Learning Robust Data-Driven Methods for Inverse Problems and Change Detection

by

Davis Gilton

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2021

Date of final oral examination: 04/29/2021

The dissertation has been approved by the following members of the Final Oral Committee:

Robert Nowak, Professor, Electrical and Computer Engineering
Rebecca Willett, Professor, Computer Sciences and Statistics, University of Chicago
William Sethares, Professor, Electrical and Computer Engineering
Stephen Wright, Professor, Computer Sciences

Acknowledgments

This thesis would not have been possible without the thoughtful and helpful mentorship of my advisor, Dr. Rebecca Willett. She has tirelessly supported my growth as a student and researcher and has taught me more than I knew existed about engineering, writing, presenting, and all that it takes to produce high-quality research in the volatile, exciting field we work in.

I am grateful for the encouragement and help of my other committee members: Robert Nowak, William Sethares, and Stephen Wright. Each time I discussed my work with any of them, I came out with valuable insights and renewed enthusiasm, which have contributed significantly to this dissertation.

Much of the work in this thesis was performed in collaboration with Greg Ongie, who has been a constant source of help, guidance, and encouragement. I cannot thank him enough for his insight, friendship, and catsitting help.

I would also like to thank my other collaborators and mentors I've been lucky enough to engage with during graduate school: Douglas Cochran and Visar Berisha, at ASU; Ed Zelnio and Christopher Paulson, at AFRL-RY; David Leifker, at Lands End; Lee Seversky and Eric Heim, at AFRL-RI; Gregory Shakhnarovich, at TTIC; Pavan Mallapragada and Joel Pfeiffer, at Microsoft. Their encouragement and help were an indispensable part of my graduate experience.

Thank you to my family, for your love and guidance throughout these several tumultuous years. And finally, thank you to Xin, Alpha, and Foxie for your love and support - I couldn't have done this without you.

Contents

List of Tables

List of Figures

Abstract

The field of image reconstruction and inverse problems in imaging have been revolutionized by the introduction of methods which *learn* to solve inverse problems. This thesis investigates a variety of methods for learning to solve inverse problems by leveraging data: first by exploring the online sparse linear bandit setting, and then by investigating modern methods for leveraging training data to learn to solve inverse problems. In addition, this thesis explores a multi-model method of leveraging human descriptions of change in time series of images to regularize a graph-cut-based change-point detection method.

Recent research into learning to solve inverse problems has been dominated by "unrolled optimization" approaches, which unroll a fixed number of iterations of an iterative optimization algorithm, replacing one or more elements of that algorithm with a neural network. These methods have several attractive properties: they can leverage even limited training data to learn accurate reconstructions, they tend to have lower runtime and require fewer iterations than more standard methods which leverage non-learned regularizers, and they are simple to implement and understand. However, learned iterative methods, like most learned inverse problem solvers, are sensitive to small changes in the data measurement model; they are uninterpretable, suffering reduced reconstruction quality if run for more or fewer iterations than were used at train time; and they are limited by memory and numerical constraints to small numbers of iterations, potentially lowering the ceiling for best available reconstruction quality using these methods.

This thesis proposes an alternative architecture design based on a Neumann series, which is attractive from a practical perspective for its sample complexity performance and ease to train compared to methods based on unrolled iterative optimization. In addition, this thesis proposes and tests two techniques to adapt arbitrary trained inverse problem solvers to different measurement models, enabling deployment of a single learned model on a variety of forward models without sacrificing performance or requiring potentially-costly new data. Finally, this thesis demonstrates how to train iterative solvers that are unrolled for an *arbitrary* number of iterations. The proposed technique for the first time permits deep iterative solvers that admit practical convergence guarantees, while allowing flexibility in trading off computation for performance.

Chapter 1

Introduction and Background

1.1 Introduction

Structure and Overview

The structure of this work is as follows:

- Chapter 1 presents an overview of learning to solve inverse problems in imaging, which acts as background for the next three chapters.
- Chapter 2 presents the Neumann network, a method to leverage limited training data for linear inverse problems in images. The Neumann network is related to other data-driven approaches to solving inverse problems in imaging, particularly learned iterative methods. The final version of this work will include some further investigation of learned denoisers and regularizers for the aforementioned iterative optimization approaches, as well as modifications to improve their data efficiency. The work in this chapter was performed jointly with Greg Ongie and Rebecca Willett.
- Chapter 3 and Chapter 4 tackle two problems in the field. First, we explore model adaptation: deep inverse solvers are almost universally brittle to changes in the measurement model, limiting the wider adaptation of these methods to highly controlled applications. Second, "unrolled" methods are based on more traditional optimization methods, but are in general uninterpretable and do not converge to useful solutions. We introduce a method based on Deep Equilibrium models [7] which permits efficient training and practical convergence guarantees for optimization algorithms with deep regularizers, a first in the literature. The work in these chapters was performed jointly with Greg Ongie and Rebecca Willett.
- In Chapter 5, a method for the linear bandit problem for sparse data is proposed and analyzed. In this case, a Bayesian method known as the Relevance Vector Machine is leveraged as part of a bandit algorithm to achieve competitive regret bounds that depend on the accuracy of the learned regularizer, with respect to the sparsity of the underlying signal. This work was performed with Rebecca Willett [21].
- Chapter 6 is a brief diversion away from learning to solve linear inverse problems into learning deep learning machinery for detecting and describing changes in *visual streams*. Visual streams are time series of images in which two successive images may be separated by viewpoint shifts, lighting changes, and other "nuisance" changes caused by gaps in time between capturing successiv frames. We demonstrate that by regularizing the changepoint method with a deep network that annotates these streams with language descriptions, the changepoint detection task becomes easier. The work in this chapter was performed jointly with Ruotian Luo, Greg Shakhnarovich, and Rebecca Willett.

The remainder of the introduction provides some background on solving inverse problems with deep networks, while outlining the setting and notation for the treatment of linear inverse problems used for the remainder of this work. As Chapter 6 studies a different problem than the other chapters, it is self-contained, and the study of prior work is in the introduction to that chapter.

1.2 Deep Regularizers for Linear Inverse Problems

The majority of this dissertation investigates various methods for solving *inverse problems*, in which we wish to reconstruct an image $x^\star$ from corrupted measurements $\mathbf{y}$. When $\mathbf{y}$ is a noisy function of $x^\star$, i.e. $\mathbf{y} = A(x^\star) + \epsilon$, where in our case $A(\cdot)$ is some non-invertible function. Inverse problems, especially *linear* inverse problems, are ubiquitous in application: common examples include image superresolution, deblurring, dehazing, MRI image reconstruction, and phase retrieval.

Concretely, we assume the following model for measurements. We observe $\mathbf{y} = \mathbf{A}x^\star + \epsilon$, where $\mathbf{y}, \epsilon \in \mathcal{R}^m$, $x^\star \in \mathcal{R}^n$, and $\mathbf{A} \in \mathcal{R}^{m \times n}$. This model encompasses multidimensional data (say, RGB images) by first vectorizing $\mathbf{x}$. Complex data can be used as well, but for simplicity we always treat complex-valued images as two-channel images - one channel for the real part, and one for the imaginary part.

A simplifying assumption this work makes is that ϵ is isotropic Gaussian noise. That is, $\mathbf{y} \sim \mathcal{N}(\mathbf{A}x^\star, \sigma^2\mathbf{I})$ for some $\sigma^2 \in \mathcal{R}$. In this case, it is simple to show that the maximum-likelihood estimate is:

$$\widehat{\mathbf{x}}_{MLE} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \tag{1.1}$$

If $\mathbf{A}$ is full-rank and $m \geqslant n$ then the above has a trivial solution $\widehat{\mathbf{x}}_{MLE} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{y}$. This solution is generally avoided for nontrivial problems, as the direct inverse is sensitive to noise if $\mathbf{A}$ is poorly-conditioned (some of its eigenvalues are much smaller than the others), and it may not lead to unique solutions for non-full-rank $\mathbf{A}$. The rest of this work assumes that $\mathbf{A}$ is either not full rank (subsampled MRI, superresolution) or is so poorly-conditioned that direct inversion is unstable (deblurring).

To alleviate the stability and non-uniqueness problems, a typical approach is to introduce a *regularizer* $r(\mathbf{x})$ which represents knowledge about desirable solutions into the reconstruction process. This can be interpreted as a maximum a posteriori estimate:

$$\widehat{\mathbf{x}}_{MAP} = \arg\min_{x} -\log(p(\mathbf{x}|\mathbf{y})) - \log(p(\mathbf{x})) = \arg\min_{x} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 - \log(p(\mathbf{x})). \tag{1.2}$$

More generally, it is typical to replace $-\log(p(\mathbf{x}))$ with some $r(\mathbf{x})$ representing prior knowledge of the distribution or structure of $\mathbf{x}$. For example, the Tikhonov regularizer [58] corresponds to $\lambda\|\mathbf{x}\|_2^2$. Common non-learned regularizers include total variation [62], sparsity in a (potentially overcomplete) basis [64, 59], or rank of some structured representation of the data [53].

This work does assume that $\mathbf{A}$ is given by the world; that is, we cannot choose $\mathbf{A}$ to be favorable in some way for reconstruction. While the field of sampling and compressed sensing are rich ones, this problem is out of scope for this thesis.

However, when training data is available, it is possible that the above "static" regularizers may be suboptimal. In several applications, large datasets are becoming available (see, for example, the fastMRI dataset [67]), which permit *learning* to solve inverse problems. A broad collection of recent works, as surveyed by [6] and [42], have explored using machine learning methods to *learn* to solve inverse problems in imaging. The below provides an exploration of some of the broad categories of learning to solve inverse problems, with particular focus on *unrolled optimization*, which can often be

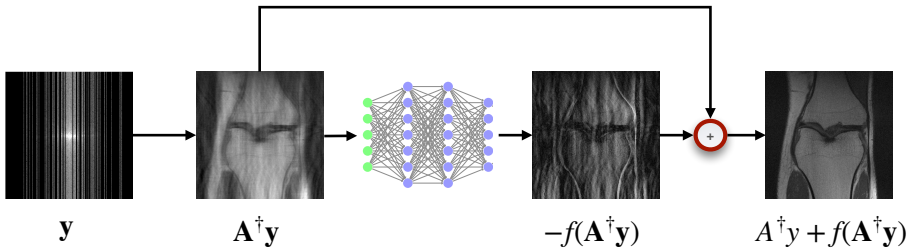


Figure 1.1: Diagram of residual learning to learn to solve inverse problems. In this case the residual network acts to correct the structured errors produced by the approximate inverse in the first step of this algorithm.

interpreted as learning a good $r(\mathbf{x})$ using training data.

Throughout, we assume we have training samples of the form $(\mathbf{x}_i, \mathbf{y}_i)$ for $i = 1, \ldots, N$, where $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \boldsymbol{\epsilon}_i$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is known and the noise $\boldsymbol{\epsilon}_i$ is treated as unknown.

1.3 Neural Networks for Inverse Problems in Imaging

There are several general categories of methods used to learn to solve inverse problems, which are reviewed below.

Residual Networks and Simple Model-Agnostic Methods

An agnostic learner uses the training data to learn a mapping from $\mathbf{y}$ to $\mathbf{x}$ with limited knowledge of $\mathbf{A}$ at any point in the training or testing process [66]. The general principle is that, given enough training data, we should be able to learn everything we need to know about $\mathbf{A}$ to successfully estimate $\mathbf{x}$. Empirically, the success of this approach appears to be highly dependent on the forward operator $\mathbf{A}$. This straightforward approach has been demonstrated on superresolution [17, 33], blind deconvolution [66], and motion deblurring [57], among others. In general, this approach requires large quantities of training data because it must not only learn the geometry of the image space containing the $\mathbf{x}$'s, but also aspects of $\mathbf{A}$.

Alternately, if $\mathbf{A}$ is known, a particularly successful approach to solving inverse problems with neural networks has been through residual learning [27].

Residual learning constructs an output of a neural network $f(\cdot)$ of the form $z = f(x) + x$. Residual networks were originally proposed in [27], but began to see widespread use in image reconstruction when introduced as a form of image denoiser by [68]. The chief insight made by this work was that training a neural network to only learn the *noise* in a noisy image is a simpler task than forcing the neural network to learn to reproduce the original image in its entirety. Here, it is expected that $f(\mathbf{x})$, the output of just the neural network, is expected to have small norm.

However, in inverse problems, it is often the case that residual learning is not possible, since $\mathbf{y}$ and $\mathbf{x}$ may not lie in the same space (as in MRI reconstruction, or more simply, superresolution, in which $\mathbf{y}$ is a different size than $\mathbf{x}$). To enable using residual learning, typically some initial reconstruction $\mathbf{A}^\dagger(\mathbf{y})$ is what is used. In this case, the residual network is of the form $\mathbf{A}^\dagger(\mathbf{y}) + f(\mathbf{A}^\dagger(\mathbf{y}))$. $\mathbf{A}^\dagger(\cdot)$ is not necessarily a pseudoinverse for $\mathbf{A}$, but must be a function that maps from the $\mathbf{y}$-space to $\mathbf{x}$-space. For concreteness, in MRI this may be a zero-filled IFFT, whereas in superresolution this may be any

number of upsampling schemes for the downsampled $\mathbf{y}$, like bicubic or nearest-neighbor interpolation. An alternate approach is to *learn* some initial linear mapping $\mathbf{A}^\dagger$, using a fully-connected layer of appropriate size, as in [70].

Training proceeds by minimizing the mean squared error between the residual reconstruction and $\mathbf{x}_i$ over all $\mathbf{x}_i$.

A number of problems have been tackled in this way: see [**quan2018compressed**, 70, 41, 31] for some examples. The residual network approach is appealing: test-time reconstruction is often very fast compared to classical iterative approaches, and training is quite straightforward. However, because little knowledge of the forward model is "built-in" to the network, resnets can require large datasets to achieve high accuracy, and they are tied to particular forward models. Since the deep network is learning to "denoise" the particular structured noise introduced by the potentially crude reconstruction $A^\dagger$, using these methods for multiple forward models requires training multiple networks. In Section 5, we address this brittleness to changes in the forward model, and introduce a method to alleviate this issue.

Decoupling Training From Reconstruction

A decoupled approach operates in two stages. In the first stage, a collection of training images $\mathbf{x}_i$ is used to learn a representation of the image space of interest. In the second stage, this learned representation is incorporated into a mapping from $(\mathbf{y}, \mathbf{x})$ to $\widehat{\boldsymbol{\beta}}$. That is, the learning takes place in a manner that is decoupled from the inverse problem at hand. Here we present two examples of this.

Learning to Reconstruct Images Via Generative Models

In the simplest method for solving inverse problems with deep generative models, it is typically assumed that the practitioner has access to a model G for $\mathbf{x}$'s that takes as input a low-dimensional latent vector $\boldsymbol{z} \in \mathbb{R}^d$ where $d < p$ and outputs $\boldsymbol{x} = G(\boldsymbol{z})$. The details of the generative model are typically not essential. The basic idea is that the images of interest lie oa low-dimensional submanifold that can be indexed by $\boldsymbol{z}$. Then solving the inverse problem corresponds to solving a constrained optimization problem:

$$\min_{\boldsymbol{z}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \text{s.t.} \mathbf{x} = G(\boldsymbol{z})$$

which gives a reconstruction of the form:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}=G(\boldsymbol{z}),\boldsymbol{z}\in\mathbb{R}^d}{\arg\min} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2. \tag{1.3}$$

Practically, solving for $\boldsymbol{z}$ is done by gradient descent (or projected gradient descent, if $\boldsymbol{z}$ is constrained to some set) on the $\boldsymbol{z}$ vector. However, the above formulation assumes that $G(\cdot)$ is a good generative model for the image set in question. For datasets with limited variety or low dimensionality, this may be a reasonable assumption, but in more complex settings like MRI reconstruction, the problem of training a good generative model is still open. This overall approach was originallydescribed in [9] with applications to compressed sensing, and has been followed up on by further work [26, 25, 15,

Figure 1.2: Solving a simple deblurring problem with CSGM and IAGAN. The forward model here is a motion blur with kernel size 7×7 and $\theta = 10$ degrees. CSGM struggles to accurately model the data distribution, while IAGAN retrains the network G to fit the measurements $\mathbf{y}$ more exactly.

28]. The original method (entitled Compressed Sensing with Generative Models, or CSGM), does not require pairs of $(\mathbf{x}_i, \mathbf{y}_i)$ for training, but does incur a significant computational cost at test time.

By contrast, the Deep Image Prior approach [60] proposes fixing the input to a *randomly-initialized* deep network, which for simplicity we also refer to as $G(\boldsymbol{z})$. However, rather than performing gradient descent on $\boldsymbol{z}$, the authors of [60] propose running gradient descent on the parameters of $G(\boldsymbol{z})$, which I denote θ. This gives an estimated solution of the form:

$$\widehat{\mathbf{x}} = G(\boldsymbol{z}; \hat{\theta}); \hat{\theta} = \arg\min_{\theta} \|\mathbf{y} - \mathbf{A}G(\boldsymbol{z}; \theta)\|_2^2. \tag{1.4}$$

Perhaps surprisingly, this method has been shown to work well for superresolution, JPEG artifact removal, and other tasks. This implies that simply constraining a reconstruction to be the output of a (deep) convolutional network is a worthwhile regularizer in itself. Unfortunately this approach is even more costly than CSGM: each reconstruction requires retraining G, which is difficult to perform in a batched fashion.

More recent work, known as Image-Adaptive GANs [29] CSGM and the Deep Image Prior approach. This work alternates between optimizing $\boldsymbol{z}$ and θ:

$$\widehat{\mathbf{x}} = G(\hat{\boldsymbol{z}}; \hat{\theta}); \hat{\boldsymbol{z}}, \hat{\theta} = \arg\min_{\boldsymbol{z}, \theta} \|\mathbf{y} - \mathbf{A}G(\boldsymbol{z}; \theta)\|_2^2. \tag{1.5}$$

While the computational downsides of Deep Image Prior are still present, this method does produce greatly improved reconstructions over CSGM, while being model-agnostic. The success of this approach indicates that learning a perfect generative model may not be neccessary to produce good solutions to inverse problems, at a significant computational cost. Comparisons between reconstructions produced by CSGM and IAGAN are provided in Fig. 1.2.

Plug-and-Play and RED

To motivate an alternative to learning a generative model, recall the MAP estimator:

$$\widehat{\mathbf{x}}_{MAP} = \arg\min_{x} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 - \log(p(\mathbf{x})).$$

To calculate $\widehat{\mathbf{x}}_{MAP}$ a reasonable approach is to choose some starting point and run gradient descent on the above objective function. The gradient step will be of the form:

$$\widehat{\mathbf{x}}_{MAP}^{(k+1)} = \widehat{\mathbf{x}}_{MAP}^{(k)} - \eta\mathbf{A}^\top(\mathbf{A}\widehat{\mathbf{x}}_{MAP}^{(k)} - \mathbf{y}) - \eta\nabla\log(p(\mathbf{x})).$$

The problematic term here is the score $\nabla\log(p(\mathbf{x}))$. [4] investigated the properties of neural networks trained to denoise data corrupted by Gaussian noise (which will be denoted $R(\mathbf{x})$), and revealed that $\nabla\log(p(\mathbf{x}))$ can be approximated by $\nabla\log(p(\mathbf{x})) \approx (R(\mathbf{x}) - \mathbf{x})/\sigma^2$. Hence we can plug in a pretrained denoising network (or some other accurate denoiser) to calculate the above $\mathbf{x}_{MAP}$.

However, the above result assumes that the denoiser is MSE-optimal and that the approximation from [4] is valid. In practice these are hard constraints, and the above algorithm tends to not perform well. However, the insight that a pretrained denoiser may be useful to "plug in" to an optimization algorithm is still valid. This is exactly the approach used by the Plug and Play and Regularization By Denoising methods.

The Plug-and-Play approach was originally proposed in [61], in which it was used in the context of the ADMM algorithm. The original method replaces a proximal operator in ADMM with an existing denoiser, but the Plug-and-Play approach has been expanded to a number of other optimization algorithms, such as the primal-dual algorithm [37], generalized approximate message passing [39], and many others (see, for example [30, 56, 10]). The simplest of these is Plug-and-Play for the proximal gradient algorithm.

A proximal gradient algorithm [51, 43] starts with an initial estimate $\mathbf{x}^{(0)}$ and step size $\eta > 0$ and then iterates between computing a gradient descent step that pushes the current estimate $\mathbf{x}^{(k)}$ to be a better fit to the data, followed by a *proximal operator* that finds an estimate in the proximity of the resulting iterate that matches some prior condition. This second step is often thought of as a denoising step, and is what is replaced in Plug-and-Play proximal gradient descent.

An alternative to Plug-and-Play, which only implicitly defines a regularizer $r(\cdot)$, is Regularization by Denoising (RED) [49]. In RED, $r(\mathbf{x}) = \mathbf{x}^\top(x - R(\mathbf{x}))/2$, where as before $R(\mathbf{x})$ is some pretrained denoiser. Under certain conditions on $R(\cdot)$ [48], the gradient $\nabla r(\mathbf{x}) = \mathbf{x} - R(\mathbf{x})$. A number of algorithms were proposed in [49, 48] for solving this problem. In general, RED and Plug-and-Play provide comparable reconstruction quality, but have different constraints on the denoisers to ensure convergence, which can be difficult to enforce or verify for general deep denoisers [50].

The key feature in both generative-model-based and learned-denoiser-based approaches is that all training takes place independently of $\mathbf{A}$ – *i.e.,* we either learn a generative model or a proximal operator using the training data, neither of which require knowledge of $\mathbf{A}$. The advantage of this approach is that once training has taken place, the learned generative model or proximal operator can be used for *any* linear inverse problem, so we do not need to re-train a system for each new inverse problem. In other words, the learning is *decoupled* from solving the inverse problem. However, it is typically required to train a number of denoisers for different noise levels, as well as choose optimization parameters by cross-validation. Some recent work has focused on automatically tuning the parameters in the

optimization method used, by training a separate agent which dynamically selects parameters during test time [63].

The flexibility of the decoupled approach comes with a high price in terms of sample complexity. To see why, note that learning a generative model or a denoising autoencoder fundamentally amounts to estimating a probability distribution and its support over the space of images; let us denote this distribution as $\phi(\mathbf{x})$. Thoroughly understanding the space of images of interest is important if our learned regularizer is to be used for linear inverse problems of which we are unaware during training.

On the other hand, when we know $\mathbf{A}$ at training time, then we only need to learn the *conditional* distribution $\phi(\mathbf{x}|\mathbf{A}\mathbf{x})$ or $\phi(\mathbf{x}|\mathbf{y})$ [19].

For example, imagine an image inpainting scenario in which we only observe a subset of pixels in the image $\mathbf{x}$. Rather than learn the distribution over the space of all possible images, we only need to learn the distribution over the space of missing pixels conditioned on the observed pixels, $\phi(\mathbf{x}|\mathbf{A}\mathbf{x})$. Of course, $\phi(\mathbf{x}|\mathbf{A}\mathbf{x})$ can be calculated from $\phi(\mathbf{x})$ and $\mathbf{A}$ using Bayes' law, but the latter distribution may lie in a much lower-dimensional space, making it easier to learn with limited data.

It is well-known that the accuracy of any estimate of $\phi(\mathbf{x})$ has a minimax rate that scales as $\mathcal{O}(N^{-\frac{\alpha}{2\alpha+p}})$ with N the number of training samples, p the dimension of $\phi(\mathbf{x})$, and α a smoothness term [18, 14, 32]. This scaling is quite restrictive, but if the conditional density only depends on a subset of size p' of the original p coordinates, the rate for estimating the conditional density function is $\mathcal{O}(N^{-\frac{\alpha}{2\alpha+p'}})$ [19].

The key point is that decoupled approaches (implicitly) require learning the full density $\phi(\mathbf{x})$, whereas a method that incorporates $\mathbf{A}$ into the learning process has the potential to simply learn the conditional density $\phi(\mathbf{x}|\mathbf{A}\mathbf{x})$, which often can be performed accurately with relatively little training data. This observation is supported by experimental results in Chapter 2, which illustrate that decoupled approaches generally require far more training samples than methods that incorporate knowledge of $\mathbf{A}$.

In addition, Plug and Play and RED tend to assume that the noise distribution at test time is relatively unstructured. Practically, this imposes constraints on what problems are best to address with these approaches: very low sampling rates in MRI or compressed sensing, or extreme distortion in deblurring tend to be more difficult to tackle with Plug and Play or RED.

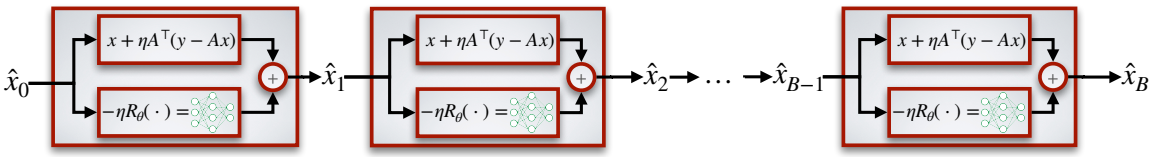


Figure 1.3: Unrolled gradient descent network. The result of B iterations of gradient descent with a fixed step size η and regularizer with gradient R, as in (**??**) is equivalent to the output of the above network, with each block corresponding to a single iteration. The network maps a linear function of the measurements, $\mathbf{x}^{(0)} = \eta\mathbf{A}^\top\mathbf{y}$, to a reconstruction $\widehat{\boldsymbol{\beta}}$ by successive application of an operator of the form $[\mathbf{I} - \eta\mathbf{A}^\top\mathbf{A}](\cdot) - \eta R(\cdot)$ and addition of $\eta\mathbf{A}^\top\mathbf{y}$. Here R is a trained neural network, and the scale parameter η is also trained.

Review of Deep Unrolling Methods

Plug and Play and RED are attractive because they may be applied for a wide range of inverse problems, but in exchange they can require learning the data distribution $\phi(\mathbf{x})$, and often do not deal well with structured noise. Rather than first learn, then use the learned network in an optimization algorithm, a recent trend in learning to solve inverse problems, called Deep Unrolling, trains a network as part of an optimization algorithm directly.

Deep Unrolling encompasses inverse problem solvers which consist of a fixed number of architecturally identical "blocks," which are often inspired by a particular optimization algorithm. These methods represent the current state of the art in MRI reconstruction, with most top submissions to the fastMRI challenge [40] being some sort of unrolled net. Unrolled networks have seen success in other imaging tasks, e.g. low-dose CT [65], light-field photography [13], and emission tomography [36].

For demonstration, a specific variant of deep unrolling methods based on gradient descent is described here, although many variants exist based on alternative optimization or fixed point iteration schemes [42]. Recall from above that if there is a known regularization function r that could be applied to an image x, then we could compute an image estimate by solving the optimization problem

$$\widehat{x} = \arg\min_{x} \frac{1}{2}\|y - Ax\|_2^2 + r(x). \tag{1.6}$$

If r is differentiable, this can be accomplished via gradient descent. That is, we start with an initial estimate $\mathbf{x}0$ such as $\mathbf{x}0 = A^\top y$ and choose a step size $\eta > 0$, such that for iteration $k = 1, 2, 3, \ldots$, we set

$$\mathbf{x}k + 1 = \mathbf{x}k + \eta A^\top (y - A\mathbf{x}k) - \eta \nabla r(\mathbf{x}k),$$

where ∇r is the gradient of the regularizer.

The block diagram for the unrolled gradient descent estimator is shown in Figure 1.3. We can now represent the gradient of the regularizer, $R(\cdot)$, with a trainable neural network. In contrast to the decoupled approach described above, unrolled optimization methods learn the regularizer (or its gradient) in the context of the forward model $\mathbf{A}$ and training observations $\mathbf{y}_i$ by minimizing the disparity between the true $\mathbf{x}_i$ and $\widehat{\boldsymbol{\beta}}(\mathbf{y}_i)$, the output of the full network (see Figure 1.3). This end-to-end training sidesteps the sample complexity challenges described in Section 1.3.

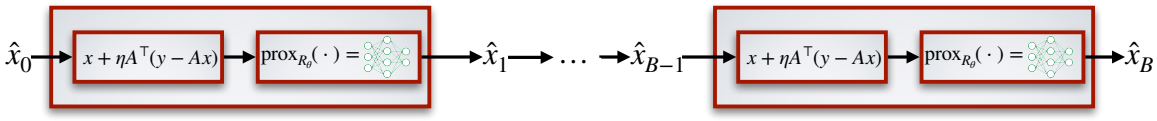


Figure 1.4: Unrolled proximal gradient descent network. The result of B iterations of proximal gradient descent with a proximal operator $\text{prox}_{R_\theta}(\mathbf{x})$ is equivalent to the output of the above network, with each block corresponding to a single iteration. The network maps a linear function of the measurements, $\mathbf{x}^{(0)} = \eta \mathbf{A}^\top \mathbf{y}$, to a reconstruction $\widehat{\boldsymbol{\beta}}$ by successive application of an operator of the form $R_\theta\left([\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A}](\cdot) + \eta \mathbf{A}^\top \mathbf{y}\right)$. Here R is a trained neural network, and the scale parameter η is also trained.

