates of the input vector, weight vector, and bias term, given the derivative from all later layers, $\partial L/\partial \overline{\hat{z}}$:

$$\frac{\partial L}{\partial \overline{\mathbf{z}}} = \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)} 0 + \frac{\partial L}{\partial \overline{\hat{z}}} \overline{\left(\frac{\partial f}{\partial \mathbf{z}}\right)} \tag{2.87}$$

$$= \frac{\partial L}{\partial \overline{\hat{z}}} \overline{\mathbf{w}} \tag{2.88}$$

$$\frac{\partial L}{\partial \overline{\mathbf{w}}} = \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)} 0 + \frac{\partial L}{\partial \overline{\hat{z}}} \overline{\left(\frac{\partial f}{\partial \mathbf{w}}\right)} \tag{2.89}$$

$$= \frac{\partial L}{\partial \overline{\hat{z}}} \overline{\mathbf{z}} \tag{2.90}$$

$$\frac{\partial L}{\partial \overline{\beta}} = \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)} 0 + \frac{\partial L}{\partial \overline{\hat{z}}} \overline{\left(\frac{\partial f}{\partial \beta}\right)} \tag{2.91}$$

$$= \frac{\partial L}{\partial \overline{\hat{z}}} \tag{2.92}$$

Because the fully connected layer will be more efficiently implemented using linear algebra libraries, we specify below the matrix form for the batch version of the fully connected function and its conjugate $\mathbb{R}$-derivatives. With batch size $M$, number of input channels $K$, and number of output channels $N$, the input matrix, weight matrix, and bias vector are $Z \in \mathbb{C}^{M \text{x} K}$, $W \in \mathbb{C}^{N \text{x} K}$, and $\beta \in \mathbb{C}^{N}$, respectively. The output matrix is $\hat{Z} \in \mathbb{C}^{M \text{x} N}$ and we represent the conjugate $\mathbb{R}$-derivative of the scalar final layer function with respect to $\hat{Z}$ as a matrix also, $\partial L/\partial \overline{\hat{Z}} \in \mathbb{C}^{M \text{x} N}$.

$$\hat{Z} = f(Z, W, \beta) = ZW^T + \beta \tag{2.93}$$

$$\frac{\partial L}{\partial \overline{Z}} = \frac{\partial L}{\partial \overline{\hat{Z}}} \overline{W} \tag{2.94}$$

$$\frac{\partial L}{\partial \overline{W}} = \frac{\partial L}{\partial \overline{\hat{Z}}}^T \overline{Z} \tag{2.95}$$

$$\frac{\partial L}{\partial \overline{\beta}} = \frac{\partial L}{\partial \overline{\hat{Z}}}^T \mathbf{1} \tag{2.96}$$

When implementing these operations, it is important to note the location of the conjugate operations as well as the non-conjugate transpose operations.

We also implemented complex versions of convolution and deconvolution layers. The math for these layers is essentially the same as the fully connected layer. The notable difference is accounting for the shared weights of the convolution kernels, just as it is in the real-valued implementations.

### Pooling

As discussed in Section 2.2.3, pooling with the max operator on complex numbers is undefined. As an alternative, we implemented the complex pooling layer to return the complex-valued input element with the largest magnitude:

$$\hat{z} = f(\mathbf{z}) = z_n, \text{ where } n = \underset{k}{\operatorname{argmax}} |z_k| \tag{2.97}$$

While this max-magnitude layer does not take into account the phase of the input vector, the layer may be combined with complex convolution and cardioid layers. For example, the network might include layer sequence: 1) convolution → 2) cardioid → 3) max magnitude pooling, and 4) another convolution. In this case, 1) the complex convolutional weights can learn to rotate the phase of desirable data towards the positive real axis, where 2) the cardioid activation will allow that value to pass through without shrinkage giving it a better chance to also pass through 3) the max-magnitude pooling layer. The subsequent convolution layers (4) can then learn to rotate the data back to its original phase as needed.

Because the local conjugate $\mathbb{R}$-derivative of an identity function is zero, the conjugate $\mathbb{R}$-derivative of the final network layer with respect to the input can be simplified.

$$\frac{\partial f}{\partial z_n} = \begin{cases} 1, & \text{if } n = \operatorname{argmax}_k z_k \\ 0, & \text{otherwise} \end{cases} \tag{2.98}$$

$$\frac{\partial f}{\partial \overline{z}_n} = 0 \tag{2.99}$$

$$\frac{\partial L}{\partial \overline{z}} = \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)} \frac{\partial f}{\partial \overline{z}} + \frac{\partial L}{\partial \overline{\hat{z}}} \overline{\left(\frac{\partial f}{\partial z}\right)} \tag{2.100}$$

$$= \begin{cases} \frac{\partial L}{\partial \overline{\hat{z}}}, & \text{if } n = \operatorname{argmax}_k z_k \\ 0, & \text{otherwise} \end{cases} \tag{2.101}$$

### Batch Normalization

Batch normalization adjusts the scale and offset of network data before being passed to an activation function [59]. By linearly adjusting the activation input such that each batch has zero mean standard deviation one, batch normalization improves the stability of the network optimization, especially with very deep networks where there is a greater chance that activations and gradients shrink to zero or explode and overflow.

The mean, variance, and linear scaling operators found within batch normalization are all well-defined operations in the complex domain. In fact, the only change to the real-valued forward function is the conjugate square operation when computing the variance:

$$\mu = \mu(\mathbf{z}) = \frac{1}{N}\sum_n z_n \tag{2.102}$$

$$\sigma^2 = \sigma^2(\mathbf{z}, \mu) = \frac{1}{N}\sum_n \overline{(z_n - \mu)}(z_n - \mu) \tag{2.103}$$

$$\hat{\mathbf{z}} = f(\mathbf{z}, \mu, \sigma^2) = \frac{\mathbf{z} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.104}$$

where $\epsilon$ is a small constant to avoid division by zero.

The complex batch normalization derivatives and associated derivations are included in the appendix, Section A.7.

### Complex Normalization

The complex normalization layer scales a complex scalar input, $z$, such that its magnitude equal one. The phase of the input is unchanged. This projects $z$ onto the unit circle.

The forward function simply divides the input by its magnitude:

$$\hat{z} = e^{i\angle z} = \frac{z}{(\overline{z}z)^{1/2}} \tag{2.105}$$

The complex batch normalization derivatives and associated derivations are included in the appendix, Section A.1.

### Dropout

A dropout layer can help prevent overfitting by randomly zeroing out activations, forcing the network to learn multiple viable paths through the network [60]. Like the real-valued dropout implementation, complex dropout is simply a masked identity function. The forward and backward implementation is straight-forward, but we can't just use two-channel real implementation because we don't want to drop the real component independent from the imaginary component.

Because the local conjugate $\mathbb{R}$-derivative of an identity function is zero, the conjugate $\mathbb{R}$-derivative of the final network layer with respect to the input can be simplified.

$$\hat{z}_n = f(z_n) = \begin{cases} \frac{1}{p} z_n, & \text{if } n \in \text{mask} \\ 0, & \text{otherwise} \end{cases} \tag{2.106}$$

$$\frac{\partial f}{\partial z_n} = \begin{cases} \frac{1}{p}, & \text{if } n \in \text{mask} \\ 0, & \text{otherwise} \end{cases} \tag{2.107}$$

$$\frac{\partial f}{\partial \overline{z}_n} = 0 \tag{2.108}$$

$$\frac{\partial L}{\partial \overline{z}} = \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)} \frac{\partial f}{\partial \overline{z}} + \frac{\partial L}{\partial \overline{\hat{z}}} \overline{\left(\frac{\partial f}{\partial z}\right)} \tag{2.109}$$

$$= \begin{cases} \frac{1}{p} \frac{\partial L}{\partial \overline{\hat{z}}}, & \text{if } n \in \text{mask} \\ 0, & \text{otherwise} \end{cases} \tag{2.110}$$

where $p$ is the probability of an activation passing through the dropout layer. As in real-valued dropout, the $1/p$ scaling factor allows the expected value of the layer output to remain the same regardless of the value of $p$.

### 2.4.5 Software

We implemented our complex neural networks as an extension to the Caffe deep learning library [38, 32]. In this section, we discuss a few of the challenges related to implementing high-performance software with support for complex numbers. While the following points were encountered while working with specific software, namely Caffe, C++, MKL/Apple BLAS, and NVIDIA CUDA, many of the concepts and caveats are transferable to other programming environments.

#### Complex Types

Modern deep learning libraries, including Caffe, have generic data structure objects that are processed by the various data processing, network layer, and network optimization components. When extending these libraries to support complex numbers, one of the first steps is to understand how complex data types might need to interact with other data types, specifically standard floating point data types. For instance, will existing data I/O layers load complex data into complex arrays or to a float array with 2x entries? If there is a mix

of complex layers and real layers as discussed in Section 2.4.2, would there be two data types used in the system, complex and float?

Caffe uses C++ templates to allow the data objects, or Blobs, to support any numeric type. Despite this flexibility, the design required that the templated type had to be the same for all components in the network. So the network layers and optimization algorithms could work with all float or all complex but not both. All complex would be problematic, as there are many values within the code, such as the learning rate, that do not make sense to be complex and would require reworking the whole system to ensure they all behaved properly as complex numbers. Thus, we had to work with all the templates set to float (or double).

Naturally, one can store complex numbers as two float values for real and imaginary. In fact, C and C++ complex types are implemented as a structure with two float values. However, to take advantage of any complex operations implemented in the language/library, such as multiplication or matrix multiplication, the two float values would need to be converted to a complex type. Converting between the two for a single number is not an issue, but for large arrays of data, it is important to be able to convert types without having to copy the data from one location to another. The good news is that according to the C++ standard section 26.4, C++14 and section 29.5, C++17, one can safely cast a pointer to an array of complex types to a pointer to an array of float values, where real and imaginary components are interleaved [61, 62]. Unfortunately, the standard provides no such guarantee in the other direction, casting from a float pointer to a complex pointer.

Because Caffe manages the data allocation within the Blob object and all template types need to be float/double, any complex data must be first allocated as an array of floats and then later converted to complex types. Unfortunately, this forced our complex Caffe code to accept the risk of an unsafe cast from float to complex, rather than the prohibitively costly allocation and copy to a new array.

Once the array pointers were cast to complex types, there were no issues converting the C++ std::complex type to complex types in C or CUDA as needed.

### Complex Operations

One of the primary benefits of deep learning libraries is that they are built to quickly compute numerical operations on large arrays of data. Modern libraries, including Caffe, rely on third party implementations of low-level numerical methods on both CPU and GPU. Specifically, Caffe interfaces to fast CPU linear algebra via BLAS libraries, such as Intel MKL or Apple Xcode Accelerate, and to GPU data processing via NVIDA CUDA and cuDNN. While BLAS libraries and CUDA support complex numbers, the associated software interface can be quite cumbersome.

As an example, the BLAS method for matrix multiply, gemm, conveniently support optional operations to transpose or conjugate-transpose and input matrix. However, this convenience does not extend to conjugating the elements of a matrix without transposing it. So for matrix multiplications that require it, as in fully connected layer Equations 2.94, a conjugate copy of the data must be made before the call to BLAS gemm.

Caffe has many layers implemented in efficient GPU code using CUDA. Using the Caffe CUDA macros, the GPU code for the low-level data processing is relatively conveniently to code and easy to read. The complex versions of the same code can be much more challenging. For instance, the log function in CUDA does not support complex types, and the complex version must be implemented based on operations to the real and imaginary components. So rather than a quick call to:

```
b = log(a);
```

we must implement the complex log method in CUDA with:

```
b.x = log( cuCabsf(a) );
b.y = atan2f(a.y, a.x);
```

Additionally, the lack of complex constructors and operator overloading makes the complex CUDA code rather difficult to read and write. As an example, the complex cardioid code to compute the conjugate $\mathbb{R}$-derivative:

$$\frac{\partial f}{\partial \overline{z}} = \frac{-\frac{i}{4}\sin(\theta)z}{\overline{z} + \epsilon} \tag{2.111}$$

is implemented in CUDA as follows:

```
cuComplex dfdcz =
    cuCdivf(
        cuCmulf(
            make_cuFloatComplex(0, -0.25f*sinf(theta)),
            z),
        cuCaddf(
            cuConjf(z),
            make_cuFloatComplex(1e-14,0)) );
```

As the demand for high-performance deep learning has grown, hardware vendors have been developing highly optimized code to run neural network computations efficiently on their CPUs and GPUs. Caffe networks gain a boost in performance when NVIDIA's cuDNN library is available to ease the computational bottleneck caused by convolution layers, but unfortunately, cuDNN does not yet support complex data types.

## 2.5 Discussion

We defer our experiments with complex deep learning to the following chapter where we apply our methods to the MR fingerprinting task. As we will see with those results, the complex cardioid activation function makes it possible for complex-valued neural networks to match and even outperform real-valued networks with two input channels. Whether or not complex data representations, complex layer functions, and complex calculus will consistently outperform real-valued neural networks is still an open question. Recently, neural networks

have consistently been demonstrating that careful modeling can be replaced by enough data and stable deep networks. That being said, there are certainly still modeling assumptions that are critical to the success of deep learning. For instance, the convolution layer is based upon a model of structural similarity across an image, which allows for an extreme reduction in the number of weights required in a neural network.

The gains shown from our proposed cardioid activation highlight the importance of activation function design on network performance. We look forward to future work continuing to develop new non-linear layers for complex data. In particular, the modified ReLU presented in [18] could be a very effective activation function. Though not explicitly mentioned in [18], this modified ReLU is the complex soft-thresholding operator, which has strong mathematical connections to enforcing the popular $L_1$ norm sparsity constraint on complex numbers [63, 64].

As deep learning continues to grow across application domains, it is critical to have the tools necessary to quickly develop and experiment with new complex-valued deep learning methods. We contribute an implementation of complex backpropagation and complex network layers as an extension to the Caffe deep learning library, including high-performance implementations on CPU with BLAS and on GPU with CUDA.

# Chapter 3

# Deep Learning for MR Fingerprinting

Tissue in the body may be characterized by how it interacts with the magnetic field during an MRI scan. Two tissue parameters, T1 and T2, are exponential time constants, e.g. $e^{-t/\mathrm{T2}}$, that describe how fast hydrogen protons in different tissues relax after being excited by an applied magnetic field. Different tissues have different relaxation properties that can be emphasized during an MRI exam. For example, T1 and T2 values allow us to discern the boundary between gray matter (T1$\simeq$ 830 ms, T2$\simeq$ 80 ms) and white matter (T1$\simeq$ 500 ms, T2$\simeq$ 70 ms) in MRI brain images [65]. These parameters also enable radiologists to differentiate between benign and malignant tissues.

Traditional MRI generates images that contain a *qualitative* visual contrast between tissues, but unfortunately, these contrast-weighted images do not provide *quantitative* and consistent values, making it difficult to perform longitudinal studies and track disease progression over time. Standard methods to estimate quantitative tissue parameters are too long to be feasible in clinical practice, as they require too many repeated scans to robustly fit accurate parameters from the data [6, 66]. Recently, Ma et al. introduced a new technique by the name of magnetic resonance fingerprinting (MRF) that computes quantitative tissue parameters, as well as system parameters, from a single pseudorandom acquisition [6].

MR fingerprinting works by scanning the subject using a pseudorandom sequence of scanner controls. The various tissues in the body will react to the controlled magnetic field changes, producing measurable signals that have unique signatures depending on their specific tissue parameters (T1, T2), applied magnetic field (B0, B1), and other system parameters. Just like a fingerprint pattern can identify a specific person, these measured signals may be decoded to determine the tissue and magnetic field parameters at each pixel location in the image.

In this chapter, we introduce relevant background material for MR fingerprinting and then detail three projects to enhance MR fingerprinting methods with deep learning. Specifically, in Section 3.2, we train complex-valued neural networks to replace the computationally heavy fingerprint matching process. In section 3.3, we construct a noise model based on *in vivo* MRF acquisitions to produce neural networks that are robust enough to accurately decode tissue parameters despite severe noise and undersampling artifacts in accelerated,

high-resolution MRF images. Finally, in section 3.4, our neural networks learn to leverage the rich information content of a single MRF scan to directly synthesis the traditional contrast-weighted images in addition to the quantitative parameter map images.

## 3.1 Background for MR Fingerprinting

To more efficiently describe our MR fingerprinting methods and experiments in this chapter, we first introduce a select set of magnetic resonance imaging concepts, as well as the MRF method itself. Naturally, there will be numerous simplifications and topics that we will not cover. For instance, we omit the discussion of gradient magnetic fields and Fourier imaging. Unless otherwise cited, the MR concepts described in this section may be found in greater detail in Nishimura's 2010 textbook Principles of Magnetic Resonance Imaging [67].

### 3.1.1 Measuring Net Magnetization

The nuclei of hydrogen ($^1$H) atoms have a magnetic moment that interacts with the various magnetic fields within an MRI scanner. When considering a collection of hydrogen atoms within a particular region, the sum of their nuclear magnetic moments may be represented by a *net magnetization* vector $\mathbf{M}$. During an MRI scan, we measure how the magnitude and orientation of this three dimensional $\mathbf{M}$ vector changes as it interacts with both the applied magnetic fields as well as the molecular structure of tissue in that location. Figure 3.1 illustrates five important aspects of net magnetization that we describe next.

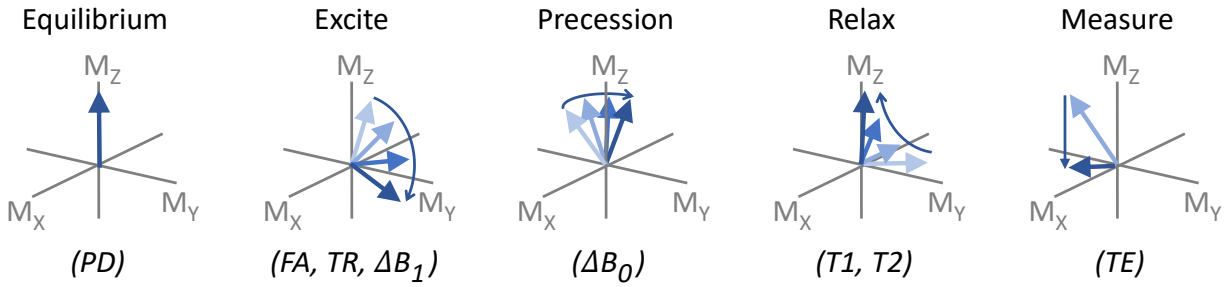


*Figure 3.1*: Five aspects related to measuring the net magnetization vector $\mathbf{M}$. From left to right: 1) *equilibrium* is $\mathbf{M}$'s initial/relaxed state; 2) the scanner *excites* $\mathbf{M}$, rotating it towards or past the $x$-$y$ plane; 3) natural *precession* over time may rotate $\mathbf{M}$ about the $z$-axis; 4) *relaxation* over time returns $\mathbf{M}$ to equilibrium; and 5) only the transverse components $\mathbf{M}$, $M_x$ and $M_y$, can be *measured* through inductive coupling. We'll later introduce the parenthetical terms $PD$, $FA$, $TR$, $\Delta B_1$, $\Delta B_0$, $T1$, $T2$, and $TE$ as they apply to each of these aspects.

**Equilibrium.** The net magnetization of a tissue sample has an equilibrium state $\mathbf{M} = M_0\hat{\mathbf{z}}$, to which $\mathbf{M}$ will return when being presented with only the main magnetic field, Figure 3.1, left. MRI scanners have one large, persistent magnetic field, $B_0$, which (ideally)

has the same strength and orientation at each location within the scanner, i.e a homogeneous magnetic field. When a sample of tissue containing hydrogen is in the scanner, before any scan begins, the net magnetization is in this equilibrium state, specifically:

1. $\mathbf{M}$ is aligned with the $B_0$ field, which by convention is the $z$ direction of an $x, y, z$ coordinate system, and

2. $\mathbf{M}$ has magnitude $M_0$, which is linearly proportional to $B_0$ as well as the number of hydrogen atoms in the sample.

**Excitation.** The scanner "excites" the hydrogen nuclei by applying a temporary radio frequency (RF) magnetic field $B_1$, which is perpendicular to $B_0$ and as a result, rotates $\mathbf{M}$ away from equilibrium, towards or past the transverse plan ($x$-$y$ plane). The $B_1$ field is induced by radio frequency (RF) field controlled by the scanner. This RF excitation creates non-zero $M_x$ and $M_y$ components of $\mathbf{M}$ and is critical to enabling two other net magnetization concepts described below, namely *relaxation* and *measurement*.

**Precession**. Precession is a natural phenomenon that causes the net magnetization vector to rotate about the axis of the magnetic field at an angular frequency proportional to the strength of the magnetic field:

$$\omega = \gamma B \tag{3.1}$$

where $\omega$ is the angular frequency and $\gamma$ is a constant called the gyromagnetic ratio. Imaging techniques in MR are based on a spatially homogeneous main magnetic field, $B = B_0$ everywhere within the scanner bore. Small changes in the main magnetic field, $\Delta B_0$, alter the precession rate and can cause imaging artifacts at positions containing this inhomogeneity.

**Relaxation.** Once the temporary $B_1$ excitation field is turned off, the net magnetization begins to relax back towards equilibrium. There are two natural relaxation mechanisms that occur simultaneously: T1 relaxation is the exponential regrowth of the $M_z$ component, while T2 relaxation is the exponential decay of the $M_{xy}$ components. A key aspect of MRI, and certainly MR fingerprinting, is that different tissues in the body have different relaxation rates. We will elaborate below on how we can leverage the different relaxation rates by measuring the net magnetization at particular times, hoping to discern between tissues types.

**Measurement.** As the magnetic moments precess about the $z$-axis, they induce a current in the scanner's RF receiver coils, allowing a measurement of the $M_x$ and $M_y$ components of the net magnetization vector. Because the two components of this RF signal are a measurement of a rotating phenomenon, it is natural to represent them as a single complex-valued number, $M_{xy}$.

### 3.1.2 Scanner Parameters

There are several scanner configurations and settings that affect the net magnetization of tissues in the scanner. We have already mentioned the strength of the main magnetic

field, which remains fixed for each scanner at $B_0$. The excitation field $B_1$ is controllable and may be changed from scan to scan or from excitation to excitation. The following are three scanner parameters that may be changed to significantly alter the $M_{xy}$ values measured during an acquisition.

**Flip angle (FA).** The flip angle is a $B_1$ field parameter that determines how much to rotate the net magnetization during excitation. Starting at equilibrium, a flip angle of 90 degrees would rotate $\mathbf{M}$ down to the transverse plane ($x$-$y$ plane) . The acquisition sequence does not have to wait for $\mathbf{M}$ to fully return to equilibrium before applying another excitation. For example, if the net magnetization is 20 degrees from $M_z$ just before an excitation, a flip angle of 90 degrees could rotate $\mathbf{M}$ to 20 degrees below the $x$-$y$ plane. It is interesting to note that in this case, T1-relaxation would then cause the negative $M_Z$ to first move towards zero before continuing to grow back to $M_z = M_0$. In addition to the flip angle, the $B_1$ field may be applied in different directions, allowing the scanner to affect the phase of $M_{xy}$.

**Repetition time (TR).** TR is the time between excitations. As seen in the above flip angle example, the timing between excitations affects the position of $\mathbf{M}$ before and after the $B_1$ rotation, which in turn will affect the measured signal. A long TR will allow the net magnetization to fully recover to $\mathbf{M} = M_0\hat{\mathbf{z}}$ before the next excitation.

**Echo time (TE).** The TE parameter defines the timing of the $M_{xy}$ measurement. Essentially, it is the time between the excitation and the measurement. As an example, if we apply a 90 degree excitation to a net magnetization at equilibrium, a very short TE will capture a very large $M_{xy}$, near $M_0$, effectively measuring the number of hydrogen atoms in the sample, see proton density below.

### 3.1.3 Tissue Parameters

**Proton density (PD).** PD is a measure of the number of hydrogen atoms, or equivalently the number of hydrogen protons, that are in a given region or pixel/voxel. The magnitude of the net magnetization vector at equilibrium, $M_0$, is directly proportional to the proton density.

**T1.** T1 is the exponential growth time constant for $M_z$ during T1-relaxation in a given tissue sample. Given the value of T1 and $M_z(0)$ at $t = 0$, the $z$-component of $\mathbf{M}$ at time $t$ is modeled by:

$$M_z(t) = M_z(0)(1 - e^{-t/T1}) \tag{3.2}$$

**T2.** T2 is the exponential decay time constant for $|M_{xy}|$ during T2-relaxation in a given tissue sample. Given the value of T2 and $M_{xy}(0)$ at $t = 0$, the transverse magnetization of $\mathbf{M}$ at time $t$ is modeled by:

$$M_{xy}(t) = M_{xy}(0)e^{-t/T2} \tag{3.3}$$

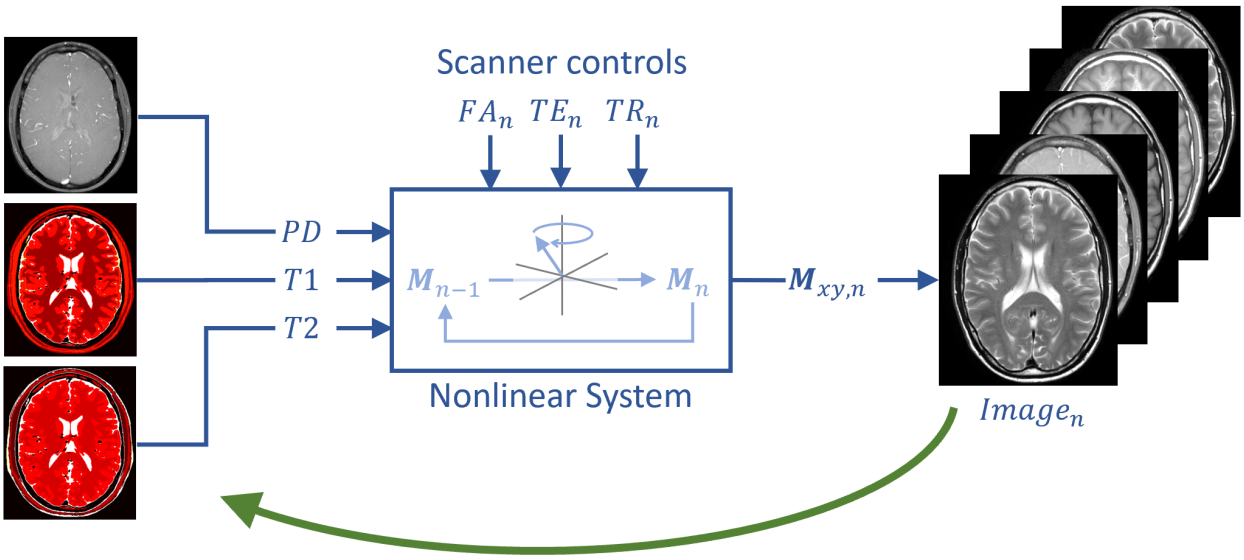


*Figure 3.2*: Dynamic system model for the nuclear spins of the tissue within an MRI scanner. Input tissue parameters (PD, T1, T2) at each location are converted to a series of $M_{xy}$ measurements, which are affected by a series of scanner control parameters. Inverting this nonlinear system would allow us to convert the output images into maps of the tissue parameters, bottom green line.

### 3.1.4 Modeling the MR System

We can view the nuclear spins of the tissue within an MRI scanner as a dynamic system, where the input values are the PD, T1, and T2 parameters of the tissue at a location within the body and the output is a series of $M_{xy}$ measurements as affected by a series of scanner controls, FA, TE, and TR, Figure 3.2. The internal state of this system is the three-dimensional net magnetization vector, $\mathbf{M}$.

It is important to note that the output images of this system are just observations of $M_{xy}$ and not the tissue parameters themselves. In order to recover these parameters, we could create a model of this system and attempt to invert it, Figure 3.2, bottom green line.

In 1946, Bloch proposed a model of how net magnetization changes over time, including the two T1 and T2 relaxation models described above [68]. The Bloch equation combines the relaxation effect with the precession of $\mathbf{M}$ about a magnetic field vector $\mathbf{B}$:

$$\frac{d}{dt}\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} -\frac{1}{T2} & \gamma B_z & -\gamma B_y \\ -\gamma B_z & -\frac{1}{T2} & \gamma B_x \\ \gamma B_y & -\gamma B_x & -\frac{1}{T1} \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{M_0}{T1} \end{bmatrix} \tag{3.4}$$

### 3.1.5 Parameter Mapping and Contrast-weighted Imaging

As noted above, the MRI system outputs the transverse component of the net magnetization but not the tissue parameter values themselves. However, we could sample the

exponential decay of T2 relaxation by setting a long TR and then systematically increasing TE for each scan. With a reasonably long TR setting, the net magnetization will essentially recover to equilibrium between excitations, simplifying the system model to:

$$|M_{xy,n}| \propto e^{-\frac{TE_n}{T2}} \tag{3.5}$$

Using a predefined series of $TE_n$ values, we can acquire a series of these $M_{xy,n}$ measurements for each pixel and then fit the data to this simplified exponential model to solve for T2. The resulting image of T2 values is called a T2 *parameter map*, Figure 3.4, right.

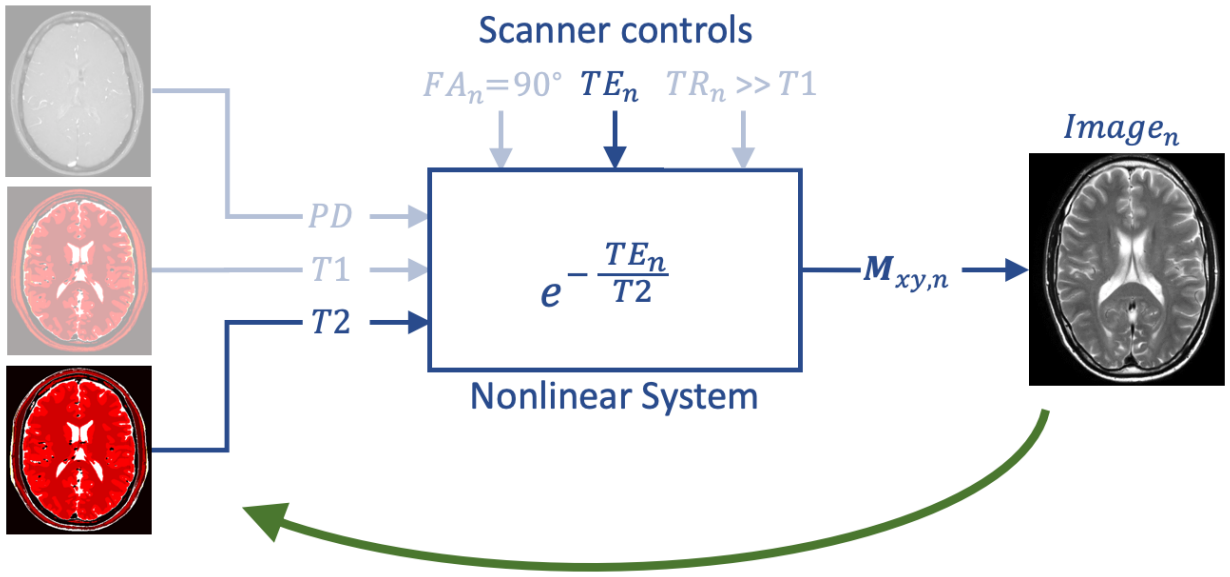


*Figure 3.3*: Simpler dynamic system model for the nuclear spins of the tissue within an MRI scanner. With a $TR$ much longer than $T1$ and a 90 degree flip angle, the tissue parameter T2 may be modeled based only on the scanner control TE.

The T1 parameter may be fit from an exponential curve in a similar manner by repeatedly changing the timing of scanning techniques called saturation-recovery or inversion-recovery [66].

Assuming a noise-free signal, it would only require measurements for two different time points to fit the exponential function and recover the T1 or T2 value at each pixel. Unfortunately, due to system noise, high-quality T1 and T2 parameter mapping with this brute force mapping techniques requires many repeated measurements, which given the long TR, would significantly increase the exam time [66].

Faster methods have been developed to do parameter mapping for T1 [9, 69], T2 [10, 70] and T1/T2 jointly [71, 72]. However, these techniques remain prohibitively long for standard clinical practice [6].