The unrolling approach can be applied to a variety of optimization algorithms beyond gradient descent. The earliest proposed unrolled inverse problem solver was [23], in which the authors proposed unrolling the Iterative Shrinkage and Thresholding Algorithm (ISTA) [8] and the coordinate descent

algorithm; further refinements of this approach were proposed in [54, 30]. More recent work has illustrated the efficacy of unrolled optimization as applied to (proximal) gradient descent [11, 16, 35], alternating directions method of multipliers [55], primal-dual methods [1], half-quadratic splitting [52, 69], block coordinate descent [46, 47, 12], alternating minimization [2], iterative reweighted least squares [3, 44], and approximate message passing [38]. In proximal gradient settings, the learned neural network is interpreted as a learned proximal operator (see Fig 1.4), whereas in the gradient descent network (see Fig 1.3), the learned neural network can be interpreted as the gradient of some regularizer. In other words, for different unrolled optimization methods the learned neural network can play different roles.

More generally, deep unrolled methods fix some number of iterations K of an existing iterative optimization scheme (typically K ranges from 5 to 10), declare that $\mathbf{x}K$ will be the estimate $\widehat{\mathbf{x}}$, and model one or more operations of the iterative scheme with a neural network, denoted $R_\theta(x)$, that can be learned with training data. We assume that all R_θ have identical weights, although other works also explore non-weight-tied variants [1].

Training attempts to minimize the cost function $\sum_{i=1}^{n} \|\widehat{\mathbf{x}}^{(K)}(y_i;\theta) - x_i^*\|_2^2$ with respect to the learnable network parameters θ. This form of training is often called "end-to-end"; that is, we do not train the network representing ∇r in isolation, but rather on the quality of the resulting estimate $\widehat{\mathbf{x}}^{(K)}$, which depends on the forward model $\mathbf{A}$.

In a related but separate subfield, [**wu2019learning**] describes a unrolling approach to learning a forward model in an imaging context, but with the goal of designing a forward model that optimizes reconstruction quality, rather than estimating a correction to the forward model from measurements.

The number of iterations is kept small for two reasons. First, at deployment, these systems are optimized to compute image estimates quickly – a desirable property we wish to retain in developing new methods. Second, it is challenging to train deep unrolled networks for many iterations due to memory limitations of GPUs because the memory required to calculate the backpropagation updates scales linearly with the number of unrolled iterations.

While for practical reasons the number of blocks B must be kept small in end-to-end training, empirically this does not appear to be an obstacle to good performance. For example, [23] notes that end-to-end training reduces the iterations of the ISTA algorithm required to achieve a fixed error rate by a factor of 20, and [16] achieve promising performance with $B = 8$ proximal gradient descent iterations. Chapter 2 proposes and demonstrates a method that resembles Deep Unrolling, but bypasses some practical issues during training to ensure better performance in the low-sample case by being based on the Neumann series expansion rather than an existing optimization method.

Another strategy to enable deeper unrollings is to perform block-by-block training as in [47]; however, this approach is unsuitable when the neural network weights are shared between blocks, which is the case in our setting. It is possible to relax the shared-weights assumption, but [2] has illustrated that in the low-sample setting, different learned weights in each block can be suboptimal.

However, tying an unrolled network to a particular B limits flexibility in deployment. If a particular datapoint only requires a rough reconstruction, or a different one requires more detail than most other datapoints, a deep unrolled system cannot accommodate these requirements. As a workaround, consider training such systems for a small number of iterations K (e.g., $K = 10$), then extracting the learned regularizer gradient ∇r, and using it within a gradient descent algorithm until convergence (i.e. for more iterations K than used in training). Our numerical results highlight how poorly this method performs in practice (See Fig. 1.5). Choosing the number of iterations K (and hence the test

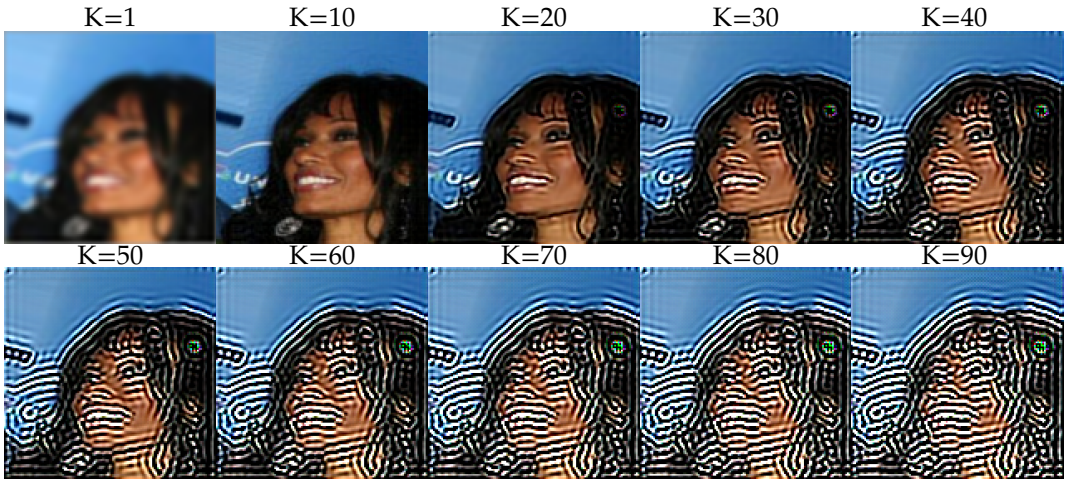


Figure 1.5: Sample images and reconstructions from Deep Unrolled Proximal Gradient Descent trained for 10 iterations performing Gaussian deblurring with Gaussian noise with $\sigma = 0.01$. Each image represents the output of iterate number K. While at 10 iterations the reconstruction quality is state-of-the-art, after 10 iterations additional iterations decrease reconstruction quality significantly. The ground truth may be viewed in the final column of Figure B.3. Chapter 4 addresses this problem, introducing a novel method to train learned iterative networks *to convergence*.

time computational budget) at training time is essential.

Chapter 4 introduces a method based on Deep Equilibrium [7] to train Deep Unrolled models with an arbitrary number of iterations. By training the full optimization process, a Deep Equilibrium-based reconstruction method can guarantee convergence, and enables adaptive computation: computation can be dynamically allocated to images which require better reconstruction quality, while less-essential images can save computation but not worry about disastrous reconstructions.

Instability in Inverse Problems

Chapter 3, on model adaptation, is motivated in part by experiments presented in [5], which show that deep neural networks trained to solve inverse problems are prone to several types of instabilities. Specifically, they showed that model drift in the form of slight changes in the forward model (even "beneficial" ones, like increasing the number of k-space samples in MRI) often have detrimental impacts on reconstruction performance. While [5] is mostly empirical in nature, a follow-up mathematical study [22] provides theoretical support to this finding, implying that instability arises naturally from training standard deep learning inverse solvers. However, recently [20] has shown that the instabilities observed in in [5] can be mitigated to some extent by adding noise to measurements during training, though such techniques are not sufficient to resolve artifacts arising from substantial model drift. To address a subset of these issues, [45] and [34] propose adversarial training frameworks that increases the robustness of inverse problem solvers. However, [45] and [34] focus on robustness to adversarial perturbations in the measurements for a fixed forward model, and do not address a global change in the forward model. Chapter 3 focuses on two methods for adapting inverse problems trained on one forward model to another without retraining.

A recent paper [24] has proposed domain adaptation techniques to transfer a reconstruction network from one inverse problem setting to another, e.g., adapting a network trained for CT reconstruction to perform MRI reconstruction. However, the focus of that approach is on adapting to *changes in the image*

distribution, whereas Chapter 3's approaches focus on *changes to the forward model* assuming the image distribution is unchanged.

Bibliography

[1] Jonas Adler and Ozan Öktem. "Learned primal-dual reconstruction". In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1322–1332.

[2] Hemant K Aggarwal, Merry P Mani, and Mathews Jacob. "MoDL: Model Based Deep Learning Architecture for Inverse Problems". In: *IEEE Transactions on Medical Imaging* (2018).

[3] Hemant Kumar Aggarwal, Merry P Mani, and Mathews Jacob. "Multi-Shot Sensitivity-Encoded Diffusion MRI using Model-Based Deep Learning (MoDL-MUSSELS)". In: *arXiv preprint arXiv:1812.08115* (2018).

[4] Guillaume Alain and Yoshua Bengio. "What regularized auto-encoders learn from the data-generating distribution". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3563–3593.

[5] Vegard Antun et al. "On instabilities of deep learning in image reconstruction and the potential costs of AI". In: *Proceedings of the National Academy of Sciences* (2020).

[6] Simon Arridge et al. "Solving inverse problems using data-driven models". In: *Acta Numerica* 28 (2019), pp. 1–174.

[7] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "Deep equilibrium models". In: *Advances in Neural Information Processing Systems*. 2019, pp. 690–701.

[8] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[9] Ashish Bora et al. "Compressed Sensing using Generative Models". In: *International Conference on Machine Learning (ICML)*. 2017, pp. 537–546.

[10] Gregery T Buzzard et al. "Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium". In: *SIAM Journal on Imaging Sciences* 11.3 (2018), pp. 2001–2020.

[11] Yunjin Chen and Thomas Pock. "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1256–1272.

[12] Il Yong Chun and Jeffrey A Fessler. "Deep BCD-Net Using Identical Encoding-Decoding CNN Structures for Iterative Image Recovery". In: *arXiv preprint arXiv:1802.07129* (2018).

[13] Il Yong Chun et al. "Momentum-Net: Fast and convergent iterative neural network for inverse problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[14] Bernard Delyon and Anatoli Juditsky. "On minimax wavelet estimators". In: *Applied and Computational Harmonic Analysis* 3.3 (1996), pp. 215–228.

[15] Manik Dhar, Aditya Grover, and Stefano Ermon. "Modeling Sparse Deviations for Compressed Sensing using Generative Models". In: *International Conference on Machine Learning*. 2018, pp. 1222–1231.

[16] Steven Diamond et al. "Unrolled Optimization with Deep Priors". In: *arXiv preprint arXiv:1705.08041* (2017).

[17] Chao Dong et al. "Learning a deep convolutional network for image super-resolution". In: *European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 184–199.

[18] David L Donoho et al. "Density estimation by wavelet thresholding". In: *The Annals of Statistics* (1996), pp. 508–539.

[19] Sam Efromovich. "Conditional density estimation in a regression setting". In: *The Annals of Statistics* 35.6 (2007), pp. 2504–2535.

[20] Martin Genzel, Jan Macdonald, and Maximilian März. "Solving Inverse Problems With Deep Neural Networks–Robustness Included?" In: *arXiv preprint arXiv:2011.04268* (2020).

[21] Davis Gilton and Rebecca Willett. "Sparse linear contextual bandits via relevance vector machines". In: *2017 International Conference on Sampling Theory and Applications (SampTA)*. IEEE. 2017, pp. 518–522.

[22] Nina M Gottschling et al. "The troublesome kernel: why deep learning for inverse problems is typically unstable". In: *arXiv preprint arXiv:2001.01258* (2020).

[23] Karol Gregor and Yann LeCun. "Learning Fast Approximations of Sparse Coding". In: *International Conference on Machine Learning (ICML)*. 2010, pp. 399–406.

[24] Yoseob Han et al. "Deep learning with domain adaptation for accelerated projection-reconstruction MR". In: *Magnetic resonance in medicine* 80.3 (2018), pp. 1189–1205.

[25] Paul Hand, Oscar Leong, and Vlad Voroninski. "Phase retrieval under a generative prior". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9136–9146.

[26] Paul Hand and Vladislav Voroninski. "Global Guarantees for Enforcing Deep Generative Priors by Empirical Risk". In: *Conference On Learning Theory*. 2018, pp. 970–978.

[27] Kaiming He et al. "Deep residual learning for image recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[28] Reinhard Heckel. "Regularizing linear inverse problems with convolutional neural networks". In: *arXiv preprint arXiv:1907.03100* (2019).

[29] Shady Abu Hussein, Tom Tirer, and Raja Giryes. "Image-adaptive gan based reconstruction". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 3121–3129.

[30] Ulugbek S Kamilov and Hassan Mansour. "Learning optimal nonlinearities for iterative thresholding algorithms". In: *IEEE Signal Processing Letters* 23.5 (2016), pp. 747–751.

[31] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. "Accurate image super-resolution using very deep convolutional networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1646–1654.

[32] John Lafferty, Han Liu, and Larry Wasserman. *Minimax theory*. [Online; accessed 07-January-2019]. 2008.

[33] Christian Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4681–4690.

[34] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. "Adversarial regularizers in inverse problems". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8507–8516.

[35] Morteza Mardani et al. "Neural proximal gradient descent for compressive imaging". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9573–9583.

[36] Abolfazl Mehranian and Andrew J Reader. "Model-based deep learning PET image reconstruction using forward-backward splitting expectation maximisation". In: *IEEE Transactions on Radiation and Plasma Medical Sciences* (2020).

[37] Tim Meinhardt et al. "Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1799–1808.

[38] Chris Metzler, Ali Mousavi, and Richard Baraniuk. "Learned D-AMP: Principled neural network based compressive image recovery". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1772–1783.

[39] Christopher A Metzler, Arian Maleki, and Richard G Baraniuk. "From denoising to compressed sensing". In: *IEEE Transactions on Information Theory* 62.9 (2016), pp. 5117–5144.

[40] Matthew J Muckley et al. "State-of-the-art Machine Learning MRI Reconstruction in 2020: Results of the Second fastMRI Challenge". In: *arXiv preprint arXiv:2012.06318* (2020).

[41] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. "Deep multi-scale convolutional neural network for dynamic scene deblurring". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, p. 3.

[42] Gregory Ongie et al. "Deep Learning Techniques for Inverse Problems in Imaging". In: *arXiv preprint arXiv:2005.06001* (2020).

[43] Neal Parikh and Stephen Boyd. "Proximal algorithms". In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[44] Aniket Pramanik, Hemant Kumar Aggarwal, and Mathews Jacob. "Off-the-grid model based deep learning (O-MoDL)". In: *arXiv preprint arXiv:1812.10747* (2018).

[45] Ankit Raj, Yoram Bresler, and Bo Li. "Improving Robustness of Deep-Learning-Based Image Reconstruction". In: *arXiv preprint arXiv:2002.11821* (2020).

[46] Saiprasad Ravishankar, Il Yong Chun, and Jeffrey A Fessier. "Physics-driven deep training of dictionary-based algorithms for MR image reconstruction". In: *Asilomar Conference on Signals, Systems, and Computers*. 2017, pp. 1859–1863.

[47] Saiprasad Ravishankar et al. "Deep dictionary-transform learning for image reconstruction". In: *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*. IEEE. 2018, pp. 1208–1212.

[48] Edward T Reehorst and Philip Schniter. "Regularization by denoising: Clarifications and new interpretations". In: *IEEE Transactions on Computational Imaging* 5.1 (2018), pp. 52–67.

[49] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The little engine that could: Regularization by denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[50] Ernest Ryu et al. "Plug-and-Play Methods Provably Converge with Properly Trained Denoisers". In: *International Conference on Machine Learning*. 2019, pp. 5546–5557.

[51] Mark Schmidt, Nicolas L Roux, and Francis R Bach. "Convergence rates of inexact proximal-gradient methods for convex optimization". In: *Advances in Neural Information Processing Systems* (*NeurIPS*). 2011, pp. 1458–1466.

[52] Uwe Schmidt and Stefan Roth. "Shrinkage fields for effective image restoration". In: *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*). 2014, pp. 2774–2781.

[53] Peter J Shin et al. "Calibrationless parallel imaging reconstruction based on structured low-rank matrix completion". In: *Magnetic resonance in medicine* 72.4 (2014), pp. 959–970.

[54] Pablo Sprechmann, Alex Bronstein, and Guillermo Sapiro. "Learning efficient structured sparse models". In: *International Conference on Machine Learning* (*ICML*). 2012, pp. 219–226.

[55] Jian Sun, Huibin Li, and Zongben Xu. "Deep ADMM-Net for compressive sensing MRI". In: *Advances in Neural Information Processing Systems* (*NeurIPS*). 2016, pp. 10–18.

[56] Yu Sun, Brendt Wohlberg, and Ulugbek S Kamilov. "An online plug-and-play algorithm for regularized image reconstruction". In: *IEEE Transactions on Computational Imaging* 5.3 (2019), pp. 395–408.

[57] Yunfei Tan et al. "Motion Deblurring Based on Convolutional Neural Network". In: *International Conference on Bio-Inspired Computing: Theories and Applications*. Springer. 2017, pp. 623–635.

[58] Andrey Nikolayevich Tikhonov. "On the stability of inverse problems". In: *Dokl. Akad. Nauk SSSR*. Vol. 39. 1943, pp. 195–198.

[59] Joel A Tropp and Stephen J Wright. "Computational methods for sparse solution of linear inverse problems". In: *Proceedings of the IEEE* 98.6 (2010), pp. 948–958.

[60] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.

[61] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. "Plug-and-play priors for model based reconstruction". In: *2013 IEEE Global Conference on Signal and Information Processing*. IEEE. 2013, pp. 945–948.

[62] Yilun Wang et al. "A new alternating minimization algorithm for total variation image reconstruction". In: *SIAM Journal on Imaging Sciences* 1.3 (2008), pp. 248–272.

[63] Kaixuan Wei et al. "Tuning-free Plug-and-Play Proximal Algorithm for Inverse Imaging Problems". In: *arXiv preprint arXiv:2002.09611* (2020).

[64] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. "Sparse reconstruction by separable approximation". In: *IEEE Transactions on signal processing* 57.7 (2009), pp. 2479–2493.

[65] Dufan Wu, Kyungsang Kim, and Quanzheng Li. "Computationally efficient deep neural network for computed tomography image reconstruction". In: *Medical physics* 46.11 (2019), pp. 4763–4776.

[66] Li Xu et al. "Deep convolutional neural network for image deconvolution". In: *Advances in Neural Information Processing Systems* (*NeurIPS*). 2014, pp. 1790–1798.

[67] Jure Zbontar et al. “fastMRI: An Open Dataset and Benchmarks for Accelerated MRI”. In: 2018. arXiv: 1811.08839.

[68] Kai Zhang et al. “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising”. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[69] Kai Zhang et al. “Learning deep CNN denoiser prior for image restoration”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2017.

[70] Bo Zhu et al. “Image reconstruction by domain-transform manifold learning”. In: *Nature* 555.7697 (2018), pp. 487–492.

Chapter 2

Neumann Networks

2.1 Introduction

The following chapter will focus on the main problem proposed in the introduction: given a limited number of samples that are obtained with a fixed number of measurements each, find the optimal reconstruction for a new data point. This is the problem of learning to solve linear inverse problems, and we will be particularly interested in their applications in imaging and image reconstruction.

Specifically, we investigate finding some $\mathbf{x}$ that minimizes the following cost, where $r(\mathbf{x})$ represents a learned regularization term, which will be elaborated on later:

$$\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + r(\mathbf{x}). \tag{2.1}$$

This chapter is structured as follows: we will examine the literature in learned methods for linear inverse problems, and then we will examine some theoretical results on optimal reconstructions, as well as sample complexity of learning certain kinds of regularizers. After that, we introduce the motivation for and architecture of the Neumann network. Several modifications of the Neumann network are also introduced, one based on patchwise regularization, which is more applicable in general.

2.2 Learning to Regularize

In this chapter we consider solving linear inverse problems in imaging in which a p-pixel image, $x^\star \in \mathbb{R}^p$ (in vectorized form), is observed via m noisy linear projections as $\mathbf{y} = \mathbf{A}x^\star + \boldsymbol{\epsilon}$, where $\mathbf{y}, \boldsymbol{\epsilon} \in \mathbb{R}^m$ and $\mathbf{A} \in \mathbb{R}^{m \times p}$. This general model is used throughout computational imaging, from basic image restoration tasks like deblurring, super-resolution, and image inpainting [24], to a wide variety of tomographic imaging applications, including common types of magnetic resonance imaging [20], X-ray computed tomography [18], radar imaging [7], among others [6]. The task of estimating $x^\star$ from $\mathbf{y}$ is often referred to as *image reconstruction*. Classical image reconstruction methods assume some prior knowledge about $x^\star$ such as smoothness [52], sparsity in some dictionary or basis [21, 37, 57], or other geometric properties [49, 55, 15, 39], and attempt to estimate a $\widehat{\mathbf{x}}$ that is both a good fit to the observation $\mathbf{y}$ and that also conforms to this prior knowledge. In general, a *regularization function* $r(\mathbf{A})$ measures the lack of conformity of $\mathbf{A}$ to this prior knowledge and $\widehat{\mathbf{x}}$ is selected so that $r(\widehat{\mathbf{x}})$ is as small as possible while still providing a good fit to the data.

However, recent work in computer vision using deep neural networks has leveraged large collections of "training" images to yield unprecedented image recognition performance [25, 28, 32], and an emerging body of research is exploring whether this training data can also be used to improve the quality of image reconstruction. In other words, *can training data be used to learn how to regularize inverse problems?* As we detail below, existing methods include using training images to learn a low-dimensional image manifold and constraining $\widehat{\mathbf{x}}$ to lie on this manifold [9] or learning a denoising autoencoder that can be treated as a regularization step (*i.e.,* proximal operator) within an iterative reconstruction scheme [41].

We propose a novel neural network architecture based on the Neumann series expansion [50, 23] that we call a *Neumann network*, we describe several of its key theoretical properties, and empirically illustrate its superior performance on a variety of reconstruction tasks. In particular,

- Neumann networks, which directly incorporate the forward operator $\mathbf{A}$ into the network architecture, can have dramatically lower sample complexity than *model-agnostic* networks that attempt to learn the entire image space. As a result, they are much more amenable to applications such as medical imaging or scientific domains where datasets may be smaller.
- Neumann networks naturally yield a block-wise structure with *skip connections* [25] emanating from each block. These skip connections appear to yield a smoother optimization landscape that is easier to train than related network architectures.
- When the images of interest lie on a union of subspaces, and when the trainable nonlinear components of the network have sufficient expressiveness/capacity, there exists a Neumann network estimator that approximates the optimal oracle estimator arbitrarily well. Furthermore, after training the Neumann network on simulated data drawn from a union of subspaces, we show the learned nonlinear components in the trained Neumann network have the form predicted by theory.
- A simple preconditioning step combined with the Neumann network further improves empirical performance.
- Modification to enforce local regularization provides superior sample complexity performance on some types of imagery.
- The empirical performance of the Neumann network on superresolution, deblurring, compressed sensing, and inpainting problems exceeds that of competing methods.

2.3 Neumann Networks

Below, we adopt the following strategy. First, we consider the setting in which the gradient of the regularizer is a linear operator and derive a simple Neumann series approximation to an optimal solution of (2.1). We then consider the overall *Neumann network* formed if R is represented by a (potentially nonlinear) neural network. In this section, we treat a nonlinear network operation as a heuristic that we justify theoretically in Section 2.5. This section also describes a simple preconditioning step that can improve the accuracy of our approach and an explicit comparison between the proposed Neumann network and the unrolled gradient descent network described in Section **??**.

Proposed Network Architecture

Our proposed network architecture is motivated by the regularized least squares optimization problem (2.1) in the special case where the regularizer r is quadratic. In particular, assume $r(\mathbf{A}) = \frac{1}{2}\mathbf{A}^\top \mathbf{R}\mathbf{A}$ so that $\nabla r(\mathbf{A}) = \mathbf{R}\mathbf{A}$ for some matrix $\mathbf{R} \in \mathbb{R}^{p\times p}$. Then a necessary condition for $x^\star$ to be a minimizer of (2.1) in this case is

$$(\mathbf{A}^\top \mathbf{A} + \mathbf{R})x^\star = \mathbf{A}^\top \mathbf{y}. \tag{2.2}$$

Assuming the matrix on the left-hand side is invertible, the solution is given by

$$x^\star = (\mathbf{A}^\top \mathbf{A} + \mathbf{R})^{-1}\mathbf{A}^\top \mathbf{y}. \tag{2.3}$$

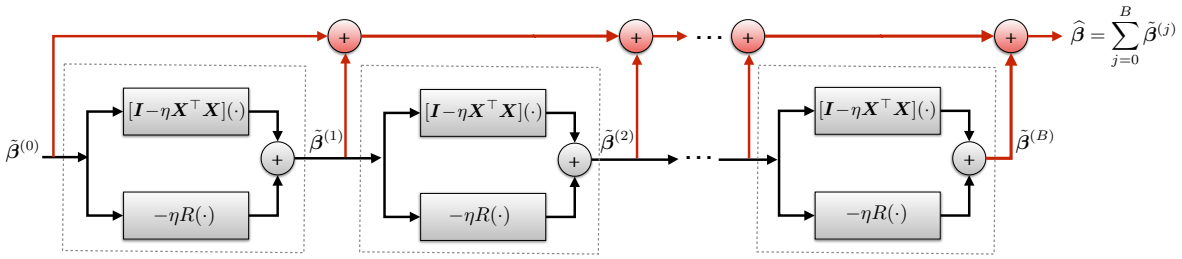


Figure 2.1: Proposed Neumann network architecture. Inspired by the Neumann series expansion for computing the inverse of an operator, a Neumann network maps a linear function of the measurements, $\widetilde{\mathbf{A}}^{(0)} = \eta \mathbf{A}^\top \mathbf{y}$ to a reconstruction $\widehat{\mathbf{x}}$ by successive application of an operator the form $[\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A}](\cdot) - \eta R(\cdot)$ while summing the intermediate outputs of each block. Here R is a trained neural network, and the scale parameter η is also trained. Unlike other networks based on unrolling of iterative optimization algorithms, the series structure of Neumann networks lead naturally to skip connections [25] (highlighted in red) that route the output of each dashed block to directly to the output layer.

In order to approximate the matrix inverse in (2.3) we consider a Neumann series expansion of linear operators [50, 23], which we now recall. Let $\mathbf{A}$ be any $p \times p$ matrix and let $\mathbf{I}$ denote the $p \times p$ identity matrix. If the *Neumann series* $\sum_{k=0}^{\infty} \mathbf{A}^k$ converges then $\mathbf{I} - \mathbf{A}$ is invertible and we have

$$(\mathbf{I} - \mathbf{A})^{-1} = \sum_{k=0}^{\infty} \mathbf{A}^k = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 \cdots \tag{2.4}$$

In particular, a sufficient condition for the convergence of the Neumann series is $\|\mathbf{A}\| < 1$ where $\|\cdot\|$ is the operator norm. We will make use of an alternative form of the same identity:

$$\mathbf{B}^{-1} = \eta \sum_{k=0}^{\infty} (\mathbf{I} - \eta \mathbf{B})^k, \tag{2.5}$$

which is obtained through a change of variables.

Applying the Neumann series expansion (2.5) to the matrix inverse appearing in (2.3), we have[1]

$$x^\star = \sum_{j=0}^{\infty} (\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A} - \eta \mathbf{R})^j (\eta \mathbf{A}^\top \mathbf{y}). \tag{2.6}$$

Truncating the series in (2.6) to $B+1$ terms, and replacing multiplication by the matrix $\mathbf{R}$ with a general mapping $R : \mathbb{R}^p \to \mathbb{R}^p$, motivates an estimator $\widehat{\mathbf{x}}$ of the form

$$\widehat{\mathbf{x}}(\mathbf{y}) := \sum_{j=0}^{B} ([\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A}](\cdot) - \eta R(\cdot))^j (\eta \mathbf{A}^\top \mathbf{y}). \tag{2.7}$$

We turn (2.7) into a trainable estimator by letting $R = R_\theta$ be a trainable mapping depending on a vector of parameters $\theta \in \mathbb{R}^q$ to be learned from training data. Specifically, in this work we assume R_θ is a neural network, where θ is a vectorized set of weights and biases that define the network. We also treat the step-size choice η as a trainable parameter. The class of estimators $\widehat{\mathbf{x}}(\mathbf{y}) = \widehat{\mathbf{x}}(\mathbf{y}; \theta, \eta)$ specified (2.7) with trainable network $R = R_\theta$ we call *Neumann networks*.

[1]The series in (2.5) is guaranteed to converge if $\|\mathbf{I} - \eta \mathbf{B}\| < 1$. Hence, the expansion in (2.6) is valid provided $\|\mathbf{I} - \eta(\mathbf{A}^\top \mathbf{A} + \mathbf{R})\| < 1$, which holds if and only if $\mathbf{A}^\top \mathbf{A} + \mathbf{R}$ is positive definite and $\eta < \|\mathbf{A}^\top \mathbf{A} + \mathbf{R}\|^{-1}$.

Observe that Neumann networks are motivated by an application of the Neumann series identity in the case where the gradient of the regularizer is a linear operator (or, equivalently, the regularizer is quadratic). However, for a general regularizer r such that $R = \nabla r$ is nonlinear, the Neumann network estimator $\widehat{\mathbf{x}}(\mathbf{y})$ in (2.7) may not be a good solution to the optimization problem (2.1). This is because the Neumann series identity (2.5) only holds for linear operators. Despite this fact, we show in the next section that a Neumann network estimator is still mathematically justified under certain model assumptions on the data distribution for which the ideal R is *piecewise linear*. For now, we simply treat the Neumann network estimator as a heuristic motivated by case where R is linear.

To see how (2.7) can be formulated as a network, observe that the terms in (2.7) have the following recursive form: let the input to the network be $\tilde{\mathbf{A}}^{(0)} := \eta \mathbf{A}^\top \mathbf{y}$ and define

$$\tilde{\mathbf{A}}^{(j)} := (\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A})\tilde{\mathbf{A}}^{(j-1)} - \eta R(\tilde{\mathbf{A}}^{(j-1)}) \tag{2.8}$$

for all $j = 1, ..., B$. Then we have $\widehat{\mathbf{x}}(\mathbf{y}) = \sum_{j=0}^{B} \tilde{\boldsymbol{\beta}}^{(j)}$.

Figure 2.1 shows a block diagram for implementing the Neumann network using the recursion (2.8). Each block with a dashed boundary in Figure 2.1 represents an application of the operator $[\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A}](\cdot) - \eta R(\cdot)$. Due to its underlying series structure, the Neumann network has several *skip connections* (highlighted in red) that route the output of each dashed block (*i.e.*, the $\tilde{\boldsymbol{\beta}}^{(j)}$'s) to the output layer, similar to those found in residual networks [25] and related architectures [28]. These skip connections are a distinguishing feature of Neumann networks compared to networks derived from unrolled optimization approaches, such as unrolled gradient descent (see Figure 1.3). We hypothesize these additional skip connections result in a smoother optimization landscape relative to other unrolling approaches, which allows for easier training via stochastic gradient descent. See Section 2.6 for empirical evidence and more discussion on this point.

Equivalence of Unrolled Gradient Descent and Neumann Network for a Linear Learned Component

Suppose the learned component R is linear, *i.e.*, $R(\mathbf{A}) = \mathbf{R}\mathbf{A}$ for some matrix $\mathbf{R} \in \mathbb{R}^{p \times p}$. The Bth iteration $\boldsymbol{\beta}^{(B)}$ of unrolled gradient descent with step size $\eta > 0$ and initialization $\boldsymbol{\beta}^{(0)} = \eta \mathbf{A}^\top \mathbf{y}$ can be expanded to obtain

$$\mathbf{A}^{(B)} = \eta \sum_{j=0}^{B} (\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A} - \eta \mathbf{R})^j \mathbf{A}^\top \mathbf{y},$$

which is precisely the form of the Neumann network estimator (2.7). Therefore, if R is linear the estimator obtained using a unrolling of gradient descent and the Neumann network estimator are the same. When R is nonlinear we no longer have this equivalence.

2.4 Modifications to the Neumann Network

In this section, some potential architectural modifications to the "vanilla" Neumann network are outlined. First, a straightforward derivation of how to include preconditioning in the Neumann network is outlined

Preconditioning

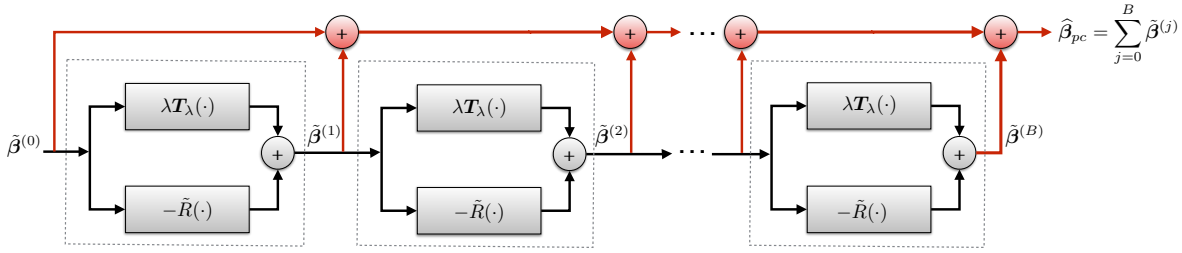


Figure 2.2: Proposed preconditioned Neumann network architecture. The network has the same basic architecture as a Neumann network, but uses a different linear component given by $\mathbf{T}_\lambda = (\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I})^{-1}$ where $\lambda > 0$ and a different initialization $\tilde{\mathbf{A}}^{(0)} = \mathbf{T}_\lambda\mathbf{A}^\top\mathbf{y}$. When the matrix inverse in $\mathbf{T}_\lambda$ is computationally prohibitive to apply, we replace all instances of $\mathbf{T}_\lambda$ with an unrolling of a fixed number of iterations of the conjugate gradient algorithm, similar to [2]. Here $\tilde{R}$ is a trained neural network, and the scale parameter λ is also learned.

Efficiently finding a solution to the linear system (2.2) using an iterative method is challenging when the matrix $\mathbf{A}^\top\mathbf{A} + \mathbf{R}$ is ill-conditioned. This suggests that the Neumann network approach, which is derived from a Neumann series expansion of the system in (2.2), may benefit from preconditioning. To begin, consider a preconditioner for (2.2).

Starting from (2.2), for any $\lambda > 0$

$$(\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I})x^\star + (\mathbf{R} - \lambda\mathbf{I})x^\star = \mathbf{A}^\top\mathbf{y}. \tag{2.9}$$

Applying $\mathbf{T}_\lambda := (\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I})^{-1}$ to both sides and rearranging terms gives

$$(\mathbf{I} - \lambda\mathbf{T}_\lambda + \tilde{\mathbf{R}})x^\star = \mathbf{T}_\lambda\mathbf{A}^\top\mathbf{y}. \tag{2.10}$$

where $\tilde{\mathbf{R}} = \mathbf{T}_\lambda\mathbf{R}$. Following the same steps used to derive the Neumann network, the modified estimator is now

$$\widehat{\mathbf{x}}_{\mathrm{pc}}(\mathbf{y}) = \sum_{\mathrm{j}=0}^{\mathrm{B}} (\lambda\mathbf{T}_\lambda(\cdot) - \tilde{\mathrm{R}}(\cdot))^{\mathrm{j}}\mathbf{T}_\lambda\mathbf{A}^\top\mathbf{y} \tag{2.11}$$

which we call a *preconditioned Neumann network*. Here $\tilde{\mathrm{R}} = \tilde{\mathrm{R}}_\theta$ is a trainable mapping depending on parameters θ. We also treat $\lambda > 0$ as a trainable parameter when gradients with respect to λ are easily calculated (more on this below).

As shown in Figure 2.2, a preconditioned Neumann network has the same basic network architecture as the standard Neumann network, except the linear component $[\mathbf{I} - \eta\mathbf{A}^\top\mathbf{A}](\cdot)$ is replaced with $[\lambda\mathbf{T}_\lambda](\cdot)$ and the learned component $[-\eta\mathrm{R}](\cdot)$ is replaced with $[-\tilde{\mathrm{R}}](\cdot)$. The preconditioned Neumann network also has a different initialization, $\tilde{\boldsymbol{\beta}}^{(0)} = \mathbf{T}_\lambda\mathbf{A}^\top\mathbf{y}$, which is the solution to the Tikhonov regularized least squares problem $\min_{\mathbf{A}} \|\mathbf{A}\mathbf{A} - \mathbf{y}\|^2 + \lambda\|\mathbf{A}\|^2$. For many inverse problems in imaging, such as deblurring, this is much more accurate approximation to the ideal solution than the matrix transpose initialization $\tilde{\boldsymbol{\beta}}^{(0)} = \eta\mathbf{A}^\top\mathbf{y}$ of the standard Neumann network. Hence, we might expect that a preconditioned Neumann network could achieve higher quality solutions with fewer blocks B. Our experiments on deblurring of natural images (see Figure 2.9(b)) support this observation.

Applying $\mathbf{T}_\lambda(\cdot)$ may be computationally prohibitive for certain large-scale inverse problems in imaging, such as those arising in CT and MRI reconstruction. To address this issue, we adapt the approach

of [2] and replace all instances of $\mathbf{T}_\lambda(\cdot)$ in the preconditioned Neumann network by an unrolling of a fixed number of iterations of the conjugate gradient (CG) algorithm [26], which approximates the application of $\mathbf{T}_\lambda(\cdot)$. Unrolling CG does not require any additional trainable parameters, and back-propagation through the CG layers can be performed via automatic differentiation. This strategy has been shown to be effective for various large-scale MRI reconstruction problems [3, 43]. Incorporating a trainable λ parameter into this approach is simple since the derivatives of the end-to-end network $\widehat{\mathbf{x}}_{\mathrm{pc}}$ with respect to λ are also easily computed by automatic differentiation. In particular, we do not need $\mathbf{T}_\lambda$ to have an analytic expression in terms of λ in order to compute derivatives.

Finally, we note other preconditioned Neumann networks could be derived by replacing $\mathbf{I}$ with a general matrix $\mathbf{S}$ such that $\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{S}$ is positive definite, e.g., $\mathbf{S} = \mathbf{D}^\top\mathbf{D}$ where $\mathbf{D}$ is a discrete approximation of the image gradient. For simplicity, $\mathbf{S} = \mathbf{I}$ in further sections.

Patchwise Regularization

Exploiting the geometry of image patches is leveraged in a myriad of image processing tools, including nonlocal means [11], dictionary learning [37, 4], BM3D [14, 39], Gaussian mixture model priors [60, 51], and more. However, state-of-the-art methods for neural network-based image reconstruction do not leverage patch geometry.

In the presence of large training datasets, this is in general a reasonable approach. By including more of the image in the network's receptive field, the network can incorporate longer-range correlation structure between areas of an image, for more accurate reconstruction. However, in the case of small datasets, learning complex correlation structure is simply not possible; recall the earlier exponential sample complexity bounds on learning prior distributions. In the small training set case, we focus on learning a prior for image *patches* – small blocks of pixels, such as 8×8 or 16×16.

Some prior work has explored leveraging patch geometry. For instance, [44, 10] both train entirely on patches but on special forward models where the operator $\mathbf{A}$ can be decomposed across patches; they split the corrupted image into multiple patches and reconstruct those independently. In contrast, this paper describes a framework in which $\mathbf{A}$ may be an global operator, such as k-space measurements in MRI or radar returns, but in which the *regularizer* may be decomposed across patches. This is similar in spirit to the patch-based decomposition of regularizers in [17], but while that prior work assumed patches lie along a low-dimensional subspace, here we learn the patch geometry from training images using a neural network.

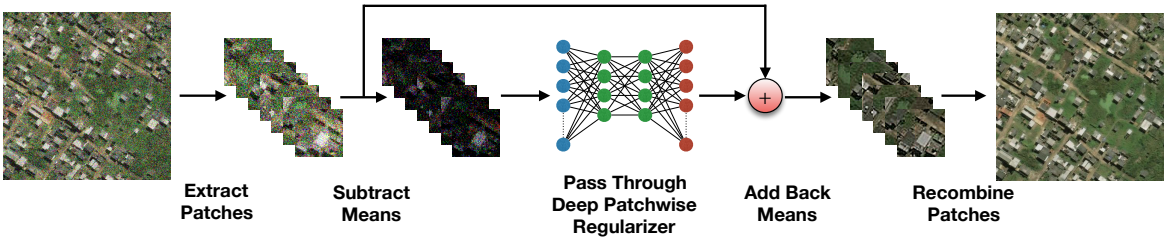


Figure 2.3: Visual representation of patchwise regularization using a deep network. The network separately processes N patches of dimension $p \times p \times 3$, where N is the number of (potentially overlapping) patches of size $p \times p$ that are used to represent the original image. By operating on small $p \times p$ patches, the regularizer learns from $N \times n_{samples}$ unique patches during training.

We utilize a flexible procedure in which a deep regularizer is applied in a patchwise manner.

Specifically, the regularizer operates by dividing the input image into overlapping patches, subtracting the mean from each patch (a standard preprocessing technique in patch-based methods [33]), and passing each mean-subtracted patch through the learned component (*e.g.*, neural network). The means from the original patches are added to the outputs of the patches, which are then recombined. Figure 2.3 is a graphical representation of the described procedure. Mathematically, this can be represented by the following expression:

$$\mathsf{R}(\mathbf{A}) = \mathrm{P}^{-1}\left(\left\{\tilde{\mathsf{R}}\left(\mathrm{P}(\boldsymbol{x}, \mathfrak{i}) - \overline{\mathrm{P}(\boldsymbol{x}, \mathfrak{i})}\right) + \overline{\mathrm{P}(\boldsymbol{x}, \mathfrak{i})}\right\}_{\mathfrak{i}=0}^{n}\right). \tag{2.12}$$

Here, $\mathrm{P}(\cdot, \mathfrak{i})$ is a patch extraction operator which outputs the $\mathfrak{i}$th patch of an image, and $\mathrm{P}^{-1}(\cdot)$ recombines a set of potentially overlapping patches, where overlapping pixels are averaged. In this case, $\tilde{\mathsf{R}}(\cdot)$ is a deep network operating on each patch individually. $\overline{\mathrm{P}(\boldsymbol{x}, \mathfrak{i})}$ denotes the average of an individual patch.

While here we propose a patch-based regularizer in the context of the Neumann network, patch-based networks may be trained alone for use in an iterative reconstruction algorithm like [47]. The former case is straightforward, as a single whole-image network can be replaced by the regularizer in Eq. 2.12. Training is simple, as gradients can flow along the patch-combination and patch-extraction operators. Training an offline patchwise regularizer typically requires learning a denoising neural network, which is trained using image patches instead of full images.

2.5 Theory

The Neumann network architecture proposed in the previous section is motivated by the Neumann series expansion of a (potentially nonlinear) operator R representing the gradient of a regularizer. Strictly speaking, this Neumann series expansion is valid (*i.e.*, corresponds to the solution of the equation (2.1)) only if R is linear. However, restricting R to to be linear severely limits the class of estimators that can be learned in our framework. In particular, if R is linear then the resulting learned estimator has to be linear, which is suboptimal for many data types.

In this section we show that a Neumann series approach is still mathematically justified for data belonging to a union of subspaces (UoS) model. The reasons for focusing on a UoS model are two-fold: First, UoS models are a natural generalization of linear subspace models and are used widely in many signal and image reconstruction problems, either as a deterministic model [19, 8, 35] or as a statistical model in the form of a Gaussian mixture model with low-rank covariances [46, 56, 27]. Second, we believe UoS models represent a reasonable trade-off between model complexity/expressiveness and analytic tractability, and provide some insight on the expected behavior of Neumann networks beyond the setting where R is linear.

To be precise, here we consider the class of *Neumann network estimators* $\widehat{\boldsymbol{x}}$ given in (2.7) that are specified by a (potentially nonlinear) mapping[2] $\mathsf{R} : \mathbb{R}^p \to \mathbb{R}^p$, step size η, and number of blocks B. We study two questions:

1. Can a Neumann network estimator be used to reconstruct images belonging to a UoS, and if so, what is an optimal choice of R?

[2]Here we do not assume R is a neural network with a particular architecture, but study the idealized case where R can represent any mapping from $\mathbb{R}^p$ to $\mathbb{R}^p$.

2. Can this R be learned using standard neural network architectures and training?

Our main result, given in Theorem 1, addresses the first question by showing there exists a Neumann network estimator with a *piecewise linear* R that gives arbitrarily small reconstruction error under mild assumptions on the subspaces and their interaction with the measurement operator. We study the second question empirically, and show that the learned R well-approximates the predicted optimal piecewise linear R in an idealized setting.

Images Belonging to a Single Subspace

Suppose the ground truth images belong to an r-dimensional subspace $S \subset \mathbb{R}^p$. Let $\mathbf{U} \in \mathbb{R}^{p \times r}$ be a matrix whose columns form an orthonormal basis for S. Assume $m \geqslant r$ and $\mathbf{AU} \in \mathbb{R}^{m \times r}$ is full rank. In other words, we assume there is no image in the subspace also in the nullspace of $\mathbf{A}$ besides the zero image[3]. Then given noise-free linear measurements of the form $y = \mathbf{A}x^\star$ of any data point $x^\star = \mathbf{U}w^\star \in S$ we can always recover $x^\star$ by applying the linear estimator

$$\widehat{x}_o(y) = \mathbf{U}(\mathbf{U}^\top \mathbf{A}^\top \mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top \mathbf{A}^\top y \tag{2.13}$$

since it is easy to check that $\widehat{x}_o(y) = x^\star$. In other words, there always exists a linear estimator that gives exact recovery of images belonging to the subspace from their noise-free linear measurements.

Our first result shows that there exists a linear Neumann network estimator of the form (2.7) (*i.e.*, a linear choice of R in (2.7)) such that for all points in the subspace the reconstruction error can be made arbitrarily small by choosing the step size η and block size B appropriately. For simplicity, we restrict ourselves to the case of noise-free measurements and where $\mathbf{A}$ has orthonormal rows.

Lemma 1. *Let $\mathbf{A} \in \mathbb{R}^{m \times p}$ be any measurement matrix with orthonormal rows, and let $S \subset \mathbb{R}^p$ be an r-dimensional subspace with orthonormal basis $\mathbf{U} \in \mathbb{R}^{r \times p}$. Suppose $m \geqslant r$ and $\mathbf{AU} \in \mathbb{R}^{m \times r}$ is full rank. Then for any $\eta \in (0, 1]$, the B-term Neumann network estimator $\widehat{x}$ with linear $R(\mathbf{A}) = \mathbf{R}\beta$ where $\mathbf{R} \in \mathbb{R}^{p \times p}$ is given by*

$$\mathbf{R} = -c_{\eta,B}\,(\mathbf{I} - \mathbf{A}^\top \mathbf{A})\mathbf{U}(\mathbf{U}^\top \mathbf{A}^\top \mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top \mathbf{A}^\top \mathbf{A} \tag{2.14}$$

for a constant $c_{\eta,B}$ depending only on η and B, satisfies the error bounds

$$\|\widehat{x}(\mathbf{A}x^\star) - x^\star\| \leqslant (1-\eta)^{B+1}\|\mathbf{A}x^\star\|. \tag{2.15}$$

for all $x^\star \in S$.

The proof of Lemma 1 is given in the Appendix of [22]. The main idea behind the proof is that with this choice of R the Neumann network terms $\tilde{\beta}^{(j)}$ simplify to

$$\tilde{\beta}^{(j)} = a_j \mathbf{A}^\top \mathbf{A}x^\star + b_j(\mathbf{I} - \mathbf{A}^\top \mathbf{A})x^\star \tag{2.16}$$

for some constants a_j and b_j that satisfy $\sum_j a_j \approx 1$ and $\sum_j b_j \approx 1$. Hence, we have $\widehat{x}(y) = \sum_{j=0}^{B} \tilde{\beta}^{(j)} \approx \mathbf{A}^\top \mathbf{A}x^\star + (\mathbf{I} - \mathbf{A}^\top \mathbf{A})x^\star = x^\star$.

[3]This assumption is met in many practical settings. For example, in an inpainting setting it is equivalent to assuming there is no image in the subspace having support contained entirely within the inpainting region. Likewise, in compressed sensing by subsampling DFT coefficients, it is equivalent to assuming there is no image in the subspace bandlimited to the set of unobserved DFT coefficients. Both of these assumptions are reasonable for subspaces spanned by natural images.

Images Belonging to a Union of Subspaces

Now we suppose that the images belong to a UoS $\cup_{k=1}^{K} \mathcal{S}_k \subset \mathbb{R}^p$ where, for simplicity, we assume each subspace $\mathcal{S}_k$ has dimension r. For all $k = 1, ..., K$ we let $\mathbf{U}_k \in \mathbb{R}^{p \times r}$ denote a matrix whose columns form an orthonormal basis for $\mathcal{S}_k$. Again, we assume $m \geqslant r$ and $\mathbf{A}\mathbf{U}_k \in \mathbb{R}^{m \times r}$ is full rank for every $k = 1, ..., K$. In other words, we assume there is no image in the UoS also in the nullspace of $\mathbf{A}$ besides the zero image.

Let $\mathbf{y} = \mathbf{A}x^\star$ be the measurements of any point $x^\star$ belonging to the UoS. If we know $x^\star$ belongs to the kth subspace, *i.e.*, $x^\star = \mathbf{U}_k w^\star$ for some $w^\star \in \mathbb{R}^p$, then similar to the single subspace case, we can apply the estimator

$$\widehat{\mathbf{x}}_o(\mathbf{y}; k) := \mathbf{U}_k(\mathbf{U}_k^\top \mathbf{A}^\top \mathbf{A}\mathbf{U}_k)^{-1}\mathbf{U}_k^\top \mathbf{A}^\top \mathbf{y} \tag{2.17}$$

since it is easy to see that $x^\star = \widehat{\mathbf{x}}_o(\mathbf{y}; k)$. We call $\widehat{\mathbf{x}}_o(\mathbf{y}; k)$ the *oracle estimator*, since it assumes knowledge of the subspace index k to which the image belongs.

We show that, under appropriate conditions on the subspaces and the measurement operator, there is a piecewise linear choice of Neumann network estimator (*i.e.*, an estimator of the form (2.7) with R piecewise linear) that recovers any image belonging to the UoS from its noise-free measurements with arbitrarily small reconstruction error. In other words, there is a Neumann network estimator that well-approximates the oracle estimator.

Specifically, we consider a piecewise linear function R^* of the form

$$R^*(\mathbf{A}) = \begin{cases} \mathbf{R}_1\mathbf{A} & \text{if } \mathbf{A} \in \mathcal{C}_1 \\ \vdots & \vdots \\ \mathbf{R}_K\mathbf{A} & \text{if } \mathbf{A} \in \mathcal{C}_K \end{cases} \tag{2.18}$$

where each $\mathbf{R}_k$ is a $p \times p$ matrix, and the regions $\mathcal{C}_k$ are disjoint and whose union is all of $\mathbb{R}^p$. The main idea behind our analysis is this: If the ground truth point $x^\star$ belongs the kth subspace, then we prove that the Neumann series summands $\tilde{\boldsymbol{\beta}}^{(0)}, \tilde{\boldsymbol{\beta}}^{(1)}, ..., \tilde{\boldsymbol{\beta}}^{(B)}$ all lie in the same region $\mathcal{C}_k$. This means that the same $\mathbf{R}_k$ is used in computing each summand, so we can write

$$\widehat{\mathbf{x}}(\mathbf{y}) = \sum_{j=0}^{B} \tilde{\boldsymbol{\beta}}^{(j)} = \sum_{j=0}^{B} (\mathbf{I} - \eta\mathbf{A}^\top\mathbf{A} - \eta\mathbf{R}_k)^j(\eta\mathbf{A}^\top\mathbf{y}).$$

Choosing $\mathbf{R}_k$ to have the same form as in the single subspace case (see Lemma 1), we then will have $x^\star = \widehat{\mathbf{x}}_o(\mathbf{y}, k) \approx \widehat{\mathbf{x}}(\mathbf{y})$.

To be exact, we specify $\mathbf{R}_k$ and $\mathcal{C}_k$ as follows. Similar to Lemma 1, we choose

$$\mathbf{R}_k = -c_{\eta,B}\,(\mathbf{I} - \mathbf{A}^\top\mathbf{A})\mathbf{U}_k(\mathbf{U}_k^\top\mathbf{A}^\top\mathbf{A}\mathbf{U}_k)^{-1}\mathbf{U}_k^\top\mathbf{A}^\top\mathbf{A},$$

for all $k = 1, ..., K$, where $c_{\eta,B} > 0$ is a constant depending only on η and B. We also define the corresponding region $\mathcal{C}_k$ as

$$\mathcal{C}_k = \{\mathbf{A} \in \mathbb{R}^p : d_{\mathbf{A},k}(\mathbf{A}) < d_{\mathbf{A},\ell}(\mathbf{A}) \text{ for all } \ell \neq k\}$$

where $d_{\mathbf{A},k}(\mathbf{A}) := \|(\mathbf{I} - \mathbf{A}\mathbf{U}_k(\mathbf{A}\mathbf{U}_k)^+)\mathbf{A}\mathbf{A}\|$ is the distance between the vector $\mathbf{A}\mathbf{A}$ and the subspace $\text{span}(\mathbf{A}\mathbf{U}_k)$. In other words, $\mathcal{C}_k$ is the set of all points whose distance to the kth subspace is smaller

than the distance to all other subspaces, as measured by the functions $d_{\mathbf{A},\ell}$ for all $\ell = 1, ..., K$.

We now state our main theorem:

Theorem 1. *Let* $\mathbf{A} \in \mathbb{R}^{m \times p}$ *be any measurement matrix with orthonormal rows, and for all* $k = 1, ..., K$ *let* $\mathbf{U}_k \in \mathbb{R}^{p \times r}$ *be an orthonormal basis for the* k*th subspace* $\mathcal{S}_k$ *with* $\dim \operatorname{span}(\mathbf{A}\mathbf{U}_k) = r$. *Suppose span*$(\mathbf{A}\mathbf{U}_k) \cap$ *span*$(\mathbf{A}\mathbf{U}_\ell) = \{0\}$ *for all* $k \neq \ell$. *Then the Neumann network estimator* $\widehat{\mathbf{x}}$ *with step size* $\eta \in (0, 1)$ *and piecewise linear* $R = R^*$ *satisfies*

$$\|\widehat{\mathbf{x}}(\mathbf{A}x^\star) - x^\star\| \leqslant (1-\eta)^{B+1}\|\mathbf{A}x^\star\|, \tag{2.19}$$

for all $x^\star \in \cup_{k=1}^K \mathcal{S}_k$.

The condition $\operatorname{span}(\mathbf{A}\mathbf{U}_k) \cap \operatorname{span}(\mathbf{A}\mathbf{U}_\ell) = \{0\}$ for all $\ell \neq k$, appearing in Theorem 1 is not overly restrictive if we take into account the statistics of natural images. For instance, this condition holds for a generic union of r-dimensional subspaces provided $m \geqslant 2r$, regardless of the number of subspaces in the union[4]. Moreover, based on results in compressive sensing using low-rank Gaussian mixture models [45, 46], we conjecture this condition can be weakened under appropriate assumptions on $\mathbf{A}$ and appropriate modification of R^*, but we do not pursue this refinement here.

Theorem 1 shows there exists a Neumann network estimator with a certain choice of R^* that well-approximates an oracle estimator for images belonging to a union of subspaces. In principle, since R^* is piecewise linear with a finite number of regions, it is realizable as a sufficiently deep neural network with ReLU activations [5]. However, this does not necessarily mean that a Neumann network estimator with R given by a ReLU network when trained on images belonging to a union of subspaces will recover $R = R^*$ specified in Theorem 1. For example, there may be other R' that yield similar training loss as R^*, or the learned component may be under-parameterized (e.g., not enough layers) in such a way that it cannot well-approximate R^*. Nevertheless, one would hope that given sufficient training data and a sufficiently expressive network architecture for the learned component, it may be possible to learn a good approximation to R^* as specified in Theorem 1. Below we illustrate that this is indeed the case for a Neumann network trained on images belonging to synthetic union of subspaces.

Finally, using the equivalence of Neumann networks and unrolled gradient descent networks estimators in the case where the learned component R is linear (see Sec. 2.3), we show that an unrolled gradient descent network as defined in (**??**) with the same piecewise linear $R = R^*$ as defined in (2.18) satisfies the error bounds as in Theorem 1:

Corollary 1. *Under the same assumptions as Theorem 1, the unrolled gradient descent estimator* $\widehat{\mathbf{x}}'(\mathbf{y}) = \boldsymbol{\beta}^{(B)}$ *with step size* $\eta \in (0, 1)$ *and* $R = R^*$ *as defined in* (2.18) *satisfies*

$$\|\widehat{\mathbf{x}}'(\mathbf{A}x^\star) - x^\star\| \leqslant (1-\eta)^{B+1}\|\mathbf{A}x^\star\|, \tag{2.20}$$

for all $x^\star \in \cup_{k=1}^K \mathcal{S}_k$,

Corollary 1 shows that the equivalence between unrolled gradient descent estimators and Neumann network estimators observed in the case where R is linear carries over to the special case where $R = R^*$ is piecewise linear and the networks are evaluated on linear measurements of points belonging to the union of subspaces.