As an alternative to time-consuming parameter mapping, standard clinical practice instead acquires what are referred to as *contrast-weighted* images. For example, a T2-weighted

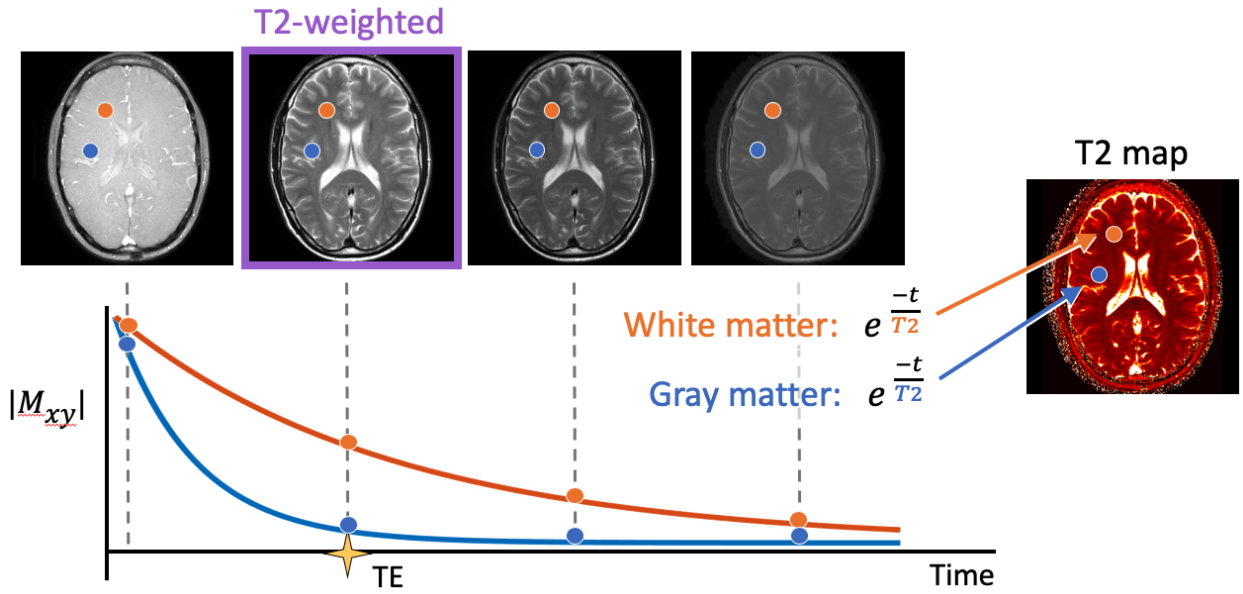


*Figure 3.4*: A T2-weighted image (top row, purple border) is a single image acquired with TE set such that the image contrast between tissues is greatest. The T2 parameter map image (far right) is an estimate, at each pixel location, of the T2 exponential time constant itself, which may be fit from a series of values acquired at different TE settings.

contrast image is acquired at a single TE, rather than a series of TE settings. The chosen TE value is selected to maximize the difference in image intensity (contrast) between tissues with different T2 values, as illustrated in the T2 relaxation plot of gray matter and white matter in Figure 3.4. Unlike parameter maps with consistent, *quantitative* values for T1 or T2, T1- and T2-weighted images show only *qualitative* visual intensity differences.

### 3.1.6 MR Fingerprinting

Rather than the slow, systematic sampling of T1 and T2 relaxation curves, MR fingerprinting proposes to simultaneously generate multiple parameter maps from a single acquisition that randomly perturbs the net magnetization vector with a seemingly erratic set of scanner controls for each excitation [6]. Rather than fitting a sequence of $M_{xy}$ measurements to an equation, as done in traditional parameter mapping describe above, MR fingerprinting compares the measured $M_{xy}$ signal evolution to a dictionary of simulated signals. As illustrated in Figure, 3.5, the dictionary signals are generated with a Bloch simulator using a specific sequence of scanner controls and many different combinations of (T1, T2) parameters. A subject is then scanned using the same sequence of scanner controls, generating a measured signal evolution at each pixel. Similar to the process of identifying a person from their fingerprint, the parameter values (T1, T2) for each pixel can be identified by matching that pixel's measured signal to the most similar signal in the dictionary of simulated signals

and their corresponding (T1, T2) parameters.

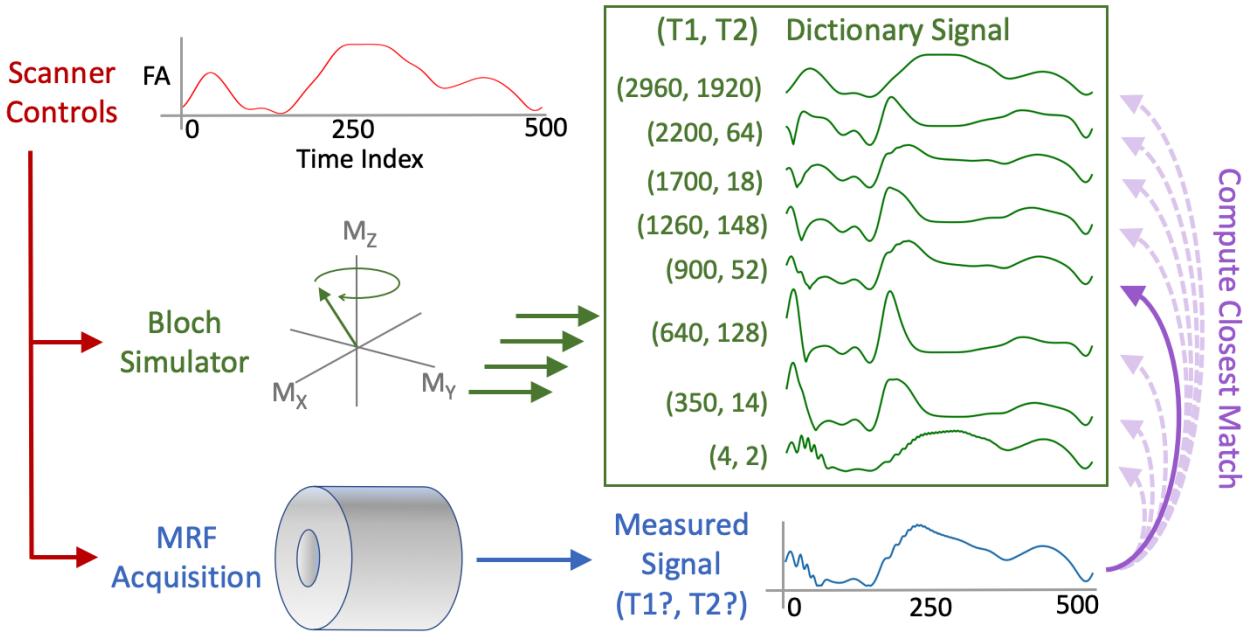


*Figure 3.5*: Illustration of the MR fingerprinting process. The same sequence of scanner controls (e.g. flip angles) is given to both a Bloch simulator and the actual scanner. Offline, the Bloch simulator generates a dictionary of signals modeling the measured evolution of the net magnetization for a predefined grid of parameter values (T1, T2). The measured signal from the scanner is compared to the dictionary signals to identify the parameters of the closest matching signal.

## 3.2 Complex-valued Deep Learning

Traditional MRI requires *many different scans* that each accentuate one of the desired parameters. Additionally, those scans only provide a *qualitative* visual contrast between tissues, e.g. *"in this image, tissues with high T1 are brighter than other tissues"*. MR fingerprinting as proposed in [6], however, simultaneously produces *quantitative* values for T1, T2, and proton density in *one single scan*. It can also provide information about system imperfections, i.e. in B0 and in B1 [6, 73, 74], Figure 3.6.

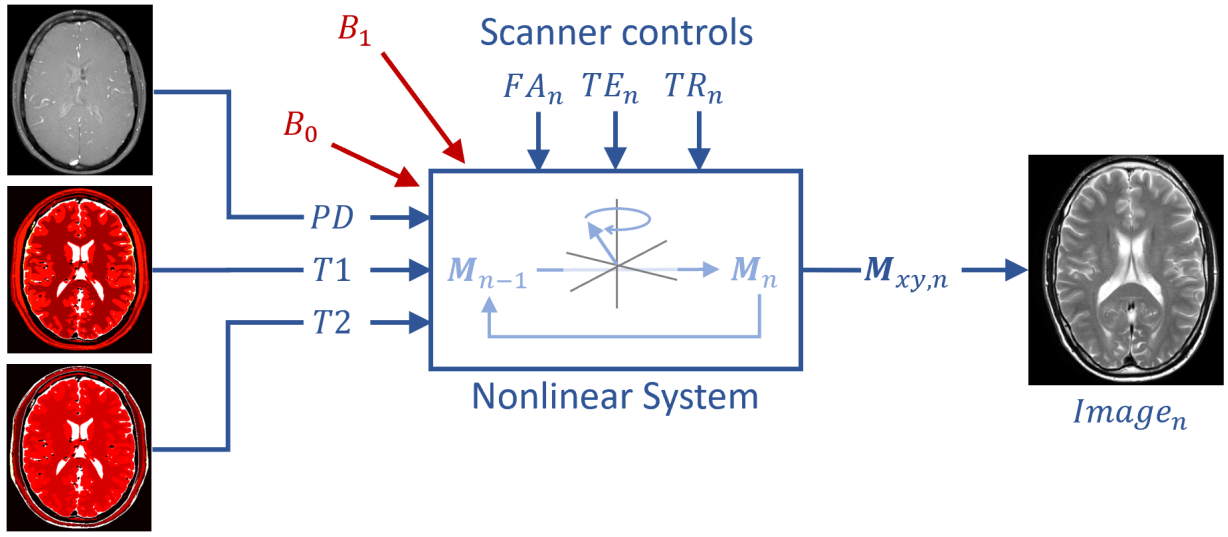


*Figure 3.6*: When considering additional parameters maps, such as magnetic field maps for B0 and B1, the dictionary for MR fingerprinting starts to grow exponentially, leading to infeasible computation times for dictionary matching techniques. We propose using neural networks to learn a fast inversion of this non-linear MR fingerprinting system.

From a machine learning perspective, the MR fingerprinting simulator provides a training set consisting of length $N$, complex-valued vectors, each of which is paired with a tuple of real-valued labels (T1, T2, B0, B1). At the test time, the MRI scanner acquires a length $N$, complex-valued signal at each pixel location that must be decoded into the tissue and system parameters for that signal. Prior MR fingerprinting works [6, 75, 76] have used a nearest neighbor search based approach to match the measured signal to a dictionary of simulated training signals. Due to the non-parametric nature of dictionary matching methods, the computation time scales linearly with the size of the dictionary and exponentially with the number of parameters, quickly becoming infeasible with a finer parameter resolution or when more parameters are required, Figure 3.7. Additional research has improved the dictionary matching efficiency by compressing the dictionary using SVD [26] or by applying group matching [77].

Rather than non-parametric dictionary matching-based methods, we propose learning a

parameterized model for solving the MR fingerprinting inverse mapping problem. Specifically, we demonstrate that feed-forward neural networks can accurately model the complex non-linear MR fingerprinting inverse mapping function with a computational efficiency that does not scale with the number of training examples, Figure 3.7.

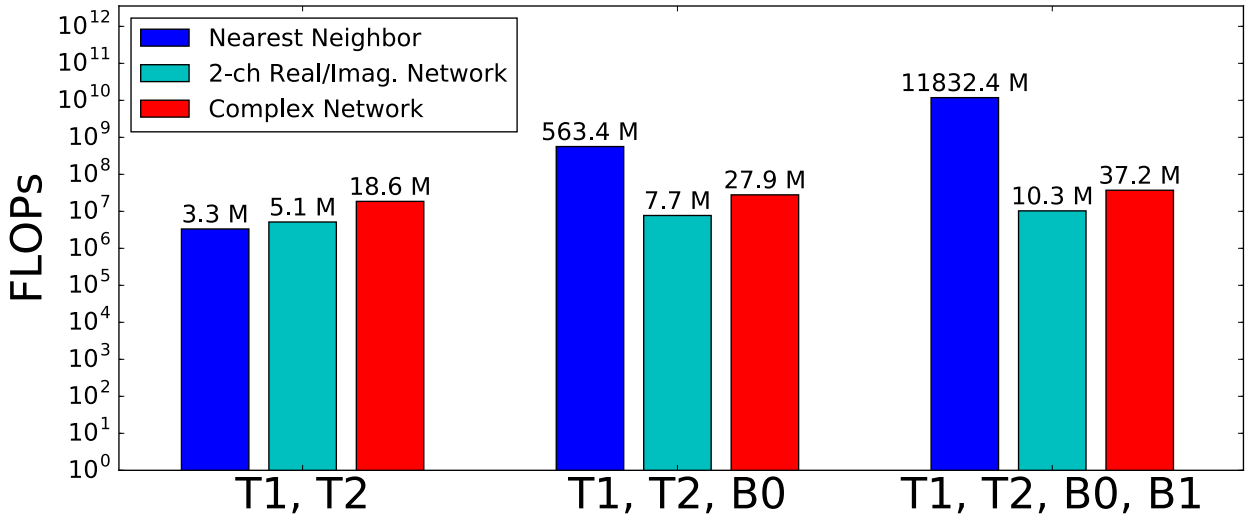


*Figure 3.7*: Comparison of floating point operations required to compute the parameters for a single pixel. Note the log scale on the y-axis, showing how dictionary matching compute times quickly become infeasible.

We also investigate using complex-valued neural networks for MR fingerprinting, since the MRF signals are inherently complex-valued. While complex-valued signals can be represented by 2-channel real signals, each channel containing real and imaginary components respectively, such a representation does not fully exploit the phase information that is captured by complex algebra. Indeed, by including the complex cardioid activation function proposed in Chapter 2, we demonstrate that complex-valued neural nets can be even more effective than real-valued networks at MRF fingerprinting.

Work by Cohen et al. [13] and Hoppe et al. [14] independently developed deep learning methods for MR fingerprinting, though they trained real-valued networks using only the magnitude of the MRF signal. Golbabaee et al. later extended a real-value version of our work to include a dimensionality reduction bottleneck within the MRF neural network [78]. Liu et al. later used deep learning to map a multi-echo acquisition to T2 parameters maps, while incorporating their neural network into the iterative reconstruction loop [15]. Nataraj et al., on the other hand, trained a non-linear regression model for MR fingerprinting rather than using either dictionary matching or neural networks [79].

### 3.2.1 Methods

We compare the following six MR fingerprinting methods:

1. Baseline: max cross-correlation dictionary matching with a T1, T2, B0 dictionary [6].
2. Real-valued neural nets with 2-channel real/imaginary inputs representing complex MRI signals, using the ReLU activation function.
3. Real-valued neural nets that are twice as wide as the second model, with 1024 and 512 feature channels in the two hidden layers.
4. Complex-valued neural nets with 1-channel complex MRI signals, using our new cardioid activation function.
5. Complex-valued neural nets using separable sigmoid activation functions (i.e. sigmoid applied to real and imaginary independently) [1].
6. Complex-valued neural nets using the siglog activation function [17].

We are particularly interested in comparing the performance of complex-valued neural networks with both the real-valued network and the real-valued network that is twice as wide. As described in Figure 2.4, the degrees of freedom of the complex network fall between the degrees of freedom of these two real networks.

**Training data.** We simulate the MRI signal with the Bloch equations and the pseudo-random pulse sequence parameters from [6] with signal length 500. We use 100,000 simulated points for training, randomly sampled with the same T1, T2, B0 density as used in the dictionary matching baseline technique.

**Testing data.** Following [75], we test our methods with the numerical MNI phantom [80, 65] with the T1, T2, and proton density values specified for each tissue type in [65]. We add a linear ramp in the B0 field across the image from -60 Hz to 60 Hz. We compute proton density from the norm of the test signal as in [6]. Although we do not include any B1 inhomogeneity in our experiments, a fourth neural network could easily be added to incorporate this or any other parameter(s). In addition to testing with a clean signal from the numerical phantom, we also tested phantom signals with complex-valued Gaussian noise added to produce a peak signal-to-noise ratio (pSNR) of 40.

**Neural networks.** Fig.3.8 shows our 3-layer neural network architecture for separate T1, T2, and B0 neural networks. The fully connected network reduces the feature channels as it gets deeper, similar to the first half of the autoencoders in [81]. Specifically, the networks consisted of three fully connected layers with 512, 256, and one output channels. The networks contained batch normalization [59] and non-linear activation between each fully connected layer. Each network terminated with an L2 loss function that penalized any error between predicted values and ground truth values for T1 and T2. The fully connected layers used Xavier weight initialization computed using the number of input channels [82].

For the real-valued networks in our experiments, we used the ReLU activation function, while for the complex networks we compared three complex-valued activation functions described in Chapter 2, namely separable sigmoid [1], siglog [17], and our complex cardioid.

We trained our networks using a stochastic gradient descent solver with momentum 0.9. We trained for 1,000,000 mini-batches of size 100 signals, scaling the learning rate by 0.9 every 10,000 iterations. We trained each network with four different initial learning rates:

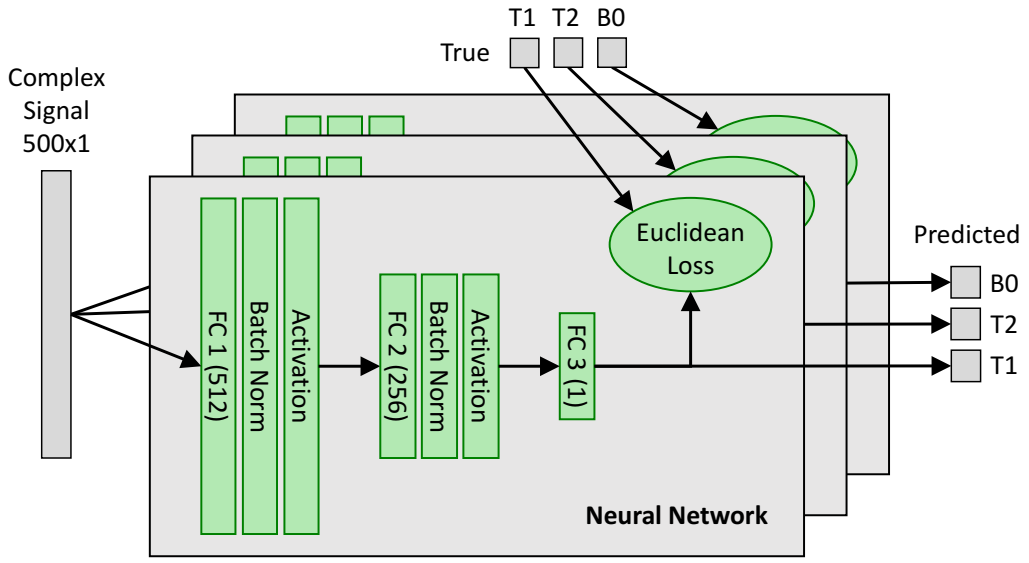


*Figure 3.8*: Fully connected neural network architecture, repeated for each desired output label (T1, T2, B0).

0.1, 0.05, 0.01, and 0.001. At test time, we selected the learning rate that produced the lowest loss on the validation dataset.

The neural networks were constructed, trained, and tested using Caffe [38] running on a cluster of NVIDIA M60 GPUs. For the complex-valued neural nets, we extend the Caffe platform with complex versions of the fully connected layer, batch normalization layer, and complex activation layers, including the $\mathbb{CR}$ calculus backpropagation for all the layer functions [32].

### 3.2.2 Results

*Table 3.1*: NRMSE results: fingerprinting from clean signals.

| **Network** | **T1** | **T2** | **$\Delta$ B0** |
|---|---|---|---|
| Dictionary matching | 10.63 | 39.78 | **1.02** |
| 2-ch real/imaginary network | 2.71 | 8.21 | 2.11 |
| 2-ch real/imaginary network 2x | 2.21 | 8.04 | 2.44 |
| **Complex (cardioid)** | **1.42** | **4.34** | 1.32 |
| Complex (separable sigmoid) | 4.72 | 9.24 | 3.33 |
| Complex (siglog) | 2.99 | 12.04 | 3.05 |

For both clean and noisy signals, we computed the normalized root mean squared error of the T1, T2, and $\Delta B0$ values predicted by each technique, as compared to the ground truth

*Table 3.2*: NRMSE results: fingerprinting from noisy signals (pSNR=40).

| **Network** | **T1** | **T2** | **Δ B0** |
|---|---|---|---|
| Dictionary matching | 12.21 | 40.38 | **1.08** |
| 2-ch real/imaginary network | 11.15 | **17.96** | 5.23 |
| 2-ch real/imaginary network 2x | 11.08 | 22.15 | 7.08 |
| **Complex (cardioid)** | **9.40** | 20.98 | 4.43 |
| Complex (separable sigmoid) | 17.31 | 33.09 | 18.83 |
| Complex (siglog) | 102.22 | 237.88 | 266.33 |

parameter values in our MNI phantom test set. Tables 3.1 and 3.2 compare the prediction accuracy at no noise and pSNR=40 noise level, respectively. Figures 3.9 and 3.10 show the corresponding reconstruction results on the MNI phantom.

Fig. 3.7 compares the computational efficiency in terms of the number of floating-point operations (FLOPs) to process parameter maps for just T1 and T2, as well as including a third parameter $\Delta B0$, and again with a fourth parameter $\Delta B1$. FLOPs were tallied for the dot product in dictionary matching and for the fully connected matrix multiplications in the neural networks. As an example, the FLOPs to process the MRF signal in the T1, T2, and $\Delta B0$ values for a single pixel are 563 million, compared to the 28 million for the complex networks. Additionally, the elapsed real-time to compute the MNI T1, T2, and $\Delta B0$ maps using dictionary matching was 45 *minutes*, compared to 11 *seconds* for the complex networks. Note that the elapsed real-time difference is even more dramatic than the number of FLOPs as the dictionary matching computations were executed on CPU with python NumPy, while the neural networks were executed on GPU with Caffe [38, 32].

We observe the following in our results:

1. A dictionary-based approach increases exponentially with more outputs and becomes infeasible. Compared to the two outputs (T1,T2), the #FLOPs increases by 171× for the three outputs (T1,T2,B0), and by 3,585× for the four outputs (T1,T2,B0,B1).
2. Inverse mapping by neural nets outperforms the traditional dictionary matching baseline on T1 and T2 values, whereas the dictionary matching approach predicts B0 values more accurately.
3. Complex-valued neural networks outperform 2-channel real-valued networks for almost all of our experiments. This advantage cannot be explained by an increased number of degrees of freedom in the complex network, Figure 2.4, as the complex network, also outperforms the 2x wide real network, suggesting that complex-valued networks can bring out information in the complex data more effectively than treating them as arbitrary two-channel real data.
4. The 2-ch real network had lower NRMSE than complex networks for T2 with pSNR=40. Upon inspecting the corresponding image quality, Figure 3.10, it can be observed that

the complex network has a visibly larger error in the CSF region, but a slightly lower error in the rest of the brain region.

5. The complex cardioid activation significantly outperformed both the separable sigmoid and siglog activation functions, allowing complex networks to not only compete with but surpass, real-valued networks.

### 3.2.3 Discussion

We propose a deep learning approach for MR fingerprinting that implements an efficient nonlinear inverse mapping function that converts complex-valued MRF signals directly into quantitative tissue and magnetic field parameters. Without compromising parameter map quality, the proposed neural network methods can produce parameter maps two orders of magnitude faster than the baseline dictionary matching methods when considering B0 maps in addition to T1 and T2. Elapsed real-time performance measurements are even more impressive than the expected computation measured in FLOPs, but perhaps real-time measurement is an unfair comparison given that the baseline dictionary matching was implemented for CPU with Python and NumPy, while the neural networks were implemented for GPU with Caffe. That being said, as many industries are investing heavily in high-performance deep learning, efficient and accessible neural network implementations will likely continue to improve.

We demonstrated that our complex cardioid activation function described in Chapter 2 enables the successful real-world application of complex-valued neural networks. With activation functions from prior work, complex neural networks struggle to produce any viable images. However, with the cardioid activation function, complex-valued networks can meet and even surpass the accuracy of real-valued networks for MR fingerprinting.

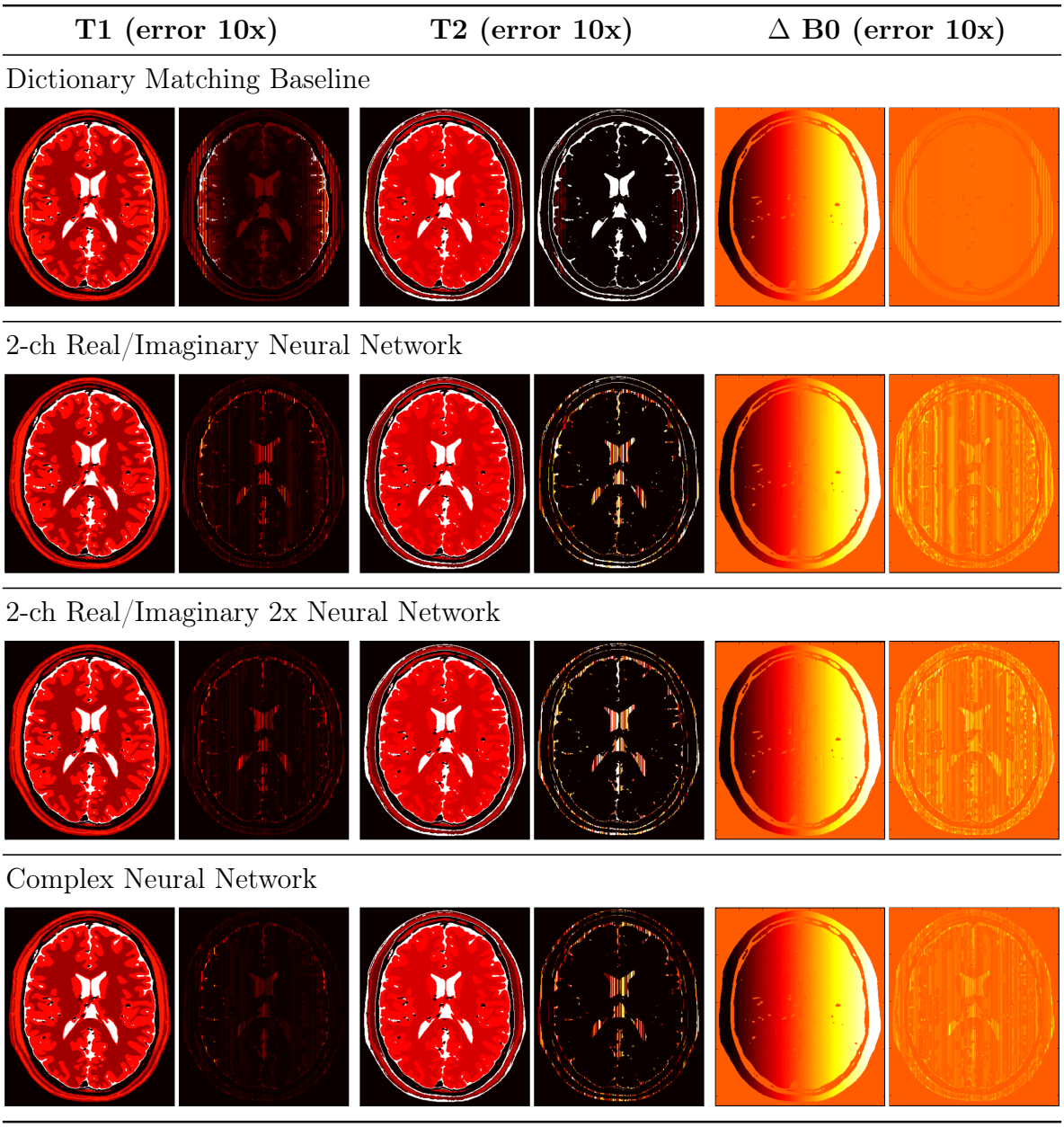


*Figure 3.9*: Numerical phantom with no added noise. Predicted quantitative parameter map images are shown adjacent to the error image. For visualization purposes, the error images are displayed at 10x the scale of the images.

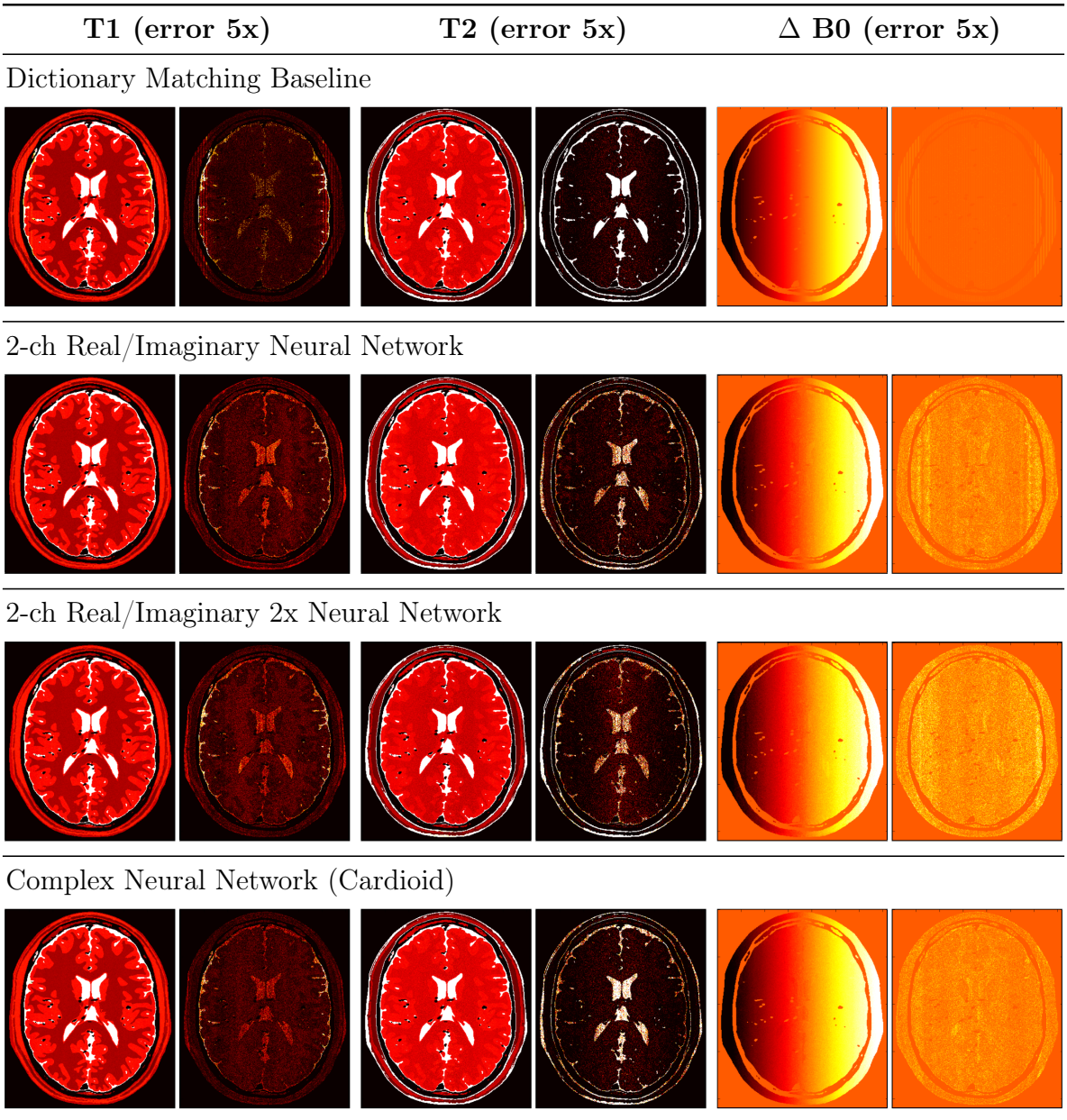


*Figure 3.10*: Numerical phantom with added noise (pSNR=40). Predicted quantitative parameter map images are shown adjacent to the error image. For visualization purposes, the error images are displayed at 5x the scale of the images.

## 3.3 MRF Training Data Synthesis

### 3.3.1 Introduction

In the previous section on deep learning MRF, we trained on and were able to reconstruct from, simulated MRF signals corrupted with complex Gaussian noise, Figure 3.10. However, when testing these trained networks on *in vivo* MRF acquisition data, the resulting T1 and T2 parameter maps have objectively poor image quality, Figure 3.11, left three columns.

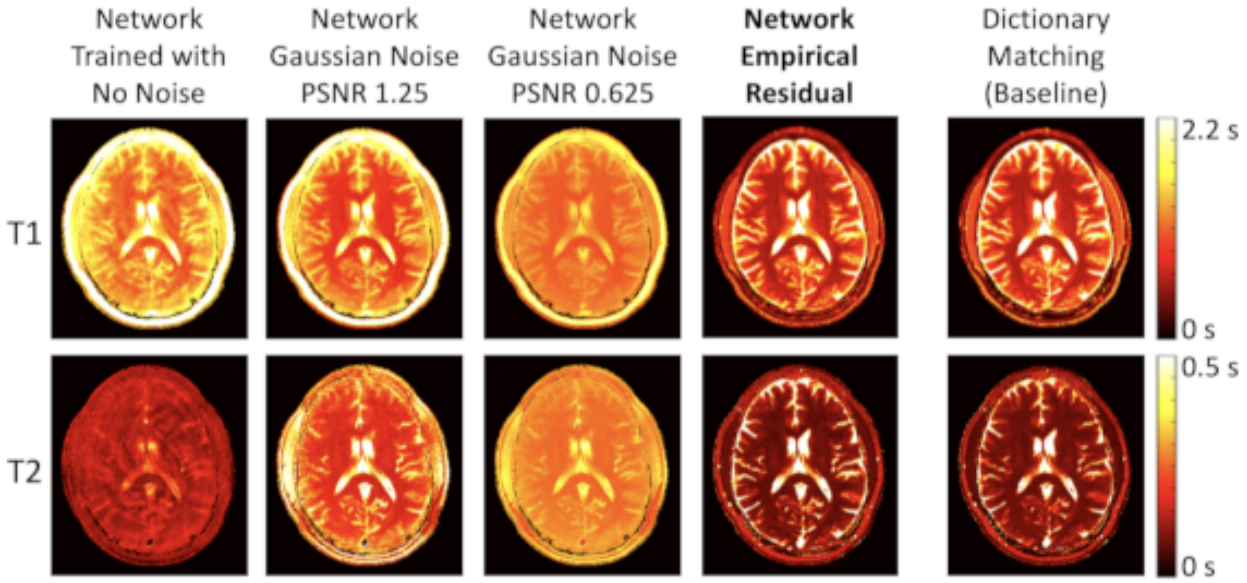


*Figure 3.11*: Parameter mapping results in different synthetic training data generation models. Results are shown for one slice of test volume one. The neural network trained with Gaussian noise fails to generate adequate parameter maps. The neural network trained with our empirical residual model (fourth column) produces parameters with the same level of quality as the dictionary matching baseline (right column).

While it is true that complex Gaussian noise is a good model for the predominant measurement noise in the MRI system, the noise in a reconstructed MR image remains complex Gaussian only when the acquisition is fully sampled on a Cartesian grid [67, 83]. The model of the imaging system becomes more complicated as we introduce non-uniform and undersampled acquisitions. With the undersampled spiral acquisition patterns often used in *in vivo* MR fingerprinting, the resulting image is affected by both measurement noise and aliasing artifacts from undersampling, Figure 3.12.

A potential solution to the inadequate Gaussian noise model is to train the MRF neural networks with *in vivo* MRF data paired with the ground truth labels, specifically *in vivo* T1 and T2 parameter maps. Unfortunately, as mentioned in Section 3.1.5, acquiring and fitting ground truth parameters maps is quite time-consuming. These long scans are further

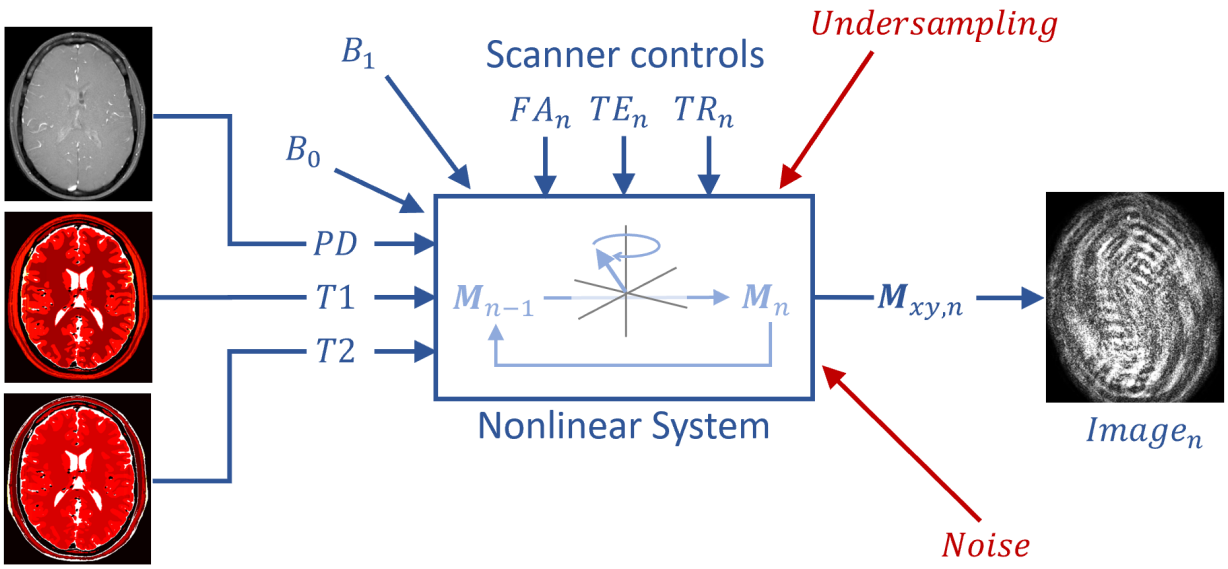


*Figure 3.12*: In our MR fingerprinting system model, we must account for both aliasing artifacts and lower SNR due to undersampling noisy data. The reconstructed image, $Image_n$, no longer fits the Gaussian noise model assumed by our previous work.

complicated by human subjects that would have to remain still across all of these acquisitions to ensure the MRF data is spatially aligned with the parameters maps.

We propose utilizing *in vivo* MRF acquisition data *without* ground truth parameters maps to build an *empirical residual model*. This residual model is based upon the subtracting a clean, simulated MRF signal from an *in vivo* MRF signal. Using this empirical residual model, we can alter simulated MRF signals to train neural networks with MRF data that is more representative of *in vivo* MRF data than when using a Gaussian noise model. The resulting trained network converts test *in vivo* MRF data into T1 and T2 parameter maps that are indistinguishable from the baseline dictionary matching parameter maps 3.11, right two columns.

### Undersampled imaging model and reconstruction

During an MRI scan, programmable linear magnetic field gradients make it possible to encode the acquired signal, $y$, in the frequency domain, or $k$-space, of the true image $\boldsymbol{m}_{true}$. When sampling a full 2D Cartesian grid in $k$-space, the forward model of the MRI system may be expressed as the following linear system:

$$\boldsymbol{y}_{full} = F_{full}\boldsymbol{m}_{true} + \boldsymbol{n} \tag{3.6}$$

where $\boldsymbol{y}_{full}$ is a length-$N$ complex-valued vector of the acquired signal ($N$ being the number of pixels in the image), $\boldsymbol{m}_{true}$ is a length-$N$ complex-valued vector of the true image, $F_{full}$ is the 2D Fourier transform operator, and $\boldsymbol{n}$ is a length-$N$ vector of zero-mean, complex-valued Gaussian noise samples.

The adjoint of the linear operator may be applied to reconstruct an image, $\boldsymbol{m}_{recon,full}$, from the fully sampled data, $\boldsymbol{y}_{full}$:

$$\boldsymbol{m}_{recon,full} = F^*_{full}\boldsymbol{y}_{full} \tag{3.7}$$

$$= F^*_{full}(F_{full}\boldsymbol{m}_{true} + \boldsymbol{n}) \tag{3.8}$$

$$= F^*_{full}F_{full}\boldsymbol{m}_{true} + F^*_{full}\boldsymbol{n} \tag{3.9}$$

$$= \boldsymbol{m}_{true} + F^*_{full}\boldsymbol{n} \tag{3.10}$$

where $F^*_{full}$ is the adjoint of the 2D Fourier transform operator, which, assuming scaling is appropriate, is the inverse 2D Fourier transform. Note that the reconstructed image is the true image plus the inverse 2D Fourier transform of the additive noise, thus zero-mean complex-valued Gaussian noise is also present in the reconstructed image. The point spread function (PSF) operator for this reconstruction system is $F^*_{full}F_{full}$, which is simply the identity operator, Figure 3.13, left. This implies that the signal from any pixel in the true image does not interfere with any other pixel location in the reconstructed image. This is not the case, however, with reconstruction systems for undersampled acquisitions.

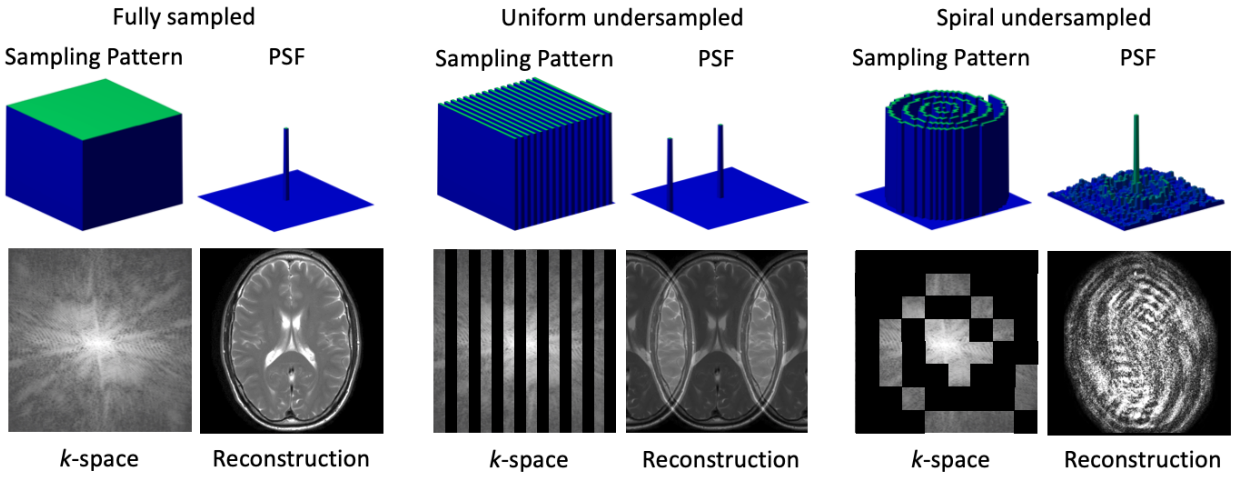


*Figure 3.13*: An illustration of different $k$-space acquisition patterns and the corresponding point spread function and example reconstructed image. Uniform undersampling of every other column in $k$-space (center) causes aliasing to manifest as a shifted copy of the image. Spiral undersampling (right) causes aliasing to manifest as artifact noise, where the original signal interferes with every other pixel location in varying amounts.

When undersampling $k$-space, the forward model of the MRI system changes to use an undersampled Fourier operator, which applies the 2D Fourier transform and then only keeps the values at the $K$ number of locations that were sampled:

$$\boldsymbol{y}_{under} = F_{under}\boldsymbol{m}_{true} + \boldsymbol{n} \tag{3.11}$$

where $\boldsymbol{y}_{under}$ is a length-$K$ complex-valued vector of the acquired signal.

The corresponding undersampled reconstruction now uses the adjoint of the undersampled Fourier operator, $F^*_{under}$:

$$\boldsymbol{m}_{recon,under} = F^*_{under}\boldsymbol{y}_{under} \tag{3.12}$$

$$= F^*_{under}(F_{under}\boldsymbol{m}_{true} + \boldsymbol{n}) \tag{3.13}$$

$$= F^*_{under}F_{under}\boldsymbol{m}_{true} + F^*_{under}\boldsymbol{n} \tag{3.14}$$

The undersampled reconstruction point spread function operator, $F^*_{under}F_{under}$, does not generally simplify to the identity operator. When undersampling every other column in $k$-space, the point spread function has two peaks offset by half a field of view, meaning that the signal from true input pixels will interfere with pixels shifted by half an image, Figure 3.13, center. More incoherent sampling patterns, such as a spiral sampling pattern, has a point spread function that affects every other pixel location. The resulting incoherent aliasing causes every pixel in the original image to interfere with every location in the reconstructed image, Figure 3.13, right.

The acquisition used to collect our *in vivo* MRF data has a different spiral undersampling pattern for each repetition, $n$. It also incorporates parallel imaging, which uses multiple RF receiver coils to measure the signal, each with different spatial sensitivities. The corresponding reconstruction uses non-Cartesian gridding with density compensation [84, 85] and RF coil combination with Philips CLEAR [86]. The forward model and reconstruction include a different undersampling operator for each spiral pattern, $F_{under,n}$, a coil sensitivity operator, $S$, and its adjoint, and a density compensation operator, $D$:

$$\boldsymbol{y}_{under,n} = F_{under,n}S\boldsymbol{m}_{true,n} + \boldsymbol{n}_{\boldsymbol{n}} \tag{3.15}$$

$$\boldsymbol{m}_{recon,under,n} = S^*F^*_{under,n}D_n\boldsymbol{y}_{under,n} \tag{3.16}$$

$$= S^*F^*_{under,n}D_n(F_{under}\boldsymbol{m}_{true,n} + \boldsymbol{n}_{\boldsymbol{n}}) \tag{3.17}$$

$$= S^*F^*_{under,n}D_nF_{under}\boldsymbol{m}_{true,n} + S^*F^*_{under,n}D_d\boldsymbol{n}_{\boldsymbol{n}} \tag{3.18}$$

### Synthetic MRF training data

When combining the forward model with the reconstruction system for fully sampled acquisitions, Equation 3.10, the true image signal is independent from the system noise. Because of this independence, labeled synthetic MRF training data may be generated for a single pixel by: 1) randomly sampling a set of parameters (T1, T2), 2) simulating the corresponding MRF signal vector, 3) randomly sampling a Gaussian noise vector, and 4) adding the noise to the simulated signal.

Because of the interference across pixels in the point spread function for undersampled spiral acquisitions, Equation 3.18, generating a synthetic MRF training vector would require simulating data at the image level rather than simply simulating the MRF evolution of one-pixel location.

In order to train neural networks to generate parameter maps despite noise coupled with aliasing, we propose using an empirical residual model. The basic idea is to create a dataset of residual signals by subtracting measured MRF signals from their dictionary matched simulated signal. These residual vectors contain an empirical combination of both the noise and interference present in the undersampled reconstruction system. Rather than Gaussian noise vectors, samples from this empirical residual dataset can be added to clean simulated MRF signals to augment the neural network training set. Figure 3.14 shows an example of how our empirical residual model matches the measured signal better than a Gaussian noise model.

### 3.3.2 Methods

#### Empirical residual model

Our empirical residual model considers three factors: residual sample, scaling sample, and random phase.

1. Residual sample: We create an empirical distribution of residual signals by comparing the measured MRF signals, $\mathbf{m}_{meas}$, from each pixel in our in-vivo training set to the dictionary matched simulated MRF signal, $\mathbf{m}_{sim}$:

$$\textbf{residual} = \boldsymbol{m}_{meas} - \alpha \boldsymbol{m}_{sim} \tag{3.19}$$

   where $\alpha$ is the signal scaling factor computed from $\text{argmin}_\alpha \, ||\boldsymbol{m}_{meas} - \alpha \boldsymbol{m}_{sim}||_2$. This scaling factor brings the measured and simulated to roughly the same intensity level, allowing us to better capture the residual. We store a collection of these empirical residual vectors, one for each measured signal in our training set. Note that the measured signal $\boldsymbol{m}_{meas}$ is a vector of complex values, thus the resulting vector will contain complex values even if the simulated signal has no imaginary components, as is the case in our simulated MRF FISP sequence.

2. Scaling sample: Proton density affects signal strength at each pixel, independent from T1, T2, and acquisition noise. For this reason, we also store samples of estimated scaling between the simulated signal and the residual for each pixel:

$$\rho = \frac{||\alpha \boldsymbol{m}_{sim}||_2}{||\boldsymbol{residual}||_2} \tag{3.20}$$

   The dominant noise is an MRI system is typically independent from the signal strength. However, this is not the case when we consider the interference from incoherent aliasing.

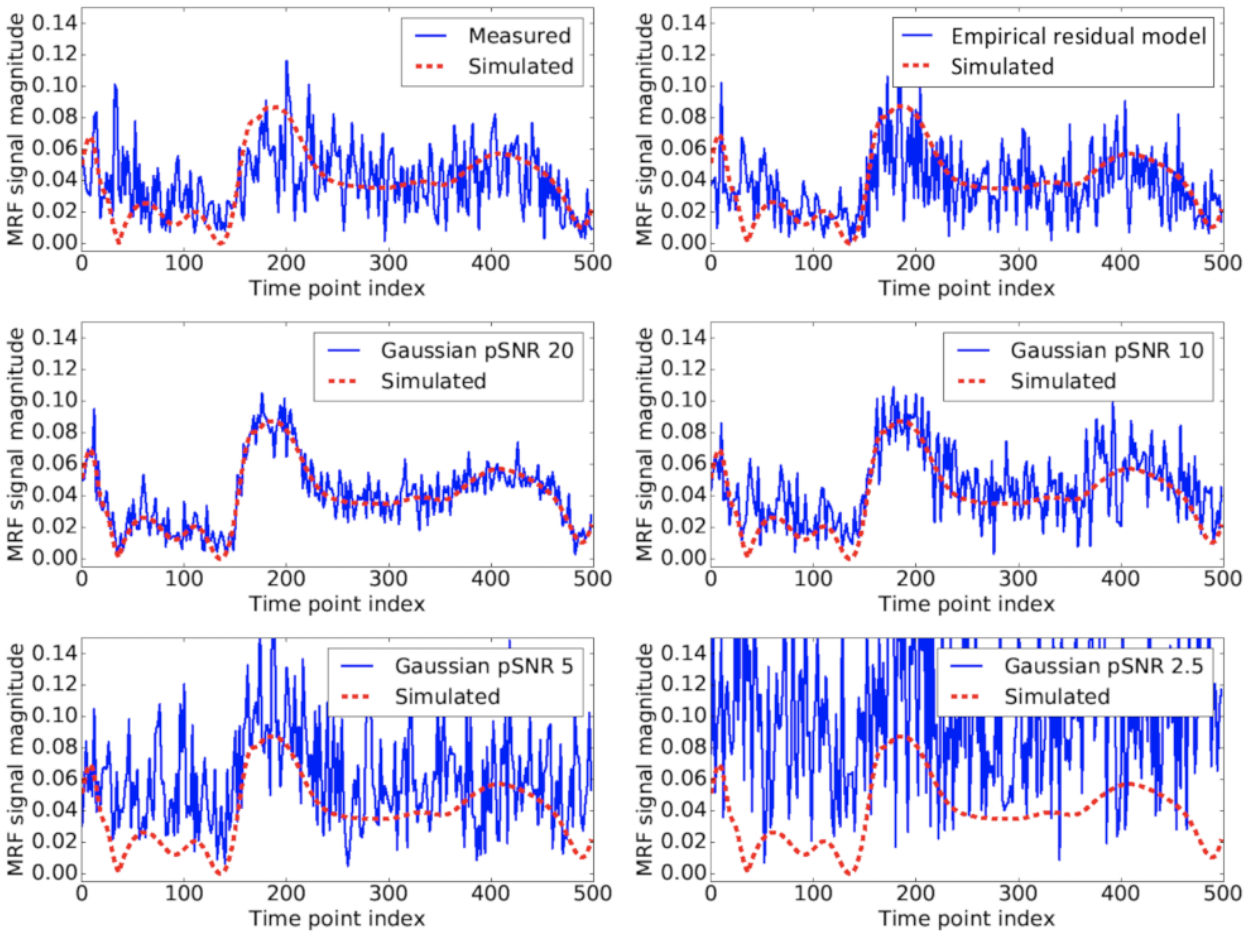


*Figure 3.14*: The magnitudes of the measured signal (top row, left) compared to simulated signals augmented with the proposed empirical residual model (top row, right) versus Gaussian noise as various noise levels (center and bottom rows). For reference, the dashed red lines show the magnitude of the associated clean simulated signal for T1 = 1280 ms, T2 = 96 ms. The dashed red line shown with the measured signal (top row, left) is the dictionary matched simulated signal, T1 = 1276 ms, T2 = 90 ms.

When undersampling, the strength of the signal from a given pixel spreads across the image. As the signal strength in the overall image increases, the aliasing energy increases, leading to interference that is signal-dependent. This is a key difference between the signal-independent Gaussian noise model. We store a collection of these scaling factors, one for each measured signal in our training set.

3. Random phase: Because the absolute phase of the measured signal is arbitrary, we also include a random phase shift to both our simulated signal and our sampled residual signal:

$$\phi \sim \mathcal{U}[0, 2\pi] \tag{3.21}$$

We sample a random scalar $\phi$ and rotate the phase entire simulated signal, $\boldsymbol{m}_{sim}$, by that angle. We sample a different random $\phi$ and similarly rotate the phase our residual signal.

We combine these three factors to create a synthetic version for each simulated training signal $\boldsymbol{m}_{sim}$:

$$\boldsymbol{m}_{synthetic} = \boldsymbol{m}_{sim} \cdot e^{j\phi_1} + \frac{1}{\rho_i} \frac{\boldsymbol{residual}_k}{||\boldsymbol{residual}_k||_2} e^{j\phi_2} \tag{3.22}$$

where $i$ and $k$ are random indices and $\phi_1$ and $\phi_2$ are random angles. In other words, each time we feed an EPG simulated training signal, $\boldsymbol{m}_{sim}$, to the neural network, we 1) randomly select a $\boldsymbol{residual}$ vector from our noise collection, 2) scale that residual vector by a random scalar sampled from our scale collection, 3) randomly rotate the phase of both the simulated signal and the scaled residual vector, and 4) add them together.

### Experimental setup

**Data acquisition and reconstruction.** With IRB approval, we scanned 12 male volunteers, ages ranging from 29 to 61 years, with a 1.5T Philips Ingenia scanner using 13 receive channels. We acquired a fingerprinting spoiled gradient echo (FISP) sequence similar to the one proposed in [22] with 500 time points, constant TE=3.3 ms and TR=20 ms, and flip angles shown in Figure 3.15. The MRF data were acquired with a spiral acquisition with two spiral readouts per TR interval, rotated at 180°. The spirals between two consecutive time points were rotated by 9°. The MRF sequence included nine or ten axial slices through the brain with a prescribed maximum in-plane resolution of 0.72x0.72 mm (FOV 230x230 mm, matrix size 320x320) and slice thickness 5 mm.

Each TR of the spiral MRF acquisition was reconstructed to image space using gridding with density compensation [84, 85] and coil combination with Philips CLEAR [86]. Figure 3.15 shows an example of the reconstructed MRF data.

To account for the arbitrary receiver gain that scales the output of any given scan, we normalize the *in vivo* MRF data by dividing by the 95th percentile of the magnitude of the

complex mean image (averaged across the 500 time points) of each MRF scan. This provides a consistent scaling when doing any preprocessing, such as the foreground mask described below. It also reduces the need for the network to be robust to a potentially large range of receiver gain values.

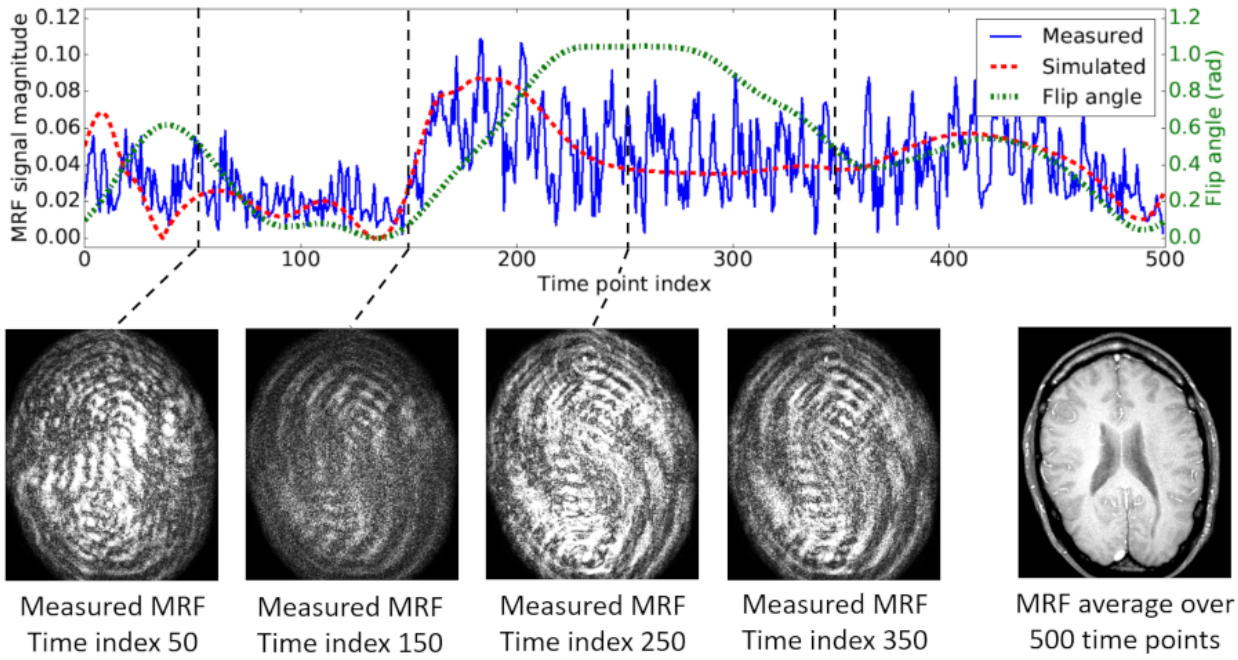


*Figure 3.15*: Empirical MRF signal. MR fingerprinting flip angle for each of the 500 time points (dashed-green line) shown with the magnitude of acquired MRF signal (solid-blue line) and the dictionary matched simulated MRF signal (dashed-red line), T1 = 1280 ms, T2 = 94 ms. The measured signal is dominated by artifacts. Also shown are magnitude images of the measured MRF signal at four different time points and the complex mean MRF across 500 time points.

**Foreground mask.** A significant portion of each image falls is the background. To avoid near-zero background points in our residual model, we create a binary foreground mask to exclude them. We automatically construct this mask by thresholding the magnitude of the complex mean image of the MRF data (after normalization to the 95th percentile) at 0.1 or greater. We use this same mask method at test time with the goal of directly mapping extremely low MRF signals to zero in all parameter maps.

**Baseline parameter maps.** We simulated a dictionary of MRF signals with the extended phase graph (EPG) algorithm [87, 88]. The dictionary consisted of 22,031 MRF signals for: T1 parameters in increments of 2 ms from 4 ms to 100 ms, increments of 10 from 100 to 1000, increments of 20 from 1000 to 2000, and increments of 40 ms from 2000 to 3000 ms; and T2 parameters in increments of 2 ms from 2 to 150, increments of 10 from 150 to 500, increments of 20 from 500 to 1000, and increments of 40 ms from 1000 to 2000 ms. Only species with T2 less than T1 were included in the dictionary. Each simulated signal in

the dictionary was scaled to have a Euclidean norm equal to one. Additional factors, such as B1 inhomogeneity and slice profile, were not included in the simulated dictionary. As in [6], we used cosine similarity to match the acquired MRF signal to the most similar signal in the simulated dictionary.

**Synthetic training data.** The simulated signals used in both the Gaussian and empirical residual models were generated in the same manner as the baseline dictionary. However rather than using T1 and T2 parameters on a uniform grid, we randomly sampled 100,000 (T1, T2) pairs using the same distribution as the dictionary. This random sampling allowed for simulated signals to be generated for (T1, T2) that fall between the dictionary grid points.

For the Gaussian noise model, zero-mean Gaussian noise was independently added to the real and imaginary channels of the simulated MRF signal, Figure 3.14. We experimented with eight different noise levels, specifically with PSNR values 0.625, 1.25, 2.5, 5, 10, 20, 40, 60, where the standard deviation of the Gaussian noise, $\sigma_{noise}$ is determined from:

$$PSNR = \frac{\max_{i,t} |signal_{i,t}|}{\sigma_{noise}} \tag{3.23}$$

where the max is taken over all 500 time points and all 100,000 samples in our simulated MRF training set.

For the empirical residual model, each training mini-batch randomly selects 1000 signals from our 100,000 EPG simulated signals and augments them according to the empirical residual model described above, Figure 3.14, top right. After applying the foreground mask to the *in vivo* MRF exam images, there were 4.7 million empirical residual signals and scaling samples available to sample from during training. We reserved two of the training MRF exams to create separate residual and scaling collections used only during validation.

We reserved two of the *in vivo* MRF exams to be used only for final testing of all of our methods.

**Ablation study and random phase.** As the empirical residual model has several components, we conducted an ablation study to better understand the impact of each of these components. Specifically, we trained and tested the empirical residual model with:

- No scale sampling (used mean scale value) and no phase rotation on either the signal or the noise
- No scale sampling (used mean scale), but phase rotation on both the signal and the residual
- Scale sampling, but no phase rotation on either the signal or the residual
- Scale sampling and phase rotation on the residual but not on the signal
- Scale sampling and phase rotation on the signal but not on the residual

The results of this study indicate that the phase rotation on the simulated signal contributed strongly to training successful networks, so we repeated our Gaussian noise model

experiments with the addition of a random phase to each signal before adding Gaussian noise, see $\phi_1$ in Equation 3.22.

**Network architecture and training.** We trained separate T1 and T2 neural networks, both with network architecture described in Figure 3.8 and [31]. Specifically, the networks consisted of three fully connected layers with 512, 256, and one output channels. The networks contained a ReLU non-linear activation between each fully connected layer and terminated with an L2 loss function that penalized any error between predicted and ground truth T1 and T2 values. The fully connected layers used Xavier weight initialization computed using the number of input channels [82]. With this shallow network, we found that batch normalization was not necessary to stabilize the optimization during training.

Because our spoiled gradient echo fingerprinting sequence is designed to be insensitive to B0, the phase of the underlying signal is consistent across all time points and does not carry any information, so we chose to use the two-channel real/imaginary network from [31], rather than a complex-valued neural network.

We trained our networks using a stochastic gradient descent solver with momentum 0.9. We trained for 100,000 mini-batches of size 1000 signals, scaling the learning rate by 0.9 every 1,000 iterations. We trained each network with four different initial learning rates: $10^{-1}$, $10^{-2}$, $10^{-3}$, and $10^{-4}$. At test time, we selected the learning rate that produced the lowest loss on the validation dataset.

The neural networks were constructed, trained, and tested using Caffe [38] running on a cluster of NVIDIA M60 GPUs.

### 3.3.3 Results

Figure 3.11 shows that the network trained with our empirical residual model performs equally well on the *in vivo* test data as dictionary matching, whereas the network with no noise and with various levels of Gaussian noise all fail to produce viable parameter maps. Table 3.3 contains the root mean squared error (RMSE) values for the different synthetic training data models. The RMSE was computed using the baseline dictionary matching T1 and T2 maps as a substitute for ground truth parameter maps.

The training sets for all networks are based upon the dataset of 100,000 simulated MRF signals, which are clean signals before any noise/residual is added. Table 3.3 also shows how each network performed on this database of clean MRF signals.

Table 3.4 contains the results of the ablation study on the empirical residual model, breaking down the various components of the model, Equation 3.22. Combining all empirical residual model components produced the best results. Sampling scaling and rotating the phase of the simulated signal by $\phi_1$ had a major impact on performance, while rotating the phase of the noise by $\phi_2$ only had an impact when included with the signal rotation, $\phi_1$.

We found that rotating the phase of the simulated signal before adding noise also significantly improves the Gaussian noise model. Figures 3.16 and 3.17 and Table 3.5 show improvements at all Gaussian noise levels when the $\phi_1$ rotation is included. In fact, the T1 map image quality for the Gaussian PSNR level 2.5 is fairly close to both the empirical

*Table 3.3*: Results comparing the Gaussian and empirical residual models. RMSE values are computed by comparing the predicted values to the T1 and T2 values found using the baseline dictionary matching method.

| | T1 (RMSE) | | T2 (RMSE) | |
|---|---|---|---|---|
| **Synthetic model** | **Clean** | **Test** | **Clean** | **Test** |
| Clean signal | **0.002** | 0.805 | **0.002** | 0.148 |
| Gaussian PSNR 60 | 0.009 | 0.896 | 0.004 | 0.157 |
| Gaussian PSNR 40 | 0.003 | 0.937 | 0.006 | 0.161 |
| Gaussian PSNR 20 | 0.005 | 1.039 | 0.013 | 0.215 |
| Gaussian PSNR 10 | 0.013 | 1.109 | 0.026 | 0.270 |
| Gaussian PSNR 5 | 0.027 | 1.143 | 0.050 | 0.317 |
| Gaussian PSNR 2.5 | 0.077 | 0.923 | 0.093 | 0.378 |
| Gaussian PSNR 1.25 | 0.223 | 0.731 | 0.165 | 0.186 |
| Gaussian PSNR 0.625 | 0.492 | 0.642 | 0.266 | 0.209 |
| Empirical residual model | 0.058 | **0.126** | 0.026 | **0.071** |

model and the dictionary matching baseline. The RMSE for the T2 Gaussian PSNR 1.25 actually drops slightly below the empirical residual model RMSE, although, the image quality as shown in 3.17 indicates that the empirical residual model best matches the baseline, especially in the gray matter regions.