[4]This is because if $\operatorname{span}(\mathbf{U}_k)$, $k = 1, ..., K$, are generic r-dimensional subspaces in $\mathbb{R}^p$, then $\mathcal{V}_k = \operatorname{span}(\mathbf{A}\mathbf{U}_k)$, $k = 1, ..., K$, are generic r-dimensional subspaces in $\mathbb{R}^m$. Since two generic subspaces are linearly independent provided the sum of their dimensions does not exceed the ambient dimension, we see that $\mathcal{V}_k$ and $\mathcal{V}_\ell$, $k \neq \ell$ collectively span a $2r$-dimensional subspace, which is only possible if their intersection is trivial.

Empirical Validation

Here we illustrate empirically that the optimal R^* predicted by Theorem 1 is well-approximated by training a Neumann network for a 1-D inpainting task on synthetic UoS data. We generate random training data belonging to a union of three 3-dimensional subspaces in $\mathbb{R}^{10}$, and train a Neumann network to inpaint five missing coordinates (*i.e.*, $\mathbf{A} \in \mathbb{R}^{5\times 10}$ restricts a vector to coordinates 1–5). We parameterize the learned component R of the Neumann Network as a 7-layer fully connected neural network with ReLU activations, which is trained by minimizing the mean squared error of the reconstruction over the training set using stochastic gradient descent (more details on this experiment can be found in the Supplementary Materials).

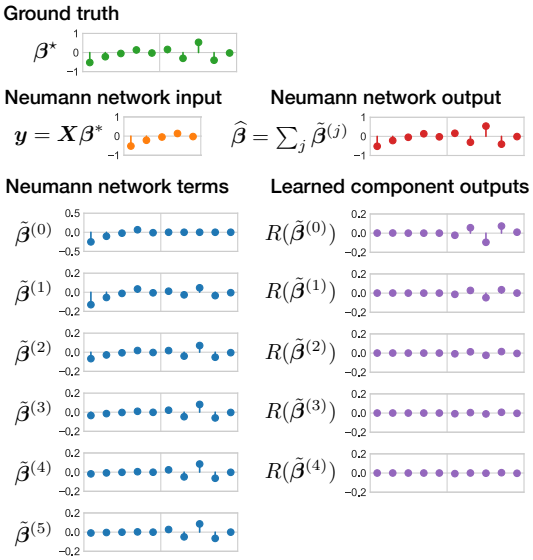


Figure 2.4: Example output of Neumann network trained on synthetic union of subspaces data for a 1-D inpainting task. Here a vector $x^\star \in \mathbb{R}^{10}$ is drawn from one of the subspaces, and its measurements $\mathbf{y} = \mathbf{A}\mathbf{A}^*$ (restriction to first five coordinates) are input into the Neumann network, which faithfully restores the missing coordinates. The output $\widehat{\mathbf{x}}$ of the Neumann network is a sum of terms $\tilde{\boldsymbol{\beta}}^{(j)}$ (shown in bottom left). As predicted by Theorem 1, the terms $\tilde{\boldsymbol{\beta}}^{(j)}$ are weighted linear combinations the projections of $x^\star$ onto the observed and unobserved coordinates. Also as predicted by Theorem 1, the outputs of the learned component $R(\tilde{\boldsymbol{\beta}}^{(j)})$ (shown in bottom right) are zero in the observed coordinates and scaled projections of $x^\star$ in the unobserved coordinates.

Figure 2.4 illustrates the output of the trained Neumann network for one specific input, including the outputs from the intermediate Neumann network terms $\tilde{\boldsymbol{\beta}}^{(j)}$ and the learned component outputs $R(\tilde{\boldsymbol{\beta}}^{(j)})$. As predicted by Theorem 1, the Neumann network terms $\tilde{\boldsymbol{\beta}}^{(j)}$ have the form $a_j\mathbf{A}^\top\mathbf{A}x^\star + b_j(\mathbf{I} - \mathbf{A}^\top\mathbf{A})x^\star$ for some constants a_j and b_j. Also, the outputs of the learned component $R(\tilde{\boldsymbol{\beta}}^{(j)})$ all lie in the null space of $\mathbf{A}$, *i.e.*, are vectors supported on coordinates $6-10$.

Figure 2.5 displays the results of a quantitative experiment to assess whether the learned component R is piecewise linear as predicted by Theorem 1. First, we test whether the learned R is approximately linear when restricted to inputs belonging to each subspace, *i.e.*, we test whether $R(x_1^\star + x_2^\star) \approx R(x_1^\star) + R(x_2^\star)$, for all $x_1^\star, x_2^\star$ belonging to the same subspace. As baselines we compare to the case where $x_1^\star$ and $x_2^\star$ belong to different subspaces, and the case where $x_1^\star$ and $x_2^\star$ are Gaussian random vectors. In Figure 2.5 we display a boxplot of the relative error $\|R(x_1^\star + x_2^\star) - R(x_1^\star) - R(x_2^\star)\|/\gamma$ of 1024 randomly generated $x_1^\star, x_2^\star$, which are normalized such that $\|x_1^\star\| = \|x_2^\star\| = \gamma$. Here we set normalization to $\gamma = 0.25$, though similar results were obtained for $\gamma \in [0.1, 0.5]$ (not shown). As predicted, the relative

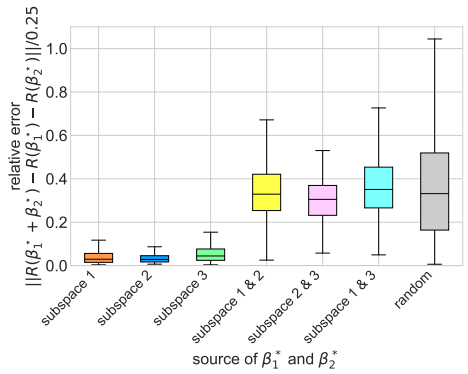


Figure 2.5: Piecewise linearity test. We measure how linear the learned R is when evaluated at two vectors drawn from the same subspace, from two different subspaces, or from two random Gaussian vectors. The plot illustrates that the learned R only behaves like a linear operator when the vectors belong the same subspace (*i.e.*, the relative error is small), which indicates the learned R is approximately piecewise linear, as predicted by Theorem 1.

error concentrates near zero in the case where $x_1^\star, x_2^\star$ belong to the same subspace, and is otherwise large, indicating the learned R is indeed approximately piecewise linear as predicted by Theorem 1.

In the Supplementary Materials we provide more empirical evidence that the R learned in this experiment closely approximates the ideal R^* predicted by Theorem 1. Specifically, we demonstrate that the learned component behaves as expected on inputs restricted to the column space and row space of the forward model $\mathbf{A}$. These experiments verify that, at least for this 1-D inpainting task on synthetic data, the ideal piecewise linear R^* predicted by Theorem 1 is well-approximated with standard neural network architectures and training.

A more systematic study involving different forward models $\mathbf{A}$ and different network architectures is needed to determine whether the ideal R^* identified in Theorem 1 is learnable more generally for large-scale imaging data and using practical architectures like convolutional neural networks. Also, our results do not address the case of noisy measurements or forward models with non-orthogonal rows, which are important considerations for many inverse problems. We leave these as open questions for future work.

Finally, while our focus in this section was on UoS models, our analysis does not rule out the applicability of Neumann networks to other non-linear models. Indeed, in the next section we show empirically that Neumann networks perform well on a variety of linear inverse problems when trained on realistic image datasets that are unlikely to be perfectly captured by a low-dimensional UoS model.

2.6 Experiments

We begin this section with a comparison of Neumann networks against other methods of solving several different inverse problems with learned components. After that, we investigate the effect of larger and smaller training sets on all methods, demonstrating that Neumann networks are robust to small training set sizes. We follow these experiments with an illustration of the effects of incorporating preconditioning into the Neumann network for deblurring, which is shown to give a gain of several dBs of PSNR, permitting smaller networks and allowing for faster training and implementation. We follow with an investigation into an MRI reconstruction problem to demonstrate the proposed methods in a large-scale setting. Finally, we explore the optimization landscape of Neumann networks relative to

unrolled gradient descent, illustrating that Neumann networks have smoother loss landscapes than unrolled Gradient Descent, while also generally achieving lower test set errors.

Datasets and Comparison Methods

In our experiments, we consider three different small-scale training sets: CIFAR10 [31], CelebA [36], and STL10 [13], and one larger-scale undersampled MRI reconstruction task.

The CIFAR10 dataset is a machine learning standard, consisting of real-world images of both man-made and natural scenes [31]. The dataset has been resized to be 32×32 pixels.

We use a subset of the aligned Celebrity Faces With Attributes (CelebA) dataset [36]. The CelebA dataset consists of human faces at a variety of angles, and the subset that is used here has been aligned so that all faces lie in the center of the image.

The STL10 dataset [13] is a curated subset of the ImageNet dataset, and was originally intended to be used with semisupervised learning problems. In our experiments, we have resized all CelebaA and STL10 images to be 64×64 pixels.

We select a subset of images of size 30,000 uniformly from each individual dataset to be used for training in the results presented below.

We use these training sets in seven different inverse problems in imaging: Block inpainting, deblurring, deblurring with additive noise ϵ of variance 0.01, superresolution (SR4 and SR10) with two different upsampling levels (4x and 10x across the entire image, respectively), and compressive sensing (CS2 and CS8) with two separate levels of compression (2x and 8x, respectively). The compressed sensing design matrices are random Gaussian matrices.

We compare Neumann networks (**NN**) and preconditioned Neumann networks (**PNN**) with four methods which can be applied to solve a variety of inverse problems:

We first compare to the gradient descent network (**GDN**), an *unrolled optimization* algorithm that is trained end-to-end. While theoretical properties GDN and NN are examined in Section 2.5, we hope to compare the qualitative and quantitative differences between the two architectures. For a fair comparison, we use an identical architecture for the nonlinear learned component in the gradient descent network and the nonlinear learned component in the Neumann network.

We also compare to MOdel-based reconstruction with Deep Learned priors (**MoDL**) [2], another unrolled algorithm containing a novel data-consistency step that performs conjugate gradient iterations inside the unrolled algorithm. MoDL is also trained end-to-end, and also shares learned parameters between the learned algorithm. In our main experimental section, we use an identical architecture for the learned component of MoDL as is used in GDN and NN.

Trainable Nonlinear Reaction Diffusion (**TNRD**) [12] is an unrolled optimization algorithm that closely resembles GDN, but with a specific, novel architecture for the learned component motivated by insights from diffusion methods for inverse problems. The learned components in each block consist of a single filter, followed by a learned nonlinearity, and then the transpose of the single filter is applied. Weights are *not* shared across blocks in TNRD.

The residual autoencoder (**ResAuto**), first proposed in [38], is an *agnostic* method. In Section 1.3, we discussed agnostic methods that learn a mapping from $\mathbf{y}$ to $\mathbf{A}$, but in these experiments, we consider a variant of an agnostic learner that learns a mapping from $\mathbf{A}^\top \mathbf{y}$ to $\widehat{\mathbf{x}}$ but does not otherwise use $\mathbf{A}$. Specifically, we construct a 12-layer convolutional-deconvolutional residual neural network

(almost twice as many layers as the network used in the Neumann network), with a channel-wise fully connected layer.

Compressed Sensing using Generative Models, (**CSGM**) [9] is a *decoupled* method which first trains a generative model for the data. After training the generative model, arbitrary inverse problems can be solved by finding the image in the range of the generator which is closest to the distorted image. As in the setup of [9], in our experiments we train three generative networks, one for each dataset.

Our final method does not incorporate training data at all into the solution of the inverse problem. We reconstruct using total-variation regularized least squares (**TV**). We minimize our objective using the algorithm of [54], with hyperparameters chosen via cross-validation over a held-out validation set for each dataset and inverse problem.

Training and Implementation

Given training pairs $\{(\mathbf{A}_i, \mathbf{y}_i)\}_{i=1}^N$, and assuming the learned component R inside the Neumann network depends smoothly on a set of parameters θ, *i.e.,* the partial derivatives $\partial_\theta R(\mathbf{A}; \theta)$ exist, we train a Neumann network $\widehat{\mathbf{x}}$ by minimizing the empirical risk $\mathcal{L}(\theta) = \sum_{i=1}^N \|\widehat{\mathbf{x}}(\mathbf{y}_i; \theta) - \mathbf{A}_i\|^2$.

In the Supplemental Materials we derive the backpropagation gradients $\partial_\theta \mathcal{L}(\theta)$ in the case where $\widehat{\mathbf{x}}$ is a Neumann network or a gradient descent network.

The learned components of NN, GDN, and MoDL have identical architectures: a 7-layer convolutional-deconvolutional neural network with a single channel-wise fully-connected layer [42], inspired by architectural choices in [16, 38, 30].

For NN, GDN, MoDL, and TNRD we used architectures with $B = 6$ blocks. The learned component is fixed per network, *i.e.,* the learned component in the first block has identical weights to the learned component in all other blocks in any given method and inverse problem, except in TNRD. While using a larger B is possible, we found that increasing B beyond 8 led to greatly increased sensitivity to SGD step size schedule choices. This phenomenon can be observed in Section 2.6. Anecdotally, we find that it is more difficult to choose SGD step sizes for GDN than for NN even for small B, and this difficulty became problematic for B greater than 6.

The ResAuto architecture imitates the architecture of [38], an approach that highly resembles the U-Net [48], but adjusted for good performance on inverse problems like superresolution, deblurring, and inpainting. Superficially, the architecture resembles an expanded version of the previously-described learned component, with 12 convolution or deconvolution layers instead of 7.

Runnable code can be found online at: github.com/dgilton/neumann_networks_code

Small-scale Experiments

In this section, a variety of methods are used to solve the previously-described inverse problems on three datasets. First, a quantitative comparison in terms of PSNR of the previously-outlined approaches on a variety of datasets and inverse problems is described in Table 2.1.

We observe that NN and GDN are competitive across all inverse problems and datasets. State-of-the-art methods like MoDL and TNRD perform quite well across all datasets, but the differences in architecture between PNN and MoDL appear to give an edge to PNN, which we hypothesize is an effect of our previously-highlighted skip connections. All methods that incorporate the forward model into the training and reconstruction process perform competitively in our small-scale experiments.

CSGM appears to suffer because of the lack of training data across all experiments. CSGM must learn the manifold associated with each dataset before being able to produce accurate reconstructions, which in our relatively sample-limited setting appears not to happen. See Figure 2.8 or the Supplement for examples of images produced by CSGM. While TV reconstructions are reasonably accurate across problems, they are not as accurate as learned approaches, especially in inpainting and compressed sensing.

		Inpaint	Deblur	Deblur+ϵ	CS2	CS8	SR4	SR10
CIFAR10	NN	28.20	36.55	29.43	33.83	**25.15**	24.48	**23.09**
	PNN	28.40	**37.83**	**30.47**	33.75	23.43	**26.06**	21.79
	GDN	27.76	31.25	29.02	**34.99**	25.00	24.49	20.47
	MoDL	28.18	34.89	29.72	33.47	23.72	24.54	21.90
	TNRD	27.87	34.84	29.70	32.74	25.11	23.84	21.99
	ResAuto	**29.05**	31.04	25.24	18.51	9.29	24.84	21.92
	CSGM	17.88	15.20	14.61	17.99	19.33	16.87	16.66
	TV	25.90	27.57	26.64	25.41	20.68	24.71	20.68
CelebA	NN	**31.06**	31.01	30.43	**35.12**	**28.38**	27.31	23.57
	PNN	30.45	**33.79**	**30.89**	32.61	26.41	**28.70**	23.74
	GDN	30.99	30.19	29.27	34.93	28.33	27.14	23.46
	MoDL	30.75	30.80	29.59	30.22	25.84	26.42	24.12
	TNRD	30.21	29.92	29.79	33.89	28.19	25.75	22.73
	ResAuto	29.66	25.65	25.29	19.41	9.16	25.62	**24.92**
	CSGM	17.75	15.68	15.30	17.99	18.21	18.11	17.88
	TV	24.07	30.96	26.24	25.91	23.01	26.83	20.70
STL10	NN	27.47	29.43	26.12	**31.98**	**26.65**	24.88	21.80
	PNN	28.00	**30.66**	**27.21**	31.40	23.43	**25.95**	**22.19**
	GDN	**28.07**	30.19	25.61	31.11	26.19	24.88	21.46
	MoDL	28.03	29.42	26.06	27.29	23.16	24.67	16.88
	TNRD	27.88	29.33	26.32	31.05	25.38	24.55	21.21
	ResAuto	27.28	25.42	25.13	19.48	9.30	24.12	21.13
	CSGM	16.50	14.04	15.59	16.67	16.39	16.58	16.47
	TV	26.29	29.96	26.85	24.82	22.04	26.37	20.12

Table 2.1: PSNR comparison for the CIFAR, CelebA, and STL10 datasets respectively. Values reported are the median across a test set of size 256.

The residual autoencoder in particular has excellent performance on certain problems like inpainting and superresolution, but is not competitive for compressed sensing and deblurring. Recall the motivation for the residual autoencoder: the closer $\mathbf{A}^\top \mathbf{y}$ is to the ground truth $x^\star$, the simpler the residual $x^\star - \mathbf{A}^\top \mathbf{y}$ that the network must learn. With this in mind, it seems reasonable that the residual autoencoder should perform well on small-scale downsampling, inpainting, and deblurring, but would fail to generate high-quality reconstructions for compressed sensing or heavy downsampling where $\mathbf{A}^\top \mathbf{y}$ is likely to be a poor approximation of $x^\star$.

The difference in performance between PNN and NN in Table 2.1 can provide some insight regarding the usage of various architectures. First, although PNN performs well for 4x superresolution, deblurring, and deblurring with noise, preconditioning is not a universal solution: inpainting and compressed sensing are perfectly conditioned and preconditioning appears to worsen performance. We see similar effects with MoDL, which performs better than NN or GDN on certain problems, but suffers especially in compressed sensing. These results further emphasize that consideration of the specific forward model at hand should be an important element of designing learned inverse problem solvers.

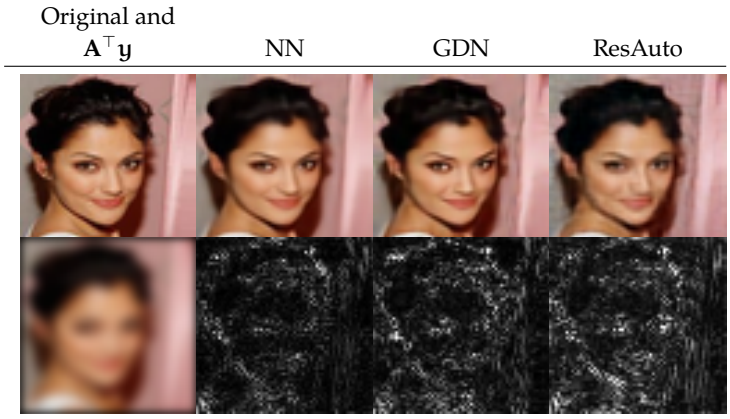


Figure 2.6: Reconstruction comparison on the CelebA dataset for the deblur plus noise problem. While the Neumann networks (NN) and gradient descent networks (GDN) perform well, the differences are most apparent the residual images in the second row, especially in the background reconstruction. Residuals are formed by displaying the norm across color channels of the error at each pixel, scaled by a factor of 6.

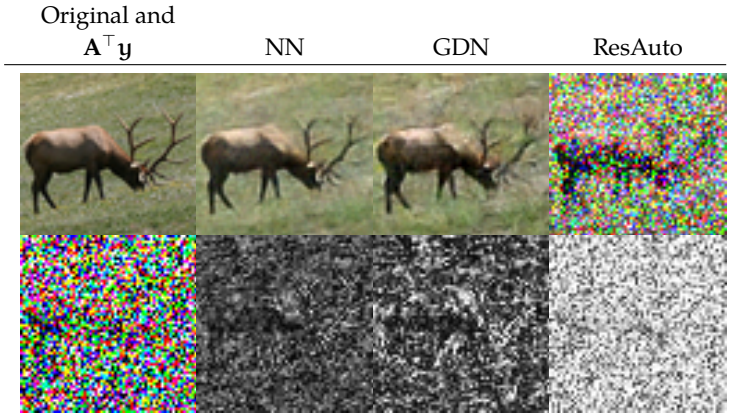


Figure 2.7: 8x compressed sensing reconstruction comparison on STL10. ResAuto fails to invert the compressed sensing problem adequately. The Gradient Descent network (GDN) reconstructs accurately but generates more artifacts than the Neumann network (NN).

Figures 2.6 and 2.7 demonstrate more qualitative and quantitative detail in some examples from several different inverse problems and all three datasets. In these figures the residuals are shown for illustrative purposes: the residuals are formed by displaying the scaled pixelwise norm across color channels of the difference $x^\star - \widehat{x}$ where $x^\star$ is the true image, and $\widehat{x}$ the estimate, scaled by a factor of 6. Magnitudes are clipped to be less than or equal to 1.

Effect of Sample Size

In section 1.3 we hypothesized that incorporating information about the forward operator would have implications for the sample sizes required to achieve particular error rates.

A coarse comparison of the presented learning-based methods at different sample sizes is provided in Figure **??**. We observe that while all methods suffer a decrease in PSNR at low sample sizes, the Neumann network has the highest-quality reconstructions at only 2,000 images, and also enjoys the largest increase of performance when going from 2,000 to 30,000 training images. Gradient descent

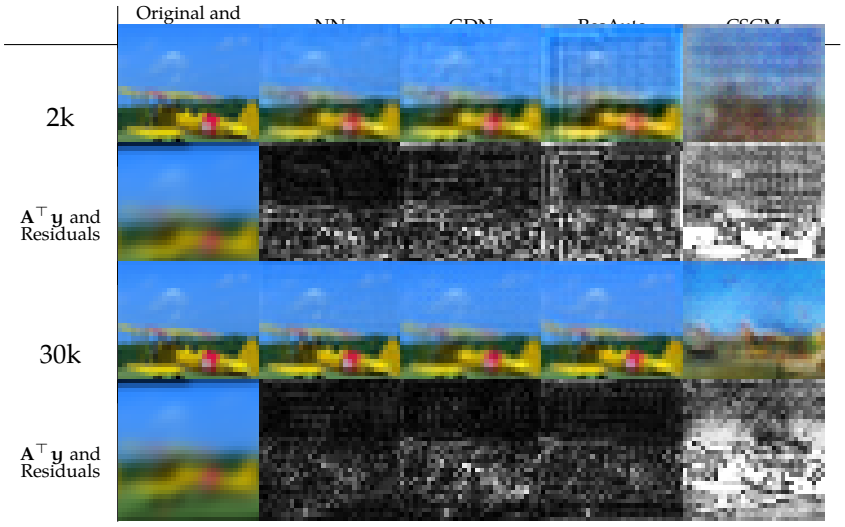


Figure 2.8: A qualitative comparison of the reconstructions produced for the deblurring problem on a single image at two different training set sizes, along with the associated residual images. Residual images are scaled by a factor of 6.

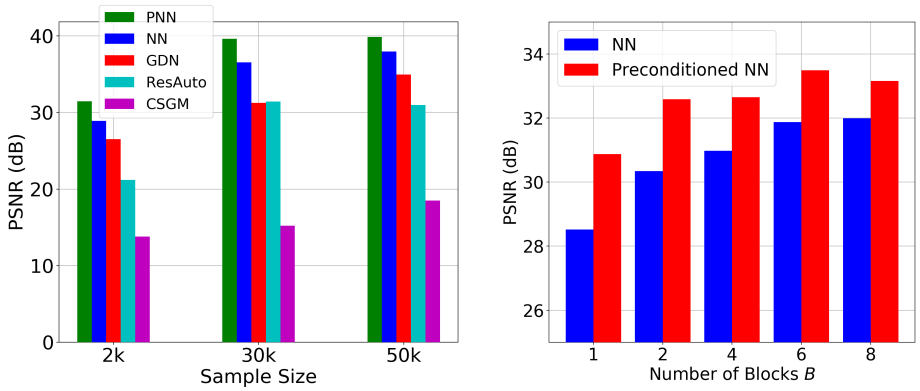


Figure 2.9: Performance comparisons. (a) Median PSNR of methods trained with different sample sizes of 2,000, 30,000, and 50,000. Neumann networks (NN) and Preconditioned Neumann networks (PNN) scale very well with training set size, with smaller marginal gains as training sizes increase. All PSNR values are for the CIFAR-10 dataset, and the inverse problem used is the previously described deblurring problem. (b) PSNR (dB) for the standard and preconditioned NN. The inverse problem in this case is deblurring with a Gaussian kernel of size 5×5 and variance $\sigma = 5.0$.

network performs well even at very low sample sizes, but artifacts are present in reconstructions at low sample sizes, visible in Figure 2.8.

Methods that do not incorporate the forward model, like ResAuto and CSGM, perform poorly in the low-sample regime, as discussed in Section 1.3. While ResAuto performs competitively at 30k iterations, there is little change between image qualities produced at these sample sizes, and even a very slight decrease in performance. CSGM improves significantly with increasing samples, but does not produce high-quality reconstructions on this inverse problem.

Effect of Preconditioning

Figure **??** illustrates the effect of preconditioning on the performance of the Neumann network with different numbers of blocks B on a deblurring task. While the original Neumann network does not surpass 32 dB PSNR with 8 blocks, the preconditioned Neumann network surpasses the original with only $B = 2$, and continues to improve as the number of blocks increases. Example images are included in the supplementary materials.

The forward problem in this case is Gaussian deblurring with $\sigma = 5.0$ and a blur kernel of size 5×5. The corresponding $\mathbf{A}$ is very poorly conditioned, and a λ of 0.01 is used in the preconditioning matrix $(\mathbf{A}^\top \mathbf{A} + \lambda \mathbf{I})^{-1}$.

Depending on the structure of $\mathbf{A}$ and how easily $(\mathbf{A}^\top \mathbf{A} + \lambda \mathbf{I})^{-1}$ can be computed, preconditioning can be computationally costly, but it appears to permit fewer Neumann network blocks for comparable performance. Since the primary resource bottleneck for training the Neumann network end-to-end is memory, fewer blocks permits faster training, or alternately, allows implementations to achieve higher performance than would otherwise be possible with fixed computational resources.

Effect of Patchwise Reglarization

In this section, we illustrate the effects of learned patch-based regularization on sample complexity, and follow these results by demonstrating that systems with local regularization are able to generalize well even when trained with a single image.

In all experiments we train and test using subsets of the SpaceNet dataset [53]. We use the AOI1 subset of images, which consists of aerial 3-band imagery of a 2544 square kilometer region of Rio de Janeiro captured by a WorldView-2 satellite system. Training images have a resolution of 50 cm, and are 416x416 pixels in size. The forward model in this case is a Gaussian blur with kernel size 9×9 and variance parameter $\sigma^2 = 2$, where noise is added after blur with standard deviation $\sigma = 0.03$. Some fraction of the dataset contains images with missing regions. We removed all such images by hand, and selected the training set from the remaining images. All numerical results shown in this section use a single held-out test set.

RED, by contrast, leverages a pretrained learned denoiser, which we denote $\hat{\mathsf{R}}(\cdot)$. The RED estimator minimizes a particular energy functional:

$$\hat{\mathbf{A}}_{\mathrm{RED}} = \arg\min_{\mathbf{A}} \|\mathbf{y} - \mathbf{A}\mathbf{A}\|_2^2 + \frac{\lambda}{2}\mathbf{A}^\top(\mathbf{A} - \hat{\mathsf{R}}(\mathbf{A})) \tag{2.21}$$

In our experiments, we use ADMM to minimize (3.7).

We use FFDNet [58] as a denoiser for RED. FFDNet is a denoising convolutional network that is designed to work with spatially-variable noise levels. We train the FFDNet by learning to denoise image patches, whose size depends on the experiment. At test time, the whole image is regularized by combining patchwise estimates in the manner illustrated by Fig. 2.3. The learned element of the Neumann network is a two-block residual network with identical structure to the residual learned component used in [40]. Other results in this chapter utilize a network based on a residual autoencoder that resembled the U-Net [48], but memory restrictions for the full-image regularizers necessitated a more lightweight network. We chose the learned component of [40] because of its empirical performance as well as prior use in an unrolled iterative algorithm.

We use both methods to demonstrate that the improvements seen below are a result of the improved sample complexity permitted by regularizing locally.

Sample Complexity

In this section, we explore our previous hypothesis that a deep regularizer that operates on small patches will have better performance in the small-sample case than a comparable regularizer that operates on larger patch sizes.