*Table 3.4*: Root mean squared error results of ablation study for the empirical residual model, comparing the effect of omitting various components of the model.

| | | | | T1 (RMSE) | T2 (RMSE) |
|---|---|---|---|---|---|
| **Residual** | **Scaling** | $\phi_1$ | $\phi_2$ | **Test** | **Test** |
| No | No | No | No | 0.805 | 0.148 |
| Yes | No | No | No | 1.422 | 0.172 |
| Yes | No | Yes | Yes | 0.498 | 0.504 |
| Yes | Yes | No | No | 1.342 | 0.213 |
| Yes | Yes | No | Yes | 0.902 | 0.148 |
| Yes | Yes | Yes | No | 0.133 | 0.074 |
| Yes | Yes | Yes | Yes | **0.126** | **0.071** |

*Table 3.5*: Results comparing synthetic models with and without rotating the phase of the simulated signal by a random $\phi_1$ before adding noise/residual.

| | T1 (RMSE) | | T2 (RMSE) | |
|---|---|---|---|---|
| **Synthetic model** | **Without $\phi_1$** | **With $\phi_1$** | **Without $\phi_1$** | **With $\phi_1$** |
| Clean | 0.805 | 0.556 | 0.148 | 0.322 |
| Gaussian PSNR 20 | 1.039 | 0.554 | 0.215 | 0.181 |
| Gaussian PSNR 10 | 1.109 | 0.427 | 0.270 | 0.201 |
| Gaussian PSNR 5 | 1.143 | 0.272 | 0.317 | 0.161 |
| Gaussian PSNR 2.5 | 0.923 | 0.168 | 0.378 | 0.119 |
| Gaussian PSNR 1.25 | 0.731 | 0.191 | 0.186 | **0.069** |
| Gaussian PSNR 0.625 | 0.642 | 0.243 | 0.209 | 0.113 |
| Empirical residual model | 0.902 | **0.126** | 0.148 | 0.071 |

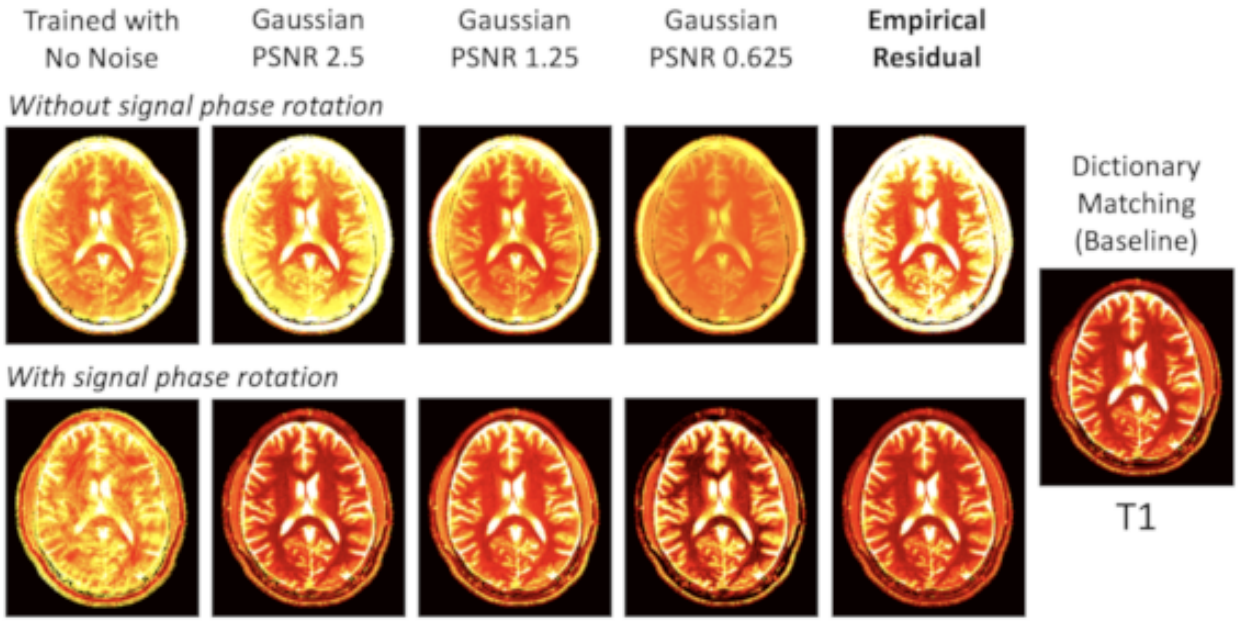


*Figure 3.16*: T1 parameter maps for one slice of test volume one, comparing the neural network predictions after training on synthetic models with and without rotating the phase of the simulated signal by $\phi_1$.

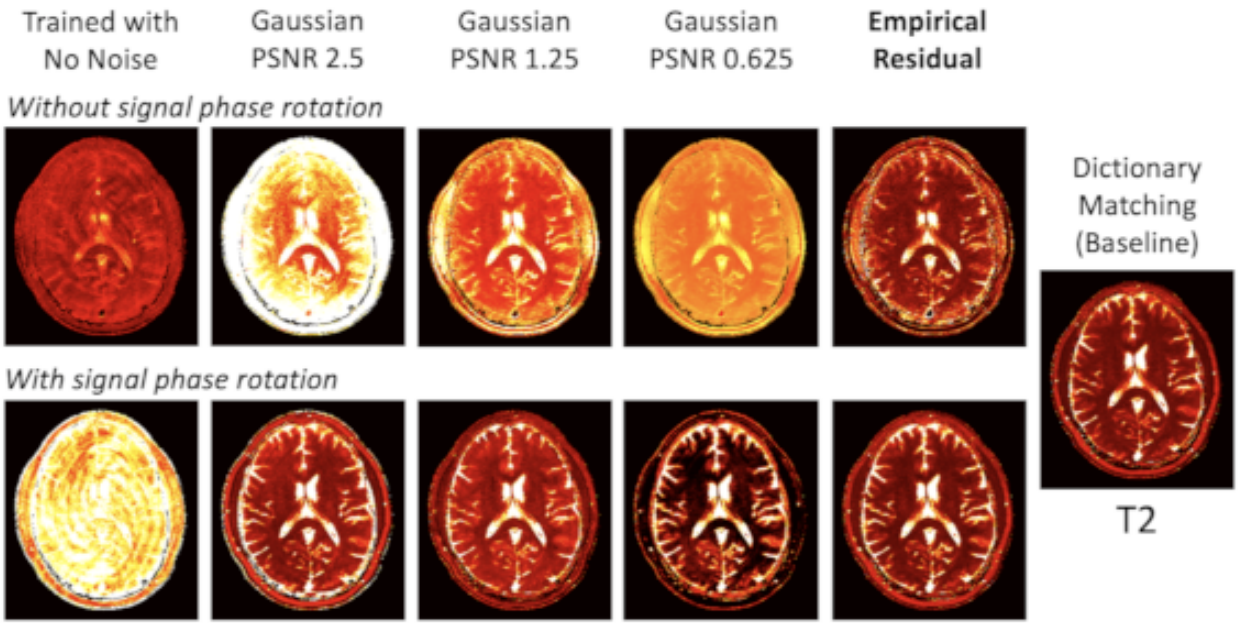


*Figure 3.17*: T2 parameter maps for one slice of test volume one, comparing the neural network predictions after training on synthetic models with and without rotating the phase of the simulated signal by $\phi_1$.

### 3.3.4 Discussion

As expected from results in Section 3.2, the networks trained with no noise or very little Gaussian noise perform quite well on the clean training signal, but these networks overfit to this clean signal and perform poorly on the *in vivo* test dataset, Table 3.3. A significant amount of Gaussian noise, down to PSNR 1.25, must be added during training in order for the test result to start surpassing the network trained on the clean signal. Unfortunately, at that point, the noise level is so high that the Gaussian model network does not produce adequate test results or even clean signal results. The network trained with the proposed empirical residual model learns a parameter mapping function that performs well on both the clean training signal and the *in vivo* test signal.

Our proposed empirical residual model augments the clean simulated MRF signal with an empirical sample of the residual (noise coupled with aliasing interference), an empirical sample of signal-to-residual ratio, and random phase rotations. When training on this empirical residual model, the network becomes robust to variations that are seen in the *in vivo* training set. Our empirical residual model samples the aliasing energy in the undersampled MRF signal that corrupts the image. This residual is signal-dependent and one of the reasons why the Gaussian noise model is not sufficient.

Through the ablation study of the empirical residual model, we found that both **scale sampling** and **randomly rotating the phase of the simulated signal** were critical to the success of the model, Table 3.4. Acquisition noise may be relatively constant across

MRF signals, but the proton density of different tissues causes the SNR to vary across the images. Additionally, in undersampled acquisitions, the signal-to-residual ratio is affected by aliasing energy which is also dependent on the signal level. By sampling from a distribution of residual scaling factors, the network can learn to be robust to the variety of signal-to-residual ratio values in MRF images. The distribution of signal-to-residual ratio in an MRF dataset may be one of the reasons that the Gaussian noise models trained a signal noise level failed. Future work could consider training with a heterogeneous set of Gaussian noise levels.

Our MRF EPG simulator does not account for an arbitrary absolute phase shift, nor does it factor in the change in phase caused by incoherent undersampling. In fact, given that our FISP MRF sequence is designed to be insensitive to B0 inhomogeneity, all of our simulated MRF signals have zero phase, i.e. they are all real-valued. Without rotating the phase of these simulated signals, our network would overfit to the homogeneous phase in the training set. Including a random signal phase in both our empirical residual model and Gaussian noise model was essential to network performances as shown in our results in Figures 3.16 and 3.17.

As with any data-driven learning, our empirical residual model is susceptible to overfitting to the training dataset. Our networks were both trained and tested on healthy volunteers, who were all scanned at the same institution. It is important for future applications of this work to train networks with empirical residual model data acquired across a variety of clinical scenarios, including anatomy, pathology, and scanner environment. It is equally important to test the trained networks using datasets that were independently acquired and representative of the application domain.

## 3.4 Direct Contrast Synthesis

MRF parameter mapping and conventional parameter mapping aim to provide consistent tissue measurements that enable quantitative comparison between exams, leading to improved post-processing and longitudinal studies [89]. However, clinicians still rely on standard contrast-weighted images, such as T1-weighted and T2-weighted images. If one could synthesize the contrast-weighted image from the MRF acquisition data, there would be a significant reduction in scan time. For example, a single four minute MRF sequence that produces quantitative maps could replace the ten minutes needed to acquired T1-weighted, T2-weighted, and FLAIR sequences, saving six minutes. Reduced exam times provide myriad clinical and financial benefits including increased patient throughput, improved patient comfort, and reduction of motion artifacts.

**Contrast synthesis via parameter maps.** One approach to generating contrast-weighted images from an MRF acquisition is to synthesize the contrast-weighted images using the MRF parameter maps and the MR signal equations [22], shown by Figure 3.18, dashed blue line. With the values from T1, T2, and proton density maps, we can simulate how the tissue at each pixel location will react to different pulse sequences, such as gradient echo, spin echo, and inversion recovery sequences. The concept of MRI synthesis dates back to 1985 [27], and techniques such as QRAPMASTER [72] have recently been shown to produce clinically viable images [90]. Unfortunately, MRI synthesis techniques from MRF parameters are still limited by biases, due to effects that are difficult to simulate and fit to, such as time varying signals, partial volume and flow. In addition, to get more accurate modeling, one must include slice profile, diffusion, magnetization transfer, and other contrast mechanisms and imaging parameters. These increase the dictionary size and can cause over-fitting and even further biases. In order to keep multi-parameter dictionaries from becoming computationally impractical, fewer values are sampled for each parameter, leading to discretization errors that limit the accuracy of the synthesized images.

**Direct contrast synthesis (DCS).** Deep learning methods excel at quickly and accurately converting an acquired image into a segmented version of that image [91, 92, 93], but they can also transform image data from one domain to another, for example converting images from day to night or from an aerial image of a city to a map rendering of that same location [94]. Recent work has shown that neural networks can also successfully transform acquired MRF signal data in the image domain into quantitative parameter maps, bypassing the computationally expensive dictionary matching [42, 31, 79, 14, 13]. In a similar manner, we propose training neural networks to extract contrast-weighted images directly from the MRF data, shown by Figure 3.18, solid red line. Using training data consisting of image-domain MRF data paired with acquired contrast-weighted images, the neural network learns to predict T1-weighted, T2-weighted, and FLAIR images from a single MRF acquisition. Our *in vivo* MRF experiments demonstrate that our direct contrast synthesis technique provides more accurate synthetic contrast images than can be generated via T1, T2, and proton density parameter maps obtained with MRF. Since the MRF sequence is broadly sensitive to the tissue parameters, this approach could be used to synthesize other contrast weighted

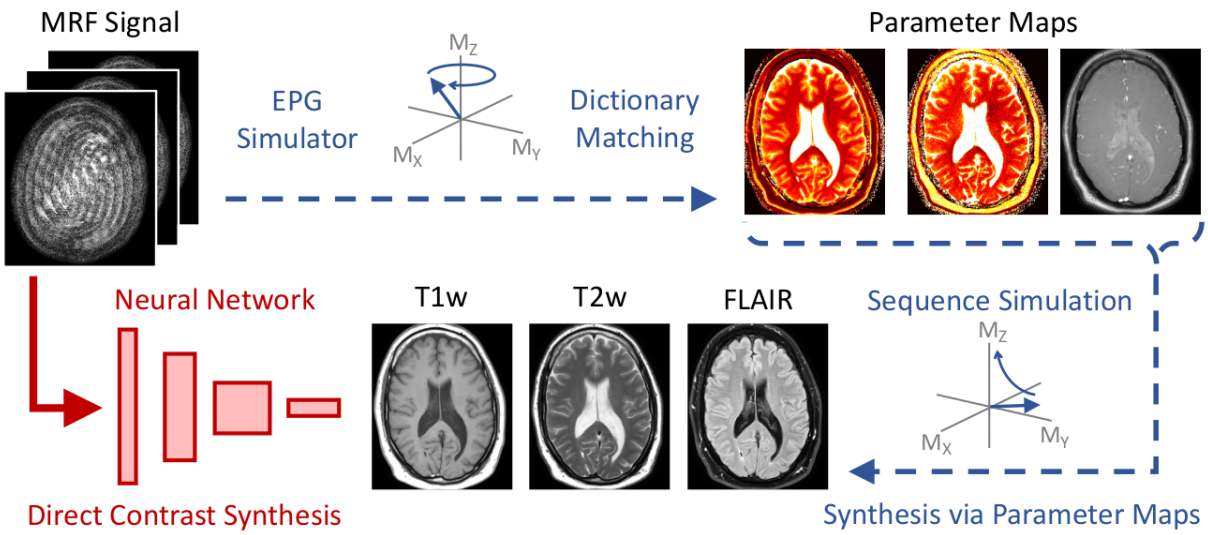


*Figure 3.18*: Two approaches to contrast synthesis from MR fingerprinting: *indirectly* via parameter maps (blue-dotted lines) versus *directly* from MRF signal (solid-red line). Direct contrast synthesis (DCS) uses a trained neural network to transform the MRF signal directly into many different contrast-weighted images. DCS bypasses two simulation steps and avoids incomplete modeling assumptions and error propagation.

images by collecting other paired data.

### 3.4.1 Methods

#### Data acquisition and preprocessing

**Acquisition and reconstruction.** With IRB approval, we scanned 21 volunteers, ages ranging from 29 to 61 years, with a 1.5T Philips Ingenia scanner using a head coil with 13 receive channels. Four consecutive axial brain sequences were acquired: T1-weighted spin echo with TE=15 ms, TR=450 ms, FA=69, two averages; T2-weighted turbo spin echo (TSE) with TE=110 ms, TR=1990 ms (twelve of the exams used TR=2215 ms), ETL 16, FA=90, two averages; FLAIR inversion recovery TSE with TE=120 ms, TR=8500 ms, TI=2500 ms, ETL=41, FA=90, two averages; and a fingerprinting spoiled gradient echo (FISP) sequence based upon [22] and [23] with 500 time points, constant TE=3.3 ms and TR=20 ms. Following [23], the MRF flip angle pattern was chosen from the best of Monte Carlo simulations with 500 time points, Figure 3.19, dashed-green line. The MRF data were acquired with a spiral acquisition with two spiral readouts per TR interval, rotated at 180°. The spirals between two consecutive time points were rotated by 9°. All of the scans were acquired with in-plane resolution of 0.72x0.72 mm (FOV 230x230 mm, matrix size 320x320) and slice thickness 5 mm. Each exam acquired the same nine or ten slices of the brain for all four acquisitions. Of the 21 subjects, 17 were scanned twice (on different days), resulting

in a total of 38 exams. FLAIR sequences were acquired for only 26 of the 38 exams.

Each TR of the spiral MRF acquisition was reconstructed to image space using gridding with density compensation [84, 85] and coil combination with Philips CLEAR [86]. Both the magnitude and phase of the reconstructed MRF signal were retained and stored as complex values. Figure 3.19 shows an example of the reconstructed MRF image for three different time points, as well as an image of the MRF signal, averaged across the temporal dimension and then reconstructed.

**Pruning and partitioning.** At training time, we rely on the fingerprinting data to be spatially aligned with each of the contrast images. If a subject moves or if the scan prescription is altered between scans, the pixels will not map to the same subject location for all four acquisitions, and the network would be trained to convert the fingerprinting data into the wrong contrast values. To avoid this issue, the alignment of all four acquisitions for each exam were visually inspected, and any exams containing patient movement were dropped from the study. Specifically, the MRF mean image (averaged across the 500 time points) showed no patient movement between any of the contrast images for 27 of the 38 exams; the remaining 11 exams were not used. Of the 27 aligned exams, 23 were used for training, two for validation, and two only for the final test results. The subjects of the two test exams were not included in any of the training or validation exams.

**Normalization.** Contrast-weighted MR images have arbitrary scaling factors for each exam. To reduce this variation in the training, validation, and test data, each contrast-weighted image was divided by a representative value of white matter in that image, producing a normalized image where white matter regions, often the largest regions, have values at or near one. The representative white matter value was automatically determined by computing a 50-bin histogram of the data and selecting the center value of the highest bin, excluding the background bins near zero. The same process was followed to normalize the MRF data, but the white matter scaling factor was found using the histogram of the magnitude of the complex mean image of the MRF data (averaged across the 500 time points). Note that more robust normalization methods will be required to normalize different anatomical regions or brain images where white matter is not the largest region.

**Foreground mask.** As the imaging FOV is larger than the scan subject, a significant portion of each image falls beyond the extent of the subject. To avoid training on near-zero background points, a binary foreground mask was created to exclude them. We automatically construct this mask by thresholding the magnitude of the complex mean image of the MRF data (after normalization) at 0.01 or greater. The binary mask was then dilated twice, with a 4-connected structuring element, to close any holes near the anatomy. This same masking method was used at test time with the goal of directly mapping extremely low MRF signals to zero in all contrast images.

### Contrast synthesis via parameter maps

**Parameter maps.** We simulated a dictionary of MRF signals with the extended phase graph (EPG) algorithm [87, 88]. The dictionary consisted of 29,981 MRF signals for: T1

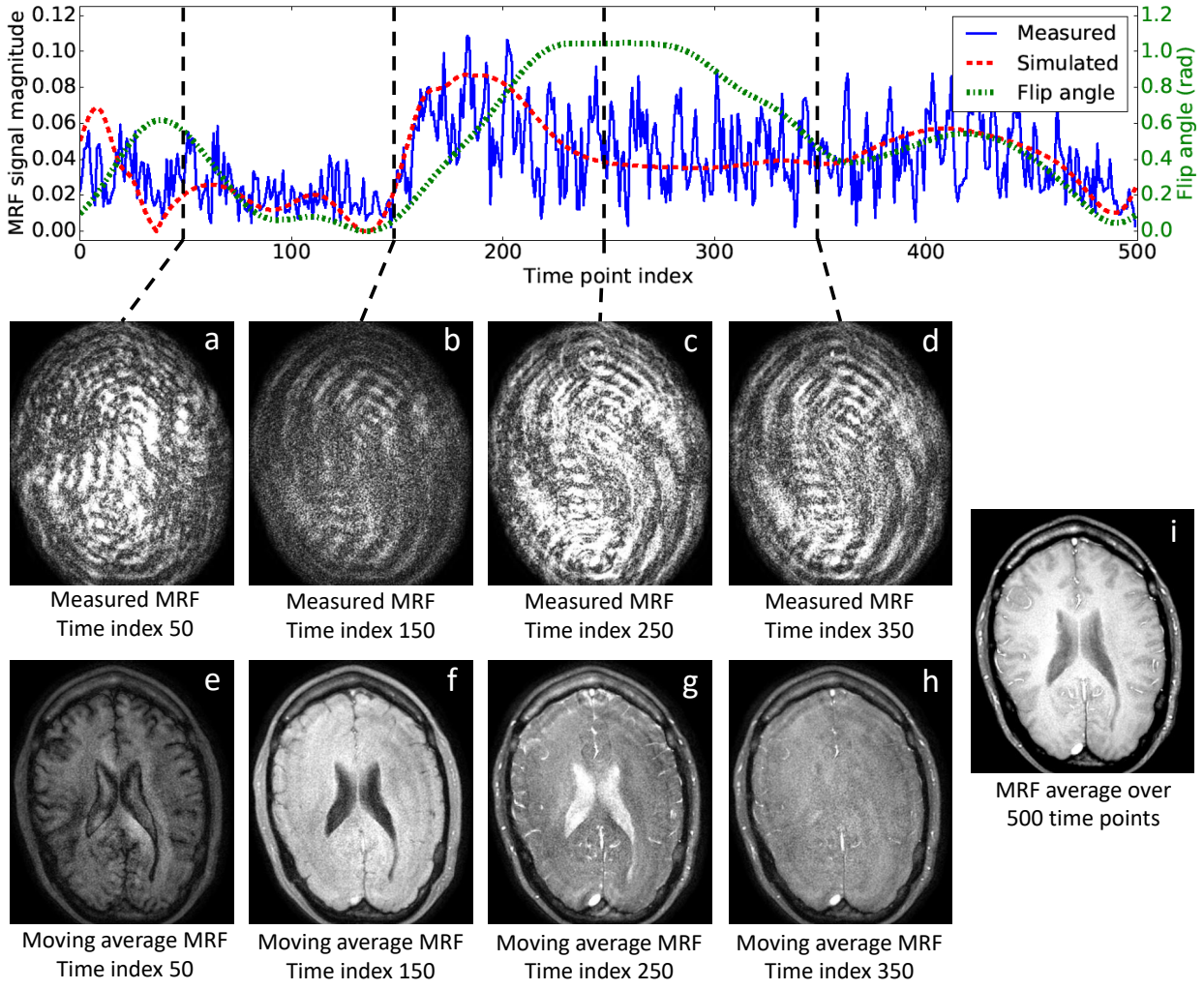


*Figure 3.19*: Top: MR fingerprinting flip angle for each of the 500 time points (dashed-green line) shown with the magnitude of acquired MRF signal (solid-blue line) and the closest simulated MRF signal (dashed-red line), T1 = 1280 ms, T2 = 94 ms. The measured signal is corrupted by both acquisition noise (signal independent) and incoherent aliasing (signal dependent). Bottom: magnitude images of the measured MRF signal at four different time points (a-d), the moving average with kernel size 32 at four different time points (e-h), and the magnitude of the complex mean MRF across 500 time points (i).

parameters in increments of 2 ms from 4 ms to 100 ms, increments of 10 from 100 to 1000, increments of 20 from 1000 to 2000, and increments of 40 ms from 2000 to 5000 ms; and T2 parameters in increments of 2 ms from 2 to 150, increments of 10 from 150 to 500, increments of 20 from 500 to 1000, and increments of 40 ms from 1000 to 2000 ms. Each simulated signal in the dictionary was scaled to have a Euclidean norm equal to one. As in [6], we used cosine similarity to match the acquired MRF signal to the nearest neighbor in the simulated dictionary. Figure 3.19, bottom left, shows one of the acquired MRF signals along with its nearest signal in the simulated dictionary. Additional factors, such as B1 inhomogeneity and slice profile, were not included in the simulated dictionary.

**Contrast sequence simulation.** The parameter maps from MRF dictionary matching were converted to T1-weighted and T2-weighted contrast images by simulating spin echo sequences using the TE, TR, and flip angle parameters specified in the data acquisition section above:

$$\text{SpinEcho}\left(PD, T1, T2, TE, TR, \theta_2\right) \tag{3.24}$$

$$= PD \sin\theta_1 \sin^2\left(\frac{\theta_2}{2}\right) \frac{1 + (\cos\theta_2 - 1)e^{-(TR-TE/2)/T1} - \cos\theta_2 e^{-TR/T1}}{1 - \cos\theta_1 \cos\theta_2 e^{-TR/T1}} e^{-\frac{TE}{T2}}$$

$$= PD \sin^2\left(\frac{\theta_2}{2}\right)\left(1 + (\cos\theta_2 - 1)e^{-(TR-TE/2)/T1} - \cos\theta_2 e^{-TR/T1}\right) e^{-\frac{TE}{T2}} \tag{3.25}$$

where $PD$ is the proton density, $\theta_1 = 90°$ is the excitation flip angle, and $\theta_2$ is the refocusing flip angle [95]. The proton density was computed by taking the magnitude of the dot product between measured and simulated MRF signals.

Likewise, parameter maps were converted to FLAIR contrast images by simulating an inversion recovery spin echo sequence [96, 97]:

$$\text{IRSpinEcho}\left(PD, T1, T2, TE, TR, TI, TE_{last}\right) \tag{3.26}$$

$$= PD \cdot \left(1 - 2e^{-\frac{TI}{T1}} + e^{-\frac{(TR-TE_{last})}{T1}}\right) e^{-\frac{TE}{T2}}$$

where $TE_{last}$ is the echo time of the last echo in the echo train, which was 234 ms for the acquired FLAIR sequences.

### Direct contrast synthesis (DCS)

**Network.** Following [31], a separate neural network was trained for each output parameter: T1-weighted, T2-weighted, and FLAIR. Because our spoiled gradient echo fingerprinting sequence is designed to be insensitive to B0, the phase of the underlying signal is consistent across all time points and does not carry any information, so we chose to use the two

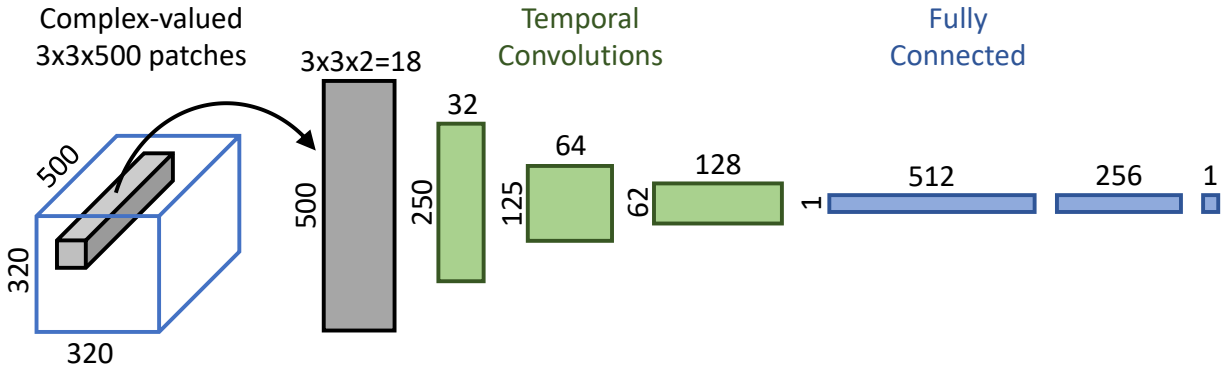


*Figure 3.20*: Patch-wise neural network architecture for direct contrast synthesis. 3x3 spatial patches are flattened and passed through three convolutional layers and then three fully connected layers, resulting in a contrast value prediction for the center of the input patch. Between each layer is a ReLU non-linear filter. The number of feature channels is shown above each block, while the size of the temporal dimensions is shown to the left. An L2 loss function is used to penalize predicted values that do not match the acquired contrast value.

channel real/imaginary network from [31], rather than a complex-valued neural network. Due to the temporal correlation across the highly undersampled MRF acquisition, temporal convolution layers, interspaced with ReLU activation layers, were added to the beginning of the network. These convolution layers gradually reduce the temporal dimension, leading to a non-linear compressed representation similar to the linear reduction of the temporal dimension by singular value decomposition in [26].

The neural networks for direct contrast synthesis were trained on approximately 10 million 3x3 patches from the *in vivo* MRF training data with no data augmentation. The 3x3 overlapping spatial patches with 500 temporal values and real and imaginary channels were flattened and passed through three temporal convolutional layers and then three fully connected layers, resulting in a contrast value prediction for the center of the input patch. The temporal convolutions used a kernel of size three and Xavier initialization computed using the number of input channels [82]. Between each layer is a ReLU non-linear activation function. The number of feature channels and the size of the temporal dimensions are shown in Figure 3.20. An L2 loss function was used to penalize predicted values that do not match the acquired contrast value. The neural networks were constructed, trained, and tested using Caffe [38] running on a cluster of NVIDIA M60 GPUs.

In addition to the patch-based network, we also trained a network using only a single MRF pixel as input. This pixel-wise network used the same architecture as the patch-wise network in Figure 3.20, except that the input is 500x2 rather than 500x18 and because of this smaller dimensional input, we reduced the number of feature channels for the three temporal convolutions from 32, 64, and 128 down to 8, 16, and 32, respectively.

To compensate for the undersampling artifacts in the MRF data as well as the low SNR due to relatively high spatial resolution, we also experimented with adding an average

pooling layer at the very beginning of the network. A pooling stride length of 1 and kernel of size 32 were used in the temporal dimension. This layer has no learned parameters and acts as a fixed *moving average* preprocessing step to denoise the input data, similar to [98]. This pooling operation was not padded, which resulted in reduced temporal dimensions from those shown in Figure 3.20, specifically 469, 235, 118, and 59 rather than 500, 250, 125, and 62.

To investigate the effect of limited data, the pixel-wise networks were trained with only five exams and again with only ten exams, in addition to being trained with all 23 training exams.

**Mini-batch selection.** During training, a mini-batch size of 1000 was used for both the pixel-wise and patch-wise networks. The points/patches of each exam were partitioned into 100 different files. While training, files were chosen at random without replacement, repeating after processing all training files. Similarly, mini-batches were randomly populated from each file without replacement, then a new random file was chosen.

**Solver.** The networks were trained using a stochastic gradient descent solver with momentum 0.9. Each network was trained with four different initial learning rates: $10^{-1}$, $10^{-2}$, $10^{-3}$, and $10^{-4}$. While training, the learning rate was scaled by 0.5 every 10,000 iterations. We trained for 40,000 and 100,000 mini-batch epochs for the patch-wise and pixel-wise networks, respectively. At test time, we selected the learning rate that produced the lowest loss on the validation dataset.

### Image comparison

**Quantitative comparison.** Root mean squared error (RMSE) quantitative values were computed for both contrast synthesis via parameter maps and direct contrast synthesis methods. RMSE values were calculated using the square root of the mean squared differences between the synthesized image and the acquired contrast image. The RMSE values are computed from all pixels in our foreground mask across all slices of the two test exams. Specifically, we compute:

$$\text{MSE}_i = \frac{1}{N_i} ||\boldsymbol{y}_i - \boldsymbol{x}_i||_2^2 \tag{3.27}$$

$$\text{RMSE} = \sqrt{\frac{1}{2} \sum_{i \in \{1,2\}} \text{MSE}_i} \tag{3.28}$$

where $\boldsymbol{y}_i \in \mathbb{R}^{N_i}$ is the vector of all $N_i$ foreground pixels in the acquired contrast-weighted images of the $i$-th test exam and $\boldsymbol{x}_i \in \mathbb{R}^{N_i}$ is the vector of all $N_i$ foreground pixels in the direct contrast synthesis images of the $i$-th test exam.

**Scaling for comparison.** In general, MRI contrast-weighted images have an arbitrary scaling factor as mentioned in the normalization preprocessing section above. The direct contrast synthesis method learns to output contrast images with the same scaling as the contrast images in the training set. Thus, because the acquired contrast-weighted images

in the test set were also normalized in the same manner as the training set, the direct contrast synthesis images do not require any scaling when comparing them via RMSE to the acquired images. However, contrast synthesis via parameter maps images does not have this normalization. To allow for the most optimistic RMSE values for these baseline images, we compute and apply the scaling factor, $\alpha$, that minimizes the sum of squared error between the contrast synthesis via parameters image data, $\boldsymbol{x}$, and the acquired contrast-weighted image data, $\boldsymbol{y}$. Specifically, we compute:

$$\alpha_i^* = \underset{\alpha}{\operatorname{argmin}} ||\boldsymbol{y}_i - \alpha \boldsymbol{x}_i||_2^2 \tag{3.29}$$

$$= \frac{\boldsymbol{y}_i^T \boldsymbol{x}_i}{\boldsymbol{x}_i^T \boldsymbol{x}_i} \tag{3.30}$$

$$\text{RMSE} = \sqrt{\frac{1}{2} \sum_{i \in \{1,2\}} \frac{1}{N_i} ||\boldsymbol{y}_i - \alpha_i^* \boldsymbol{x}_i||_2^2} \tag{3.31}$$

where $\boldsymbol{y}_i \in \mathbb{R}^{N_i}$ is the vector of all $N_i$ foreground pixels in the acquired contrast-weighted images of the $i$-th test exam, $\boldsymbol{x}_i \in \mathbb{R}^{N_i}$ is the vector of all $N_i$ foreground pixels in the contrast synthesis via parameter maps images of the $i$-th test exam, and $\alpha_i$ is the computed scaling factor for the $i$-th test exam.

### 3.4.2 Results

The training times for a single learning rate of one contrast-weighted network were 4.5 hours and 1.5 hours for the patch-wise and pixel-wise networks, respectively, when running on a single GPU of an NVIDIA M60 card. With that same GPU at test time, a single slice was processed in 1.5 seconds and 0.4 seconds, for the patch-wise and pixel-wise networks, respectively. Example comparisons between the contrast synthesis via parameter maps and the proposed direct contrast synthesis are shown in Figure 3.21. Direct contrast synthesis consistently produced lower root mean squared error and higher qualitative image quality than contrast synthesis via parameter maps, which contain significant artifacts, especially in the vasculature and cerebrospinal fluid (CSF). Our pixel-wise network has better qualitative image quality than that of the patch-wise network, which appears smoother. However, the patch-wise network does provide better results for regions near the skull, as well as lower root mean squared error, Table 3.6.

Figure 3.22 shows the T1, T2, and proton density parameter maps estimated from the MRF data using dictionary matching, and used to generate the contrast-weighted images shown in Figure 3.21, left column. The noisy T2 maps lead to noisy T2-weighted and FLAIR images synthesized via parameter maps. The high values for vasculature in the generated proton density map amplify errors in the synthesized contrast images, but even if the proton density scaling factor is removed from Equation 3.25, the synthesized T2-weighted images still contain erroneous bright vessels, Figure 3.22, bottom.