		8×8	16×16	64×64	Full Image
RED	100	30.01 ± 1.14	29.46 ± 1.16	29.33 ± 1.07	29.32 ± 1.18
	500	31.33 ± 1.22	30.19 ± 1.10	30.96 ± 1.27	29.51 ± 1.29
	1000	33.47 ± 1.26	31.47 ± 1.26	31.12 ± 1.21	30.03 ± 1.32
NN	100	32.64 ± 1.26	31.22 ± 1.35	28.88 ± 1.14	27.08 ± 1.00
	500	32.87 ± 1.24	31.93 ± 1.46	31.09 ± 1.34	27.81 ± 1.01
	1000	32.90 ± 1.25	32.20 ± 1.44	32.19 ± 1.44	29.85 ± 1.17

Table 2.2: PSNR (dB) comparison for deblurring across several training set sizes (*i.e.,* number of training images) with different regularizer patch sizes. Results are the mean plus and minus the standard deviation of PSNR over a test set of size 64.

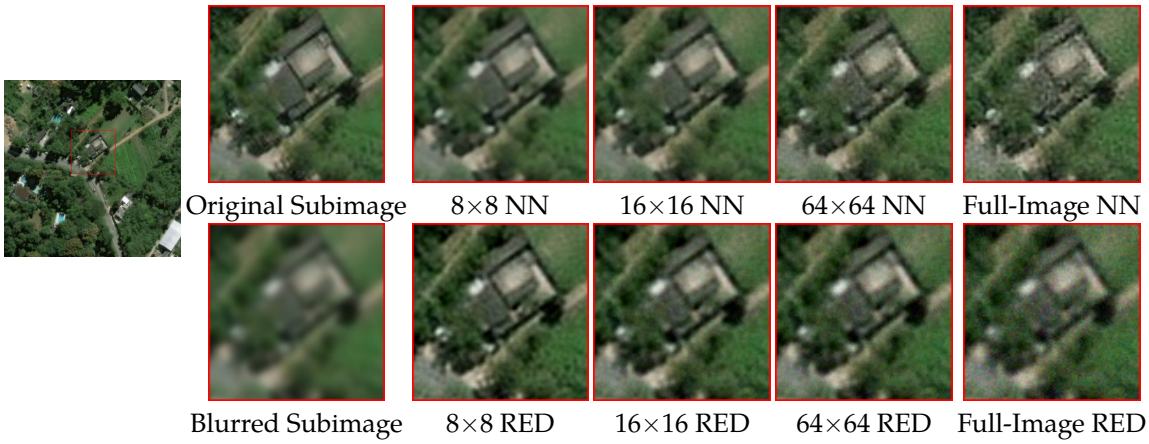


Figure 2.10: Zoomed-in visual comparison of reconstruction quality for a variety of regularization patch sizes for both the Neumann Network (NN) and Regularization by Denoising (RED). Larger patch sizes result in visual artifacts and reduced PSNR. Sub-images are 100×100 pixels, representing a 50 m $\times$ 50 m area on the ground.

In Table 2.2, the empirical results support our sample complexity hypothesis. While both NN and RED enjoy higher PSNR with increasing training set sizes, decreasing the size of the patches that are input to the regularizer has a dramatic effect on PSNR. **For the Neumann network, shrinking the patch size from 64×64 to 8×8 has comparable effect on PSNR as a $10\times$ increase in training set size.** A qualitative, visual comparison of detail in a test image reconstructed by the networks trained on 1000 images is given in Figure 2.10.

Single-Image Training

We demonstrate that enforcing locality in a learned regularizer permits training on a single ground truth image while still enjoying competitive reconstruction accuracy. In this experiment, the training set consists only of a single clean image. Training was performed with identical parameter settings

as prior experiments. The number of training steps was 1000. The held-out test set was the same as previous experiments.

Table 2.3 contains test PSNR results. The results shown here are for a single instance of this experiment: while any training image may be chosen, we found empirically it is beneficial to train on an image with a variety of visual features. Fig. 2.11 contains some sample reconstructions of an image

	8×8	Full Image
RED	26.60 ± 1.54	21.02 ± 0.57
NN	31.90 ± 1.42	18.34 ± 1.31

Table 2.3: PSNR (dB) comparison for single-training-image reconstruction. When there is just one training image, local regularization does not overfit, unlike full-image regularization. Results are the mean plus and minus the standard deviation over test set of size 64.

Figure 2.11: Single-image Training Reconstruction. Local regularization enables competitive reconstruction quality with a single training image, while unrestricted training on a single image results in a poor, noisy reconstruction. The inverse problem is Gaussian deblurring with kernel size 9×9, variance $\sigma^2 = 2$, and noise level 0.03.

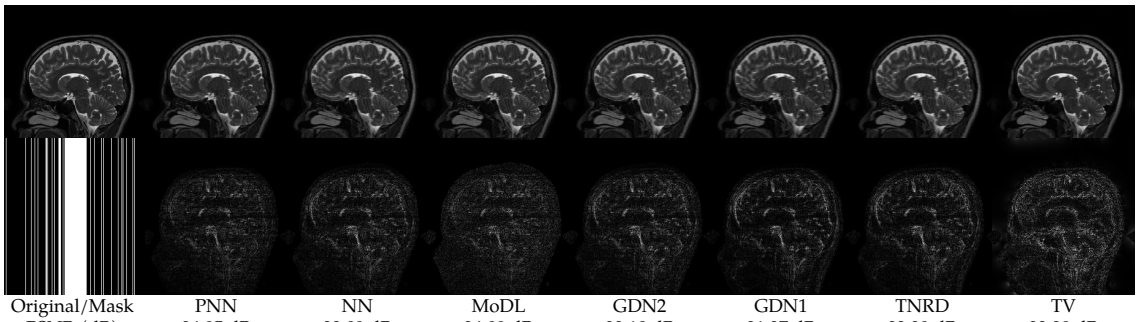
from the test set. We find using patchwise regularizers produces high-quality reconstructions, while regularizers operating on the full image result in noisy artifacts that are likely a result of overfitting to the single training image.

MRI Experiments

In this section we provide results of multi-coil MRI reconstruction from undersampled measurements. Full training and test data is the data used for the experiments in [2], consisting of 12-coil Cartesian sampled k-space data of dimension $232 \times 208 \times 12$ with known coil sensitivity maps. The size of the training set is 360 such acquisitions across 4 subjects, with testing being performed on 40 images from one, separate subject who was not used for training. The sum-of-squares reconstruction is treated as ground truth. Further details of the data acquisition can be found in [2].

All experiments are for $4\times$ undersampling, although we differ from [2] in that we train on a fixed k-space undersampling mask. The undersampling mask is fully sampled in the center 0.15 fraction of frequencies, with the remaining frequencies being sampled according to a random Gaussian pattern. The mask is visualized in figure 2.12.

For the MRI experiments we follow the precedent set by [2] in our choice of learned component, using only a simple five-layer convolutional network with 64 filters per layer and ReLU nonlinearities



Original/Mask	PNN	NN	MoDL	GDN2	GDN1	TNRD	TV
PSNR (dB)	34.95 dB	33.09 dB	34.09 dB	33.18 dB	31.37 dB	32.39 dB	32.29 dB
Test Time (sec)	16.3 sec	5.5 sec	14.3 sec	5.7 sec	3.1 sec	4.0 sec	349.2 sec

Figure 2.12: A comparison of MRI reconstruction quality for a variety of trainable and non-trainable image reconstruction methods. The 12-coil data is undersampled by a factor of $4\times$ and Gaussian noise with $\sigma = 0.01$ is added in k-space. The reconstructions are displayed in the first row, while the second row contains the residual images scaled by a factor of 4. PSNR is displayed next to the method name, while below each method name is the mean time required to reconstruct a single MRI image in seconds. GDN2 denotes the Gradient Descent network using the same initialization as the preconditioned Neumann network, while GDN1 uses the same initialization as the Neumann network.

for all architectures other than TNRD. The TNRD architecture follows the architecture proposed in [12]. The Neumann network results presented here are for the preconditioned Neumann network (PNN), and the number of blocks for GDN, PNN, MoDL, and TNRD is fixed to be 5. The preconditioning operator in PNN is implemented through 10 conjugate gradient iterations, identically to [2]. We compare to GDN with the same initialization as NN (GDN1) and as PNN (GDN2) to study the effect of different initializations on GDN.

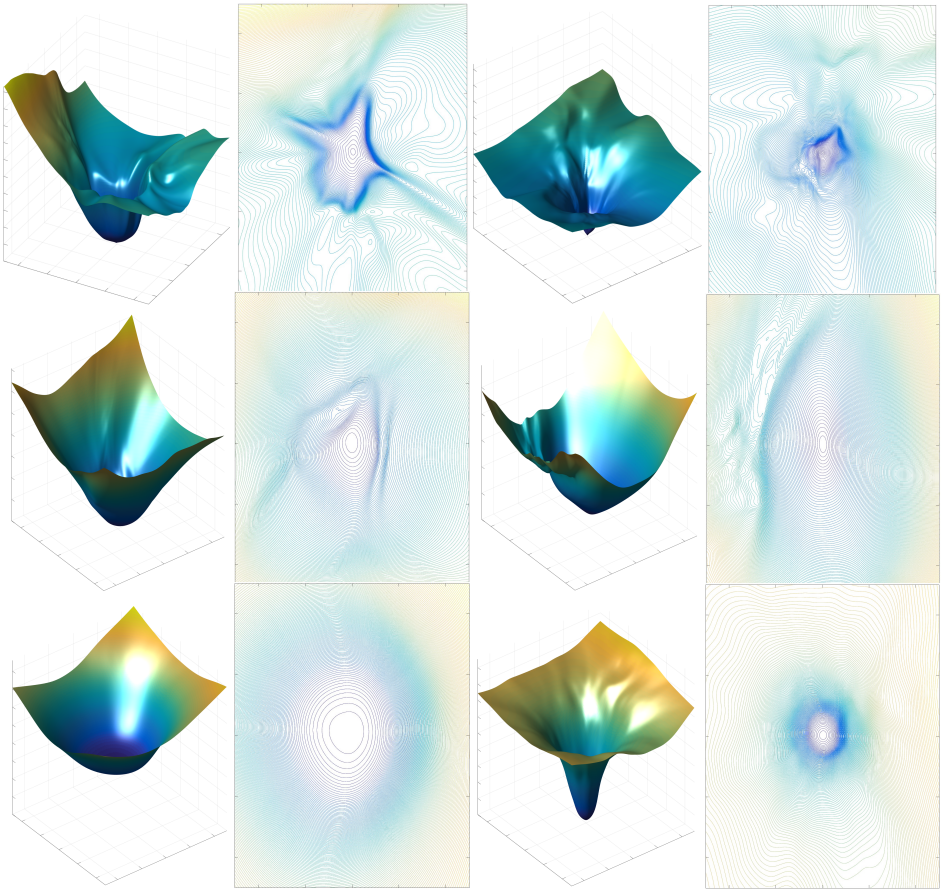
We observe that unrolled optimization approaches are advantageous in this setting compared to the more traditional TV-regularized reconstruction. Preconditioning, both to improve initialization as in GDN2, and incorporated into the architectures, as in PNN and MoDL, improves PSNR significantly in this setting.

A major benefit of learned reconstruction methods is their test time, which is displayed beneath the method name and PSNR in Figure 2.12. We note that all learned approaches reconstruct an order of magnitude faster than the agnostic TV approach. Although preconditioning incurs an additional cost in terms of test time, the performance increase is substantial for MoDL and PNN.

Optimization Landscapes

The performance of the Neumann networks (NN) and Gradient Descent networks (GDN) are very similar across a range of problems and datasets, but NN slightly outperforms GDN consistently. We hypothesize this is due to differences in the connectivity of their network architectures and the effect this has on training.

Specifically, both NN and GDN contain connections across blocks, but differ mainly in their *direction* and *extent*. Adjacent blocks in both networks share residual connections as in a ResNet [25] (the inclusion of the identity $\mathbf{I}$ in the linear part $[\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A}](\cdot)$ of each block of NN and GDN is a residual connection). However, the main difference is that NN contain additional "skip" connections that connect each block with the final layer, similar to architectures like DenseNets [28]. Recent work [34] has highlighted the role of residual connections in the optimization landscape of deep architectures, implying that residual connections "smooth" the optimization landscape. Specifically, fewer local



(a) Neumann Loss Surface (b) Neumann Contour (c) GDN Loss Surface (d) GDN Contour

Figure 2.13: Optimization landscapes and contour plots. (a) The optimization landscape associated with the training loss of the Neumann network around the center optimal point. (b) The associated contour plot. (c) The optimization landscape associated with the training loss of the gradient descent network. (d) The associated contour plot. Neumann network landscapes tend to have wider basins around the minimizer and be steeper outside the basin of the minimizer, which are both more favorable to practical optimization by SGD. Figures use the CIFAR-10 dataset and, from top to bottom, deblurring inverse problem with $\sigma = 2.5$, $10\times$ superresolution, and compressed sensing with $8\times$ compression.

minima tend to be present, and those minima tend to be wide, as opposed to sharp. In addition, the authors of [34] note that skip connections from intermediate or early layers of deep networks to final layer tend to provide stronger smoothing effects on optimization landscapes than residual connections alone. Hence, we might expect the additional skip connections present in NN also lead to a smoother optimization landscape.

In Figure 2.13 we illustrate the optimization landscapes using the method of [34], which proposes a procedure for projecting the loss landscape of very high-dimensional models into two dimensions for visualization purposes. Suppose that the fully-trained network has a set of parameters which is vectorized $\widehat{\theta} \in \mathbb{R}^K$. We draw two independent standard Gaussian vectors v_1, v_2 with dimension K, and normalize them in the manner described in [34] that accounts for the scaling ambiguity of ReLU networks. Then we compute the test and training set error for parameters given by $\widehat{\theta} + \tau(iv_1 + jv_2)$ for step size $\tau > 0$ and integers i, j. The plots above are generated for $i, j \in \{-125, ..., 125\}$ and $\tau = 0.01$. We demonstrate plots for three different forward models: deblurring, compressed sensing with $8\times$

compression, and $10\times$ superresolution.

Figure 2.13 illustrates several attractive properties of the NN and GDN. Local minima appear to be rare in the neighborhood of the trained minima for GDN and NN. While neither is convex, it is interesting to note that the NN landscapes seem to have much wider basins around minima and higher slope outside this main basin. GDN's optimization landscape appears to require a search around a low-slope landscape until finding a region of high curvature in the deblurring and compressed sensing case, and contains more local minima than NN.

In addition, experimental evidence and some theory indicate that wider local minima have better generalization properties [29]. This does not indicate that one architecture should perform better than another, but if both networks achieve similar training error, wider local minima may translate to better test performance.

2.7 Notes for Practitioners

In this section, we will address certain issues that may arise when implementing the ideas presented in this chapter. In particular, this section will focus on some of the choices made during design and implementation of these algorithms.

Normalizing and the use of η

In the time since this chapter was written and published, one of the most common questions raised by practitioners is on the role of η in the expression:

$$\widehat{\mathbf{x}}(\mathbf{y}) = \sum_{j=0}^{B} ([\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A}](\cdot) - \eta R(\cdot))^j (\eta \mathbf{A}^\top \mathbf{y}).$$

In the publicly-released code, the above is implemented with the following expression:

$$\widehat{\mathbf{x}}(\mathbf{y}) = \sum_{j=0}^{B} ([\mathbf{I} - \eta \mathbf{A}^\top \mathbf{A}](\cdot) - R(\cdot))^j (\eta \mathbf{A}^\top \mathbf{y}),$$

which is identical to the original formulation except that $\eta R(\cdot)$ has been replaced by $R(\cdot)$. In practice, this simple change can improve results significantly, especially for measurement models that require choosing very small η. In this case, the $R(\cdot)$ network "absorbs" the η term, and we assume that it can learn to produce outputs on the correct scale during training.

In practice, some measurement models require $\eta = 0.0001$ or smaller, since $\mathbf{A}^\top \mathbf{A}$ is expansive without multiplying by η. In this case, multiplying $R(\cdot)$ by 0.0001 can make learning difficult, as the final reconstruction will not depend strongly on $R(\cdot)$ so backpropagation can be difficult unless the learning rate is quite high.

Hyperparameter Choices

Another common question related to the practice of iterative reconstruction networks is what to do when the network is not training well. Specifically, which parameters are best to tune?

In this chapter and the others, empirically it appears that the most important parameters are: learning rate, η, and finally the initial weights of $R(\cdot)$. A typical tuning procedure will first choose η (or other architecture-specific parameters) by setting $R(\cdot)$ to 0 and choosing η so that the final reconstruction is at least not worse than $\mathbf{A}^\top y$. Beyond that, as with many optimization problems, the key issues are step size and the initial point. Empirically, smaller weights are better; $R(\cdot)$ having small norm appears to be valuable.

Choices for $R(\cdot)$

Typical choices for $R(\cdot)$ are: a U-net [48], DnCNN [59], or a very small 3-layer convolutional network. This section summarizes some benefits and downsides of each, based largely on practical experience.

The U-net is a convolutional-deconvolutional network, with connections between layers that use the same scale. In the best case, they produce high-quality reconstructions, and have inbuild residual connections, which can help significantly in practice when the optimal form of $R(\cdot)$ is something like a denoiser. However, the convolution-transpose layers can, for certain types of images and at certain resolutions, result in undesirable "speckling" artifacts. These artifacts are a side-effect of mismatches between the stride used in a convolution-transpose layer and its kernel width, and are well-known in the superresolution community. Sometimes it may be beneficial to add an extra convolutional layer at the end of the U-net to repair these speckling artifacts. All the same, most state-of-the art systems (as of 2021) leverage a U-Net as regularizer.

The DnCNN is a simple architecture, consisting simply of a number of residual connections and convolutional layers that maintain the spatial resolution of the initial image, sidestepping the need for upsampling or downsampling used in the U-net. As such, the characteristic artifacts mentioned previously cannot exist, at the cost of higher number of parameters and an increased memory footprint (since the intermediate layers of the network work on the same spatial resolution as the initial layers).

Very small networks are typically used in methods like TNRD or Deep Primal Dual Networks [1], where the iterative algorithm is iterated up to 20 times and $R(\cdot)$ may not be shared among layers. This necessitates smaller individual $R(\cdot)$ for practical reasons. Deep Primal Dual networks are vying for the state-of-the-art in medical image reconstruction utilizing this style. The downside of this approach is that the final reconstruction algorithm is essentially one massive end-to-end network, making interpretation impossible. However, if the end goal is massive parameterization for good performance and data is plentiful, this has been demonstrated to be a successful approach.

Bibliography

[1] Jonas Adler and Ozan Öktem. "Learned primal-dual reconstruction". In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1322–1332.

[2] Hemant K Aggarwal, Merry P Mani, and Mathews Jacob. "MoDL: Model Based Deep Learning Architecture for Inverse Problems". In: *IEEE Transactions on Medical Imaging* (2018).

[3] Hemant Kumar Aggarwal, Merry P Mani, and Mathews Jacob. "Multi-Shot Sensitivity-Encoded Diffusion MRI using Model-Based Deep Learning (MoDL-MUSSELS)". In: *arXiv preprint arXiv:1812.08115* (2018).

[4] M. Aharon, M. Elad, and A. Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *IEEE Transactions on signal processing* 54.11 (2006), pp. 4311–4322.

[5] Raman Arora et al. "Understanding Deep Neural Networks with Rectified Linear Units". In: *International Conference on Learning Representations (ICLR)*. 2018.

[6] Harrison H Barrett and Kyle J Myers. *Foundations of image science*. John Wiley & Sons, 2013.

[7] Richard E Blahut. *Theory of remote image formation*. Cambridge University Press, 2004.

[8] Thomas Blumensath. "Sampling and reconstructing signals from a union of linear subspaces". In: *IEEE Transactions on Information Theory* 57.7 (2011), pp. 4660–4671.

[9] Ashish Bora et al. "Compressed Sensing using Generative Models". In: *International Conference on Machine Learning (ICML)*. 2017, pp. 537–546.

[10] Emrah Bostan, Ulugbek S Kamilov, and Laura Waller. "Learning-based image reconstruction via parallel proximal algorithm". In: *IEEE Signal Processing Letters* 25.7 (2018), pp. 989–993.

[11] A. Buades, B. Coll, and J.-M. Morel. "A non-local algorithm for image denoising". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. IEEE. 2005, pp. 60–65.

[12] Yunjin Chen and Thomas Pock. "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1256–1272.

[13] Adam Coates, Andrew Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning". In: *International Conference on Artificial Intelligence and Statistics (AIStats)*. 2011, pp. 215–223.

[14] Kostadin Dabov et al. "Image denoising with block-matching and 3D filtering". In: *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*. Vol. 6064. International Society for Optics and Photonics. 2006, p. 606414.

[15] Aram Danielyan, Vladimir Katkovnik, and Karen Egiazarian. "BM3D frames and variational image deblurring". In: *IEEE Transactions on Image Processing* 21.4 (2012), pp. 1715–1728.

[16] Chao Dong et al. "Image super-resolution using deep convolutional networks". In: *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2016), pp. 295–307.

[17] Weisheng Dong et al. "Compressive sensing via nonlocal low-rank regularization". In: *IEEE Transactions on Image Processing* 23.8 (2014), pp. 3618–3632.

[18] IA Elbakri and JA Fessler. "Statistical image reconstruction for polyenergetic X-ray computed tomography". In: *IEEE Transactions on Medical Imaging* 21.2 (2002), pp. 89–99.

[19] Ehsan Elhamifar and René Vidal. "Sparse subspace clustering". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 2790–2797.

[20] Jeffrey A Fessler. "Model-based image reconstruction for MRI". In: *IEEE Signal Processing Magazine* 27.4 (2010), pp. 81–89.

[21] Mário AT Figueiredo and Robert D Nowak. "A bound optimization approach to wavelet-based image deconvolution." In: *IEEE International Conference on Image Processing (ICIP)*. 2005, pp. 782–785.

[22] Davis Gilton, Greg Ongie, and Rebecca Willett. "Neumann Networks for Linear Inverse Problems in Imaging". In: *IEEE Transactions on Computational Imaging* (2019).

[23] Israel Gohberg and Seymour Goldberg. *Basic operator theory*. Birkhäuser, 2013.

[24] R. C. Gonzalez and R. E. Woods. *Digital image processing*. 3rd. Pearson, 2007.

[25] Kaiming He et al. "Deep residual learning for image recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[26] Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*. Vol. 49. 1. NBS Washington, DC, 1952.

[27] Antoine Houdard, Charles Bouveyron, and Julie Delon. "High-dimensional mixture models for unsupervised image denoising (HDMI)". In: *SIAM Journal on Imaging Sciences* 11.4 (2018), pp. 2815–2846.

[28] Gao Huang et al. "Densely connected convolutional networks." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, p. 3.

[29] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[30] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. "Accurate image super-resolution using very deep convolutional networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1646–1654.

[31] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[32] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. "Fractalnet: Ultra-deep neural networks without residuals". In: *arXiv preprint arXiv:1605.07648* (2016).

[33] Marc Lebrun et al. "Secrets of image denoising cuisine". In: *Acta Numerica* 21 (2012), pp. 475–576.

[34] Hao Li et al. "Visualizing the loss landscape of neural nets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 6389–6399.

[35] Guangcan Liu et al. “Robust recovery of subspace structures by low-rank representation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2013), pp. 171–184.

[36] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[37] Julien Mairal et al. “Online dictionary learning for sparse coding”. In: *International Conference on Machine Learning (ICML)*. ACM. 2009, pp. 689–696.

[38] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 2802–2810.

[39] Willem Marais and Rebecca Willett. “Proximal-Gradient methods for Poisson image reconstruction with BM3D-Based regularization”. In: *IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. 2017, pp. 1–5.

[40] Morteza Mardani et al. “Neural proximal gradient descent for compressive imaging”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 9573–9583.

[41] Tim Meinhardt et al. “Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1799–1808.

[42] Deepak Pathak et al. “Context encoders: Feature learning by inpainting”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2536–2544.

[43] Aniket Pramanik, Hemant Kumar Aggarwal, and Mathews Jacob. “Off-the-grid model based deep learning (O-MoDL)”. In: *arXiv preprint arXiv:1812.10747* (2018).

[44] Chen Qin et al. “Convolutional recurrent neural networks for dynamic MR image reconstruction”. In: *IEEE transactions on medical imaging* 38.1 (2018), pp. 280–290.

[45] Hugo Reboredo et al. “Compressive classification”. In: *2013 IEEE International Symposium on Information Theory*. IEEE. 2013, pp. 674–678.

[46] Francesco Renna et al. “Reconstruction of signals drawn from a Gaussian mixture via noisy compressive measurements”. In: *IEEE Transactions on Signal Processing* 62.9 (2014), pp. 2265–2277.

[47] Yaniv Romano, Michael Elad, and Peyman Milanfar. “The little engine that could: Regularization by denoising (RED)”. In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[48] 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 (MICCAI)*. Springer. 2015, pp. 234–241.

[49] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[50] Paul Schafheitlin. *Die theorie der Besselschen funktionen*. Vol. 4. BG Teubner, 1908.

[51] Afonso M Teodoro, Jose M Bioucas-Dias, and Mario AT Figueiredo. “A convergent image fusion algorithm using scene-adapted gaussian-mixture-based denoising”. In: *IEEE Transactions on Image Processing* 28.1 (2018), pp. 451–463.

[52] Andrey Nikolayevich Tychonoff and VY Arsenin. "Solution of ill-posed problems". In: *Winston & Sons, Washington* (1977).

[53] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. "Spacenet: A remote sensing dataset and challenge series". In: *arXiv preprint arXiv:1807.01232* (2018).

[54] Yilun Wang et al. "A new alternating minimization algorithm for total variation image reconstruction". In: *SIAM Journal on Imaging Sciences* 1.3 (2008), pp. 248–272.

[55] Rebecca M Willett and Robert D Nowak. "Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging". In: *IEEE Transactions on Medical Imaging* 22.3 (2003), pp. 332–350.

[56] Jianbo Yang et al. "Compressive sensing by learning a Gaussian mixture model from measurements". In: *IEEE Transactions on Image Processing* 24.1 (2015), pp. 106–119.

[57] Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity". In: *IEEE Transactions on Image Processing* 21.5 (2012), pp. 2481–2499.

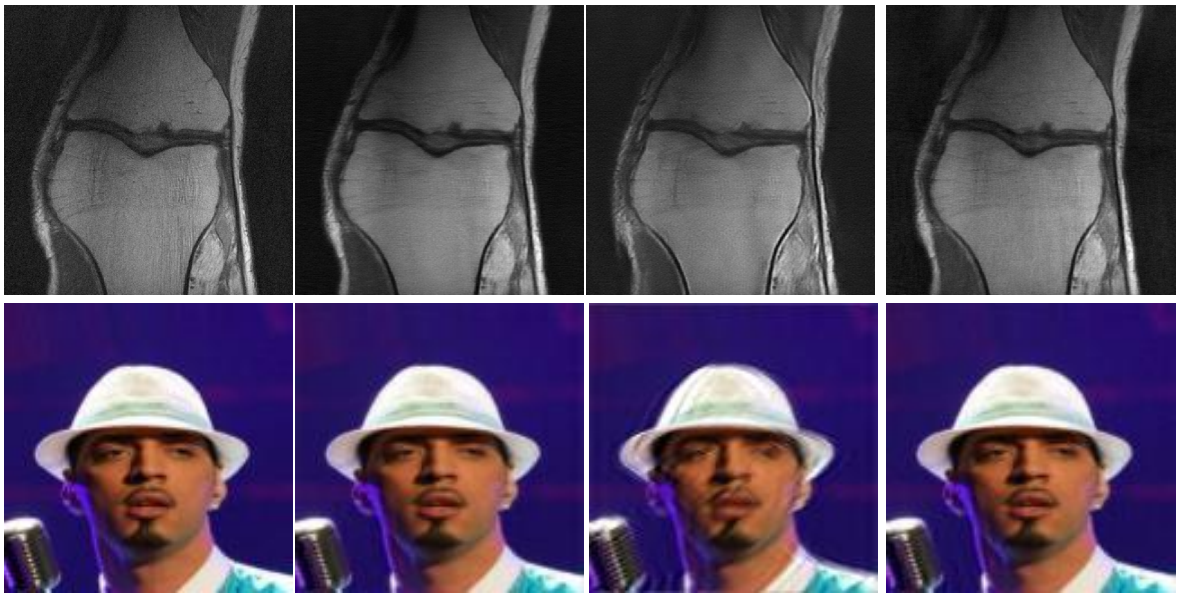
[58] Kai Zhang, Wangmeng Zuo, and Lei Zhang. "FFDNet: Toward a fast and flexible solution for CNN-based image denoising". In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4608–4622.