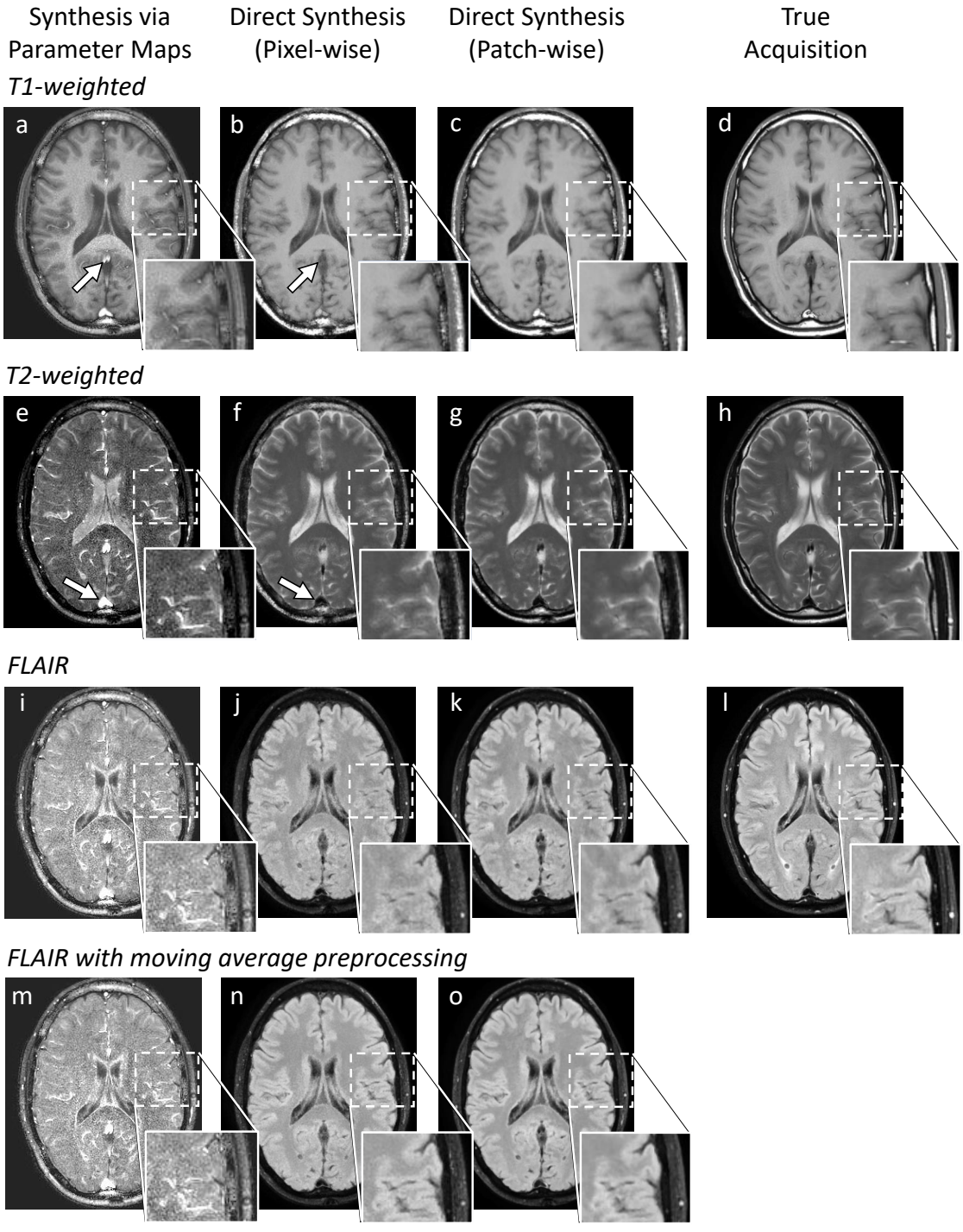


*Figure 3.21*: Results for both contrast synthesis via parameters and direct synthesis on test exam number one for T1-weighted (a-c), T2-weighted (e-g), FLAIR (i-k), and FLAIR from an MRF signal preprocessed with a moving average filter (m-o). Results are shown from training direct contrast synthesis networks with MRF pixels (b,f,j,n) and MRF 3x3 spatial patches (c,g,k,o). Note that the synthesis via parameters method presents inconsistent vessel contrast (white arrows) for all three contrast-weighted images, most noticeably in the superior sagittal sinus.

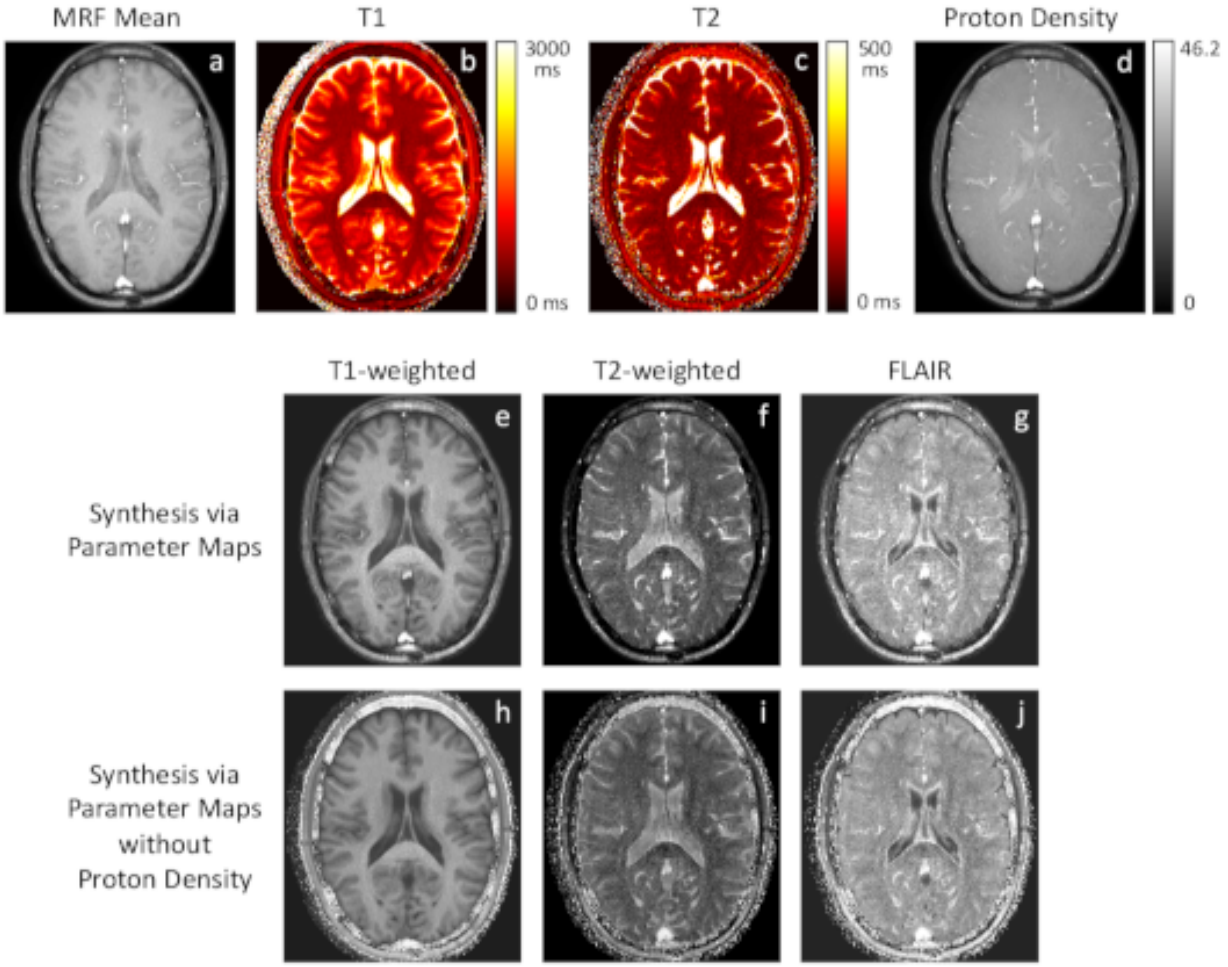


*Figure 3.22*: Quantitative maps for one slice of test exam number one computed as part of the contrast synthesis via parameters method. The MRF data, shown as a mean over the 500 time points (a), is matched to the nearest neighbor in a simulated MRF dictionary to produce T1 maps (b) and T2 maps (c). These maps, as well as the computed proton density (c), are then used to indirectly synthesize contrast-weighted images )(e-g). To illustrate the effect of the proton density values, contrast-weighted images synthesized via parameter maps are also shown without the proton density scaling factor (h-j).

*Table 3.6*: Root mean squared error (RMSE) quantitative results for both contrast synthesis via parameter maps and direct contrast synthesis (DCS) methods. The second column contains the number of training exams used for each result. †The FLAIR results were trained with 11 exams rather than 23.

| Method \ RMSE | Training exams | T1w | T2w | FLAIR |
|---|---|---|---|---|
| Synthesis via parameter maps | N/A | 0.374 | 0.614 | 0.522 |
| DCS pixel-wise | 5 | 0.317 | 0.429 | 0.175 |
| DCS pixel-wise | 10 | 0.294 | 0.404 | 0.164 |
| DCS pixel-wise | 23† | 0.290 | 0.399 | 0.163 |
| DCS pixel-wise, moving average | 23† | 0.289 | 0.402 | 0.175 |
| DCS patch-wise | 23† | 0.221 | 0.300 | **0.136** |
| DCS patch-wise, moving average | 23† | **0.211** | **0.294** | 0.140 |

Figure 3.23 and Table 3.6 show the improved image quality as the number of training exams is increased. With more training data, noise and artifacts decrease in the resulting reconstruction, most noticeably in the CSF of the T1-weighted images. Figure 3.21 shows little difference between training and testing results when training with 23 exams, which indicates that we are not overfitting to specific subjects. That being said, our training and test sets both came from healthy volunteers scanned at the same institution, so further study is required to ensure that our networks are robust to various clinical scenarios.

### 3.4.3 Discussions and conclusions

Our direct contrast synthesis results demonstrate the feasibility of generating many contrast-weighted images from a single MRF acquisition, which could significantly reduce exam times. By learning to directly convert the MRF signal into desired contrast-weighted images, a trained neural network bypasses the two simulation steps required to synthesize contrast images via parameter maps. Our DCS network produces more accurate image contrast and contains far fewer artifacts compared to synthesizing via MRF parameter maps. These image quality gains are most prevalent in regions containing vasculature, as flow is particularly difficult to model when synthesizing via parameter maps [99].

**Synthesis via parameter maps.** The sources of error in the baseline contrast synthesis via parameter maps include: 1) factors that were not included in the dictionary simulation; 2) approximations and error propagation in synthesis simulation; and 3) noise and artifacts from high spatial resolution and under-sampling of the MRF acquisition, which also impacts the direct contrast synthesis method.

Contrast synthesis via parameter maps could be improved by modeling more effects during simulation, specifically B1 inhomogeneity, slice profile, flow, partial volume, and fat. Unfortunately, including more simulation parameters quickly becomes a combinatorial problem, forcing the dictionary to explode in size or severely sacrifice parameter resolution and range.

One benefit of using a simulation model to synthesize contrast-weighted images is it

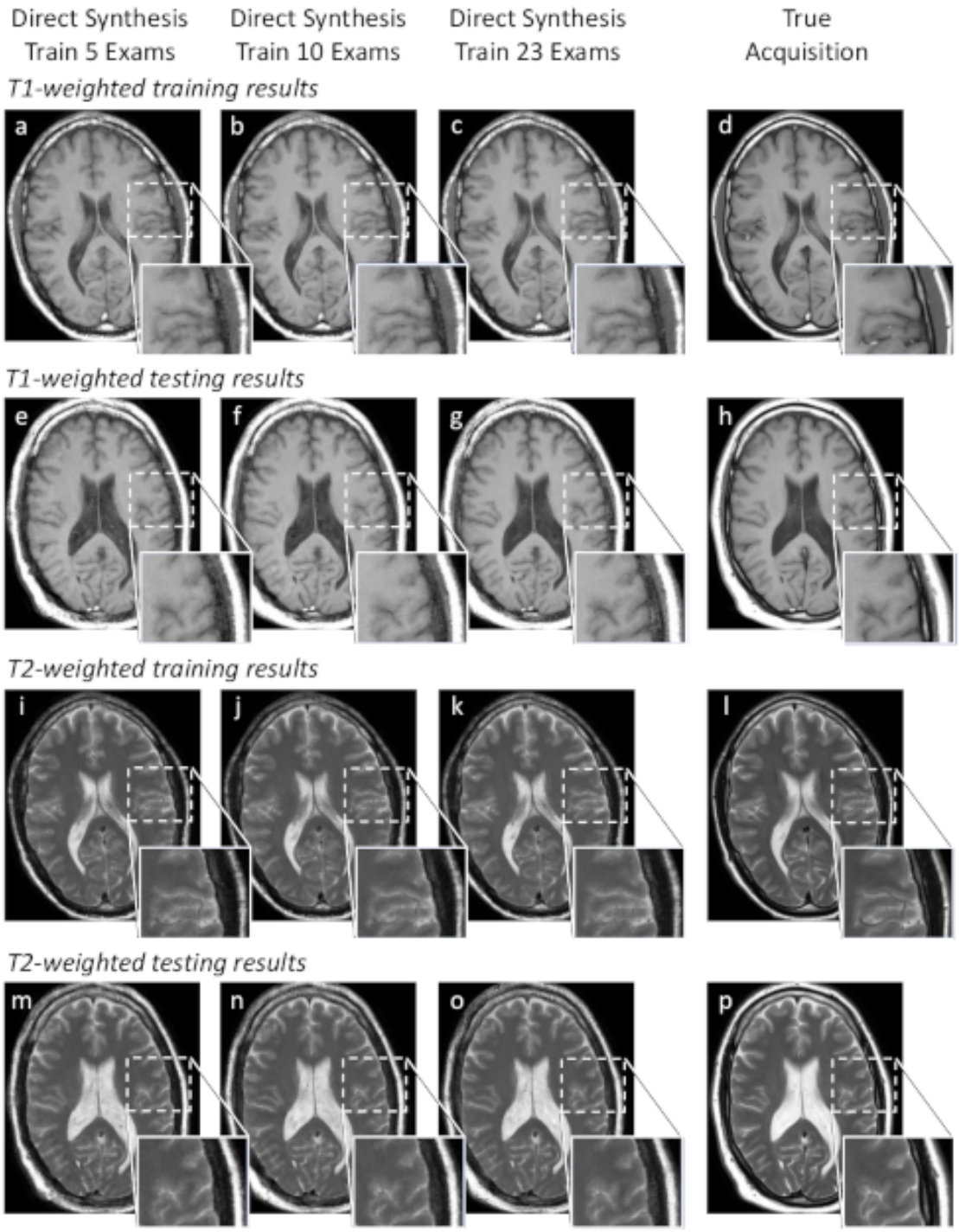


*Figure 3.23*: Training and testing results from pixel-wise direct contrast synthesis networks trained using 5, 10, and 23 exams. Note how the noise and artifacts are reduced as the number of training exams increases. The first and third rows show the network output using an MRF exam from our training data, while the second and fourth rows show results for test exam number two, which was never used for training or architecture/hyperparameter selection.

inherently comes with a means to analyze how parameter values affect the contrast value at each pixel. When shifting to a deep learning model for contrast synthesis, this straightforward analysis is lost to the complexities of a trained neural network. Golbabaee, et al. [78] begin to address this problem for MRF parameter map networks, but analysis and explainability remains an open challenge in deep learning.

**Neural network training data.** Direct contrast synthesis uses the training data to learn a mapping from MRF signals to contrast-weighted images without explicitly modeling any of the aforementioned effects. The DCS training data also includes a natural prior on the distribution of these effects, for instance, partial volume proportions at tissue interfaces. While Figure 3.23 shows that even a limited number of training exams can approximate the correct image contrast, more training data is required to improve this empirical prior distribution. As with all data-driven algorithms, care must be taken to acquire a representative training set to avoid overfitting to specific anatomy, pathology, or scanning environment.

Collecting training data for DCS is relatively straight-forward, especially in comparison to the time-intensive acquisitions to obtain "ground truth" parameter maps. Assuming a cooperative subject who can limit movement between sequences, the training data only requires adding an MRF sequence with matching acquisition geometry (spatial resolution, slice thickness, FOV) onto the standard clinical protocol. Training data will need to be curated to ensure all series are aligned, but registration algorithms could be incorporated to detect alignment errors, as well as correct in-plane motion.

**Network architecture.** The pixel-wise networks presented in this work are small, efficient, and train on the 10 million different MRF signals in our training set. The convolutions along the temporal dimension take advantage of the structure of the MRF signal across repetition time points and gradually reduce the 500 temporal values to a single contrast-weighted pixel value. However, given the acquisition noise and undersampling, the pixel-wise network does not quite reproduce the image quality of the contrast-weighted acquisitions. Training on individual pixel locations does not take advantage of the spatial similarity in the desired images. Our patch-based network begins to incorporate neighboring pixel signals and, in some areas, is able to resolve finer structure, but the patches are still treated independently, and the spatial information is all combined in the first convolution layer.

Image-to-image convolutional networks, such as U-net [92], would learn to preserve spatial structure throughout the network, but a default implementation would treat the temporal dimension as multiple channels and immediately combine them, essentially attempting to learn the MRF signal in the first layer. A hybrid approach that combines an image-to-image network with temporal convolutions could be beneficial. This combined approach would have to be conscious of computing hardware limits on this larger network, and any training at the image level would require a significantly larger dataset; while there are 10 million pixel locations in our training data, there are only 200 images.

**Acquisition and reconstruction.** Our MRF sequence was acquired with 0.72 mm in-plane spatial resolution in order to synthesize contrast-weighted images with spatial resolution seen in clinical protocols. However, this improved spatial resolution comes with the cost of lower SNR in the MRF data and residual noise in the synthesized images. To re-

duce the impact of lower SNR and undersampling artifacts, our methods introduced both moving average preprocessing and temporal convolutions in our neural network. To further improve image quality, future work could combine DCS with recent developments in MRF sequence design, such as improved MRF encoding capability [100], and integrate DCS with reconstruction methods, such as compressed sensing and low rank [101, 70, 24, 25, 102, 103].

# Chapter 4

# Empirical Effect of Gaussian Noise in Undersampled MRI Reconstruction

## 4.1 Introduction

Undersampling in Fourier-based medical imaging provides a variety of clinical benefits including shorter exam times, reduced motion artifacts, and the ability to capture fast moving dynamics, such as cardiac motion. Undersampling reduces acquisition time by collecting fewer measurements in the frequency domain than required by the Nyquist rate. However, undersampling causes two specific challenges for the reconstruction system, namely, an **underdetermined system**[1] of linear equations and **lower SNR** (signal-to-noise ratio) due to reduced measurement time. When reconstruction algorithms are able to overcome these challenges, undersampling can benefit a variety of Fourier-based imaging modalities, including MRI with parallel imaging or compressed sensing [104, 11], computed tomography (CT) with reduced or gated acquisition views [105, 106], and positron emission tomography (PET) with multiplexed or missing detectors [107, 108]. Undersampling for acceleration is becoming the mainstream approach for fast imaging. In fact, this year, two of the major MRI manufacturers have announced products that leverage undersampling and a compressed sensing reconstruction that have been approved by the FDA. While the tools and analysis discussed in this chapter apply generally to Fourier-based medical imaging with Gaussian noise, we will direct our numerical modeling, examples, and experiments to the application of compressed sensing MRI.

When designing an undersampled reconstruction system, the primary concern is often focused on compensating for the underdetermined system caused by sub-Nyquist sampling, for example choosing a sparse representation for compressed sensing. However, we should

[1] In the context of this chapter, we specify fully determined and underdetermined as follows: for a fixed Cartesian $k$-space (frequency space) grid with predefined field of view and spatial resolution parameters, **fully determined** means having at least one measured sample for each $k$-space grid location and **underdetermined** means at least one $k$-space location has zero samples, in which case we have more unknowns (image pixels) than equations (one per acquired $k$-space location).

not overlook the fact that collecting fewer measurements in practice leads to overall lower SNR in the acquired data. If the measurements are too noisy, the low SNR will lead to poor reconstructed image quality even if the reconstruction system was fully determined. On the other hand, with high SNR measurements, the resulting image quality will be limited by how well the reconstruction can constrain the underdetermined system. The effects of the underdetermined system and the lower SNR are coupled during the reconstruction process, making it difficult to analyze one without the other. It is important, however, to analyze how both issues impact the reconstruction system in order to determine the empirical limits of undersampling and gain insight on how to improve undersampled acquisition and reconstruction when targeting specific applications.

Compressed sensing theory has provided us with extensive analysis of the bounds for the successful signal recovery from undersampled data. Candès [28] describes a bound on the squared error of the recovered signal limited by the undersampling rate and the sparsity of the data. He also shows that this bound scales linearly with the variance of the noise in the measured data. Candès and Plan [29] provide a more general compressed sensing theory that addresses a combination of practical concerns. For instance, they derive the bounds on the squared error of the recovered signal for systems with Fourier encoding matrices, noise measurements, and approximately sparse signals. Unfortunately, while the squared error is an important tool in measuring the similarity between signals, it often fails to provide a good measure of perceptual image quality. Wainwright [30] improves upon the squared error definition of success by studying the undersampling rates and sparsity levels for which there is a high probability of successfully recovering the support of the sparse signal.

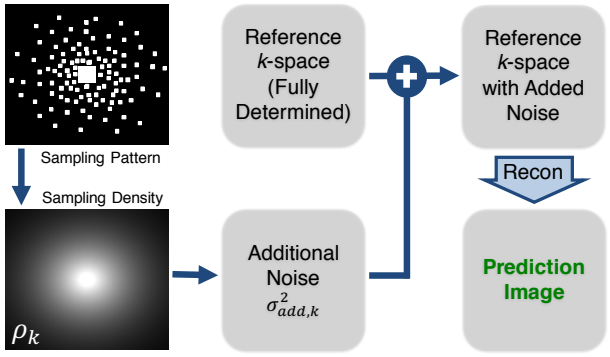


*Figure 4.1*: Prediction of Image Quality: The process to add the proper amount of noise to fully-sampled reference $k$-space and reconstruct an image affected by lower SNR due to reduced acquisition time but not affected by an underdetermined system. The expected measurement time at each $k$-space location, $\tau_k$, associated with the given sampling pattern is used to calculate the amount of noise (zero mean, complex Gaussian with variance $\sigma^2_{add,k}$) to add to each position in the reference $k$-space. This $k$-space with added noise is then processed by the reconstruction algorithm to produce the prediction image.

While it is important to have theory showing that reconstruction techniques are mathematically founded, when testing a reconstruction algorithm on a new undersampled clinical dataset and the image results are unacceptable, it is difficult to leverage the theoretical bounds to understand the cause of the failure. Conversely, when an undersampled reconstruction is successful at a certain undersampling rate, it is natural to then ask, how much further can we push undersampling? In this case, it is difficult to translate theoretic analysis, such as time constants for polylogarithmic bounds [29], into practice. Our goal in this chapter is to provide the tools to empirically analyze the effects of lower SNR from reduced measurement time using a reconstruction system that is fully determined, rather than underdetermined. To this end, we present the **image quality prediction** process (Fig. 4.1). The image quality prediction process takes a Nyquist-sampled (fully determined) reference dataset and adds the proper amount of noise in order to mimic the lower SNR produced by a given undersampling pattern. By reconstructing this noisy, but still Nyquist-sampled dataset, we have a **prediction image** that has been affected by lower SNR from reduced measurement time but not by artifacts from an underdetermined reconstruction. The image quality prediction process gives us the following three benefits:

- Comparing the prediction image to the reference reconstruction allows us to see the impact of lower SNR from reduced measurement time on the reconstruction system.
- Comparing the prediction image to the underdetermined reconstruction, we are able to assess the added effect of the underdetermined system on the reconstructed image.
- The prediction image provides a better estimate of undersampled image quality than over-optimistically comparing an underdetermined reconstruction to a fully-sampled reference reconstruction.

As exemplified in Fig. 4.2, for a given clinical application and undersampling pattern, pulse sequence and reconstruction developers can use the image quality prediction process to determine if low SNR, rather than the underdetermined system, is the limiting factor for a successful reconstruction. Specifically, an unsatisfactory prediction image indicates that the undersampled acquisition contains more noise than the reconstruction can handle. On the other hand, a high-quality prediction image and poor results from the underdetermined reconstruction indicate that the constraints on the underdetermined system are not adequate for the limited number of samples acquired. Once developers understand the limiting factor in a given undersampling application, they can then recommend changes to the acquisition protocol to adjust the measurement SNR or the undersampling rate. Developers can also appropriately focus their efforts on improving the reconstruction algorithm to better account for the noise distribution or to improve the reconstruction constraints, such as the sparsity model.

Before describing the details of the image quality prediction process, we first specify how measurement time affects SNR, specifically when undersampling. We complete this section by introducing a weighted least squares optimization that generalizes the reconstruction process for both undersampled data and the fully determined prediction data.

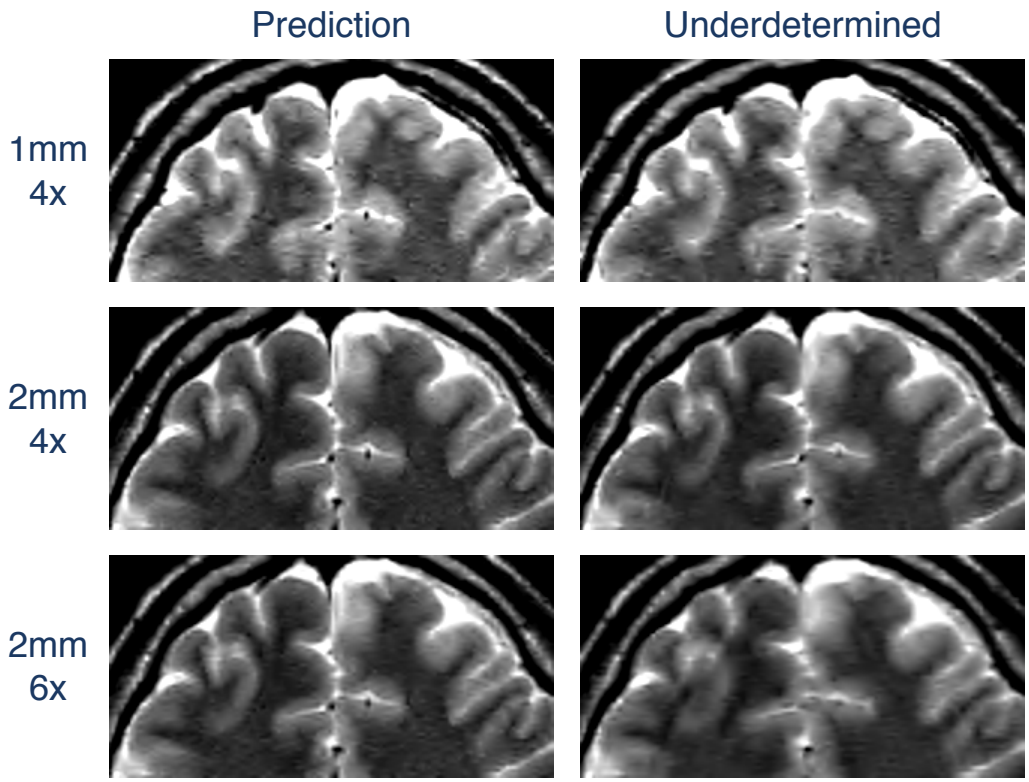


*Figure 4.2*: Using the image quality prediction process to adjust scan parameters. This 2D fast spin echo acquisition with 1 mm slice thickness and 4x undersampling produces poor reconstruction image quality (top right). The corresponding prediction image (top left) also has poor image quality, indicating that noise is the limiting factor. Increasing to 2 mm slice thickness (center row) reduces the noise and produces higher image quality in both the prediction and the underdetermined reconstruction. Further accelerating the scan with 6x undersampling (bottom row), the prediction image quality is significantly higher than the reconstruction image quality, indicating that the underdetermined system is the limiting factor for those scan parameters.

### 4.1.1 Measurement Time and SNR

For MRI reconstruction, we can model the signal $s$ as the discrete Fourier transform of the unknown target image object $\boldsymbol{m}$:

$$s_k = (F\boldsymbol{m})_k \tag{4.1}$$

where $F$ is the multidimensional discrete Fourier transform operator and $k$ is the $k$-th location in $k$-space. However, each measurement $s_k$ comes with an associated noise. We can model the noisy measurement $Y_k$ as:

$$Y_k \sim \mathcal{N}\Big(Re(s_k),\, \sigma^2_{acq}/\tau_k\Big) + i\mathcal{N}\Big(Im(s_k),\, \sigma^2_{acq}/\tau_k\Big) \tag{4.2}$$

where $Y_k$ is a random variable drawn from a complex-valued Gaussian distribution with mean $s_k$ and variance defined by the system noise variance, $\sigma^2_{acq}$, scaled by one over the measurement time, $\tau_k$, as described in [83]. With this definition, we assume that the signal is deterministic based on our model, the signal is independent from the noise, and that the noise is independent and identically distributed. In cases where these assumptions do not hold, additional care may be taken to adjust the data to this model, for example, pre-whitening coil channels in parallel imaging or accounting for echo time variation in fast spin echo acquisitions.

Again following [83], we define SNR as the signal intensity divided by the standard deviation of the noise and note that from (4.2) we see that the SNR for measured data at the $k$-th location in $k$-space scales with $1/\sqrt{\tau_k}$:

$$SNR = \frac{signal}{\sqrt{noise\ var.}} = \frac{s_k}{\sqrt{\sigma^2_{acq}/\tau_k}} \tag{4.3}$$

As an example, if we double measurement time at each $k$-space location (e.g. acquire two samples rather than one), the modeled signal remains the same, the noise variance is reduced by a factor of 2, and the SNR increases by a factor of $\sqrt{2}$.

We model the measurement time at the $k$-th location in $k$-space, $\tau_k$, as the acquisition time per sample times the number of samples:

$$\tau_k = \tau_{acq} n_k \tag{4.4}$$

Without loss of generality, we assume a fixed acquisition time for every sample, $\tau_{acq}$, defined by the acquisition parameters and the number of samples, $n_k$, that may vary across $k$-space locations.

The measurement time $\tau_k$ is not necessarily equal for all $k$-space locations. Variable density sampling across $k$-space can be a natural effect of certain acquisition techniques, such as radial sampling. Variable density sampling may also be desired to take advantage of the higher energy in the low-frequency regions to improve SNR (similar to Weiner filtering) or

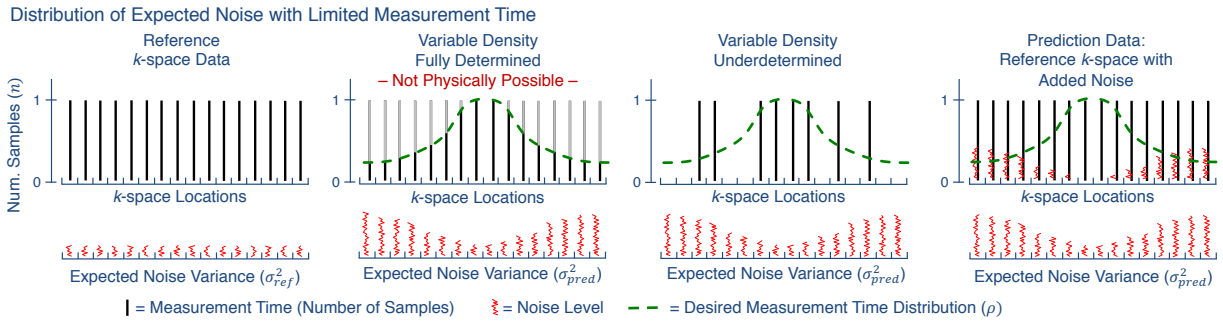


*Figure 4.3*: With limited measurement time, the sampling density distribution $\tau$ (dashed green line) may fall below one unit of measurement time. For systems with a minimum measurement time, fractional samples (second column) are not possible / do not contribute to a reduction in scan time, and we are forced to sample below the Nyquist rate (third column) to meet the required measurement time limit. To simulate the infeasible Nyquist-sampled, fully determined acquisition (second column), the image quality prediction process adds noise to a fully determined reference acquisition (fourth column). Note that all three of these datasets have the same distribution of expected noise variance across $k$-space (bottom row).

to account for asymptotic incoherence [109]. A variable density distribution of measurement time generates a corresponding distribution of expected noise variance across $k$-space. Lower sampling density at the high frequency $k$-space locations results in higher variance at these locations, generating a colored (blue) noise distribution, rather than the white noise associated with a uniform acquisition time distribution. It is this colored noise that is coupled with the underdetermined system effects during image reconstruction.

Note that in this work, we are not attempting to determine the optimal sampling density, but rather provide a tool to help analyze the effects of the chosen sampling density as well as other acquisition and reconstruction parameters.

### 4.1.2 Undersampling and Expected Measurement Time

Fast and/or short acquisitions require a limit on the total measurement time. Unfortunately, some systems and applications have constraints on the minimum measurement time at a single $k$-space location. In this case, it is not feasible to sample $k$-space such that the reconstruction system is fully determined (Fig. 4.3, second column). Undersampling is required in order to meet the measurement time constraints without sacrificing other scan requirements, such as spatial resolution, that are defined by the desired measurement time distribution (Fig. 4.3, dashed green curve). Without loss of generality, we will define the system's minimum measurement time to be one sample of duration $\tau_{acq}$ and any shorter acquisition times, $\tau_k < \tau_{acq}$, are infeasible.