[59] Kai Zhang et al. "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising". In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[60] Daniel Zoran and Yair Weiss. "From learning models of natural image patches to whole image restoration". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 479–486.

Chapter 3

Model Adaptation for Inverse Problem Solvers



(a) Ground truth (b) No model drift (c) Model drift (d) Model drift w/model adaptation

Figure 3.1: Small perturbations in measurements for deep learning-based image reconstruction operators can lead to both subtle and obvious artifacts in reconstructions across problems and domains. In the top row, we present results for undersampled MRI reconstruction of knee images, and the second row illustrates deblurring images of human faces. (a) Ground truth image. (b) No model drift. Training and test data correspond to same model, A_0, yielding accurate reconstruction via learned model. (c) Model drift but no model adaptation. Training assumes model A_0 but at test time we have model A_1. Reconstruction using trained network *without model adaptation* gives significant distortions. (d) Model drift and model adaptation. Training assumes model A_0 but at test time we have model A_1. Reconstruction using model adaptation prevents distortions and compares favorably to the setting without model drift. The MRI example demonstrates our Reuse and Regularize method (Alg. 2), and the deblurring example demonstrates our Parameterize and Perturb method (Alg. 1). Experimental details are in Section 3.4.

3.1 Introduction

Repeated studies have illustrated that neural networks can be trained to solve inverse problems in imaging, including problems such as image reconstruction in MRI, inpainting, superresolution, deblurring, and more. Recent reviews and tutorials on this topic [4, 24] have described various approaches to this problem. For concreteness, we focus on the case of *linear* inverse problems in imaging. In the general framework of interest, an unknown n-pixel image (in vectorized form) $x \in \mathcal{R}^n$ (or $\mathbb{C}^n$) is observed via m noisy linear measurements $y \in \mathbb{R}^m$ (or $\mathbb{C}^m$) according to the model

$$y = A_0 x + \varepsilon, \tag{3.1}$$

where the matrix $A_0 \in \mathcal{R}^{m \times n}$ (or $\mathbb{C}^{m \times n}$) is the *forward model* and ε represents a vector of noise. The goal is to recover x from y.

In this paper, we focus on the setting in which the forward model A_0 is known and used during training. Past work has illustrated that leveraging knowledge of A_0 during training can reduce the sample complexity [12]. This paradigm is particularly common in applications such as medical imaging,

where A_0 represents a model of the imaging system. For instance, in magnetic resonance imaging (MRI), A_0 reflects which k-space measurements are collected.

Unfortunately, these methods can be surprisingly fragile in the face of *model drift*, which occurs when, at test time, we are provided samples of the form

$$y = A_1 x + \varepsilon' \tag{3.2}$$

for some new forward model $A_1 \neq A_0$ and/or a change in the noise distribution (*i.e.,* the noise ε' is distributed differently than ε). That is, assume we have trained a solver that is a function of both the original forward model A_0 and a learned neural network. One might try to reconstruct x from y using this solver, but it will perform poorly because it is using a misspecified model (A_0 instead of A_1). Alternatively, we might attempt to use the same general solver where we replace A_0 with A_1 but leave the learned component intact. In this case, the estimate x computed from y may also be poor, as illustrated in [3] and [15]. The situation is complicated even further if we do not have a precise model of A_1 at test time.

These are real challenges in practice. For example, in MRI reconstruction there is substantial variation in the forward model depending on the type of acquisition – e.g., Cartesian versus non-Cartesian k-space sampling trajectories, different undersampling factors, different number of coils and coil sensitivity maps, magnetic field inhomogeneity maps, and other calibration parameters [10] – all which need to be accounted for during training and testing. A network trained for one of these forward models may need to be retrained from scratch in order to perform well on even a slightly different setting (e.g., from six-fold to four-fold undersampling of k-space). Furthermore, training a new network from scratch may not always be feasible after deployment due to a lack of access to ground truth images. This could be either due to privacy concerns of sharing patient data between hospitals and researchers, or because acquiring ground truth images is difficult for the new inverse problem.

This leads us to formulate the problem of *model adaptation*: given a reconstruction network trained on measurements from one forward model adapt/retrain/modify the network to reconstruct images from measurements reflecting a new forward model. We consider a few variants of this problem: (a) the new forward model A_1 is known, along with one or more unlabeled training samples y_i reflecting A_1, and (b) A_1 is unknown or only partially known, and we only have one or more unlabeled training samples reflecting A_1. These training samples are unlabeled in the sense that they are not paired with "ground truth" images used to generate the y_i's. Our proposed model adaptation methods allow a reconstruction network to be trained for a known forward model and then adapted to a related forward model without access to ground truth images, and without knowing the exact parameters of the new forward model.

Model drift as stated above is a particular form of *distribution drift*, in which the distribution of $Y|X = x$ changes between training and deployment and we know Y has a linear dependence on X before and after the drift (even if we do not know the parameters of those linear relationships, represented as A_0 and A_1). For example, if we assume a noise model $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, then the training distribution is $Y|X = x \sim \mathcal{N}(A_0 x, \sigma^2 I)$ and the distribution at deployment (assuming the same noise model) is $Y|X = x \sim \mathcal{N}(A_1 x, \sigma^2 I)$. In general, distribution drift challenges may be addressed using transfer learning [25, 35, 33] and domain adaptation [23, 18, 34]. One of the methods we explore in the body of the paper, Parameterize and Perturb, shares several features with transfer learning methodology.

However, since in our setting we have a specific form of distribution drift, it is possible to design more targeted methods with better performance, as illustrated by existing specialized methods for image inpainting [9], as well as our general-purpose Reuse and Regularize method (detailed below).

3.2 Problem Formulation

Here we formalize the problem of *model adaptation* as introduced above.

Suppose we have access to an estimator $\widehat{x} = f_0(y)$ that has been designed/trained to solve the inverse problem

$$y = A_0 x + \varepsilon, \quad x \sim P_X, \ \varepsilon \sim P_{N_0} \tag{P0}$$

where A_0 is a known (linear) forward model, P_X denotes the distribution of images x and P_{N_0} denotes the distribution of the noise ϵ. We assume the trained estimator "solves" the inverse problem in the sense that it produces an estimate $\hat{x} = f_0(y)$ such that the mean-squared error (MSE) $\mathbb{E}_{x,\epsilon}[\|\hat{x} - x\|^2]$ is small.

Now assume that the forward model has changed from A_0 to a new model A_1 and/or the noise distribution has changed from P_{N_0} to a new noise distribution P_{N_1}, resulting in the new inverse problem

$$y = A_1 x + \varepsilon', \quad x \sim P_X, \ \varepsilon' \sim P_{N_1}. \tag{P1}$$

We consider both the case where A_1 is *known* (*i.e.*, we have an accurate estimate of A_1) and the case where A_1 is partially unknown, in the sense that it belongs to a class of parameterized forward models, *i.e.*, $A_1 \in \{A(\sigma) : \sigma \in \mathcal{R}^q\}$, where the parameters $\sigma \in \mathcal{R}^q$ are unknown.

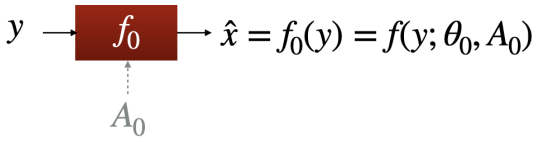
The goal of *model adaptation* is to adapt/retrain/modify the estimator $\widehat{x} = f_0(y)$ that was designed to solve the original inverse problem (P0) to solve the new inverse problem (P1). We will consider two variants of this problem:

- *Model adaptation without calibration data:* In this setting, we assume access to only one measurement vector y generated according to (P1).
- *Model adaptation with calibration data:* In this setting we assume access to a new set of measurement vectors $\{y_i\}_{i=1}^N$ generated according to (P1), but without access to the paired ground truth images (i.e., the corresponding x_i's).

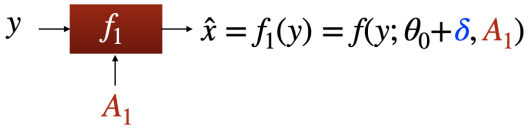
While the above discussion centers around a general estimator $\widehat{x} = f_0(y)$, we are particularly interested in estimators that combine a trained trained deep neural network component depending on a vector of weights/parameters θ_0, along with the original forward model A_0; we will call such an estimator a *reconstruction network*. Specifically, we assume that the forward model A_0 (or other derived quantities, such as its transpose $A_0^\top$, psuedo-inverse $A_0^\dagger$, etc.) is embedded in the reconstruction network, either in an initialization layer and/or in multiple subsequent layers. This is the case for networks based on unrolling of iterative algorithms (see, for example, [**gregor2010learning**, 32, 21, 24, 4], and references therein), in which A_0 appears repeatedly in the network in "data-consistency" layers that approximately re-project the intermediate outputs of the network onto the set of data constraints $\{x \in \mathcal{R}^n : A_0 x = y\}$. In general, we will assume the reconstruction network can be parametrized as $f_0(\cdot) = f(\cdot; \theta_0, A_0)$ where $\theta_0 \in \mathcal{R}^p$ is the vector of pre-trained neural network weights/parameters and A_0 is the original forward model.

Finally, to simplify the presentation, we will assume an additive white Gaussian noise model for both (P0) and (P1), *i.e.*, $\mathsf{P}_{\mathsf{N}_0} = \mathcal{N}(0, \sigma_0^2 \mathrm{I})$ and $\mathsf{P}_{\mathsf{N}_1} = \mathcal{N}(0, \sigma_1^2 \mathrm{I})$ with known variances σ_0^2 and σ_1^2. In this case the negative log-likelihood of x given y under measurement model (P1) is $\frac{1}{2\sigma_1^2}\|A_1 x - y\|_2^2$, which justifies our use of quadratic data-consistency terms in the development below.

(a) Original reconstruction network.



(b) Model adaptation by Parametrize and Perturb (P&P).



(c) Model adaptation by Reuse & Regularize (R&R).

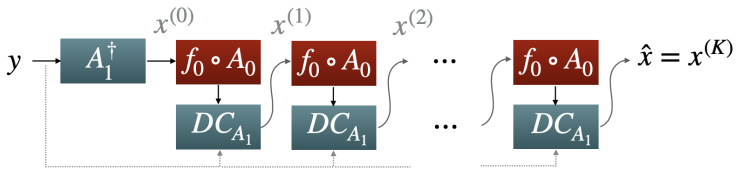


Figure 3.2: Three basic paradigms of reconstruction under "model drift". (a) If the training data is generated using the model $y = A_0 x + \varepsilon$, this can be used to learn a reconstruction network $f(y; \theta_0, A_0)$ which is parameterized by weights or parameters θ_0 and may also explicitly depend on forward model A_0. (b) **Parametrize and Perturb (P&P):** If at test time we are presented with data corresponding to the model $y = A_1 x + \varepsilon'$, we may not only use the new forward model A_1 but also learn a perturbation δ to the original network parameters θ_0 to compensate for the model drift. (c) **Reuse and Regularize (R&R):** Alternatively to P&P, we may reuse the pre-trained network f_0 as an implicit regularizer in an iterative model-based reconstruction scheme. The proposed scheme alternates between applying $f_0 \circ A_0$, which denoises and/or removes artifacts, and a data-consistency step (denoted by DC_{A_1} above) that enforces the estimated image $\hat{x}$ satisfies $A_1 \hat{x} \approx y$.

The feasibility of model adaptation

To compute an accurate reconstruction under the original forward model, A_0, the learned solver must reconstruct components of the image that lie in the null space $\mathsf{N}(A_0)$: for superresolution, these are high-frequency details lost during downsampling, and in inpainting, these are the pixels removed by A_0.

Reconstructing under a different forward model, A_1, requires reconstructing different components of the image in the null space $\mathsf{N}(A_1)$. The general intuition behind model adaptation is that if A_0 and A_1 are similar, then the mapping represented by f_0 can inform the new mapping that we need to learn from image components in the range of A_1 to components in $\mathsf{N}(A_1)$. For example, in an inpainting setting, the learned network not only represents the missing pixels, but it also represents some function of the observed pixels that are relevant to filling in the missing pixels. Thus if A_1 has a similar null space

(*e.g.,* an offset in the collection of missing pixels), it is reasonable to expect that the original network has learned to represent some information about image components in the null space of A_1 but not in the null space of A_0. As the null spaces of A_0 and A_1 get further apart, model adaptation becomes less effective. This is similar to the widely-noted behavior of transfer learning, where transfer learning efficacy depends on the similarity of the training and target distributions. This intuition is supported by our empirical results, which illustrate that when A_0 and A_1 correspond to different blur kernels or perturbed k-space sampling patterns in MRI, the learned mapping f_0 does contain information about image components in the null space of A_1 that can be leveraged to improve reconstruction accuracy, even without additional training samples drawn using the model A_1.

3.3 Proposed Approaches

We propose two distinct model adaptation approaches, *Parameterize & Perturb* (*P&P*) and *Reuse & Regularize* (*R&R*), as detailed below.

Parametrize and Perturb: A transfer learning approach

Let f_0 be a reconstruction network trained to solve inverse problem (P0). Suppose we can explicitly parameterize f_0 both in terms of the trained weights/parameters θ_0 and the original forward model A_0, *i.e.,* we may write $f_0(\cdot) = f(\cdot\,; \theta, A_0)$. Given a new measurement vector y under the measurement model (P1), a "naive" approach to model adaptation is to simply to replace substitute the new forward model A_1 for A_0 in this parametrization, and estimate the image as $f(y; \theta_0, A_1)$. However, as illustrated in Figure 3.1, this can lead to artifacts in the reconstruction due to model mismatch.

Instead, we propose estimating the image as $f(y; \theta_1, A_1)$ where θ_1 is a perturbed set of of network parameters obtained by solving the optimization problem:

$$\min_{\theta} \|y - A_1 f(y; \theta, A_1)\|_2^2 + \mu \|\theta - \theta_0\|_2^2. \tag{3.3}$$

where $\mu > 0$ is a tunable parameter. The first term enforces data consistency, i.e., the estimated image $\widehat{x}$ should satisfy $y \approx A_1 \widehat{x}$, while the second term $\|\theta - \theta_0\|_2^2$ ensures the retrained parameters θ stay close to the original network parameters θ_0. This term is necessary to avoid degenerate solutions, which we demonstrate in the appendix. Our use of a proximity term of this form is also inspired in part by its success in other transfer learning applications (see, *e.g.,* [36]).

If the forward model A_1 is also unknown, we propose optimizing for it as well in the above formulation, which gives:

$$\min_{\theta, A \in \mathcal{A}} \|y - A f(y; \theta, A)\|_2^2 + \mu \|\theta - \theta_0\|_2^2. \tag{3.4}$$

where $\mathcal{A}$ denotes a constraint set. We assume the forward model is parameterized such that the constraint set is given by $\mathcal{A} = \{A(\sigma) : \sigma \in \mathbb{R}^q\}$, where $A(\sigma)$ denotes a class of of forward models parametrized by a vector $\sigma \in \mathcal{R}^q$ with $q \ll m \cdot n$ (e.g., in a blind deconvolution setting, $A(\sigma)$ corresponds convolution with an unknown kernel σ). In particular, we propose optimizing over the parameters σ, which is possible with first-order methods such as stochastic gradient descent, provided the map $\sigma \mapsto A_\sigma$ is first-order differentiable.

Algorithm 1: Parameterize & Perturb (P&P)

Input Original forward model A_0, new forward model A_1, pre-trained reconstruction network $f_0(\cdot) = f(\cdot; \theta_0, A_0)$, regularization parameter μ, new measurements y.

1: Modify the reconstruction network f_0 by internally changing A_0 to A_1, to obtain the estimate $f(y; \theta_0, A_1)$

2: Fine-tune the network weights as $\theta_1 = \theta_0 + \delta$ where δ is a perturbation learned by solving (3.3)

The preceding discussion focused on the case of reconstructing single measurement vector y at test time, *i.e.*, model adaptation without calibration data. Additionally, we consider a P&P approach in the case where we have access to calibration data $y_1, ..., y_N$ generated according to (P1). In this case we propose retraining the network by minimizing the sum of data-consistency terms over the calibration set:

$$(\theta_1, A_1) = \underset{\theta, A \in \mathcal{A}}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \|y_i - A f(y_i; \theta, A)\|_2^2 + \lambda \|\theta - \theta_0\|_2^2. \tag{3.5}$$

At deployment, we propose using the retrained network $\widehat{x} = f(y; \theta_1, A_1)$ as our estimator.

It is worth noting that the P&P model adaptation technique presented above bears similarities to the deep image prior (DIP) approach to solving inverse problems as introduced in [13]. However, P&P differs from DIP in two key aspects: First, in DIP the reconstruction network is initialized with random weights, whereas in P&P we start with a network whose initial weights θ_0 are obtained by training to solve the initial inverse problem (P0). Second, we explicitly enforce proximity to the initial weights to prevent overfitting to the data, and do not rely on early stopping heuristics as is the case DIP. The P&P approach also shares similarities to the "fine-tuning" step proposed in the Σ-net MRI reconstruction framework [17], where a loss similar to (3.3) is minimized to enforce data consistency at test time. However, different from P&P, the fine-tuning approach in [17] regularizes the reconstruction by minimizing the loss between initial reconstruction and the new network output in the SSIM metric. As demonstrated in Figure 3.1, this initial reconstruction can have severe artifacts in certain settings due to model mismatch, in which case enforcing proximity in image space to an initial reconstruction is less justified.

Reuse & Regularize: Model adaptation without retraining

One drawback of the P&P approach is that it requires fine-tuning the network for each input y, which is computationally expensive relative to a feed-forward reconstruction approach. Additionally, the P&P approach is somewhat indirect, relying only on the inductive bias of the network architecture and its original parameter configuration to impart a regularization effect for the new inverse problem (P1). Here we propose a different model adaptation approach that does not require retraining the original reconstruction network, and explicitly makes use of the fact that the original network is designed to solve (P0).

Suppose we are given a reconstruction network $f_0(y)$ trained to solve (P0). The key idea we exploit is that the composition of f_0 with the original forward model A_0, should act as an *auto-encoder*, *i.e.*, if we define the map $g : \mathbb{R}^d \to \mathbb{R}^d$ by $g(x) = f_0(A_0 x)$ then by design we should have $g(x) \approx x$ for any image x sampled from the image distribution P_X. See Figure 3.3 for an illustration in the case of undersampled

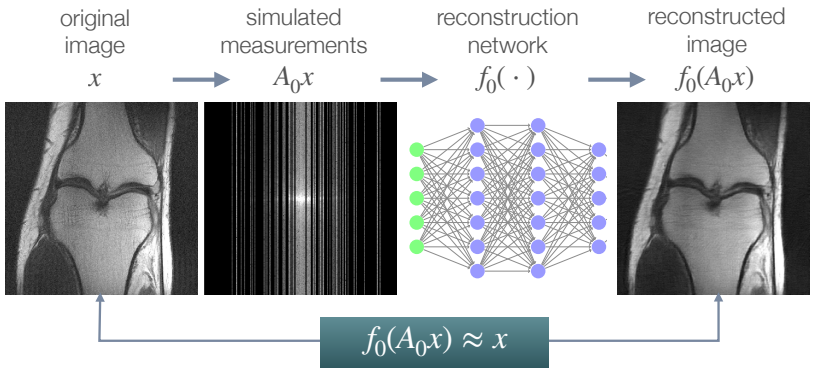


Figure 3.3: Illustration of the auto-encoding property of the map $f_0 \circ A_0$ as used in the proposed R&R model adaptation approach, illustrated in an undersampled MRI reconstruction setting.

MRI reconstruction.

Given this fact, one simple approach to reconstructing a measurement vector y under (P1) is to start from an initial guess, *e.g.*, the least squares solution $x^{(0)} = A_1^\dagger y$, and attempt to find a fixed-point of $g(\cdot)$ by iterating:

$$x^{(k+1)} = g(x^{(k)}), \quad k = 0, 1, 2, \ldots \tag{3.6}$$

However, this approach only uses knowledge of the new forward model A_1 in the initialization step. Also, unless we can guarantee the map $g(\cdot)$ is non-expansive (i.e., its Jacobian is 1-Lipschitz), these iterations could diverge.

Instead, building off the intuition that g acts as an auto-encoder, we propose using g as a regularizer in an iterative model-based reconstruction scheme. In particular, we adopt a regularization-by-denoising (RED) approach, which allows one to convert an arbitrary denoiser/de-artifacting map into a regularizer [28]. The RED approach is motivated by the following cost function:

$$\min_x \frac{1}{2}\|A_1 x - y\|_2^2 + \lambda \rho(x) \tag{3.7}$$

where the function $\rho(x) := x^\top(x - g(x))$ can be interpreted as a regularizer induced by the map $g(x)$ and $\lambda > 0$ is a regularization parameter. Under appropriate conditions on the function g, one can show $\nabla\rho(x) = x - g(x)$. This fact is used in [28] to derive a variety of iterative algorithms based on first-order methods (see also [27] for further analysis of RED, including convergence guarantees).

Algorithm 2: Reuse & Regularize (R&R)

Input Pre-trained reconstruction network $f_0(\cdot)$, original forward model A_0, new forward model A_1, regularization parameter $\lambda > 0$, max iterations K, new measurements y.

1: $x \leftarrow A_1^\dagger y$ ▷ *least-squares initialization* **for** $k = 1, 2, \ldots, K$ **do**

2: $z \leftarrow f_0(A_0 x)$ ▷ *regularize using pre-trained network*

3: $x \leftarrow (A_1^\top A_1 + \lambda I)^{-1}(A_1^\top y + \lambda z)$ ▷ *data consistency*

4: **return** x

For simplicity, we focus on a RED approach with proximal gradient descent (see, e.g., [26]) as the

base algorithm with stepsize $\tau > 0$. This results in an alternating scheme:

$$z^{(k)} = (1-\tau)x^{(k)} + \tau g(x^{(k)})$$
$$x^{(k+1)} = \arg\min_x \frac{1}{2\lambda}\|A_1 x - y\|_2^2 + \frac{1}{2\tau}\|x - z^{(k)}\|_2^2$$

The x-update above has the closed-form expression

$$x^{(k+1)} = \left(A_1^\top A_1 + \tfrac{\lambda}{\tau} I\right)^{-1} \left(A_1^\top y + \tfrac{\lambda}{\tau} z^{(k)}\right) \tag{3.8}$$

For simplicity of implementation and to reduce the number of tuning parameters, we fix the stepsize to $\tau = 1$ in all our experiments. We summarize these steps in Algorithm 2.

Note that in the limit as $\lambda \to \infty$, Algorithm 2 reduces to the fixed-point scheme (3.6), and in the limit as $\lambda \to 0$ Algorithm 2 will return the initialization $x = A_1^\dagger y$. In general, the output from Algorithm 2 will interpolate between these two extremes: x will be an approximate fixed point of g and will approximately satisfy data consistency, *i.e.*, $y \approx A_1 x$.

For certain types of forward models the x-update in (3.8) can be computed efficiently (e.g., if A_1 corresponds to a 2-D discrete convolution with circular boundary conditions, then $A_1^\top A_1$ diagonalizes under the 2-D discrete Fourier transform). However, in general, the matrix inverse $(A_1^\top A_1 + \lambda I)^{-1}$ may be expensive to apply. Therefore, in practice we propose approximating (3.8) with a fixed number of conjugate gradient iterations.

A notable aspect of the R&R approach is that it has potential to improve the accuracy of network-based reconstructions *even in the absence of model drift*, *i.e.*, even if $A_1 = A_0$. This is because data-consistency is not guaranteed by certain reconstruction networks (e.g., U-Nets). However, we are less likely to see a benefit in the case where the reconstruction network already incorporates data-consistency layers, such as networks inspired by unrolling iterative optimization algorithms. We explore this aspect empirically in Section 3.4 in the context of MRI reconstruction.

Algorithm 3: Reuse & Regularize with fine-tuning (R&R+)

Input Pretrained reconstruction network $f_0(\cdot) = f(\cdot;\theta_0)$, original forward model A_0, new forward model A_1, regularization parameter λ, new measurements y.

1: Construct an estimator $\widehat{x}(y;\theta_0)$ by unrolling K iterations of Algorithm 2

2: Fine-tune the network weights as $\theta_1 = \theta_0 + \delta$ where δ is a perturbation learned by approximately minimizing the cost (3.10) via SGD.

3: **return** $x = \widehat{x}(y;\theta_1)$

The R&R approach can also be extended the case where the new forward model A_1 depends on unknown parameters. First, we define an estimator $\hat{x}(y;A_1)$ by unrolling of a fixed number of iterates of Algorithm 2, *i.e.*, we take $\hat{x}(y;A_1) = x^{(K)}$ where $x^{(K)}$ is the Kth iterate of Algorithm 2 with input y for some small fixed value of K (e.g., $K = 5$). Supposing A_1 belongs to a parameterized class of forward models $A(\sigma)$, i.e., $A_1 = A(\sigma_1)$ for some set of parameters σ_1, we propose estimating σ_1 by minimizing

data-consistency of the estimator:

$$\tilde{\sigma}_1 = \arg\min_{\sigma} \|A(\sigma)\hat{x}(y; A(\sigma)) - y\|_2^2 \tag{3.9}$$

The resulting image estimate is then taken to be $\hat{x}(y; \tilde{A}_1)$ where $\tilde{A}_1 = A(\tilde{\sigma}_1)$.

Finally, we also consider a combination of the P&P and R&R approaches where we additionally fine-tune the weights θ_0 of the reconstruction network f_0 embedded in the unrolled R&R estimator. Writing this estimator as $\widehat{x}(y; \theta_0)$, similar to the P&P approach we propose "fine-tuning" the weights θ_0 by approximately minimizing the cost function

$$\min_{\theta} \|A_1\widehat{x}(y; \theta) - y\|_2^2, \tag{3.10}$$

to obtain the updated network parameters $\theta_1 = \theta + \delta$ where δ is some small perturbation. The estimated image is then given by $x = \widehat{x}(y; \theta_1)$. We call this approach R&R+. Empirically we see consistent improvement in reconstruction accuracy from R&R+ over R&R without any fine-tuning (see Figures 3.7 and 3.8). However, this comes at the additional computational cost of having to retrain the reconstruction network parameters at test time.

3.4 Experiments

In this section we empirically demonstrate our approach to model adaptation on three types of inverse problems with two example reconstruction network architectures. We have chosen these comparison points for their simplicity and to illustrate the broad applicability of our proposed approaches. In particular, our approaches to model adaptation are not tied to a specific architectural design.

Methods and datasets used

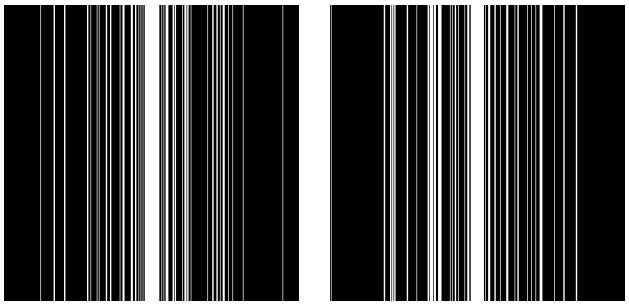
We demonstrate our approaches on three inverse problems: motion deblurring, superresolution, and undersampled single-coil MRI reconstruction.

For motion deblurring, our initial model A_0 corresponds to a 10° motion blur with a 7×7 kernel, and A_1 is a 20° motion blur with a 7×7 kernel, with angle given with respect to the horizontal axis. In superresolution, our initial model is a bilinear downsampling with rate $2\times$, and A_1 corresponds to $2\times$ bicubic downsampling.

MRI reconstruction is performed with a $6\times$ undersampling of k-space in the phase encoding direction for both A_0 and A_1. The sampling maps are shown in Fig 3.4.

We use two datasets in our experiments. First, for motion deblurring and superresolution, we train and test on 128x128-pixel aligned photos of human faces from the CelebA dataset [19].

The data used in the undersampled MRI experiments were obtained from the NYU fastMRI Initiative [38]. The primary goal of the fastMRI dataset is to test whether machine learning can aid in the reconstruction of medical images. We trained and tested on a subset of the single-coil knee dataset, which consist of simulated single-coil measurements. In all tests, we use complex-valued data, which interfaces with our deep networks by treating the real and imaginary parts of the images as separate channels. We measure reconstruction accuracy with respect to the center 320×320 pixels of the complex IFFT of the fully-sampled k-space data. For the purpose of visualization, we display only the magnitude images in the following sections.



(a) Original k-space sampling pattern (A_0)

(b) Resampled k-space sampling pattern (A_1)

Figure 3.4: Visualization of k-space masks used for MRI experiments. Each mask represents a 6-fold Cartesian undersampling with 4% of the center k-space lines fully sampled, and the remaining lines sampled according to a Gaussian variable density scheme. The A_1 mask contains the same center lines, but the higher frequency k-space lines are sampled separately.

Learning rates and regularization parameters (*i.e.,* μ in Algorithm 1 and λ in Algorithm 2) were tuned via cross-validation on a hold out validation set of 512 images for CelebA, and 64 MR images for fastMRI. Batch sizes were fixed in advance to be 128 for the motion blur and superresolution settings, and 8 for the MRI setting. Hyperparameters were tuned via grid search on a log scale. For R&R, we use $K = 5$ iterations in the main loop of Algorithm 2. During training, we add Gaussian noise with $\sigma = 0.01$ to all measurements, as suggested by [11] to improve robustness.

We compare the performance of two reconstruction network architectures across all datasets. First, we utilize the U-Net architecture [29]. Our U-Net implementation takes as input the adjoint of the measurements under the forward model $A_0^\top y$ or $A_1^\top y$, which is then passed through several CNN layers before obtaining a reconstructed image $\widehat{x}$.

We also utilize the MoDL architecture [1], a learned architecture designed for solving inverse problems with known forward models. MoDL is an iterative or "unrolled" architecture, which alternates between a data-consistency step and a trained CNN denoiser, with weights tied across unrolled iterations. We use a U-Net architecture as the denoiser in our implementation of MoDL, ensuring that the overall number of parameters (except for a learned scaling factor in MoDL) is the same in both architectures.

To compare to deep learning-based approaches which do not require training on particular forward models, we compare to the Image-Adaptive GAN (IAGAN) [16] and to Regularization by Denoising (RED) [28]. IAGAN leverages a pretrained GAN to reconstruct arbitrary linear measurements by fitting the latent code input to the GAN, while also tuning the GAN parameters in a way similar to our proposed P&P approach and the Deep Image Prior approach [13].

RED requires only a pretrained denoiser, which we implement by pretraining a set of residual U-Net denoisers on the fastMRI and CelebA training sets, with a variety of different Gaussian noise levels. Specifically, we train 15 denoisers for each problem setting, with σ ranging from 10^{-4} to 10^{1} on a logarithmic scale. All results shown are tuned on the validation set to ensure the optimal denoisers are used.

We also compare to a penalized least squares approach with total variation regularization [30], a classical approach that does not use any learned elements. While more complex regularizers are possible, total variation (TV) is used because of its status as a simple, widely-used conventional baseline.

		Baselines					
		TV	RED		Train w/A_0 Test w/A_0	Train w/A_0 Test w/A_1	Train w/A_1 Test w/A_1
Blur		27.61	30.23	U-Net	34.15	25.42	33.98
				MoDL	36.25	23.91	36.13
SR		28.33	28.59	U-Net	30.74	26.3	31.22
				MoDL	31.32	22.27	31.98
MRI		25.09	27.76	U-Net	31.51	27.47	32.33
				MoDL	31.88	22.82	31.79
		Proposed Model Adaptation Methods					
		Known A_1			Unknown A_1		
		P&P (Alg. 1)	R&R (Alg. 2)	R&R+ (Alg. 3)	P&P (Alg. 1)	R&R (Alg. 2)	R&R+ (Alg. 3)
Blur	U-Net	33.01	32.11	**33.50**	29.18	27.67	30.05
	MoDL	30.08	33.82	**34.73**	29.89	27.81	27.94
SR	U-Net	28.00	29.95	**29.99**	27.77	26.98	29.35
	MoDL	24.59	28.18	**29.83**	23.14	24.93	25.29
MRI	U-Net	29.07	29.71	**31.43**	28.92	28.06	29.54
	MoDL	30.63	30.25	**31.44**	26.64	23.46	27.67

Table 3.1: Comparison of performance of various baseline methods for inverse problems across a variety of datasets and forward models. The metric presented is the mean PSNR. SSIM values can be found in Table A.2

Parametrizing forward models

Both of our proposed model adaptation methods permit the new forward model to be unknown during training, provided it has a known parametrization. In this case, the parameters describing the forward model are learned along with the reconstruction. Here we describe the parametrizations of the forward models that are used.

For the deblurring task, the unknown blur kernel is parametrized as a 7x7 blur kernel, initialized with the weights used for the ground-truth kernel during the initial stage of training. Practically, this is identical to a standard convolutional layer with a fixed initialization and only one learned kernel.

A similar approach is used for superresolution. The forward model can be efficiently represented by strided convolution, and the adjoint is represented by a standard "convolution transpose" layer, again with the weights initialized to match the forward operator in the initial pre-training phase.

In the case of MRI, we use two choices of A_1, depending on whether we assume A_1 is fully known or not. In the case A_1 is fully known, we utilize another $6\times$ undersampled k-space mask, but with resampled high-frequency lines. We display the original and new k-space sampling masks in Figure 3.4. To illustrate the utility of our approach under miscalibration of the forward model in an MRI reconstruction setting, we also consider a unknown random perturbation of the original k-space lines, which we attempt to learn during reconstruction. The vertical k-space lines are still fully sampled, as are the center 4% of frequencies, but all high frequency lines are perturbed uniformly at random with a continuous value from -2 to 2. We wish to emphasize that this experiment is not meant to reflect clinical practice, since such miscalibration of k-space sampling locations is not typically encountered in anatomical imaging with Cartesian k-space sampling trajectories. However, we include this experiment simply to illustrate that our approach could be extended to unknown parametric changes in the forward model in an MR reconstruction setting.

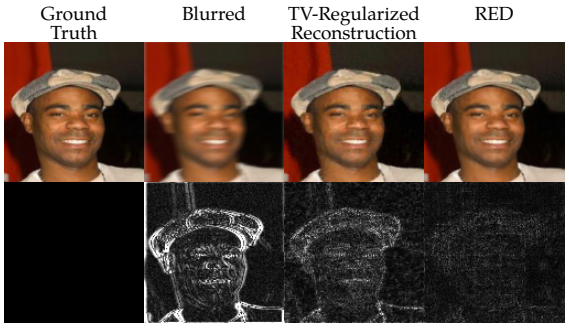


Figure 3.5: Comparison figures for the deblurring methods in Figure 3.7. We present the ground truth, the blurred image (with Gaussian noise with $\sigma = 0.01$ added), a total variation (TV) regularized reconstruction, and a comparison to Regularization by Denoising (RED), a model-agnostic method leveraging a deep denoiser. Below each of the above is the residual image, multiplied by $5\times$ for ease of visualization.

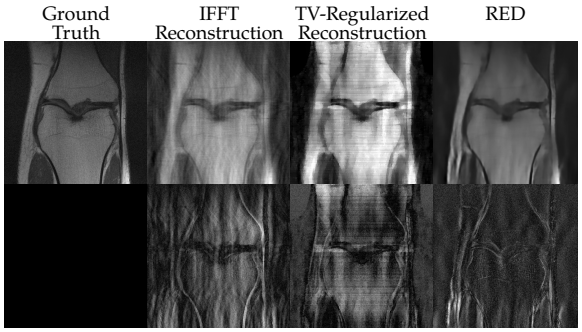


Figure 3.6: Comparison figures for the MRI reconstruction methods in Figure 3.8. We present the IFFT with all k-space data maintained, the naïve IFFT reconstruction after k-space masking, a total variation (TV) regularized reconstruction (with PSNR 27.3 dB), and a RED reconstruction (with PSNR 28.4 dB). We also present the residuals relative to the fully-sampled IFFT, multiplied by $5\times$ for ease of visualization.

Main results

In Table 3.1 we present our main results. We present sample reconstructions for the deblurring problem and MRI reconstruction problem in Figs. 3.7 and 3.8. For reference, the ground truth, inputs to the networks, a total variation regularized reconstruction, and a RED reconstruction are presented in Figs. 3.5 and 3.6. We also provide in the Appendix a table of SSIM values as well as the full version of Table 3.1, which contains the standard deviations of PSNR.

While the magnitude of the improvements vary across domains and problems, we find that retraining the network with the proposed model adaptation techniques significantly improve performance by several dBs in the new setting. This effect is particularly striking in the case of MRI reconstruction with MoDL, where the "naive" approach of replacing A_0 with A_1 in the network gives catastrophic results (a roughly 9 dB drop in reconstruction PSNR), while the proposed model adaptation approaches give reconstruction PSNRs within 1-2 dB of the baseline approach of training and testing with the same forward model in the case where A_1 is known.

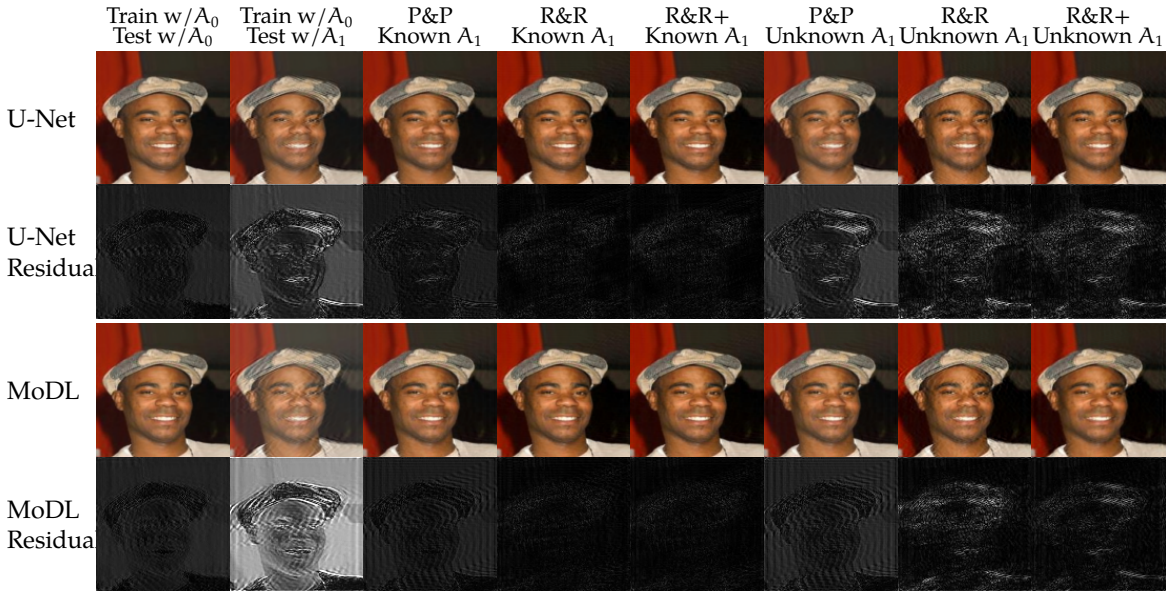


Figure 3.7: Visual examples of reconstruction quality for the motion deblurring inverse problem solved by U-Net and MoDL, as well as the associated residuals. Each residual is multiplied by 5 for ease of inspection. The initial forward model A_0 is a 7x7 motion blur with angle 10°, and the A_1 model has a 7x7 motion blur kernel with angle 20°. The analogous figure for the superresolution problem, and further examples, are available in the Appendix. Best viewed electronically.

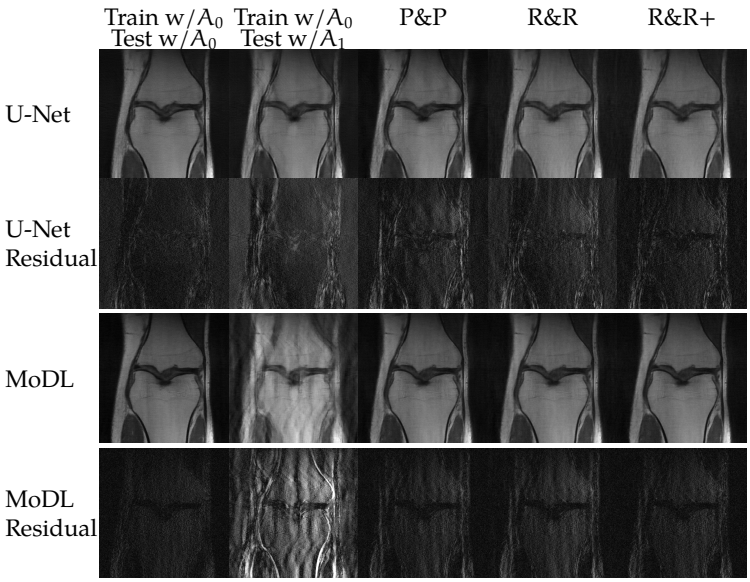


Figure 3.8: Visual examples of different reconstruction approaches for the MRI inverse problem under model drift, along with associated residuals. All residual images are scaled by 5x for ease of inspection. Best viewed electronically.

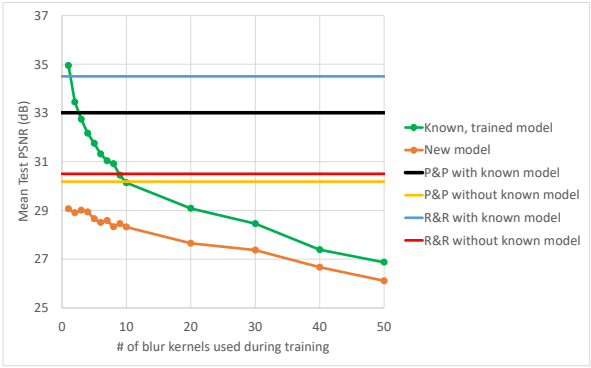


Figure 3.9: Naïvely learning to deblur with a single network and multiple blur kernels sacrifices performance on all blurs. In **green**, the test-time accuracy of a network trained to deblur multiple blurs, tested on a known kernel. In **orange**, the same network, but tested on a new blur that was not used during training. In **black**, our proposed P&P approach (Alg. 1), with a known model, and in **yellow** the same with a learned forward model. **Blue** and **red** show the performance of our R&R approach (Alg. 2), with and without a known forward model.

Learning multiple forward models

In this section we explore an alternative approach to model adaptation. In this setting, we assume that a set of candidate forward models are known during training time. During test time, a single forward model is used for measurement, but the test-time forward model is not known during training. This case represents the setting where the forward model might be parametrized, and so a reasonable approach may be to train the learned network using a number of different forward models to improve robustness.

In simple settings, training on multiple models might be reasonable. However, when the forward model parameterization is high-dimensional, learning to invert all possible forward models may be difficult.

We demonstrate this setting with a deblurring example, in which the same network is trained using a number of blur kernels. The blur kernels are the same kernels used for comparisons in [14]. For consistency, we resize all 50 blur kernels to 7x7, and normalize the weights to sum to 1. We compare reconstruction accuracy when the ground truth blur kernel is included in the set of kernels used for training, as well as when the reconstruction network has never seen data blurred with the testing kernel.

The results are shown in Fig 3.9. Experimentally, we find that training on multiple blur kernels simultaneously incurs a performance penalty as the number of blur kernels used in training increases. In this setting, where the forward model has many degrees of freedom and data is limited, attempting to learn to solve all models simultaneously is worse than transferring a single learned model, even in the absence of further ground truth data for calibration.

Adapting to variable sampling rates in single-coil MRI

A particular concern raised in [3] is related to the stability of a learned solver with respect to the level of undersampling at measurement time. In particular, the authors of that work observe that an image reconstruction system trained to recover images sampled at a particular rate would experience a

	2×	4×	6×	8×	12×
Single-Model	35.74	32.53	31.51	**30.69**	**29.48**
6× No adaptation	27.02	30.20	31.51	27.76	26.15
6× R&R	35.11	**32.61**	**31.73**	29.40	27.34
Multi-Model	33.99	31.62	30.48	29.25	28.35
Multi-Model R&R	**35.80**	32.35	30.81	29.60	28.61

Table 3.2: Comparison of reconstruction PSNR for a variety of MRI acceleration factors for several different approaches. "Multi-model" refers to a U-Net trained for reconstruction on all shown sampling rates, whereas each column of the "Single-Model" results represents a network trained for that particular sampling pattern. 6× refers to the network shown in other experiments. Training a single-sample model consistently performs well for that particular forward model, but at the cost of lower performance on other accelerations, even for higher sampling rates. The multi-model approach sacrifices performance on any one forward model, but most of the difference can be removed by augmenting the multi-model network with our R&R method.

degradation in reconstruction accuracy for *higher* sampling rates than the one the system was trained on.

In Fig. 3.10 we explore this problem in the MRI setting using a U-Net as the reconstruction method, and demonstrate that our R&R method can adapt to this setting as well. By using R&R during inference, the learned network was trained at a 6× accceleration acquisition setting, but was safely deployed for other accelerations *without significant degradation in reconstruction quality*, and *comparing favorably to networks trained explicitly for other sampling rates*.

For comparison purposes, we also train a U-Net using multiple sampling masks. During training, the multiple-model solver is trained to reconstruct MRIs that are measured using the five different sampling patterns demonstrated in Fig. 3.10. We present the mean PSNR on the test set in Table 3.2, along with the mean test PSNR for applying R&R to the multiple-model solver, assuming at test time that the sampling pattern is known. Reconstructions from the multiple-model solver can be found in the Appendix. We observe that training with multiple models means that at test time all models produce reasonable reconstructions, but at the cost of reconstruction quality compared to networks trained for single sampling patterns.

In this experiment, we also observe an interesting side-effect of R&R: when R&R is used to "adapt" to a forward model A_0 that the original network was trained on, we tend to see an improvement in reconstruction quality. This effect is most pronounced for the U-Net trained to reconstruct multiple sampling patterns, but is also true for the "dedicated" solver, demonstrated in Fig. 3.10.

Model adaptation under variable model overlap

In this section we explore how varying the distance between the forward models A_0 and A_1 affects reconstruction quality, and how our proposed R&R method deals with different amounts of overlap between A_0 and A_1. The forward model under investigation is 6× single-coil MRI reconstruction.

To explore variable levels of model drift in the single-coil MRI reconstruction case, we vary which k-space frequencies are sampled in a Cartesian pattern. Specifically, we construct a list of "non-sampled" frequencies and a list of "sampled" frequencies under A_0. We create A_1 by swapping n "sampled" frequencies for n frequencies in the original "non-sampled" list, to ensure that the new A_1 contains exactly n frequencies that were not sampled under A_0. We do not swap the 4% center frequencies in any test.

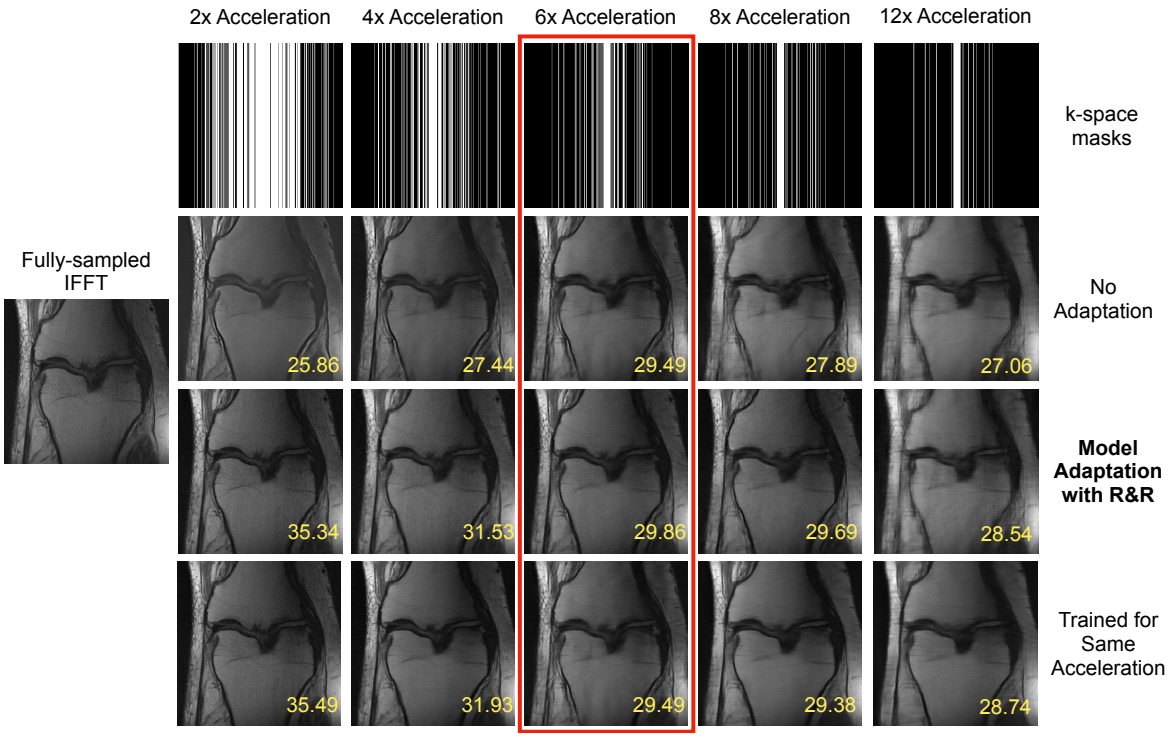


Figure 3.10: Using the R&R model adaptation approach permits using a U-Net trained for $6\times$ acceleration on MRI reconstruction across a range of acceleration parameters. The various k-space sampling patterns used in these experiments are shown in the top row. Without adaptation (second row), the reconstruction quality decreases when changing the acceleration factor, *even when more k-space measurements are taken*, as originally observed in [3]. The R&R reconstructions (third row) compare favorably to the performance of networks trained on each particular k-space sampling pattern (bottom row). The PSNR of each image is presented in dB in yellow on each image.

In Fig. 3.11 we plot $\mathfrak{n}$ vs the mean PSNR over 10 separate instantiations of the above experiment for a no-adaptation approach as well as our R&R method. We run 10 separate instances since the frequencies that are swapped, as well as what frequencies they are swapped to, is random, introducing some variance to the process. We also visually represent the maximum and minimum PSNR across all instances with shading.

Note that the new A_1 in this subsection is different than the A_1 used in other MRI experiments in Table A.1 or Fig. 3.8. The A_1 in Fig. 4b corresponds to a new random selection of frequencies, some of which might also be represented in A_0 (in Fig. 4a). In contrast, the A_1 used for the results in Fig. 3.11 may be "harder" for model adaptation because the new frequencies were explicitly chosen to have zero joint support with those in A_0. In this experiment, the number of changed frequencies acts as a proxy for the difference between $N(A_0)$ and $N(A_1)$, the null spaces of A_0 and A_1.

Sample complexity

Our other experiments assume that model adaptation is performed at the level of individual test samples. However, in the case where we have access to a *calibration set* of measurements under the new forward model A_1 that we can leverage to retrain the network using the P&P approach.

In the transfer learning setting, a key concern is the size of the transfer learning set necessary

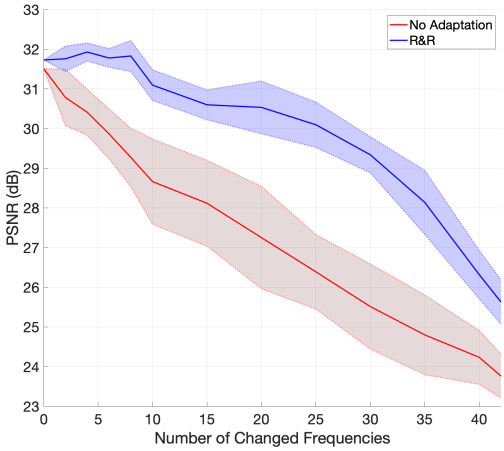


Figure 3.11: Comparison of the mean PSNR for the R&R method and no adaptation for single-sample MRI reconstruction vs. the number of frequencies that differ between A_0 and A_1. The shaded areas represent the standard deviation of mean test PSNR over 10 runs, since frequencies are replaced randomly.

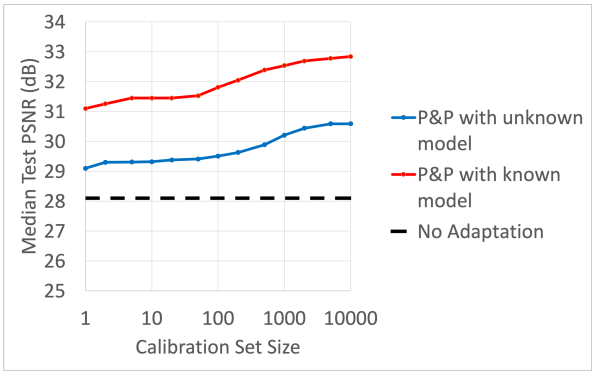


Figure 3.12: Performance of the P&P model adaptation approach for motion deblurring as a function of the number of calibration samples (blurred images) under the new forward model. Both of our approaches outperform a naive approach ("No Adaptation"), even without exact knowledge of the new forward model.

to achieve high-quality results. In this section we compare the performance of P&Pacross different calibration set sizes.

In Fig. 3.12 we explore the effect of the number of samples observed under the new forward model on the adapted model. We observe that even without knowing the forward model, a single calibration sample is sufficient to give improvement over the "naive" method that replaces A_0 with A_1 without further retraining. When the forward model is known during calibration and testing, a single example image can result in a 2 dB improvement in PSNR.

Model-blind reconstruction with generative networks

Recent work [8, 2, 5, 16] has explored solving inverse problems using generative networks, which permit reconstruction under arbitrary forward models assuming an expressive enough generative network. In particular, [2] and [5] consider the case where the forward model is either partially or

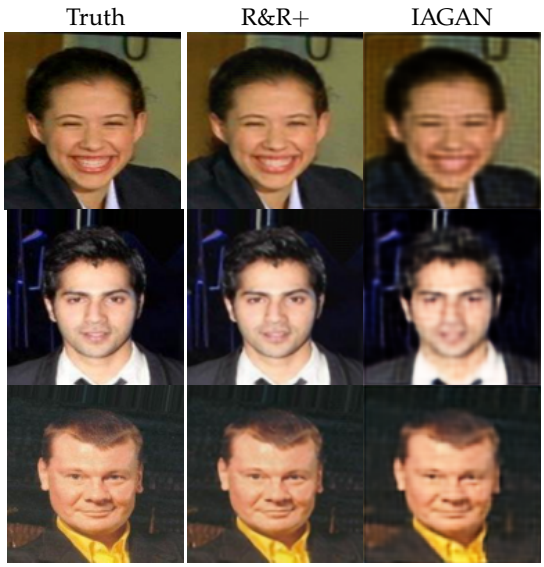


Figure 3.13: Comparison of model adaptation (R&R+) with a model-blind GAN-based reconstruction approach (IAGAN [16]) for 2× super-resolution. While a GAN-based approach only requires learning a single generative network for all forward models, our results suggest that a network trained for a specific forward model with the same number of training samples gives better reconstructions. Best viewed electronically.

entirely unknown, and hence may be learned by parameterizing and jointly optimizing over both the forward model and the latent code for the generative network.

In Fig. 3.13 we provide an illustration of reconstructions obtained by the method of [16], compared to our proposed R&R approach. In our demonstration, as in [16], the generative network under consideration is a pretrained Boundary Equilibrium GAN (BEGAN) [6]. The reconstruction quality is higher when a model-specific network is used, especially when examining fine details and textures.

In the absence of (x_i, y_i) pairs, a generative approach may be reasonable. However, learning the data manifold in its entirety requires a great deal of data at minimum, along with a sufficiently large and well-tuned generator. The authors of [37] also note this fundamental limitation: for smaller or simpler applications, learning a high-quality GAN is straightforward, but for more complex applications it is difficult to train GAN models that are sufficiently accurate to rely on for high-quality reconstructions.

3.5 Discussion and Conclusion

This paper explores solutions to the fragility of learned inverse problem solvers in the face of model drift. We demonstrate across a range of simple, practical applications that using a learned image reconstruction network in settings even slightly different than they were trained in results in significant reconstruction errors, both subtle and obvious. We propose two model adaptation procedures: the first is based on a transfer learning approach that attempts to learn a perturbation to the pre-trained network parameters, which we call Parametrize and Perturb (P&P); the second reuses the network as an implicitly defined regularizer in an iterative model-based reconstruction scheme, which we call Reuse and Regularize (R&R). We also look at a hybrid approach combining these techniques we call R&R+.

We show that our model adaptation techniques enable retraining/reuse of learned solvers under

a change in the forward model, even when the change in forward model is not known. In addition, we demonstrate that just learning to invert a variety of forward models at once is not necessarily the solution to the problem of model drift: directly training on many forward models empirically appears to cause reconstruction quality to fall across all learned models. We also show that our approach is superior to one that requires learning a model of the entire image space via a generative model.