Undersampling (Fig. 4.3, third column) avoids acquiring fractional samples by measuring either one or zero samples at each $k$-space location. A binary undersampling pattern can be

constructed to fit the desired sampling density, whether it be uniform or variable density. This technique of constructing a continuous output with discrete inputs is analogous to pulse-width modulation in digital signal generation and to digital halftoning in computer graphics.

At first, it may appear that the SNR using these binary undersampling patterns is the same as a fully-sampled acquisition because at the $k$-space locations where we collect a measurement, it has the same variance, $\sigma^2_{acq}$, as any fully-sampled measurement. Also, at locations where we don't measure any signal, we also don't collect any noise. However, the Fourier transform effectively averages the measured $k$-space locations with the zeros from the missing measurements, scaling the SNR by the square root of the sampling density (4.3). To model this averaging effect based on the density of binary sampling patterns, we consider the expected measurement time at each $k$-space location, $\tau_k$.

We model the expected measurement time for random undersampling patterns by considering the generation of a random sampling pattern. The binary value for each location in the pattern may be determined by drawing a random sample from a Bernoulli distribution. To generate a pattern with a particular sampling density, the mean parameter of each Bernoulli distribution is set to the desired fractional measurement time, $\rho_k$, for that location. Specifically, let us model $T_k$ as a Bernoulli random variable representing the measurement time at a single location in $k$-space. The expected value of $T_k$ is $\tau_{pred,k}$:

$$Sample_k \sim Bern(\rho_k) \tag{4.5}$$

$$T_k = \tau_{acq} n_k Sample_k \tag{4.6}$$

$$\tau_{pred,k} = \mathbb{E}[T_k] = \tau_{acq} n_k \rho_k \tag{4.7}$$

$\tau_{pred,k}$ gives us the expected measurement time per $k$-space location, which in turn leads us to the expected noise variance per $k$-space location, $\sigma^2_{pred,k} = \sigma^2_{acq}/\tau_{pred,k}$.

### 4.1.3 Image Quality Prediction

Using the expected measurement time described in the previous section, the image quality prediction process generates an image that shows the empirical effects of reduced measurement time without any effects of an underdetermined system caused by undersampling. This process, as depicted in Fig. 4.1, creates the prediction image by adding noise (based on the expected measurement time of a specific undersampling pattern) to a fully-sampled reference $k$-space dataset and then passing that adjusted $k$-space through the regularized weighted least squares reconstruction algorithm described in the following section.

The first step in the prediction process is to determine the expected measurement time at each $k$-space location, $\tau_{pred,k}$, for the given undersampling pattern. For random sampling patterns, this sampling density distribution is readily available, as it is the same distribution that generated the sampling pattern. When the sampling density is not explicitly or analytically available, the measurement time distribution may be approximated from the sampling

pattern with local averaging, Voronoi diagrams, or other techniques used in sampling density compensation.

From the measurement time distribution, we calculate how much noise needs to be added to the fully sampled (fully determined) reference $k$-space dataset to match the equivalent statistical noise produced by the given undersampling pattern. To simulate an undersampled acquisition with Gaussian noise variance $\sigma^2_{pred,k} = \sigma^2_{acq}/\tau_{pred,k}$, we simply add complex-valued Gaussian noise to the reference $k$-space based on the expected measurement time distribution, $\tau_{pred,k}$ (4.7), (Fig. 4.3, right). Given that $\sigma^2_{ref} = \sigma^2_{acq}/\tau_{ref}$ is the Gaussian noise variance measured from the reference data, we can calculate the variance of the complex Gaussian noise, $\sigma^2_{add,k}$, to add to location $k$ in the reference $k$-space:

$$\sigma^2_{pred,k} = \sigma^2_{ref} + \sigma^2_{add,k} \tag{4.8}$$

$$\sigma^2_{add,k} = \sigma^2_{pred,k} - \sigma^2_{ref} \tag{4.9}$$

$$= \left(\frac{\tau_{ref}}{\tau_{pred,k}} - 1\right)\sigma^2_{ref} \tag{4.10}$$

where $\tau_{ref} = \tau_{acq} n_{ref}$ (4.4) and $n_{ref}$ is the number of samples acquired in the reference data. The detailed derivation between (4.9) and (4.10) may be found in Appendix B.1. Often $n_{ref} = 1$, however, the reference data may be acquired using many samples, for example the number of averages might equal two or, in the case of our first two experiments, $n_{ref} = 144$ (Fig. 4.4).

Note that with variable density sampling patterns, $\tau_{pred,k}$, is not constant across $k$-space, and thus, the variance of the added noise, $\sigma^2_{add,k}$, will also vary across $k$-space.

The noise to add at each point in $k$-space is drawn from a complex-valued, zero-mean Gaussian distribution with variance equal to the $\sigma^2_{add,k}$ for that $k$-space location. This noise is simply added to the reference $k$-space to produce fully determined $k$-space with the noise distribution matching that of the undersampled data (see Fig. 4.3, right).

The final step in the image quality prediction process is to pass the reference $k$-space with added noise through the regularized weighted least squares reconstruction algorithm described in the following section, producing a prediction image that gives an estimate of the reconstruction image quality assuming no effect from an underdetermined reconstruction system.

### 4.1.4 Weighted Least Squares Reconstruction

We require a consistent reconstruction formulation that supports standard fully-sampled and undersampled data as well as the prediction data. To this end, we use a maximum *a posteriori* (MAP) formulation of MRI reconstruction that leads, in general, to a regularized weighted least squares optimization. Equations (4.1) and (4.2) combine to give us a Gaussian

likelihood probability of measuring a signal $y_k$ given an image object $\boldsymbol{m}$:

$$P(y_k|\boldsymbol{m}) = \frac{1}{\sqrt{2\pi\sigma^2_{acq}/\tau_k}} exp\left(\frac{-|y_k - (F\boldsymbol{m})_k|^2}{2\sigma^2_{acq}/\tau_k}\right) \tag{4.11}$$

With this Gaussian likelihood and assuming a general prior probability on our image data $P(\boldsymbol{m})$, the resulting MAP formulation leads to a weighted-least squares optimization:

$$\boldsymbol{m}^* = \underset{\boldsymbol{m}}{\text{argmax}}\, P(\boldsymbol{m}|\boldsymbol{y}) \tag{4.12}$$

$$= \underset{\boldsymbol{m}}{\text{argmax}}\, P(\boldsymbol{y}|\boldsymbol{m})P(\boldsymbol{m}) \tag{4.13}$$

$$= \underset{\boldsymbol{m}}{\text{argmin}}\, \frac{1}{2}\sum_{k=1}^{N_P} \tau_k|y_k - (F\boldsymbol{m})_k|^2 - \log P(\boldsymbol{m}) \tag{4.14}$$

$$= \underset{\boldsymbol{m}}{\text{argmin}}\, \frac{1}{2}||W\boldsymbol{y} - WF\boldsymbol{m}||_2^2 - \log P(\boldsymbol{m}) \tag{4.15}$$

where $\boldsymbol{m}$ is the vectorized image with $N_P$ number of pixels; $\boldsymbol{y}$ is the vectorized acquired $k$-space locations with $N_P$ number of elements; $F$ is the $N_P$x$N_P$ multidimensional discrete Fourier transform operator; and $W$ is an $N_P$x$N_P$ diagonal matrix with $W_{k,k} = \sqrt{\tau_k}$ values along the diagonal. The detailed derivation between (4.13) and (4.14) may be found in the Appendix B.2.

In this work, we will use a Laplacian-based prior to promote sparsity, $(-\log P(\boldsymbol{m}) = \lambda||\Psi(\boldsymbol{m})||_1)$, where $\Psi$ is a sparsity transform function and $\lambda$ is the Laplace prior parameter. This $\ell_1$ regularized weighted least squares (WLS) optimization does not have an analytic solution, and finding the solution requires a non-linear reconstruction algorithm. In general, we can solve this optimization using an iterative algorithm, such as FISTA [110] or ADMM [111].

This optimization framework, given the proper weight values described below, generalizes the reconstruction of a) fully sampled, b) undersampled, and c) image quality prediction datasets.

a) **Fully sampled weights:** The least squares weights for a fully sampled dataset (both uniform and variable density sampling) are simply equal to the square root of the measurement time, $W_{k,k} = \sqrt{\tau_k} = \sqrt{\tau_{acq} n_k}$, for the $k$-th sample. Assuming, again, that the acquisition time per sample is constant across $k$-space, $\tau_{acq}$ may be pulled out of the $\ell_2$ norm term, simplifying the weights to be equal to the square root of the number of samples, $W_{k,k} = \sqrt{n_k}$.

   Note that with $n_k$ constant across $k$-space and a uniform prior $P(\boldsymbol{m})$, the MAP optimization becomes the standard least squares optimization:

$$\boldsymbol{m}^* = \underset{\boldsymbol{m}}{\text{argmin}}\, \frac{1}{2}||\boldsymbol{y} - F\boldsymbol{m}||_2^2 \tag{4.16}$$

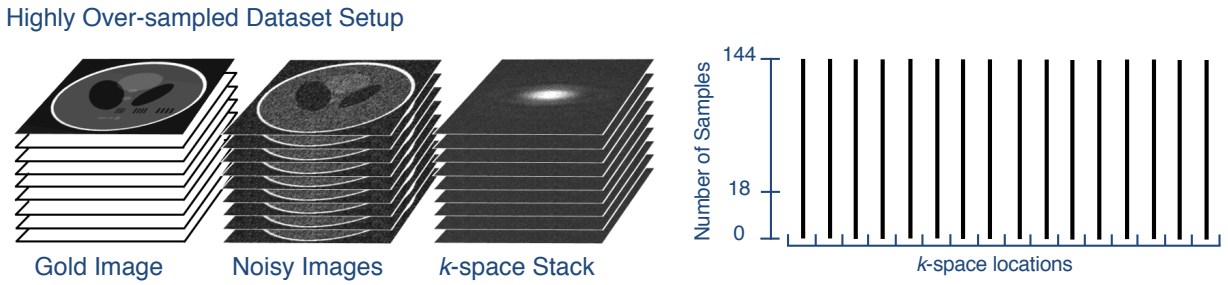


*Figure 4.4*: Experimental setup allowing us to choose the number of acquisition samples (from 0 to 144) at each $k$-space location. Noise is added to 144 copies of the input gold image. These noisy images are then Fourier transformed to create a stack of $k$-space images with 144 samples available at each $k$-space location. Note that for the tomato dataset, there is no gold image and the $k$-space stack comes directly from the 144 scanner acquisitions of the tomato.

b) **Undersampled weights:** When undersampling, the weights, $W_{k,k}$, are simply set to one or zero depending on whether or not that $k$-space location has been sampled (assuming the same measurement time at each sampled location). With these binary weights, the operator $W$ in (4.15) becomes the undersampling operator defined by the binary sampling pattern. With a Laplacian-based prior, the MAP reconstruction becomes the standard Lasso optimization [64] commonly used in compressed sensing. In addition to strictly binary undersampling patterns, the WLS optimization also allows for undersampling patterns that have zero measurement time at certain locations and a range of measurement times across the remaining locations, for example, an acquisition with undersampled high frequencies and over-sampled low frequencies.

c) **Prediction weights:** The prediction data is designed to simulate the noise variance from the expected measurement time for a given sampling density $\rho_k$, leading us to WLS weights $W_{k,k} = \sqrt{\tau_{pred,k}} = \sqrt{\tau_{acq} n_{ref,k} \rho_k}$, which may be simplified to $W_{k,k} = \sqrt{\rho_k}$ assuming constant sampling time and constant number of samples per location in the fully-sampled reference data.

## 4.2 Methodology

In this institutional review board-approved study, we acquired MRI data by scanning two healthy, adult volunteers.

### 4.2.1 Effect of Measurement Time Distribution

To better understand the effects of reduced measurement time and undersampling and to test our image quality prediction process, we created an experiment that enables us to

compare the reconstructions of 1) a fully determined dataset, 2) an underdetermined dataset, and 3) the corresponding prediction data, all using the same total measurement time and sampling density distribution.

The foundation of this experiment is a "stack" of 144 fully-sampled $k$-space images (Fig. 4.4). Each entry in the $k$-space stack is a different noisy acquisition of the same object slice. With 144 samples available at each of the $N$ $k$-space locations, we are able to select a subset of these samples to simulate acquiring a specific number of samples at each $k$-space location based on a desired measurement time distribution.

We used two different datasets for this experiment. The first dataset was the classic Shepp-Logan digital phantom [112] with a slight modification to add a set of parallel dark bars that will help analyze spatial resolution. This phantom was chosen because it has an explicitly sparse representation (many true zero values) in the finite differences domain (often seen in total variation (TV) reconstructions), implying that we can use compressed sensing to find a proper solution to the underdetermined system of equations caused by undersampling. As seen in Candes, Romberg, and Tao [113], the Shepp-Logan phantom, without noise, may be perfectly recovered after severe undersampling. To analyze how noise propagates through the reconstruction system, we generated a different instance of complex-valued, zero-mean, Gaussian noise to add to 144 copies of the $k$-space for the Shepp-Logan phantom. The second dataset is 144 actual MRI acquisitions of a tomato at a single slice location. This data was acquired on a 3T scanner (Siemens Healthineers, Erlangen, Germany) using a T1-weighted gradient echo sequence with 10 ms TE, 35 ms TR, 12 degree FA, 90 mm FOV, 2 mm slice thickness, and 192x192 acquisition matrix. Only the body coil was used during acquisition to both simplify the reconstruction model and ensure that each of the 144 acquisitions had relatively low SNR.

For both datasets, we selected a subset of the full stack of $k$-space samples based on three different sampling distributions, as depicted in the top row of Fig. 4.5: **reference**, using all $144N$ samples (where $N$ is the number of $k$-space locations); **fully determined**, selecting only $18N$ samples according to either a uniform or variable density sampling distribution across $k$-space locations; and **underdetermined**, selecting $18N$ samples and following the same density distribution but collecting all 144 samples at $N/8$ randomly chosen $k$-space locations and collecting zero samples for the remaining locations. We also reconstructed both datasets using the **image quality prediction** process to add noise to the $144N$ reference dataset to simulate the noise level from the $18N$ fully determined dataset.

For all reconstructions, the selected $k$-space samples were averaged at each $k$-space location to create a single $k$-space image to be reconstructed ($\boldsymbol{y}$, from equations (4.12) and (4.15)).

We reconstructed all data using our implementation of ADMM, formulated for the regularized weighted least squares optimization, with the weights equal to the number of measurements acquired at each $k$-space location, as specified in section 4.1.4. For the digital phantom dataset, we used isotropic total variation as the sparsity model. For the single channel MRI acquired data, we used Daubechies-4 wavelets with translation invariant cycle spinning [114] as the sparsity model.

### 4.2.2 Effects of Measurement Noise and Undersampling Rate

Given enough acquisition time, we can satisfy a given sampling density distribution by either Nyquist sampling $k$-space or by undersampling. Both of these sampling patterns produce similar distributions of expected noise variance in our data, but undersampling incurs an additional cost from having an underdetermined system of equations. In this experiment, we will extend the over-sampled stack experiment above to take a closer look at the effect of measurement noise and undersampling rate on reconstruction image quality. We accomplish this by varying both the measurement noise level and the undersampling rate and then comparing the mean squared error (MSE) images reconstructed from variable density **fully determined** data and from variable density **underdetermined** data.

As in the measurement time experiment above, we have a stack of $k$-space data, and we generate an output image by reconstructing a subset of $k$-space samples, selected according to an either a fully determined variable density sampling pattern or an underdetermined pattern following the same measurement time distribution.

In this experiment, the $k$-space stack is generated from copies of a single relatively high SNR (31.3 dB) *in vivo* head acquisition. Similar to the Shepp-Logon $k$-space stack, we added to $k$-space a sample of complex-valued, Gaussian noise with zero mean and a given standard deviation. We executed the experiment using three different values for the added noise standard deviation (1, 5, 8) and four undersampling rates (2x, 4x, 8x, and 12x undersampled).

The head dataset for this experiment is an axial slice of a 3D fully-sampled, spoiled gradient echo dataset acquired on a 1.5T scanner (GE Healthcare, Waukesha, WI) with 8 receive channels, 5 ms TE, 12 ms TR, 20 degrees FA, 184x230 mm FOV, 1 mm slice thickness, and 256x256 acquisition matrix. This multi-channel dataset was preprocessed, using ESPIRiT coil sensitivity maps [115], to combine the data into a single-channel, allowing us to use a simpler reconstruction model for this experiment. This head dataset has relatively high SNR, so we were able to experiment with very low noise and subsequently experiment with higher noise levels by adding Gaussian noise to the $k$-space stack for this dataset. This head dataset also provides a real example of an image that is only approximately sparse in the wavelet transform domain.

We repeated these 12 experiments (three noise levels by four undersampling rates) 100 times, each time reconstructing the fully determined data and the underdetermined data, as well as the corresponding prediction data. We then plotted the resulting MSE values (relative to the original head image) (Fig. 4.7).

### 4.2.3 Image Quality Prediction

We demonstrate the image quality prediction process by comparing the output of the actual undersampled reconstruction to both the generated prediction image and the fully determined reference image. We executed this experiment for two *in vivo* fully-sampled MRI datasets using increasingly aggressive retrospective undersampling rates.

### *in vivo* Knee

The *in vivo* knee dataset is an axial slice of a 3D fully-sampled, fast spin echo dataset acquired on a 3T scanner (GE Healthcare, Waukesha, WI) with 8 receive channels, 25 ms TE, 1550 ms TR, echo train length of 40, 160 mm FOV, 0.6 mm slice thickness, and 320x320 acquisition matrix. This dataset was collected by Epperson et al [5] and is available at [116].

The two retrospective undersampling patterns used were 4x and 12x undersampled, variable density Poisson disc. Both patterns fully-sampled the center of $k$-space to allow for ESPIRiT auto-calibration [115]. Neither the reference data nor the undersampling patterns included the corners of $k$-space, a common acquisition acceleration.

The optimization equation for this parallel imaging, compressed sensing reconstruction is an extension of equation (4.15), modified to include parallel imaging and a Laplacian prior:

$$\min_{\boldsymbol{m}} \frac{1}{2}\|W\boldsymbol{y} - WFS\boldsymbol{m}\|_2^2 + \lambda\|\Psi\boldsymbol{m}\|_1 \tag{4.17}$$

where $\boldsymbol{m}$ is the vectorized image with $N_P$ number of pixels; $\boldsymbol{y}$ is the vectorized acquired multi-channel $k$-space data with $N_C N_P$ number of elements ($N_P$ is the number of pixels, $N_C$ is the number of coils); $F$ is the $N_C N_P$x$N_C N_P$ 2D Fourier transform operator for each coil independently; $S$ is the $N_C N_P$x$N_P$ block diagonal sensitivity maps generated with ESPIRiT calibration; $\Psi$ is the sparsity transform; $\lambda$ is the regularization parameter; and $W$ is the $N_C N_P$x$N_C N_P$ diagonal weight matrix.

Note that the reconstruction process now includes the parallel imaging coil combination operator $S^H$. With the addition of parallel imaging, the undersampled reconstruction system is now both ill-conditioned and underdetermined. Previous works have provided tools to empirically analyze the noise propagation through the ill-conditioned parallel imaging system, for example by computing the geometry-factor [117] or with Monte Carlo simulations with added noise [118]. The image quality prediction process will empirically show the effect of lower SNR due to reduced measurement time on the compressed sensing and parallel imaging reconstruction without any effect from an underdetermined or ill-conditioned system. The actual underdetermined reconstruction will then produce an image affected by similar lower SNR as well as the effects from the ill-conditioned and underdetermined parallel imaging and compressed sensing system.

The sparsity filter (associated with $\Psi$) used within the reconstruction was wavelet soft-thresholding using Daubechies-4 with translation invariant cycle spinning [114].

The regularized weighted least squares optimization for both prediction and underdetermined reconstruction used our implementation of the ADMM algorithm. The only difference between the two reconstructions was the appropriate change to the weights as specified in section 4.1.4. Specifically, the weights for the prediction reconstruction were the square root of the sampling density, $\rho_k$, at each $k$-space location and the actual underdetermined reconstruction weights were binary with ones for acquired locations and zeros elsewhere.

The image quality prediction process requires an understanding of the existing noise level in the fully-sampled reference data ($\sigma^2_{full}$). Ideally, this noise level could be obtained from

an explicit measurement of the received signal using the coils on the same scanner, prior to the actual exam. In our experiments, we measured the noise level from the reference data directly by Fourier transforming the (multi-channel) $k$-space data and measuring the variance of the values from a 11x11 background patch in each coil image. The noise level was measured and applied independently for each coil channel.

A direct inverse 2D Fourier transform followed by coil combination ($\boldsymbol{m}_{ref} = S^H F^{-1} \boldsymbol{y}_{ref}$) was used on the reference $k$-space to generate the fully-sampled reference image for comparison (Fig. 4.8 top).

***in vivo* Head**

The *in vivo* head dataset is an axial 2D fast spin echo dataset acquired on a 3T scanner (Siemens Healthineers, Erlangen, Germany) with 12 receive channels, 91 ms TE, 6000 ms TR, echo train length of 11, 195x220 mm FOV, and 286x320 acquisition matrix. The 12 coil channels were reduced to 4 channels with Siemens coil compression. Multiple slices were acquired at slice thicknesses of 1 mm and 2 mm. The phase encodes lines were retrospectively undersampled at 4x and 6x acceleration using a 1D variable density Poisson disc sampling with the center 24 lines fully sampled. This dataset was processed in the same manner as the *in vivo* knee dataset above.

### 4.2.4 Regularization Parameter

The regularization parameter, $\lambda$, in the regularized weighted least-squares (WLS) optimization balance the least-squares data consistency term with the $\ell_1$ sparsity term. Due to the WLS optimization formulation, the expected value of the least squares term is the same for both the prediction process and the actual underdetermined reconstruction. This indicates that we should use the same sparsity regularization parameter for both prediction and underdetermined reconstructions.

In order to confirm that the same regularization parameter value works for both prediction and underdetermined reconstructions, we repeated the *in vivo* head experiment above (with 9x undersampling) using a variety of $\lambda$ values. Specifically, we used $\lambda$ values of 0.002, 0.02, and 0.2 for the prediction process and the underdetermined reconstruction. This experiment is also designed to show that the image quality prediction process functions properly for different choices of the regularization parameter value.

## 4.3 Results

The following three results are shared across all of our experiments:

1. the **prediction** image has equivalent or worse image quality than the **reference** image,
2. the **undersampled** reconstruction image has equivalent or worse image quality than the **prediction** image,

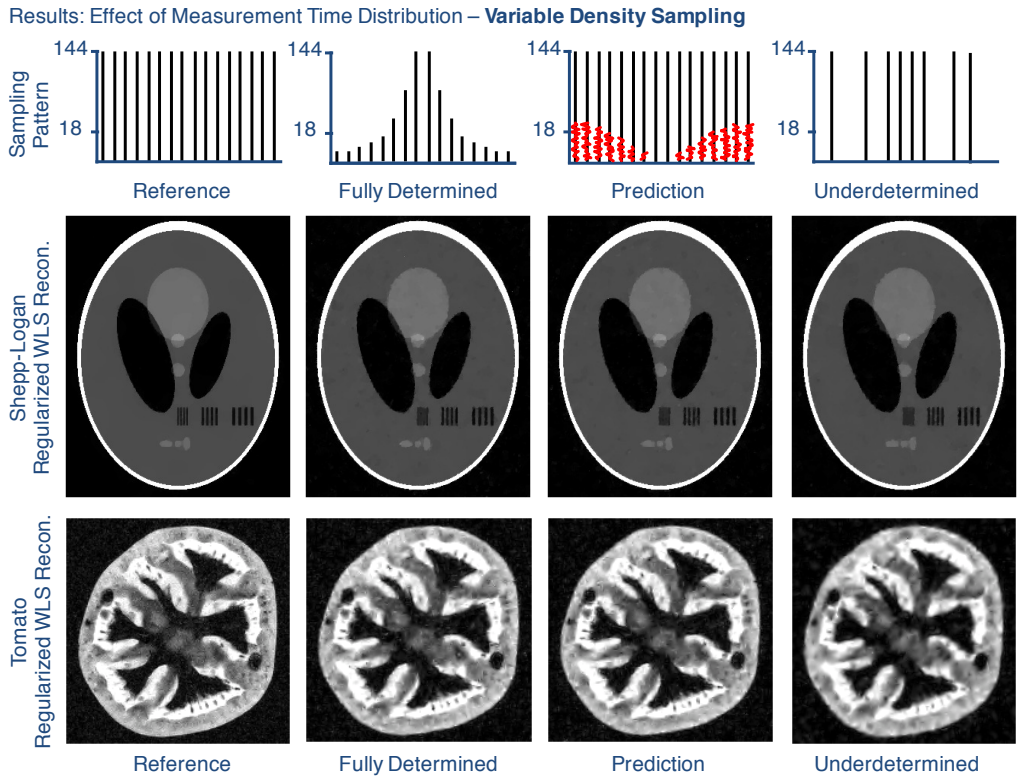


*Figure 4.5*: Results from the effect of the measurement time distribution experiment using variable density sampling. Each column uses a different set of $k$-space samples; from left to right: **over-sampled reference**, using all 144 samples at each $k$-space location; **variable density, fully determined**, using 1/8 of the total samples following a variable density distribution; **prediction data**, using fully-sampled $k$-space with noise added to simulate the variable density, fully determined dataset; **variable density (randomly sampled), underdetermined**, using 1/8 of the total samples and following the same variable density distribution, but only using either 144 or zero samples at each location. *Row 1*: Illustration of how measurement time is distributed across $k$-space. *Row 2*: WLS reconstruction of the Shepp-Logan data, regularized with total variation. *Row 3*: WLS reconstruction of the tomato $k$-space data, regularized with wavelets.

3. the **prediction** image for a given sampling density has equivalent image quality to the **fully determined** image with the same sampling density.

### 4.3.1 Effect of Measurement Time Distribution

Fig. 4.5 shows the results of our experiment to test the effect of various measurement time distributions on reconstruction image quality. For both the Shepp Logan digital phantom and the MRI acquisition of the tomato, the fully determined images with reduced measurement time show lower image quality than the images reconstructed from the reference acquisition

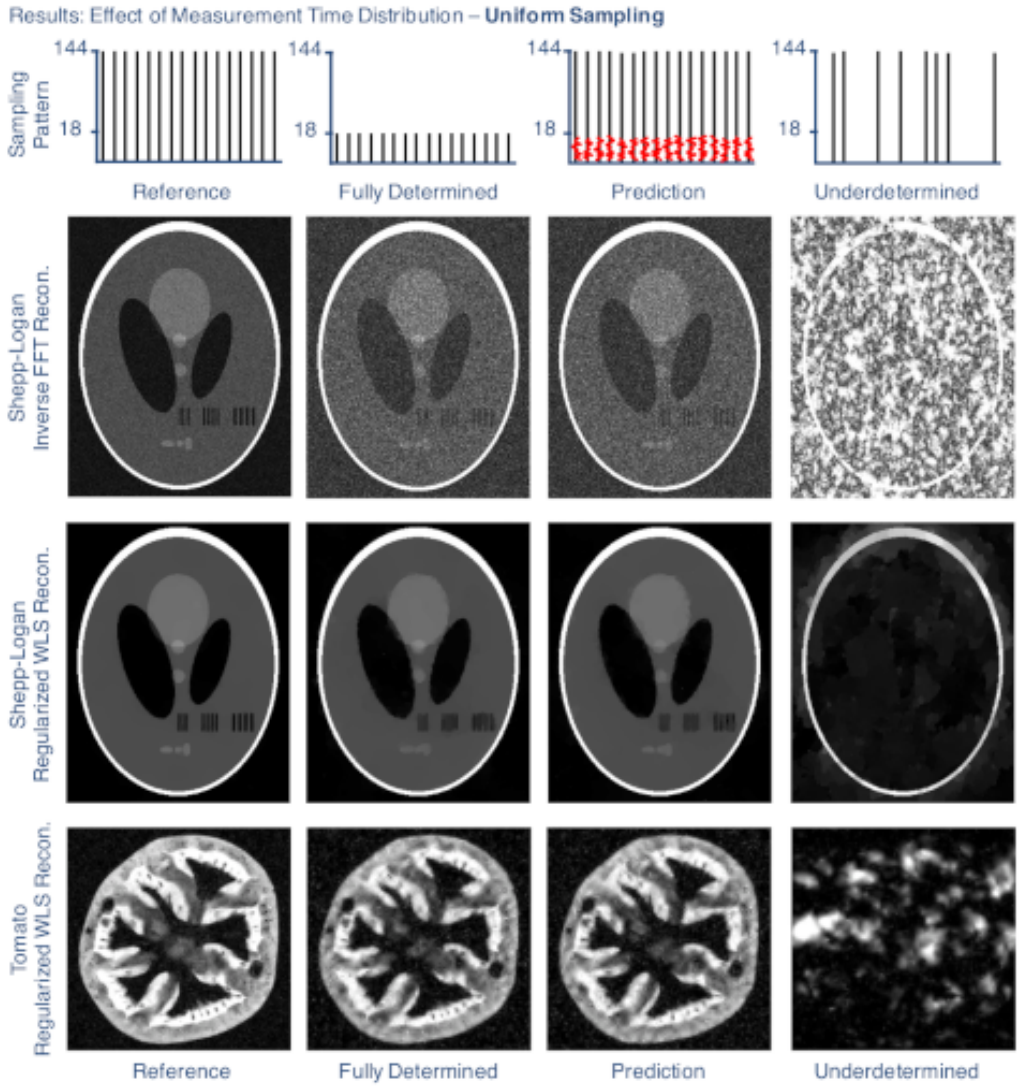


*Figure 4.6*: Results from the effect of the measurement time distribution experiment using uniform density sampling. Each column uses a different set of $k$-space samples; from left to right: **oversampled reference**, using all 144 samples at each $k$-space location; **uniform density, fully determined**, using 1/8 of the total samples following a uniform density distribution; **prediction data**, using fully-sampled $k$-space with noise added to simulate the uniform density, fully determined dataset; **uniform density (randomly sampled), underdetermined**, using 1/8 of the total samples and following the same uniform density distribution, but only using either 144 or zero samples at each location. *Row 1*: Illustration of how measurement time is distributed across $k$-space. *Row 2*: Direct inverse Fourier transform reconstruction of the Shepp-Logan data. *Row 3*: WLS reconstruction of the Shepp-Logan data, regularized with total variation. *Row 4*: WLS reconstruction of the tomato $k$-space data, regularized with wavelets.

data. As seen specifically in the blurred spatial vertical bars, the fully determined images did not fully recover from the limited acquisition time despite not having any corruption from an underdetermined systems of equations.

The variable density underdetermined Shepp Logan reconstruction (Fig. 4.5, third row, right) was successful and has nearly identically image quality to the fully determined reconstruction but still lower image quality than the reference reconstruction. This indicates that the underdetermined reconstruction recovered well from the underdetermined system, but still could not completely recover from the lower SNR due to the reduced measurement time. For the acquired tomato dataset, however, the underdetermined image quality (Fig. 4.5, bottom, right) is lower than the prediction and fully determined image quality, indicating that the reconstruction could not completely recover from the underdetermined system. This is not a surprising result because the tomato image is not sufficiently sparse in the wavelet transform domain, especially when compared to the explicit sparsity of the Shepp Logan phantom in the finite differences domain.

Fig. 4.5 also shows the results of the image quality prediction process for the same two datasets and sampling distributions. The second and third columns in this figure show that the fully determined reconstructions have essentially identical image quality to their corresponding prediction images. This verifies that the image quality prediction process closely simulates the noise level and reconstructed image quality of the associated fully determined acquisitions.

Similar results from the same experiment with uniform density sampling are shown in Figure 4.6.

### 4.3.2 Effects of Measurement Noise and Undersampling Rate

By varying the input noise level and the undersampling rate, we see the differences in the resulting MSE for the reconstructions of the fully determined data and underdetermined data, both with the same measurement time distribution. Fig. 4.7 shows that for a fixed noise level and increasing undersampling rate, the MSE of the fully determined images increases, showing that the reduced measurement time affects image quality despite no undersampling. Also, as we increase the undersampling rate, the MSE of the underdetermined images increases significantly faster than the fully determined images. This gap in image quality shows the degrading effect of the underdetermined reconstruction increasing as the undersampling rate increases and the sparsity transform can no longer adequately model the image in a sufficiently sparse representation.

As seen in Fig. 4.7, the MSE of the prediction images matches the MSE of the fully determined reconstructions for all noise levels and undersampling rates, indicating that the image quality prediction process is consistently simulating the expected noise level for the given sampling density.

The results from this experiment help us to see that when the image quality of the prediction image is unacceptable, the actual undersampled reconstruction will also be unacceptable (i.e. higher MSE). In this situation, the low SNR of the acquisition is the limiting factor

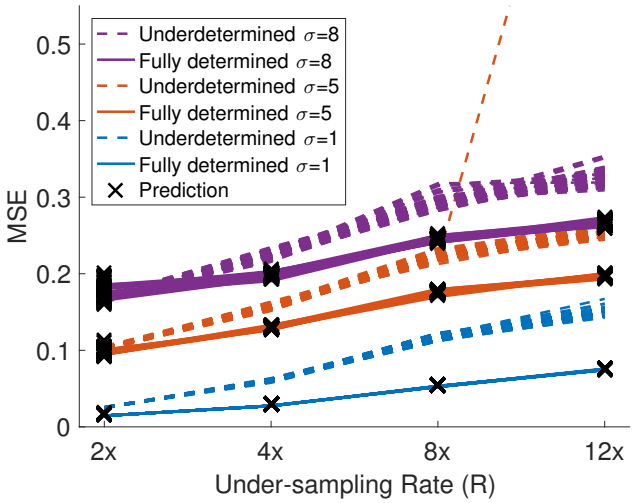


*Figure 4.7*: Results of our experiment to compare fully determined and underdetermined reconstructions with the same total measurement time across four different undersampling rates (2x, 4x, 8x, 12x) for three different noise levels (added noise standard deviations 1.0, 5.0, 8.0). Mean squared error (MSE) values are plotted for each of the 100 repetitions of the same experiment. Note that for the 2x undersampling rate, the fully determined and underdetermined reconstructions have essentially the same MSE. As the undersampling rate increases, the underdetermined system produces an increasingly worse MSE than the fully determined system. Note that one of the 100 underdetermined reconstructions at $\sigma = 5$ and $R = 12x$ failed to converge. This outlier is consistent with compressed sensing theory and practice where the reconstruction may fail to converge at higher undersampling rates.

in the reconstruction, not the artifacts due to the underdetermined system. To improve the reconstruction in this case, steps should be taken to adjust the acquisition parameters to increase the SNR or better handle the expected noise levels (e.g. reducing spatial resolution, decreasing undersampling rate, or improving the image prior $P(\boldsymbol{m})$).

### 4.3.3 Image Quality Prediction

Fig. 4.2 shows the prediction and underdetermined reconstruction images for the *in vivo* head experiment. This figure illustrates how the prediction image may be used to gain insight into the causes of poor undersampled image quality and adjust scan parameters, such as slice thickness, as needed.

Fig. 4.8 shows the reference, prediction, and underdetermined reconstruction images for the *in vivo* experiment using the knee dataset and various undersampling rates. Fig. 4.8 shows the following three qualitative results: 1) the reference image has better image quality than the prediction images; 2) the prediction images have better image quality than the corresponding underdetermined images; and 3) the underdetermined images are more

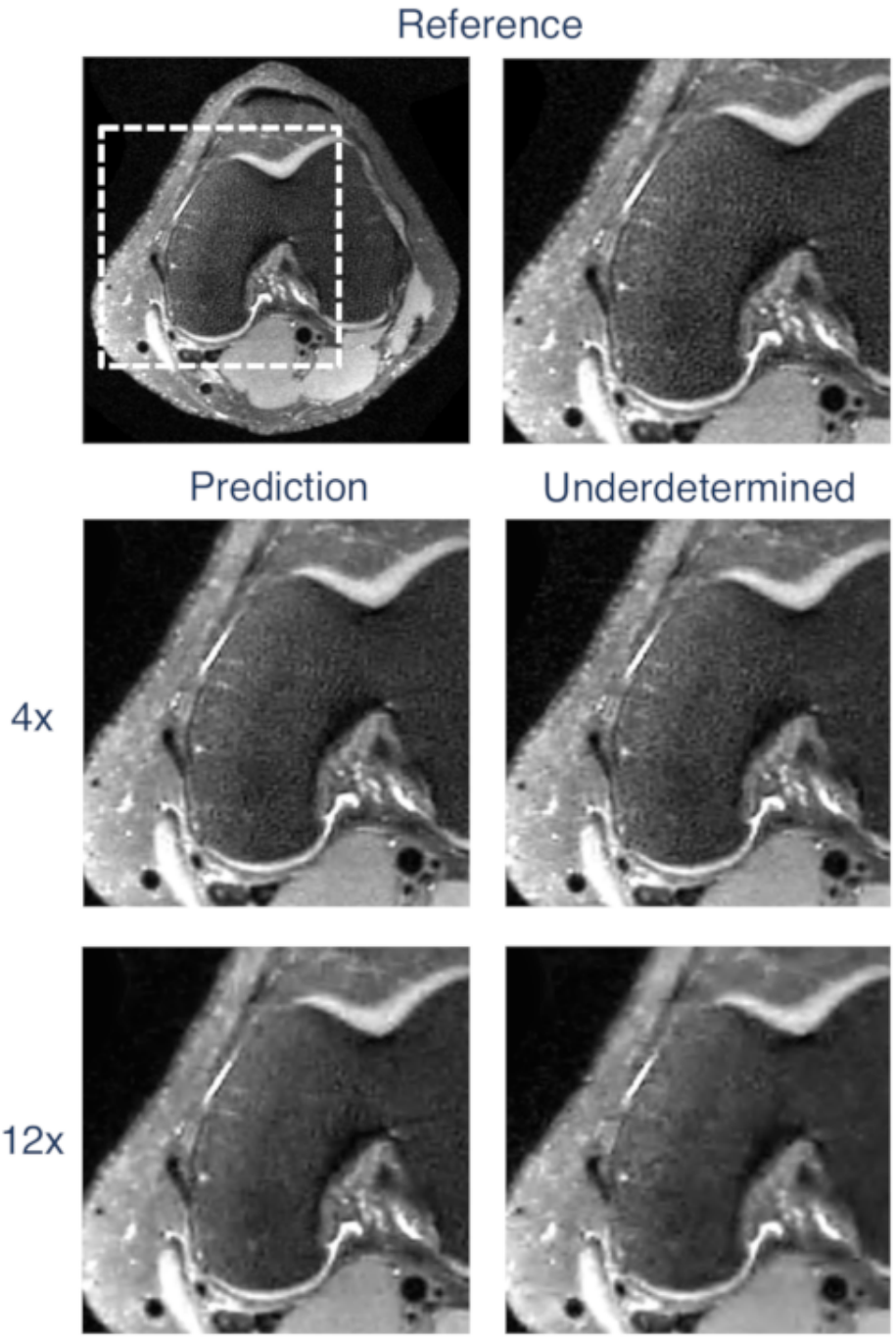


*Figure 4.8*: Results from *in vivo* image quality prediction experiment. Fully determined reference axial knee (top) followed by prediction (left column) and actual underdetermined reconstruction (right column) for two different undersampling rates: 4x and 12x. Images are zoomed and cropped to show image quality detail.

similar in image quality to the prediction images than the reference image. That the reference images look better than the prediction images is expected because the prediction process adds more noise to the fully sampled reference data. That the prediction images look better than the underdetermined image is expected because the underdetermined reconstruction had to find a proper solution to an underdetermined system of equations in addition to recovering from the lower SNR from reduced measurement time. Finally, the prediction image provides a better estimate of reconstruction image quality than the reference image.

With a reasonable amount of undersampling, the 4x underdetermined images only have slightly lower image quality than the prediction images. As undersampling increases to 12x, the image quality gap between the underdetermined and prediction images increases. These results are consistent with our effect of measurement noise and undersampling rate experiment when increasing the sampling rate (section 4.3.2).

### 4.3.4 Regularization Parameter

Fig. 4.9 shows the prediction and reconstruction results for the head dataset with 9x undersampling and wavelet regularization using three different regularization values ($\lambda$). *Low*, *medium*, and *high* $\lambda$ values of 0.002, 0.02, and 0.2, respectively, were used for the prediction process and actual reconstruction. The resulting prediction and reconstruction images match up well in terms of image quality. Specifically, the *low* $\lambda$ images are both fairly noisy; the *high* $\lambda$ images are both over-smoothed; and the *medium* $\lambda$ images both fall in-between, representing the best image quality of each column. These results illustrate that selecting the regularization parameter for the prediction process can give us a reasonable expectation for the image quality of the reconstruction process using the same regularization parameter.

## 4.4 Discussion

The presented **image quality prediction** process provides an empirical upper bound on undersampled image quality, which serves as a better metric for evaluating the effectiveness of a reconstruction algorithm than direct comparison to a fully-sampled reference reconstruction. The prediction process enables an analysis of a reconstruction algorithm's ability to handle lower SNR due to reduced measurement time without any effect from an underdetermined system. By simulating the effect of lower SNR without any underdetermined effects, the prediction process allows us to determine whether a reconstruction is actually limited by our sparse recovery or simply limited by low acquisition signal to noise ratio. Comparison of the prediction image to the reference reconstruction provides a means to assess the effects of lower SNR on reconstruction image quality. Comparison of the prediction image to the underdetermined reconstruction enables us to analyze what artifacts are introduced when undersampling is used rather than fully determined following the same measurement time distribution. The image quality prediction results and analysis are consistent with our exper-

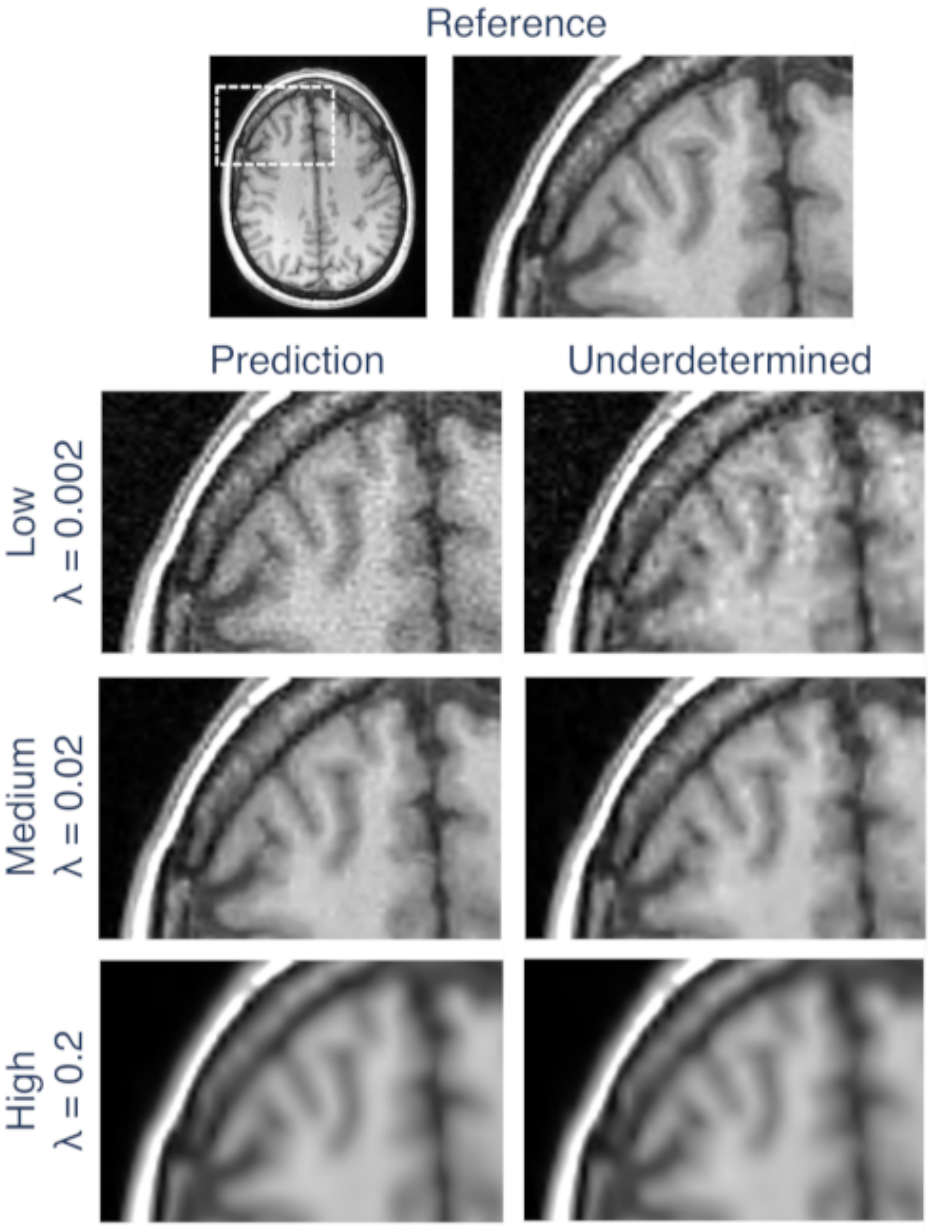


*Figure 4.9*: Results from sparsity regularization parameter experiment. Fully determined reference axial head (top) followed by prediction (left column) and actual underdetermined reconstruction (right column) for three regularization values ($\lambda$). The $\lambda$ values 0.002, 0.02, and 0.2 were used for the prediction process and the underdetermined reconstruction. Images are zoomed and cropped to show image quality detail.

iments using our highly over-sampled datasets to explicitly compare reconstruction results from variable density fully determined and underdetermined data.

An additional benefit of the prediction process is that it may be used to compare and tune different reconstruction algorithms or parameters, assessing how different reconstruction systems handle the lower SNR due to reduced measurement time in addition to comparing the actual undersampled reconstructions.

A limitation of the image quality prediction process is that it requires a fully-sampled reference dataset. Access to a fully-sampled acquisition is not always possible, especially in cases with 3D or 4D dynamic imaging, where long, fully-sampled acquisition times are not practical. Also, the image quality prediction process can isolate the effects of low SNR from the effects from an underdetermined system, but it cannot do the contrary, i.e. it cannot isolate the effects from an underdetermined system from the effects of low SNR. Future work could investigate the effects of an underdetermined systems using *in vivo* data by reconstructing fully-sampled reference datasets that are highly over-sampled to have minimal input noise, $\sigma_{ref}$. Of course, the effects of the underdetermined system would still be dependent on the image content, which varies significantly across clinical applications.

While developing the image quality prediction process, we use a maximum *a posteriori* formulation to derive a **general weighted least-squares optimization framework** that accounts for both uniform and variable density sampling patterns, with undersampling as a special case using binary weights. This weighted least squares formulation adjusts the standard least squares term to account for the colored noise arising from the distribution of expected measurement time across $k$-space locations. Future work could develop methods to similarly incorporate the effects of measurement time distribution into the sparsity regularization term, allowing the sparsity filters to better recover from colored noise in addition to incoherent aliasing.

# Bibliography


[1] Tohru Nitta. An extension of the back-propagation algorithm to complex numbers. *Neural Netw.*, 10(9):1391–1415, November 1997.

[2] Michael Markl, Philip J Kilner, and Tino Ebbers. Comprehensive 4d velocity mapping of the heart and great vessels by cardiovascular magnetic resonance. *Journal of Cardiovascular Magnetic Resonance*, 13(1):7, 2011.

[3] Timothy J O'shea and Nathan West. Radio machine learning dataset generation with gnu radio. In *Proceedings of the GNU Radio Conference*, volume 1, 2016.

[4] Rudrasis Chakraborty, Jiayun Wang, and Stella X. Yu. Sur-real: Frechet mean and distance transform for complex-valued deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[5] Kevin Epperson, Anne Marie Sawyer, Michael Lustig, Marcus Alley, Martin Uecker, Patrick Virtue, Peng Lai, and Shreyas Vasanawala. Creation of fully sampled MR data repository for compressed sensing of the knee. In *2013 Meeting Proceedings*. Section for Magnetic Resonance Technologists, April 2013.

[6] Dan Ma, Vikas Gulani, Nicole Seiberlich, Kecheng Liu, Jeffrey L Sunshine, Jeffrey L Duerk, and Mark A Griswold. Magnetic resonance fingerprinting. *Nature*, 495(7440):187–192, 2013.

[7] Bruce Fischl and Anders M Dale. Measuring the thickness of the human cerebral cortex from magnetic resonance images. *Proceedings of the National Academy of Sciences*, 97(20):11050–11055, 2000.

[8] Michael Markl, Alex Frydrychowicz, Sebastian Kozerke, Mike Hope, and Oliver Wieben. 4d flow mri. *Journal of Magnetic Resonance Imaging*, 36(5):1015–1036, 2012.

[9] Daniel R Messroghli, Aleksandra Radjenovic, Sebastian Kozerke, David M Higgins, Mohan U Sivananthan, and John P Ridgway. Modified look-locker inversion recovery (molli) for high-resolution t1 mapping of the heart. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 52(1):141–146, 2004.

[10] Alois M Sprinkart, Julian A Luetkens, Frank Träber, Jonas Doerner, Jürgen Gieseke, Bernhard Schnackenburg, Georg Schmitz, Daniel Thomas, Rami Homsi, Wolfgang Block, et al. Gradient spin echo (grase) imaging for fast myocardial t2 mapping. *Journal of Cardiovascular Magnetic Resonance*, 17(1):12, 2015.

[11] Michael Lustig, David Donoho, and John M. Pauly. Sparse MRI: The application

of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[12] Morteza Mardani, Enhao Gong, Joseph Y Cheng, Shreyas Vasanawala, Greg Zaharchuk, Marcus Alley, Neil Thakur, Song Han, William Dally, John M Pauly, et al. Deep generative adversarial networks for compressed sensing automates mri. *arXiv preprint arXiv:1706.00051*, 2017.

[13] Ouri Cohen, Bo Zhu, and Matthew S Rosen. Mr fingerprinting deep reconstruction network (drone). *Magnetic Resonance in Medicine*, 80(3):885–894, 2018.

[14] Elisabeth Hoppe, Gregor Körzdörfer, Tobias Würfl, Jens Wetzl, Felix Lugauer, Josef Pfeuffer, and Andreas K Maier. Deep learning for magnetic resonance fingerprinting: A new approach for predicting quantitative parameter values from time series. In *GMDS*, pages 202–206, 2017.

[15] Fang Liu, Li Feng, and Richard Kijowski. Mantis: Model-augmented neural network with incoherent k-space sampling for efficient mr parameter mapping. *Magnetic resonance in medicine*, 82(1):174–188, 2019.

[16] Tristan Needham. *Visual complex analysis*. Oxford University Press, 1998.

[17] G. M. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):330–334, May 1992.

[18] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

[19] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[20] Joan Bruna, Soumith Chintala, Yann LeCun, Serkan Piantino, Arthur Szlam, and Mark Tygert. A theoretical argument for complex-valued convolutional networks. *arXiv preprint arXiv:1503.03438*, 139, 2015.

[21] Titouan Parcollet, Mirco Ravanelli, Mohamed Morchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. Quaternion recurrent neural networks. *arXiv preprint arXiv:1806.04418*, 2018.

[22] Yun Jiang, Dan Ma, Nicole Seiberlich, Vikas Gulani, and Mark A. Griswold. Mr fingerprinting using fast imaging with steady state precession (fisp) with spiral readout. *Magnetic Resonance in Medicine*, 74(6):1621–1631, 2015.

[23] K Sommer, T Amthor, M Doneva, P Koken, J Meineke, and P Börnert. Towards predicting the encoding capability of mr fingerprinting sequences. *Magnetic resonance imaging*, 41:7–14, 2017.

[24] Mariya Doneva, Thomas Amthor, Peter Koken, Karsten Sommer, and Peter Börnert. Matrix completion-based reconstruction for undersampled magnetic resonance fingerprinting data. *Magnetic Resonance Imaging*, 41:41–52, 2017.

[25] Mike Davies, Gilles Puy, Pierre Vandergheynst, and Yves Wiaux. A compressed sensing framework for magnetic resonance fingerprinting. *SIAM Journal on Imaging Sciences*, 7(4):2623–2656, 2014.

[26] Debra F McGivney, Eric Pierre, Dan Ma, Yun Jiang, Haris Saybasili, Vikas Gulani,

and Mark A Griswold. SVD compression for magnetic resonance fingerprinting in the time domain. *IEEE Transactions on Medical Imaging*, 33(12):2311–2322, 2014.

[27] Stuart A Bobman, Stephen J Riederer, James N Lee, Steven A Suddarth, Henry Z Wang, Burton P Drayer, and James R MacFall. Cerebral magnetic resonance image synthesis. *American Journal of Neuroradiology*, 6(2):265–269, 1985.

[28] Emmanuel J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9ÃŘ10):589 – 592, 2008.

[29] Emmanuel J Candès and Yaniv Plan. A probabilistic and RIPless theory of compressed sensing. *IEEE Transactions on Information Theory*, 57(11):7235–7254, 2011.

[30] M.J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using $\ell_1$ -constrained quadratic programming (lasso). *Information Theory, IEEE Transactions on*, 55(5):2183–2202, May 2009.

[31] Patrick Virtue, Stella X. Yu, and Michael Lustig. Better than real: Complex-valued neural networks for MRI fingerprinting. In *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017.

[32] Patrick Virtue. Complex-valued Caffe Source Code, pvirtue/caffe: 2017 ICIP MR Fingerprinting. [Online]. Available from: https://github.com/pvirtue/caffe/tree/complex, May 2017. [Accessed 28th June 2018].

[33] Patrick Virtue, Jonathin I Tamir, Mariya Doneva, Stella X Yu, and Michael Lustig. Learning contrast synthesis from MR fingerprinting. In *2018 Scientific Meeting Proceedings*, page 676. International Society for Magnetic Resonance in Medicine, 2018.

[34] Patrick Virtue, Jonathin I Tamir, Mariya Doneva, Stella X Yu, and Michael Lustig. Direct contrast synthesis for magnetic resonance fingerprinting. In *Proceedings of the 2018 ISMRM Workshop on Machine Learning*, page 676. International Society for Magnetic Resonance in Medicine, 2018.

[35] Patrick Virtue and Michael Lustig. The empirical effect of gaussian noise in undersampled MRI reconstruction. *Tomography: a journal for imaging research*, 3(4):211, 2017.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[38] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[39] Timothy J O'Shea, Johnathan Corgan, and T Charles Clancy. Convolutional radio modulation recognition networks. In *International Conference on Engineering Applications of Neural Networks*, pages 213–226. Springer, 2016.

[40] Sizhe Chen, Haipeng Wang, Feng Xu, and Ya-Qiu Jin. Target classification using the deep convolutional networks for SAR images. *IEEE Transactions on Geoscience and*

*Remote Sensing*, 54(8):4806–4817, 2016.

[41] Jun Ding, Bo Chen, Hongwei Liu, and Mengyuan Huang. Convolutional neural network with data augmentation for SAR target recognition. *IEEE Geoscience and Remote Sensing Letters*, 13(3):364–368, 2016.

[42] Ouri Cohen, Bo Zhu, and Matthew Rosen. Deep learning for fast MR fingerprinting reconstruction. In *2017 Scientific Meeting Proceedings*, page 688. International Society for Magnetic Resonance in Medicine, 2017.

[43] Khalid Youssef, Louis Bouchard, Karen Haigh, Jan Silovsky, Bishal Thapa, and Chris Vander Valk. Machine learning approach to rf transmitter identification. *IEEE Journal of Radio Frequency Identification*, 2(4):197–205, 2018.

[44] Szu-Wei Fu, Ting-yao Hu, Yu Tsao, and Xugang Lu. Complex spectrogram enhancement by convolutional neural network with multi-metrics learning. In *Machine Learning for Signal Processing (MLSP), 2017 IEEE 27th International Workshop on*, pages 1–6. IEEE, 2017.

[45] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[46] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[47] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016.

[48] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[49] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[50] Rahul Singh, Abhishek Chakraborty, and BS Manoj. Graph fourier transform based on directed laplacian. In *Signal Processing and Communications (SPCOM), 2016 International Conference on*, pages 1–5. IEEE, 2016.

[51] Michael Maire, Takuya Narihira, and Stella X Yu. Affinity cnn: Learning pixel-centric pairwise relations for figure/ground embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 174–182, 2016.

[52] Md. Faijul Amin and Kazuyuki Murase. Single-layered complex-valued neural network for real-valued classification problems. *Neurocomput.*, 72(4-6):945–955, January 2009.

[53] Charles F Cadieu and Bruno A Olshausen. Learning intermediate-level representations of form and motion from natural movies. *Neural computation*, 24(4):827–866, 2012.

[54] W. Wirtinger. Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Mathematische Annalen*, 97(1):357–375, 1927.

[55] K. Kreutz-Delgado. The Complex Gradient Operator and the CR-Calculus. *ArXiv*

*e-prints*, June 2009.

[56] DH Brandwood. A complex gradient operator and its application in adaptive array theory. In *IEE Proceedings F-Communications, Radar and Signal Processing*, volume 130, pages 11–16. IET, 1983.

[57] H. Leung and S. Haykin. The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 39(9):2101–2104, Sep 1991.

[58] M. S. Kim and C. C. Guest. Modification of backpropagation networks for complex-valued signal processing in frequency domain. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 27–31 vol.3, June 1990.

[59] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015.

[60] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[61] ISO. *International Standard ISO/IEC 14882:2014 - Programming Language C++.* International Organization for Standardization, Geneva, Switzerland, 2014.

[62] ISO. *International Standard ISO/IEC 14882:2017 - Programming Language C++.* International Organization for Standardization, Geneva, Switzerland, 2017.

[63] David L Donoho, Iain M Johnstone, Jeffrey C Hoch, and Alan S Stern. Maximum entropy and the nearly black object. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 41–81, 1992.

[64] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):pp. 267–288, 1996.

[65] Berengere Aubert-Broche, Alan C Evans, and Louis Collins. A new improved version of the realistic digital brain phantom. *NeuroImage*, 32(1):138–145, 2006.

[66] Adrian P Crawley and R Mark Henkelman. A comparison of one-shot and recovery methods in t1 imaging. *Magnetic resonance in medicine*, 7(1):23–34, 1988.

[67] Dwight G Nishimura. *Principles of Magnetic Resonance Imaging*. Standford Univ., 2010.

[68] F. Bloch. Nuclear induction. *Phys. Rev.*, 70:460–474, Oct 1946.

[69] Kelvin Chow, Jacqueline A Flewitt, Jordin D Green, Joseph J Pagano, Matthias G Friedrich, and Richard B Thompson. Saturation recovery single-shot acquisition (sasha) for myocardial t1 mapping. *Magnetic resonance in medicine*, 71(6):2082–2095, 2014.

[70] Mariya Doneva, Peter Börnert, Holger Eggers, Christian Stehning, Julien Sénégas, and Alfred Mertins. Compressed sensing reconstruction for magnetic resonance parameter mapping. *Magnetic Resonance in Medicine*, 64(4):1114–1120, 2010.

[71] Sean CL Deoni, Brian K Rutt, and Terry M Peters. Rapid combined t1 and t2 mapping using gradient recalled acquisition in the steady state. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in*

*Medicine*, 49(3):515–526, 2003.

[72] J.B.M. Warntjes, O. Dahlqvist Leinhard, J. West, and P. Lundberg. Rapid magnetic resonance quantification on the brain: Optimization for clinical usage. *Magnetic Resonance in Medicine*, 60(2):320–329, 2008.

[73] Guido Buonincontri and Stephen J Sawiak. Mr fingerprinting with simultaneous b1 estimation. *Magnetic resonance in medicine*, 76(4):1127–1135, 2016.

[74] Gregor Körzdörfer, Yun Jiang, Peter Speier, Jianing Pang, Dan Ma, Josef Pfeuffer, Bernhard Hensel, Vikas Gulani, Mark Griswold, and Mathias Nittka. Magnetic resonance field fingerprinting. *Magnetic resonance in medicine*, 81(4):2347–2359, 2019.

[75] Eric Y Pierre, Dan Ma, Yong Chen, Chaitra Badve, and Mark A Griswold. Multiscale reconstruction for MR fingerprinting. *Magnetic Resonance in Medicine*, 2015.

[76] Jakob Assländer, Steffen J Glaser, and Jürgen Hennig. Pseudo steady-state free precession for MR-fingerprinting. *Magnetic Resonance in Medicine*, 2016.

[77] Stephen F Cauley, Kawin Setsompop, Dan Ma, Yun Jiang, Huihui Ye, Elfar Adalsteinsson, Mark A Griswold, and Lawrence L Wald. Fast group matching for MR fingerprinting reconstruction. *Magnetic Resonance in Medicine*, 74(2):523–528, 2015.

[78] Mohammad Golbabaee, Dongdong Chen, Pedro A Gómez, Marion I Menzel, and Mike E Davies. Geometry of deep learning for magnetic resonance fingerprinting. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7825–7829. IEEE, 2019.

[79] G. Nataraj, J. F. Nielsen, C. Scott, and J. A. Fessler. Dictionary-free mri perk: Parameter estimation via regression with kernels. *IEEE Transactions on Medical Imaging*, PP(99):1–1, 2018.

[80] D Louis Collins, Alex P Zijdenbos, Vasken Kollokian, John G Sled, Noor J Kabani, Colin J Holmes, and Alan C Evans. Design and construction of a realistic digital brain phantom. *IEEE Transactions on Medical Imaging*, 17(3):463–468, 1998.

[81] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[82] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[83] Albert Macovski. Noise in MRI. *Magnetic Resonance in Medicine*, 36(3):494–497, 1996.

[84] J. D. O'Sullivan. A fast sinc function gridding algorithm for fourier inversion in computer tomography. *IEEE Transactions on Medical Imaging*, 4(4):200–207, Dec 1985.

[85] J. I. Jackson, C. H. Meyer, D. G. Nishimura, and A. Macovski. Selection of a convolution function for fourier inversion using gridding [computerised tomography application]. *IEEE Transactions on Medical Imaging*, 10(3):473–478, Sep 1991.

[86] Peter B Roemer, William A Edelstein, Cecil E Hayes, Steven P Souza, and Otward M Mueller. The nmr phased array. *Magnetic resonance in medicine*, 16(2):192–225, 1990.

[87] J Hennig. Multiecho imaging sequences with low refocusing flip angles. *Journal of Magnetic Resonance (1969)*, 78(3):397–407, 1988.

[88] Matthias Weigel. Extended phase graphs: dephasing, rf pulses, and echoes-pure and simple. *Journal of Magnetic Resonance Imaging*, 41(2):266–295, 2015.

[89] Yun Jiang, Dan Ma, Kathryn E Keenan, Karl F Stupic, Vikas Gulani, and Mark A Griswold. Repeatability of magnetic resonance fingerprinting t1 and t2 estimates assessed using the ismrm/nist mri system phantom. *Magnetic Resonance in Medicine*, 78(4):1452–1457, 2017.

[90] L.N. Tanenbaum, A.J. Tsiouris, A.N. Johnson, T.P. Naidich, M.C. DeLano, E.R. Melhem, P. Quarterman, S.X. Parameswaran, A. Shankaranarayanan, M. Goyen, and A.S. Field. Synthetic mri for clinical neuroimaging: Results of the magnetic resonance image compilation (magic) prospective, multicenter, multireader trial. *American Journal of Neuroradiology*, 2017.

[91] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[92] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[93] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[94] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[95] Matt A Bernstein, Kevin F King, and Xiaohong Joe Zhou. *Handbook of MRI pulse sequences*. Elsevier, 2004.

[96] GM Bydder and IR Young. MR imaging: Clinical use of the inversion recovery sequence. *Journal of Computer Assisted Tomography*, 9(4):659–675, 1985.

[97] John N Rydberg, Stephen J Riederer, Charlotte H Rydberg, and Clifford R Jack. Contrast optimization of fluid-attenuated inversion recovery (flair) imaging. *Magnetic resonance in medicine*, 34(6):868–877, 1995.

[98] Xiaozhi Cao, Congyu Liao, Zhixing Wang, Ying Chen, Huihui Ye, Hongjian He, and Jianhui Zhong. Robust sliding-window reconstruction for accelerating the acquisition of mr fingerprinting. *Magnetic Resonance in Medicine*, 78(4):1579–1588, 2017.

[99] Pedro A Gómez, Miguel Molina-Romero, Guido Buonincontri, Marion I Menzel, and Bjoern H Menze. Designing contrasts for rapid, simultaneous parameter quantification and flow visualization with quantitative transient-state imaging. *arXiv preprint arXiv:1901.07800*, 2019.

[100] K Sommer, T Amthor, M Doneva, P Koken, J Meineke, and P Börnert. Towards predicting the encoding capability of mr fingerprinting sequences. *Magnetic Resonance Imaging*, 41:7–14, 2017.

[101] Michael Lustig, David Donoho, and John M Pauly. Sparse mri: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–

1195, 2007.

[102] Jakob Assländer, Martijn A Cloos, Florian Knoll, Daniel K Sodickson, Jürgen Hennig, and Riccardo Lattanzi. Low rank alternating direction method of multipliers reconstruction for mr fingerprinting. *Magnetic Resonance in Medicine*, 79(1):83–96, 2018.

[103] Bo Zhao, Kawin Setsompop, Elfar Adalsteinsson, Borjan Gagoski, Huihui Ye, Dan Ma, Yun Jiang, P Ellen Grant, Mark A Griswold, and Lawrence L Wald. Improved magnetic resonance fingerprinting reconstruction with low-rank and subspace modeling. *Magnetic Resonance in Medicine*, 79(2):933–942, 2018.

[104] Daniel K. Sodickson and Warren J. Manning. Simultaneous acquisition of spatial harmonics (SMASH): Fast imaging with radiofrequency coil arrays. *Magnetic Resonance in Medicine*, 38(4):591–603, 1997.

[105] Jiayu Song, Qing H Liu, G Allan Johnson, and Cristian T Badea. Sparseness prior based iterative image reconstruction for retrospectively gated cardiac micro-CT. *Medical physics*, 34(11):4476–4483, 11 2007.

[106] Emil Y. Sidky, Chien-Min Kao, and Xiaochuan Pan. Accurate image reconstruction from few-views and limited-angle data in divergent-beam CT. *Journal of X-Ray Science and Technology*, 14(2):119–139, 01 2006.

[107] G. Chinn, P.D. Olcott, and C.S. Levin. Improving SNR with a maximum likelihood compressed sensing decoder for multiplexed PET detectors. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2010 IEEE*, pages 3353–3356, Oct 2010.

[108] S. Valiollahzadeh, Tingting Chang, J.W. Clark, and O.R. Mawlawi. Image recovery in PET scanners with partial detector rings using compressive sensing. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, pages 3036–3039, Oct 2012.

[109] Ben Adcock, Anders C Hansen, Clarice Poon, and Bogdan Roman. Breaking the coherence barrier: A new theory for compressed sensing. *arXiv preprint arXiv:1302.0561*, 2013.

[110] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[111] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[112] Lawrence A Shepp and Benjamin F Logan. The Fourier reconstruction of a head section. *Nuclear Science, IEEE Transactions on*, 21(3):21–43, 1974.

[113] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[114] R.R. Coifman and D.L. Donoho. Translation-invariant de-noising. In Anestis Antoniadis and Georges Oppenheim, editors, *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, pages 125–150. Springer New York, 1995.

[115] Martin Uecker, Peng Lai, Mark J Murphy, Patrick Virtue, Michael Elad, John M Pauly, Shreyas S Vasanawala, and Michael Lustig. ESPIRiT - an eigenvalue approach

to autocalibrating parallel MRI: Where SENSE meets GRAPPA. *Magnetic Resonance in Medicine*, 71(3):990–1001, 2014.

[116] MRI data repository. [Online]. Available from: http://mridata.org/, c2013-. [Accessed 2nd September, 2017].

[117] Klaas P Pruessmann, Markus Weiger, Markus B Scheidegger, Peter Boesiger, et al. SENSE: sensitivity encoding for fast MRI. *Magnetic Resonance in Medicine*, 42(5):952–962, 1999.

[118] Per Thunberg and Per Zetterberg. Noise distribution in sense-and grappa-reconstructed images: a computer simulation study. *Magnetic Resonance Imaging*, 25(7):1089–1094, 2007.

# Appendix A

# Chapter 2 Derivations

This appendix contains the definitions and derivations of the forward pass function and $\mathbb{CR}$ backpropagation gradients for the network layers referenced in Chapter 2. For all layers functions derived in this section, we define the local $\mathbb{CR}$ gradients of the function with respect to its input: $\partial f/\partial z$ and $\partial f/\partial \overline{z}$. During backpropagation, all layers return the conjugate $\mathbb{R}$-derivative of final layer, $L$, with respect to the input $z$:

$$\frac{\partial L}{\partial \overline{z}} = \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)}\frac{\partial f}{\partial \overline{z}} + \frac{\partial L}{\partial \overline{\hat{z}}}\overline{\left(\frac{\partial f}{\partial z}\right)} \tag{A.1}$$

where $\partial L/\partial \overline{\hat{z}} \in \mathbb{C}$ is the derivative received from the layers later in the network architecture.

## A.1 Complex Normalization

Derivation for the layer to normalize a complex scalar input $z$ by scaling its magnitude to one.

Forward function:

$$\hat{z} = n(z) = e^{i\measuredangle z} \tag{A.2}$$

$$= \frac{z}{m(z)} \tag{A.3}$$

$$m = m(z) = (\overline{z}z)^{1/2} \text{ (see complex magnitude, Section 2.4.2)} \tag{A.4}$$

Local $\mathbb{CR}$ derivatives of $n$:

$$\frac{\partial n}{\partial z} = \frac{1}{m(z)} + z\frac{-1}{m(z)^2}\frac{\partial m}{\partial z} \tag{A.5}$$

$$= \frac{1}{m(z)} + z\frac{-1}{\overline{z}z}\frac{1}{2}\frac{\overline{z}}{m(z)} \tag{A.6}$$

$$= \frac{1}{m(z)} - \frac{1}{2}\frac{1}{m(z)} \tag{A.7}$$

$$= \frac{1}{2}\frac{1}{m(z)} \tag{A.8}$$

$$\frac{\partial n}{\partial \overline{z}} = -\frac{z}{m(z)^2}\frac{\partial m}{\partial \overline{z}} \tag{A.9}$$

$$= -\frac{z}{\overline{z}z}\frac{1}{2}\frac{z}{m(z)} \tag{A.10}$$

$$= -\frac{1}{2}\frac{z}{\overline{z}m(z)} \tag{A.11}$$

Conjugate $\mathbb{R}$-derivative of final layer, $L$, with respect to the conjugates of the input $z$, given the derivative from all later layers, $\partial L/\partial \overline{\hat{z}} \in \mathbb{C}$:

$$\frac{\partial L}{\partial \overline{z}} = \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)}\frac{\partial n}{\partial \overline{z}} + \frac{\partial L}{\partial \overline{\hat{z}}}\overline{\left(\frac{\partial n}{\partial z}\right)} \tag{A.12}$$

$$= \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)}\frac{1}{2m(z)} + \frac{\partial L}{\partial \overline{\hat{z}}}\frac{-z}{2\overline{z}m(z)} \tag{A.13}$$

$$= \frac{1}{2m(z)}\left(\overline{\left(\frac{\partial L}{\partial \overline{\hat{z}}}\right)} - \frac{\partial L}{\partial \overline{\hat{z}}}\frac{z}{\overline{z}}\right) \tag{A.14}$$

## A.2 Phase

To compute the phase of complex scalar $z$, we can normalize $z$ to have unit magnitude and then take the log and divide by $i$:

$$\hat{z} = f(z) = \frac{1}{i}\log e^{i\angle z} = -i\log n(z) \tag{A.15}$$

$$n = n(z) = e^{i\measuredangle z} = \frac{z}{(\overline{z}z)^{1/2}} \text{ (see complex normalization, Section A.1)} \tag{A.16}$$

Local $\mathbb{CR}$ derivatives of $f(z)$:

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial n}\frac{\partial n}{\partial z} + \frac{\partial f}{\partial \overline{n}}\overline{\left(\frac{\partial n}{\partial \overline{z}}\right)}, \text{ but } \frac{\partial f}{\partial \overline{n}} = 0 \tag{A.17}$$

$$= \frac{-i}{n(z)}\frac{1}{2}\frac{1}{|z|} \tag{A.18}$$

$$= \frac{-i}{2}\frac{|z|}{z}\frac{1}{|z|} \tag{A.19}$$

$$= \frac{-i}{2z} \tag{A.20}$$

$$\frac{\partial f}{\partial \overline{z}} = \frac{\partial f}{\partial n}\frac{\partial n}{\partial \overline{z}} + \frac{\partial f}{\partial \overline{n}}\overline{\left(\frac{\partial n}{\partial z}\right)}, \text{ but } \frac{\partial f}{\partial \overline{n}} = 0 \tag{A.21}$$

$$= \frac{-i}{n(z)}\frac{-1}{2}\frac{n(z)}{\overline{z}} \tag{A.22}$$

$$= \frac{i}{2\overline{z}} \tag{A.23}$$

## A.3 Separable Sigmoid

Derivation for the separable sigmoid layer function that applies a sigmoid function, $g$, to the real and imaginary parts of the complex scalar input $z$ and places the respective results into the real and imaginary components of the complex scalar output $\hat{z}$.

The forward function:

$$\hat{z} = f(z) = g\big(\operatorname{Re}(z)\big) + ig\big(\operatorname{Im}(z)\big) \tag{A.24}$$

$$g = g(z) = \frac{1}{1 + e^{-z}} \tag{A.25}$$

$$Re = \mathrm{Re}(z) = \frac{1}{2}(z + \overline{z}) \text{ (see real component layer, Section 2.4.2)} \tag{A.26}$$

$$Im = \mathrm{Im}(z) = \frac{1}{2i}(z - \overline{z}) \text{ (see imaginary component layer, Section 2.4.2)} \tag{A.27}$$

Local $\mathbb{CR}$ derivatives of $g$ and $f$, as well as a reminder of the derivatives for the real and imaginary component functions:

$$h(z) \coloneqq \frac{\partial g}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2} = g(z)(1 - g(z)) \qquad \frac{\partial g}{\partial \overline{z}} = 0 \tag{A.28}$$

$$\frac{\partial \,\mathrm{Re}}{\partial z} = \frac{1}{2} \qquad \frac{\partial \,\mathrm{Re}}{\partial \overline{z}} = \frac{1}{2} \tag{A.29}$$

$$\frac{\partial \,\mathrm{Im}}{\partial z} = \frac{1}{2i} \qquad \frac{\partial \,\mathrm{Im}}{\partial \overline{z}} = \frac{i}{2} \tag{A.30}$$

$$\frac{\partial f}{\partial z} = \frac{\partial g}{\partial Re}\frac{\partial \,\mathrm{Re}}{\partial z} + \frac{\partial g}{\partial \overline{Re}}\overline{\left(\frac{\partial \,\mathrm{Re}}{\partial \overline{z}}\right)} + i\frac{\partial g}{\partial Im}\frac{\partial \,\mathrm{Im}}{\partial z} + \frac{\partial g}{\partial \overline{Im}}\overline{\left(\frac{\partial \,\mathrm{Im}}{\partial \overline{z}}\right)} \tag{A.31}$$

$$= h\big(\,\mathrm{Re}(z)\big)\frac{1}{2} + 0 + ih\big(\,\mathrm{Im}(z)\big)\frac{1}{2i} + 0 \tag{A.32}$$

$$= \frac{1}{2}\Big(h\big(\,\mathrm{Re}(z)\big) + h\big(\,\mathrm{Im}(z)\big)\Big) \tag{A.33}$$

$$\frac{\partial f}{\partial \overline{z}} = \frac{\partial g}{\partial Re}\frac{\partial \,\mathrm{Re}}{\partial \overline{z}} + \frac{\partial g}{\partial \overline{Re}}\overline{\left(\frac{\partial \,\mathrm{Re}}{\partial z}\right)} + i\frac{\partial g}{\partial Im}\frac{\partial \,\mathrm{Im}}{\partial \overline{z}} + \frac{\partial g}{\partial \overline{Im}}\overline{\left(\frac{\partial \,\mathrm{Im}}{\partial z}\right)} \tag{A.34}$$

$$= h\big(\,\mathrm{Re}(z)\big)\frac{1}{2} + 0 + ih\big(\,\mathrm{Im}(z)\big)\frac{i}{2} + 0 \tag{A.35}$$

$$= \frac{1}{2}\Big(h\big(\operatorname{Re}(z)\big) - h\big(\operatorname{Im}(z)\big)\Big) \tag{A.36}$$

## A.4 Siglog

Derivations for the siglog layer function with complex scalar input $z$ and complex scalar output $\hat{z}$.

We refer to this activation as *siglog* because it is equivalent to applying the sigmoid operator to the log of the input magnitude and then restoring the phase:

$$\operatorname{siglog}(z) = g(\log(|z|))e^{-i\measuredangle z}, \text{ where } g(z) = \frac{1}{1+e^{-z}} \tag{A.37}$$

$$= \frac{1}{1+e^{-\log(|z|)}}\frac{z}{|z|} \tag{A.38}$$

$$= \frac{1}{1+\frac{1}{|z|}}\frac{z}{|z|} \tag{A.39}$$

$$= \frac{z}{1+|z|} \tag{A.40}$$

The forward function including positive constant that can control the steepness, $c$, and the scale, $r$, of the function:

$$\hat{z} = f(z; r, c) = \frac{z}{c + \frac{1}{r}|z|} \tag{A.41}$$

Local $\mathbb{CR}$ derivatives of $f$:

$$\frac{\partial f}{\partial z} = \frac{c + \frac{1}{r}m(z) - z\frac{1}{r}\frac{1}{2}\frac{\overline{z}}{m(z)}}{\big(c + \frac{1}{r}m(z)\big)^2} \tag{A.42}$$

$$= \frac{c + \frac{1}{r}m(z) - \frac{1}{r}\frac{1}{2}m(z)}{\big(c + \frac{1}{r}m(z)\big)^2} \tag{A.43}$$

$$= \frac{c + \frac{1}{r}\frac{1}{2}m(z)}{\big(c + \frac{1}{r}m(z)\big)^2} \tag{A.44}$$

$$\frac{\partial f}{\partial \overline{z}} = -\frac{z}{\left(c + \frac{1}{r}m(z)\right)^2}\frac{1}{r}\frac{1}{2}\frac{z}{m(z)} \tag{A.45}$$

$$= -\frac{1}{2r}\frac{z^2}{m(z)}\frac{1}{\left(c + \frac{1}{r}m(z)\right)^2} \tag{A.46}$$

## A.5 iGaussian

Derivation for the inverted Gaussian with complex scalar input $z$ and complex scalar output $\hat{z}$.

The forward function warps the magnitude of $z$ by the inverted Gaussian function, $g(z)$, and then restores the phase by multiplying by the normalized version of $z$, $n(z)$:

$$\hat{z} = f(z;\sigma^2) = (1 - e^{-\overline{z}z/2\sigma^2})e^{-i\measuredangle z} \tag{A.47}$$

$$= g(z;\sigma^2)n(z) \tag{A.48}$$

$$g = g(z;\sigma^2) = 1 - e^{-\overline{z}z/2\sigma^2} \text{ (inverted Gaussian)} \tag{A.49}$$

$$n = n(z) = \frac{z}{(\overline{z}z)^{\frac{1}{2}}} \text{ (see complex normalization, Section A.1)} \tag{A.50}$$

Local $\mathbb{CR}$ derivatives of $g$ and $f$:

$$\frac{\partial g}{\partial z} = e^{-\overline{z}z/2\sigma^2}\frac{\overline{z}}{2\sigma^2} \tag{A.51}$$

$$\frac{\partial g}{\partial \overline{z}} = e^{-\overline{z}z/2\sigma^2}\frac{z}{2\sigma^2} \tag{A.52}$$

$$\frac{\partial f}{\partial z} = n(z)\frac{\partial g}{\partial z} + g(z)\frac{\partial n}{\partial z} \tag{A.53}$$

$$\frac{\partial f}{\partial \overline{z}} = n(z)\frac{\partial g}{\partial \overline{z}} + g(z)\frac{\partial n}{\partial \overline{z}} \tag{A.54}$$

## A.6 Cardioid

Derivation for the complex cardioid layer function with complex scalar input $z$ and complex scalar output $\hat{z}$.

The forward function scales the magnitude of $z$ by a cardioid function:

$$\hat{z} = f(z) = \frac{1}{2}\Big(1 + \cos\big(\measuredangle z\big)\Big)z \tag{A.55}$$

$$= \frac{1}{2}\Big(1 + \cos\big(-i\log e^{i\measuredangle z}\big)\Big)z \tag{A.56}$$

$$= \frac{1}{2}\Big(1 + \cos\big(-i\log n(z)\big)\Big)z \tag{A.57}$$

$$= \frac{1}{2}z + \frac{1}{2}\cos\big(-i\log n(z)\big)z \tag{A.58}$$

$$n = n(z) = e^{i\measuredangle z} = \frac{z}{(\overline{z}z)^{1/2}} \text{ (see complex normalization, Section A.1)} \tag{A.59}$$

Local $\mathbb{CR}$ derivatives of $f$:

$$\frac{\partial f}{\partial z} = \frac{1}{2} + \frac{1}{2}\cos\big(-i\log n(z)\big) + z\frac{-1}{2}\sin\big(-i\log n(z)\big)\frac{-i}{n(z)}\frac{\partial n}{\partial z} \tag{A.60}$$

$$= \frac{1}{2} + \frac{1}{2}\cos\big(-i\log n(z)\big) + z\frac{1}{2}\sin\big(-i\log n(z)\big)\frac{i}{n(z)}\frac{1}{2}\frac{1}{(\overline{z}z)^{1/2}} \tag{A.61}$$

$$= \frac{1}{2} + \frac{1}{2}\cos\big(-i\log n(z)\big) + z\frac{1}{2}\sin\big(-i\log n(z)\big)i\frac{(\overline{z}z)^{1/2}}{z}\frac{1}{2}\frac{1}{(\overline{z}z)^{1/2}} \tag{A.62}$$

$$= \frac{1}{2} + \frac{1}{2}\cos\big(-i\log n(z)\big) + \frac{i}{4}\sin\big(-i\log n(z)\big) \tag{A.63}$$

$$= \frac{1}{2} + \frac{1}{2}\cos\big(\measuredangle z\big) + \frac{i}{4}\sin\big(\measuredangle z\big) \tag{A.64}$$

$$\frac{\partial f}{\partial \overline{z}} = z\frac{-1}{2}\sin\big(-i\log n(z)\big)\frac{-i}{n(z)}\frac{\partial n}{\partial \overline{z}} \tag{A.65}$$

$$= z\frac{-1}{2}\sin\big(-i\log n(z)\big)\frac{-i}{n(z)}\frac{-1}{2}\frac{z}{\overline{z}(\overline{z}z)^{1/2}} \tag{A.66}$$

$$= z\frac{-1}{2}\sin\big(-i\log n(z)\big)i\frac{(\overline{z}z)^{1/2}}{z}\frac{1}{2}\frac{z}{\overline{z}(\overline{z}z)^{1/2}} \tag{A.67}$$

$$= \frac{-i}{4}\sin\big(-i\log n(z)\big)\frac{z}{\overline{z}} \tag{A.68}$$

$$= \frac{-i}{4}\sin\big(\measuredangle z\big)\frac{z}{\overline{z}} \tag{A.69}$$

## A.7 Batch Normalization

Complex version of batch normalization introduced in [59] for input vector $\mathbf{z} \in \mathbb{C}^N$ and output vector $\hat{\mathbf{z}} \in \mathbb{C}^N$. The partial derivative $\partial L/\partial \overline{\hat{\mathbf{z}}}$ will be passed in from later layers. Note: to simplify the derivations, the functions below do not include scale and bias parameters, $\gamma$ and $\beta$ from [59]. These parameters are simply scalar versions of the fully connected linear layer weights, which could be applied to each component of the output vector, $\hat{\mathbf{z}}$.

Forward functions, including the functions to compute the mean and variance of $\mathbf{z}$:

$$\mu = \mu(\mathbf{z}) = \frac{1}{N}\sum_n z_n \tag{A.70}$$

$$\sigma^2 = \sigma^2(\mathbf{z}, \mu) = \frac{1}{N}\sum_n \overline{(z_n - \mu)}(z_n - \mu) \tag{A.71}$$

$$\hat{\mathbf{z}} = f(\mathbf{z}, \mu, \sigma^2) = \frac{\mathbf{z} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{A.72}$$

Local $\mathbb{CR}$ derivatives for each function:

$$\frac{\partial \mu}{\partial \mathbf{z}} = \frac{\mathbf{1}}{N} \qquad \frac{\partial \mu}{\partial \overline{\mathbf{z}}} = \mathbf{0} \tag{A.73}$$

$$\frac{\partial \sigma^2}{\partial \mathbf{z}} = \frac{1}{N}(\overline{\mathbf{z} - \mu}) \qquad \frac{\partial \sigma^2}{\partial \overline{\mathbf{z}}} = \frac{1}{N}(\mathbf{z} - \mu) \tag{A.74}$$

$$\frac{\partial \sigma^2}{\partial \mu} = -\frac{1}{N}\sum_n \overline{z_n - \mu} = 0 \qquad \frac{\partial \sigma^2}{\partial \overline{\mu}} = -\frac{1}{N}\sum_n (z_n - \mu) = 0 \tag{A.75}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{z}} = \frac{I}{\sqrt{\sigma^2 + \epsilon}} \qquad \frac{\partial \mathbf{f}}{\partial \overline{\mathbf{z}}} = \mathbf{0} \tag{A.76}$$

$$\frac{\partial \mathbf{f}}{\partial \mu} = \frac{-\mathbf{1}}{\sqrt{\sigma^2 + \epsilon}} \qquad \frac{\partial \mathbf{f}}{\partial \overline{\mu}} = \mathbf{0} \tag{A.77}$$

$$\frac{\partial \mathbf{f}}{\partial \sigma^2} = -\frac{1}{2}(\mathbf{z} - \mu)(\sigma^2 + \epsilon)^{-3/2} \qquad \frac{\partial \mathbf{f}}{\partial \overline{\sigma^2}} = \mathbf{0} \tag{A.78}$$

Derivatives of final layer, $L$, with respect to the mean and variance functions, given the derivative from all later layers $\partial L / \partial \overline{\hat{\mathbf{z}}}$:

$$\frac{\partial L}{\partial \sigma^2} = \sum_n \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}_n}}\right)} \frac{\partial f_n}{\partial \sigma^2} + \frac{\partial L}{\partial \overline{\hat{z}_n}} \overline{\left(\frac{\partial f_n}{\partial \overline{\sigma^2}}\right)} \tag{A.79}$$

$$= -\frac{1}{2}\sum_n \overline{\left(\frac{\partial L}{\partial \overline{\hat{z}_n}}\right)} (z_n - \mu)(\sigma^2 + \epsilon)^{-3/2} \tag{A.80}$$

$$\frac{\partial L}{\partial \overline{\sigma^2}} = \sum_n \overline{\left(\frac{\partial L}{\partial \hat{\overline{z}}_n}\right)} \frac{\partial f_n}{\partial \overline{\sigma^2}} + \frac{\partial L}{\partial \hat{\overline{z}}_n} \overline{\left(\frac{\partial f_n}{\partial \sigma^2}\right)} \tag{A.81}$$

$$= -\frac{1}{2} \sum_n \frac{\partial L}{\partial \hat{\overline{z}}_n} \overline{(z_n - \mu)} (\sigma^2 + \epsilon)^{-3/2} \tag{A.82}$$

$$\frac{\partial L}{\partial \overline{\mu}} = \sum_n \overline{\left(\frac{\partial L}{\partial \hat{\overline{z}}_n}\right)} \frac{\partial f_n}{\partial \overline{\mu}} + \frac{\partial L}{\partial \hat{\overline{z}}_n} \overline{\left(\frac{\partial f_n}{\partial \mu}\right)} + \overline{\left(\frac{\partial L}{\partial \overline{\sigma^2}}\right)} \frac{\partial \sigma^2}{\partial \overline{\mu}} + \frac{\partial L}{\partial \overline{\sigma^2}} \overline{\left(\frac{\partial \sigma^2}{\partial \mu}\right)} \tag{A.83}$$

$$= \sum_n \frac{\partial L}{\partial \hat{\overline{z}}_n} \frac{-1}{\sqrt{\overline{\sigma^2 + \epsilon}}} \tag{A.84}$$

Note that $\partial f / \partial \mu$ will not be required for the computation of $\partial L / \partial \overline{z_n}$ below.

Finally, the derivative of final layer, $L$, with respect to the conjugate of the input vector components, $\overline{z_n}$:

$$\frac{\partial L}{\partial \overline{z_n}} = \overline{\left(\frac{\partial L}{\partial \hat{\overline{\mathbf{z}}}}\right)} \frac{\partial \mathbf{f}}{\partial \overline{z_n}} + \frac{\partial L}{\partial \hat{\overline{\mathbf{z}}}} \overline{\left(\frac{\partial \mathbf{f}}{\partial z_n}\right)} + \frac{\partial L}{\partial \sigma^2} \frac{\partial \sigma^2}{\partial \overline{z_n}} + \frac{\partial L}{\partial \overline{\sigma^2}} \overline{\left(\frac{\partial \sigma^2}{\partial z_n}\right)} + \frac{\partial L}{\partial \mu} \frac{\partial \mu}{\partial \overline{z_n}} + \frac{\partial L}{\partial \overline{\mu}} \overline{\left(\frac{\partial \mu}{\partial z_n}\right)} \tag{A.85}$$

$$= 0 + \frac{\partial L}{\partial \hat{\overline{z}}_n} \frac{1}{\sqrt{\overline{\sigma^2 + \epsilon}}} \tag{A.86}$$

$$- \frac{1}{2} \sum_m \overline{\left(\frac{\partial L}{\partial \hat{\overline{z}}_m}\right)} (z_m - \mu)(\sigma^2 + \epsilon)^{-3/2} \frac{1}{N} (z_n - \mu)$$

$$- \frac{1}{2} \sum_m \frac{\partial L}{\partial \hat{\overline{z}}_m} \overline{(z_m - \mu)} (\sigma^2 + \epsilon)^{-3/2} \frac{1}{N} (z_n - \mu)$$

$$+ 0 + \sum_m \frac{\partial L}{\partial \overline{\hat{z}_m}} \frac{-1}{\sqrt{\sigma^2 + \epsilon}} \frac{1}{N}$$

$$= \frac{1}{\sqrt{\sigma^2 + \epsilon}} \tag{A.87}$$

$$\left( \frac{\partial L}{\partial \overline{\hat{z}_n}} - \frac{1}{2N} \sum_m \left( \overline{\left( \frac{\partial L}{\partial \overline{\hat{z}_m}} \right)} \frac{z_m - \mu}{\sqrt{\sigma^2 + \epsilon}} \right) \frac{z_n - \mu}{\sqrt{\sigma^2 + \epsilon}} \right.$$

$$\left. - \frac{1}{2N} \sum_m \left( \frac{\partial L}{\partial \overline{\hat{z}_m}} \frac{\overline{z_m - \mu}}{\sqrt{\sigma^2 + \epsilon}} \right) \frac{z_n - \mu}{\sqrt{\sigma^2 + \epsilon}} - \frac{1}{N} \sum_m \frac{\partial L}{\partial \overline{\hat{z}_m}} \right)$$

$$= \frac{1}{\sqrt{\sigma^2 + \epsilon}} \left( \frac{\partial L}{\partial \overline{\hat{z}_n}} - \frac{1}{2N} \sum_m \left( \overline{\left( \frac{\partial L}{\partial \overline{\hat{z}_m}} \right)} \hat{z}_m \right) \hat{z}_n - \frac{1}{2N} \sum_m \left( \frac{\partial L}{\partial \overline{\hat{z}_m}} \overline{\hat{z}_m} \right) \hat{z}_n - \frac{1}{N} \sum_m \frac{\partial L}{\partial \overline{\hat{z}_m}} \right) \tag{A.88}$$

$$= \frac{1}{\sqrt{\sigma^2 + \epsilon}} \left( \frac{\partial L}{\partial \overline{\hat{z}_n}} - \frac{\hat{z}_n}{N} \operatorname{Re} \left( \sum_m \frac{\partial L}{\partial \overline{\hat{z}_m}} \overline{\hat{z}_m} \right) - \frac{1}{N} \sum_m \frac{\partial L}{\partial \overline{\hat{z}_m}} \right) \tag{A.89}$$

## A.8 Euclidean Loss

Derivation for the complex Euclidean loss layer with length $N$ complex vector input $\mathbf{z}$ and real scalar output $L \in \mathbb{R}$.

Forward function, given the fixed ground truth vector, $\mathbf{z}_{gt}$:

$$L = L(\mathbf{z}; \mathbf{z}_{gt}) = \frac{1}{2} ||\mathbf{z} - \mathbf{z}_{gt}||_2^2 \tag{A.90}$$

$$= \frac{1}{2} (\mathbf{z} - \mathbf{z}_{gt})^H (\mathbf{z} - \mathbf{z}_{gt}) \tag{A.91}$$

$$= \frac{1}{2}\sum_{n=1}^{N}(\overline{z_n} - \overline{z_{gt,n}})(z_n - z_{gt,n}) \tag{A.92}$$

$$= \frac{1}{2}\sum_{n=1}^{N}\overline{z_n}z_n - \overline{z_{gt,n}}z_n - \overline{z_n}z_{gt,n} + \overline{z_{gt,n}}z_{gt,n} \tag{A.93}$$

Conjugate $\mathbb{R}$-derivative of $L$, for both an individual component, $z_n$, and for the full input vector, $\mathbf{z}$:

$$\frac{\partial L}{\partial \overline{z_n}} = \frac{1}{2}(z_n - z_{gt,n}) \tag{A.94}$$

$$\frac{\partial L}{\partial \overline{\mathbf{z}}} = \frac{1}{2}(\mathbf{z} - \mathbf{z}_{gt}) \tag{A.95}$$

To show that the complex-valued Euclidean loss can be computed by a real-valued implementations, we compare the values computed by the forward and backward passes of each version with the assumption that the real methods are operating on 2x length vectors with the real components concatenated to the imaginary components. The forward pass derivation for $\mathbf{z} = \mathbf{x} + i\mathbf{y}$, given the fixed ground truth vector, $\mathbf{z}_{gt} = \mathbf{x}_{gt} + i\mathbf{y}_{gt}$ is as follows:

$$\mathbf{e} \coloneqq \mathbf{z} - \mathbf{z}_g t \tag{A.96}$$

$$= \mathbf{e}_x + i\mathbf{e}_y \tag{A.97}$$

$$L = L(\mathbf{z}; \mathbf{z}_{gt}) = \frac{1}{2}||\mathbf{z} - \mathbf{z}_{gt}||_2^2 \tag{A.98}$$

$$= \frac{1}{2}\mathbf{e}^H\mathbf{e} \tag{A.99}$$

$$= \frac{1}{2}\sum_{n=1}^{N}\overline{e_n}e_n \tag{A.100}$$

$$= \frac{1}{2}\sum_{n=1}^{N}(e_{x,n} - ie_{y,n})(e_{x,n} + ie_{y,n}) \tag{A.101}$$

$$= \frac{1}{2}\sum_{n=1}^{N} e_{x,n}^2 + e_{y,n}^2 \tag{A.102}$$

$$= \frac{1}{2}|||[\mathbf{e}_x^T, \mathbf{e}_y^T]^T||_2^2 \tag{A.103}$$

$$= \frac{1}{2}|||[\mathbf{x}^T - \mathbf{x}_{gt}^T, \mathbf{x}^T - \mathbf{x}_{gt}^T]^T||_2^2 \tag{A.104}$$

$$= \frac{1}{2}|||[\mathbf{x}^T, \mathbf{y}^T]^T - [\mathbf{x}_{gt}^T, \mathbf{y}_{gt}^T]^T||_2^2 \tag{A.105}$$

$$= L_{real}([\mathbf{x}^T, \mathbf{y}^T]^T; [\mathbf{x}_{gt}^T, \mathbf{y}_{gt}^T]^T) \tag{A.106}$$

The backward pass of a real-valued Euclidean loss layer with input $[\mathrm{Re}(\mathbf{z}), \mathrm{Im}(\mathbf{z}]$ produces the real and imaginary components of the complex Euclidean conjugate $\mathbb{R}$-derivative, but with a 2x scaling factor:

$$\frac{\partial L_{real}}{\partial x_n} = x_n - x_{gt,n} \tag{A.107}$$

$$\frac{\partial L_{real}}{\partial \mathbf{x}} = \mathbf{x} - \mathbf{x}_{gt} \tag{A.108}$$

$$\frac{\partial L_{real}}{\partial [\mathbf{x}^T, \mathbf{y}^T]^T} = [\mathbf{x}^T - \mathbf{x}_{gt}^T, \mathbf{y}^T - \mathbf{y}_{gt}^T]^T \tag{A.109}$$

$$= [\mathrm{Re}(\mathbf{z} - \mathbf{z}_{gt})^T, \mathrm{Im}(\mathbf{z} - \mathbf{z}_{gt})^T]^T \tag{A.110}$$

$$= \left[ \mathrm{Re}\left(2\frac{\partial L}{\partial \overline{\mathbf{z}}}\right)^T, \mathrm{Im}\left(2\frac{\partial L}{\partial \overline{\mathbf{z}}}\right)^T \right]^T \tag{A.111}$$

# Appendix B

# Chapter 4 Derivations

## B.1 Variance of Added Gaussian Noise

Derivation between equations (4.9) and (4.10) to determine the variance of the Gaussian noise to add during the image quality prediction process:

$$\sigma^2_{add,k} = \sigma^2_{under,k} - \sigma^2_{ref} \tag{4.9}$$

$$= \frac{\sigma^2_{acq}}{\tau_{pred,k}} - \sigma^2_{ref}$$

$$= \frac{\tau_{ref}}{\tau_{ref}} \frac{\sigma^2_{acq}}{\tau_{pred,k}} - \sigma^2_{ref}$$

$$= \frac{\tau_{ref}}{\tau_{pred,k}} \sigma^2_{ref} - \sigma^2_{ref}$$

$$= \left( \frac{\tau_{ref}}{\tau_{pred,k}} - 1 \right) \sigma^2_{ref} \tag{4.10}$$

## B.2 Weighted Least Squares

Derivation of the maximum likelihood formulation of MRI reconstruction optimization. This derivation applies directly to the likelihood term of the MAP derivation described in section 4.1.4, specifically between equations (4.13) and (4.14):

$$\boldsymbol{m}^*_{MLE}$$

$$= \underset{\boldsymbol{m}}{\operatorname{argmax}} P(\boldsymbol{y}|\boldsymbol{m})$$

$$= \underset{\boldsymbol{m}}{\operatorname{argmin}} - \log P(\boldsymbol{y}|\boldsymbol{m})$$

$$= \underset{\boldsymbol{m}}{\operatorname{argmin}} - \log \prod_{k=1}^{N_P} P(y_k|\boldsymbol{m})$$

$$= \underset{\boldsymbol{m}}{\operatorname{argmin}} - \sum_{k=1}^{N_P} \log P(y_k|\boldsymbol{m})$$

$$= \underset{\boldsymbol{m}}{\operatorname{argmin}} - \sum_{k=1}^{N_P} \log \frac{1}{\sqrt{2\pi\sigma_{acq}^2/\tau_k}} exp\left(\frac{-|y_k - s_k|^2}{2\sigma_{acq}^2/\tau_k}\right)$$

$$= \underset{\boldsymbol{m}}{\operatorname{argmin}} \sum_{k=1}^{N_P} \log \frac{-1}{\sqrt{2\pi\sigma_{acq}^2/\tau_k}} + \frac{|y_k - (F\boldsymbol{m})_k|^2}{2\sigma_{acq}^2/\tau_k}$$

$$= \underset{\boldsymbol{m}}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{N_P} \tau_k |y_k - (F\boldsymbol{m})_k|^2$$