The proposed P&P, R&R, and R&R+ model adaptation approaches each have different trade-offs, and may be useful in different scenarios. In general, we observe that R&R+ produces superior reconstructions over R&R and P&P, but incurs significant computation and time costs associated with network retraining specified in (3.10). The P&P approach also incurs similar costs associated with network retraining. However, when a calibration set is available (as in Section 3.4), the P&P approach only needs to be retrained once, and computation cost at deployment matches the original solver. However, we observe two significant benefits of the R&R approach. First, empirically we observe that only few iterations of R&R (see Algorithm 2) tend to be required to give accurate results (namely, 5 iterations in all our experiments), which increases computational cost by only a constant factor relative to the original reconstruction network. In addition, in the R&R approach only one new parameter is introduced, in contrast to several parameters related to the optimization required for P&P and R&R+. Finally, our experiments suggest that the improvement offered by R&R+ tends to be marginal relative to the improvement seen by going from no adaptation to R&R. Therefore, in situations where reconstruction time is crucial, model adaptation by R&R may be preferred over R&R+.

One surprising benefit of the R&R approach is that even in the absence of model drift (i.e., $A_0 = A_1$) the reconstruction accuracy improves relative to the output from the reconstruction network. This is because R&R iteratively modifies the output of the network to enforce data-consistency at test time. This may potentially resolve the issue raised in [31] about whether learned image reconstruction networks are truly "solving" a given inverse problem, *i.e.,* give a well-defined inverse map of the measurement model. However, to show this would require a much more detailed analysis of the estimator defined by the R&R approach that is beyond the scope of this work.

Adapting learned inverse problem solvers to function under new forward models is just one step towards robustifying these powerful approaches. Our approach for unknown A_1 assumes an explicit parametrization of the forward model, but such a parametrization is not always straightforward or realistic. How best to adapt to complex changes in the forward model that are not easily parametrized is an important open question for future work; see [20] for one recent approach to learning non-parametric (and potentially non-linear) changes to forward models in an iterative reconstruction framework.

While this work focused on linear inverse problem, many of the principles introduced in this work extend also to non-linear inverse problems. For example, the R&R approach, which is based on the regularization-by-denoising technique (RED), is readily adapted to non-linear problems amenable to a RED approach, which includes phase retrieval [22] among others.

Our empirical evidence suggests that successful model adaptation is possible provided the nullspace (or approximate nullspaces) of A_0 and A_1 are close in some sense. However, in settings where nullspaces of A_0 and A_1 are far apart, model adaptation may lead to artifacts or hallucinated details in the reconstructions. In order to understand these limitations of model adaptation, recent methods introduced to quantify hallucinations induced by neural-network based reconstructions may prove to be useful [7].

Finally, while we focused our attention on model drift, an important open problem is how to adapt to simultaneous model and data distribution drift, and the extent to which these effects can be treated

independently. We hope to address these questions in future work.

3.6 Notes for Practitioners

In this section, we will address certain issues that may arise when implementing the ideas presented in this chapter. In particular, this section will focus on some of the choices made during design and implementation of these algorithms.

R&R

As in prior chapters, the regularized inverse used in R&R can be implemented with a conjugate gradient algorithm, and in all the experiments used in this chapter, that is the approach that is used. However, it is not necessary to ensure that R&R can be backpropagated through unless R&R+ is being used, so non-differentiable approaches can also be used.

The regularized inverse in R&R is why Alg 2 is ordered as presented. If the order is swapped, the first step of the R&R algorithm is: $z \leftarrow (A_1^\top A_1 + \lambda I)^{-1}(A_1^\top y + \lambda x)$. This is fine if A_1 is known. However, in the case where A_1 is unknown, typically it is initialized to be A_0. In this case, the previous expression is:

$$z \leftarrow (A_0^\top A_0 + \lambda I)^{-1}(A_0^\top (A_0 x^\star + \epsilon) + \lambda x).$$

Since we typically initialize x_0 with $A_0^\top y$ (or the zero vector):

$$z \leftarrow (A_0^\top A_0 + \lambda I)^{-1}(A_0^\top (1+\lambda) y).$$

This term is highly sensitive when y is measured under A_1. If the overlap between A_1 and A_0 is small, performing this inverse can create artifacts in z. For this reason, we first pass the reconstruction through $f_0(A_0 \cdot)$, followed by the data-fit term.

P&P and R&R+

Tuning P&P and R&R+ can be a delicate process. In particular, choosing the learning rate is crucial. Empirically it appears sufficient to choose the learning rate based on a small validation set of 2-5 images. A general tip is that the learning rates used for P&P and R&R+ should be around three orders of magnitude smaller than the learning rates used to train the original network. Higher will result to overfitting to the measurements, but too low will not actually improve the reconstructions. As in Deep Image Prior and IAGAN, early stopping is helpful here, to prevent the aforementioned overfitting to a particular measurement.

A note on practicality: R&R+ with iterative networks like MoDL is likely too computationally expensive to be practical. The memory requirements to train what is essentially MoDL inside MoDL quickly become prohibitive.

Bibliography

[1] Hemant K Aggarwal, Merry P Mani, and Mathews Jacob. "MoDL: Model Based Deep Learning Architecture for Inverse Problems". In: *IEEE Transactions on Medical Imaging* (2018).

[2] Rushil Anirudh et al. "An unsupervised approach to solving inverse problems using generative adversarial networks". In: *arXiv preprint arXiv:1805.07281* (2018).

[3] Vegard Antun et al. "On instabilities of deep learning in image reconstruction and the potential costs of AI". In: *Proceedings of the National Academy of Sciences* (2020).

[4] Simon Arridge et al. "Solving inverse problems using data-driven models". In: *Acta Numerica* 28 (2019), pp. 1–174.

[5] Kalliopi Basioti and George V Moustakides. "Image Restoration from Parametric Transformations using Generative Models". In: *arXiv preprint arXiv:2005.14036* (2020).

[6] David Berthelot, Thomas Schumm, and Luke Metz. "Began: Boundary equilibrium generative adversarial networks". In: *arXiv preprint arXiv:1703.10717* (2017).

[7] Sayantan Bhadra et al. "On hallucinations in tomographic image reconstruction". In: *arXiv preprint arXiv:2012.00646* (2020).

[8] Ashish Bora et al. "Compressed Sensing using Generative Models". In: *International Conference on Machine Learning (ICML)*. 2017, pp. 537–546.

[9] Alhussein Fawzi et al. "Image inpainting through neural networks hallucinations". In: *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. Ieee. 2016, pp. 1–5.

[10] Jeffrey A Fessler. "Model-based image reconstruction for MRI". In: *IEEE signal processing magazine* 27.4 (2010), pp. 81–89.

[11] Martin Genzel, Jan Macdonald, and Maximilian März. "Solving Inverse Problems With Deep Neural Networks–Robustness Included?" In: *arXiv preprint arXiv:2011.04268* (2020).

[12] Davis Gilton, Greg Ongie, and Rebecca Willett. "Neumann Networks for Linear Inverse Problems in Imaging". In: *IEEE Transactions on Computational Imaging* (2019).

[13] R. C. Gonzalez and R. E. Woods. *Digital image processing*. 3rd. Pearson, 2007.

[14] Michal Hradiš et al. "Convolutional neural networks for direct text deblurring". In: *Proceedings of BMVC*. Vol. 10. 2. 2015.

[15] Shady Abu Hussein, Tom Tirer, and Raja Giryes. "Correction Filter for Single Image Super-Resolution: Robustifying Off-the-Shelf Deep Super-Resolvers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1428–1437.

[16] Shady Abu Hussein, Tom Tirer, and Raja Giryes. "Image-adaptive gan based reconstruction". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 3121–3129.

[17] Hammernik Kerstin et al. "Σ-net Systematic Evaluation of Iterative Deep Neural Networks for Fast Parallel MR Image Reconstruction". In: *arXiv preprint arXiv:1912.09278* (2019). arXiv: 1912.09278.

[18] Da Li et al. "Deeper, broader and artier domain generalization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5542–5550.

[19] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[20] Sebastian Lunz et al. "On learned operator correction in inverse problems". In: *SIAM Journal on Imaging Sciences* 14.1 (2021), pp. 92–127.

[21] Morteza Mardani et al. "Neural proximal gradient descent for compressive imaging". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9573–9583.

[22] Christopher Metzler, Phillip Schniter, Ashok Veeraraghavan, et al. "prDeep: robust phase retrieval with a flexible deep network". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3501–3510.

[23] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. "Domain generalization via invariant feature representation". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 10–18.

[24] Gregory Ongie et al. "Deep Learning Techniques for Inverse Problems in Imaging". In: *arXiv preprint arXiv:2005.06001* (2020).

[25] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[26] Neal Parikh and Stephen Boyd. "Proximal algorithms". In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[27] Edward T Reehorst and Philip Schniter. "Regularization by denoising: Clarifications and new interpretations". In: *IEEE Transactions on Computational Imaging* 5.1 (2018), pp. 52–67.

[28] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The little engine that could: Regularization by denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[29] 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 (MICCAI)*. Springer. 2015, pp. 234–241.

[30] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[31] Emil Sidky et al. "Do CNNs solve the CT inverse problem". In: *IEEE Transactions on Biomedical Engineering* (2020).

[32] Jian Sun, Huibin Li, and Zongben Xu. "Deep ADMM-Net for compressive sensing MRI". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 10–18.

[33] Chuanqi Tan et al. "A survey on deep transfer learning". In: *International conference on artificial neural networks*. Springer. 2018, pp. 270–279.

[34] Mei Wang and Weihong Deng. “Deep visual domain adaptation: A survey”. In: *Neurocomputing* 312 (2018), pp. 135–153.

[35] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[36] LI Xuhong, Yves Grandvalet, and Franck Davoine. “Explicit inductive bias for transfer learning with convolutional networks”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2825–2834.

[37] Raymond A Yeh et al. “Semantic image inpainting with deep generative models”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5485–5493.

[38] Jure Zbontar et al. “fastMRI: An Open Dataset and Benchmarks for Accelerated MRI”. In: 2018. arXiv: 1811.08839.

Chapter 4

Deep Equilibrium Architectures for Learned Optimization in Inverse Problems

4.1 Introduction

A collection of recent efforts surveyed in [26] consider the problem of using training data to solve inverse problems in imaging. Specifically, imagine we observe a corrupted set of measurements y of an image $x^\star$ under a linear measurement operator A with some noise ε according to

$$y = Ax^\star + \varepsilon. \tag{4.1}$$

Our task is to compute an estimate of $x^\star$ given measurements y and knowledge of A. This task is particularly challenging when the inverse problem is *ill-posed*, *i.e.*, when the system is underdetermined or ill-conditioned, in which case simple methods such as least squares estimation may not have a unique solution or may produce estimates that are highly sensitive to noise.

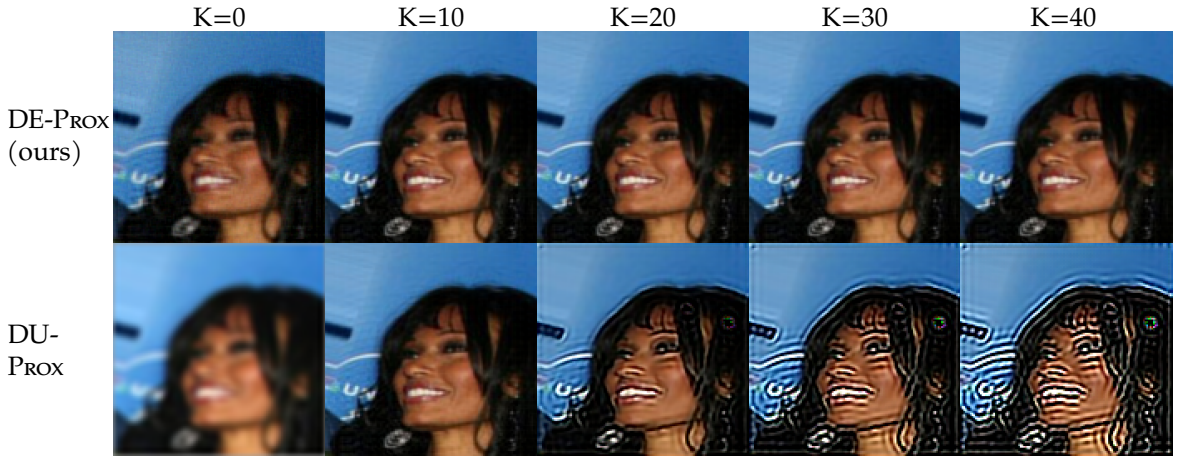


Figure 4.1: Deep Unrolling (DU) methods are state-of-the-art deep networks for image reconstruction that unroll iterative optimization algorithms for a fixed number of iterations K. As shown above in an illustrative example with Deep Unrolled Proximal Gradient Descent (DU-PROX), these methods do not allow flexible operation at test time: unrolling for K iterations where K was not used at training results in severe artifacts. By utilizing Deep Equilibrium networks (DE-PROX above), our method trains inverse solvers to return good reconstructions *at convergence*, instead of at an arbitrary number of iterations, resulting in a flexible, higher-performing image reconstruction technique.

Decades of research has explored geometric models of image structure that can be used to regularize solutions to this inverse problem, including [34, 31, 11] and many others. More recent efforts have focused instead on using large collections of training images, $\{x_i^*\}_{i=1}^n$, to *learn* effective regularizers.

One particularly popular and effective approach involves augmenting standard iterative inverse problem solvers with learned deep networks. This approach, which we refer to as **deep unrolling (DU)**, is reviewed in 4.2. The basic idea is to build an architecture that mimics a small number of iterations of an iterative algorithm. In practice, the number of iterations is quite small (typically 5-10) because of issues stability, memory, and numerical issues arising in backpropagation. This paper sidesteps this key limitation of deep unrolling methods with a novel approach based on **deep equilibrium models (DEMs)** [4], which are designed for training arbitrarily deep networks. The result is a novel approach to training networks to solve inverse problems in imaging that **yields a consistent improvement in performance above state-of-the-art alternatives** and where **the computational budget can be selected at test time** to optimize context-dependent tradeoffs between accuracy and computation. The key empirical findings, which are detailed in 4.6, are illustrated in Fig. 4.2.

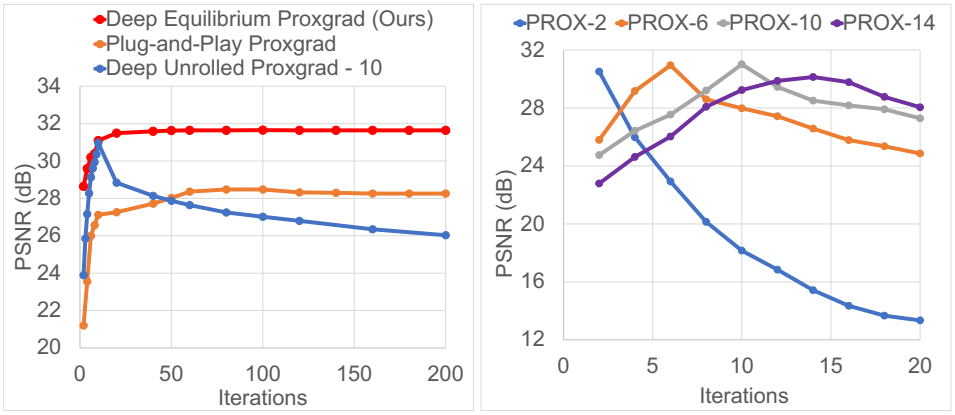


Figure 4.2: (a) PSNR of reconstructed images for an MRI reconstruction problem as a function of iterations used to compute the reconstruction. Unrolled methods are optimized for a fixed computational budget during training, and running additional steps at test time yields a significant drop in performance. Our deep equilibrium methods can achieve the same PSNR for the optimal computational budget of an unrolled method, but can trade slightly more computation time for a significant increase in the PSNR, allowing a user to choose the desired computational budget and reconstruction quality. (b) Standard unrolled deep optimization networks typically require choosing some fixed number of iterates during training. Deviating from this fixed number at test time incurs a significant penalty in PSNR. The forward model here is 8x accelerated single-coil MRI reconstruction, and the unrolled algorithm is unrolled proximal gradient descent with K iterates, labeled PROX-K (Fig. 4.4). For further experimental details see 4.6.

Contributions

This chapter presents a novel approach to machine learning-based methods for solving linear inverse problems in imaging. Unlike most state-of-the-art methods, which are based on unrolling a small number of iterations of an iterative reconstruction scheme ("deep unrolling"), the proposed method is based on deep equilibrium models that correspond to a potentially infinite number of iterations. This framework yields more accurate reconstructions that the current state-of-the-art across a range of inverse problems and gives users the ability to navigate a tradeoff between reconstruction computation time and accuracy at test time. Furthermore, because our formulation is based on finding a fixed points of a operator, we can use standard acceleration techniques to speed test time computations – something that is not possible with deep unrolling methods. In addition, our approach inherits provable convergence guarantees depending on the "base" algorithm used to select a fixed point equation for the deep equilibrium framework. Experimental results also show that our proposed initialization for Deep Equilibrium Models based on pre-training is superior to random initialization, and the proposed approach is more robust to noise than past methods. Overall, the proposed DEM approach is a unique bridge between conventional fixed-point methods in numerical analysis and learning-based techniques for inverse problems in imaging.

4.2 Relationship to Prior Work

Review of Deep Unrolling Methods

Deep unrolling methods for solving inverse problems in imaging consist of a fixed number of architecturally identical "blocks," which are often inspired by particular optimization algorithm. These methods represent the current state-of-the-art in MRI reconstruction, with most top submissions to the fastMRI challenge [25] being some sort of unrolled net. Deep unrolling architectures have also been successfully applied to other inverse problems in imaging, such as low-dose CT [38], light-field photography [10], and emission tomography [23].

We describe here a specific deep unrolling method based on the gradient descent algorithm, although many other variants exist based on alternative optimization or fixed point iteration schemes [26]. Suppose we have a known regularization function r that could be applied to an image x; *e.g.,* in Tikhonov regularization, $r(x) = \frac{\lambda}{2}\|x\|_2^2$ for some scalar $\lambda > 0$. Then we could compute an image estimate $\widehat{x}$ by solving the optimization problem

$$\widehat{x} = \arg\min_x \frac{1}{2}\|y - Ax\|_2^2 + r(x). \tag{4.2}$$

If r is differentiable, this can be accomplished via gradient descent. That is, we start with an initial estimate $\mathbf{x}0$ such as $\mathbf{x}0 = A^\top y$ and choose a step size $\eta > 0$, such that for iteration $k = 1, 2, 3, \ldots$, we set

$$\mathbf{x}k + 1 = \mathbf{x}k + \eta A^\top (y - A\mathbf{x}k) - \eta \nabla r(\mathbf{x}k),$$

where ∇r is the gradient of the regularizer.

The basic idea behind deep unrolled methods is to fix some number of iterations K (typically K ranges from 5 to 10), declare that $\mathbf{x}K$ will be our estimate $\widehat{x}$, and model ∇r with a neural network, denoted $R_\theta(x)$, whose weights θ can be learned from training data. For example, we may define the unrolled gradient descent estimate to be $\widehat{x}^{(K)}(y;\theta) := x^{(K)}$ where $x^{(0)} = A^\top y$ and for $k = 0, \ldots, K-1$ we have the recursive update

$$\mathbf{x}k + 1 = \mathbf{x}k + \eta A^\top (y - A\mathbf{x}k) - \eta R_\theta(\mathbf{x}k). \tag{4.3}$$

Training attempts to minimize the cost function $\sum_{i=1}^n \|\widehat{x}^{(K)}(y_i;\theta) - x_i^*\|_2^2$ with respect to the network weights θ. This form of training is often called "end-to-end"; that is, we do not train the network R_θ that replaces ∇r in isolation, but rather on the quality of the resulting estimate $\widehat{x}^{(K)}$, which depends on the forward model A. Above we assume that all instances of R_θ have identical weights θ, although other works explore variants where the R_θ has iteration dependent weights [1].

The number of iterations in deep unrolling methods is kept small for two reasons. First, at deployment, these systems are optimized to compute image estimates quickly – a desirable property we wish to retain in developing new methods. Second, it is challenging to train deep unrolled networks for many iterations due to memory limitations of GPUs because the memory required to calculate the backpropagation updates scales linearly with the number of unrolled iterations.

As one potential workaround, suppose we train a deep unrolled method for small number of iterations K (e.g., $K = 5$), then extract the learned regularizer gradient R_θ and at inference time run the iterative scheme (4.3) until convergence (*i.e.,* for more iterations K than used in training). Our

numerical results highlight how poorly this approach performs in practice (section 4.6). Choosing a sufficiently large number of iterations K (and hence the computational budget for inference) at training time is essential. As we illustrate in Fig. **??**, one cannot deviate from the choice of K used in training and expect good performance.

Review of Deep Equilibrium Models

In [4], the authors propose a method for training arbitrarily-deep networks given by the repeated application of a single layer. More precisely, consider an L-layer network with input y and weights θ. Letting x^k denote the output of the k^{th} hidden layer, we may write

$$x^{k+1} = f_\theta^{(k)}(x^k; y) \quad \text{for} \quad k = 0, \ldots, L-1$$

where k is the layer index and $f_\theta^{(k)}$ is a nonlinear transformation such as inner products followed by the application of a nonlinear activation function. Recent prior work explored forcing this transformation at each layer to be the same (i.e. *weight tying*), so that $f_\theta^{(k)} = f_\theta$ for all k and showed that such networks still yield competitive performance [12, 5]. Under weight tying, we have the recursion

$$x^{k+1} = f_\theta(x^k; y). \tag{4.4}$$

The limit of x^K as $K \to \infty$, provided it exists, is a fixed point of the operator $f_\theta(\cdot, y)$. In [4] the authors show that the network weights θ can be learned with constant memory using implicit differentiation, bypassing computation and numerical stability issues associated with related techniques on large-scale problems [9, 16]. This past work focused on sequence models and time-series tasks, assuming that each f_θ was a single layer of a neural network, and did not explore the image reconstruction task that is the focus of this paper. Following the posting of a preprint of this paper, [17] propose "fixed-point networks" using a strategy similar to ours, independently verifying the potential of this framework in image reconstruction.

Plug-and-Play and Regularization by Denoising Methods

Initiated by [36], a collection of methods based on the *plug-and-play* (PnP) framework have been proposed, allowing denoising algorithms to be used as priors for model-based image reconstruction. The starting point of PnP is to write reconstructed image as the minimizer of a cost function given by a sum of a data-fit term and a regularizer as in (4.2). Applying alternating directions method of multipliers (ADMM, [7, 8]) to this minimization problem gives a collection of update equations, one of which has the form

$$\arg\min_x \frac{1}{2\sigma}\|z - x\|_2^2 + r(x),$$

where $r(x)$ is the regularizer and $\sigma > 0$ is a parameter; this update can be considered as a "denoising" of the image z. PnP methods replace this explicit optimization step with a "plugged-in" denoising method. Notably, some state-of-the-art denoisers (e.g., BM3D [11] and U-nets [30]) do not have an explicit r associated with them, but nevertheless empirically work well within the PnP framework. A related framework called *Regularization by Denoising* (*RED*) [29] is based on a similar philosophy as

PnP, but instead considers an explicit regularizer of the form

$$r(x) = x^\top(x - \rho(x)),$$

where $\rho(x)$ corresponds to an image denoising function.

Recent PnP and RED efforts focuses on using training data to *learn* denoisers [24, 32, 41, 35, 19]. In contrast to the unrolling methods described in section 4.2, these methods are *not* trained end-to-end; rather, the denoising module is trained independent of the inverse problem at hand (*i.e.,* independent of the forward model A). As described by [26], decoupling the training of the learned component from A results in a reconstruction system that is flexible and does not need to be re-trained for each new A, but can require substantially more training samples to achieve the reconstruction accuracy of a method trained end-to-end for a specific A.

4.3 Proposed Approach

Our approach is to design an iteration map $f_\theta(\cdot\,; y)$ so that a fixed-point $x^{(\infty)}$ satisfying

$$x^{(\infty)} = f_\theta(x^{(\infty)}; y) \tag{4.5}$$

is a good estimate of the image $x^\star$ given its measurements y.

Here we describe choices of f_θ (and hence of the implicit infinite-depth neural network architecture) that explicitly account for the forward model A and generally for the inverse problem at hand. Specifically, we propose choosing f_θ based on different optimization algorithms applied to regularized least squares problem (4.2). This approach is similar to a DU approach (see 4.2), but where the number of iterations is effectively infinite – a paradigm that has been beyond the reach of all previous deep unrolling architectures for solving inverse problems in imaging. Below we consider three specific choices of f_θ, but we note that many other options are possible.

Deep Equilibrium Gradient Descent (DE-Grad)

Connecting the unrolled gradient descent iterations in (4.3) with the deep equilibrium model in (4.4), we let

$$f_\theta(x; y) = x + \eta A^\top(y - Ax) - \eta R_\theta(x). \tag{4.6}$$

Recall that in this setting R_θ is a trainable network that replaces the gradient of the regularizer. See Figure 4.3 for a block diagram illustrating this choice of f_θ.

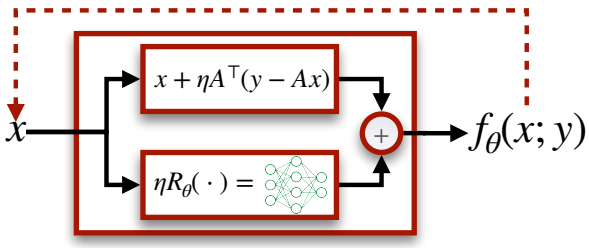


Figure 4.3: Deep Equilibrium Gradient Descent (DE-Grad)

Deep equilibrium proximal gradient (DE-Prox)**:** Proximal gradient methods [27] use a *proximal operator* associated with a function h:

$$\text{prox}_h(x) = \arg\min_u \frac{1}{2}\|u - x\|_2^2 + h(u). \tag{4.7}$$

Specifically, the proximal gradient descent algorithm applied to the optimization problem in (4.2) yields the iterates

$$xk+1 = \text{prox}_{\eta r}(xk + \eta A^\top(y - Axk)),$$

where $\eta > 0$ is a step size. Similar to the deep unrolling approach of [21], we consider replacing $\text{prox}_{\eta r}$ with a trainable network $R_\theta : \mathcal{R}^n \to \mathcal{R}^n$, which gives the iteration map

$$f_\theta(x; y) = R_\theta(x + \eta A^\top(y - Ax)). \tag{4.8}$$

See Figure 4.4 for a block diagram illustrating this choice of f_θ.

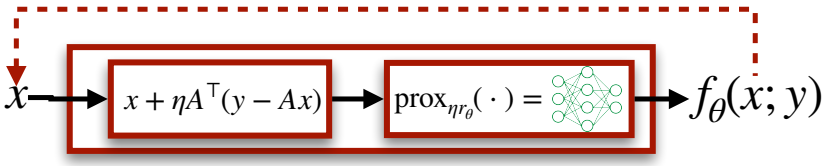


Figure 4.4: Deep Equilibrium Proximal Gradient (DE-Prox)

Deep equilibrium alternating direction method of multipliers (DE-ADMM)**:** The Alternating Directions Method of Multipliers (ADMM, [7]) is an efficient first-order algorithm for large-scale constrained optimization problems. ADMM can be applied to the uncontrained optimization problem (4.2) by rewriting it as the equivalent constrained problem

$$\min_{x,z} \frac{1}{2}\|y - Ax\|_2^2 + r(z) \text{ subject to } z = x.$$

The augmented Lagrangian (in its "scaled form" – see [7]) associated with this problem is given by

$$L_\alpha(x, z, u) := \frac{1}{2}\|y - Ax\|_2^2 + r(z) + \frac{1}{2\alpha}\|z - x + u\|_2^2$$

where u is an additional auxiliary variable and $\alpha > 0$ is a user-defined parameter. The ADMM iterates are then

$$\begin{aligned} z^{(k+1)} &= \arg\min_z L_\alpha(xk, z, u^{(k)}) \\ x^{(k+1)} &= \arg\min_x L_\alpha(x, z^{(k+1)}, u^{(k)}) \\ u^{(k+1)} &= u^{(k)} + z^{(k+1)} - x^{(k+1)}, \end{aligned} \tag{4.9}$$

Here the z- and x-updates simplify as

$$\begin{aligned} z^{(k+1)} &= \text{prox}_{\alpha r}(x^{(k)} - u^{(k)}) \\ x^{(k+1)} &= (I + \alpha A^\top A)^{-1}(\alpha A^\top y + z^{(k+1)} + u^{(k)}). \end{aligned}$$

As in the DE-Prox approach, $\text{prox}_{\alpha r}(\cdot)$ can be replaced with a learned network, denoted R_θ. Making this replacement, and substituting $z^{(k+1)}$ directly into the expressions for $x^{(k+1)}$ and $u^{(k+1)}$ gives:

$$\begin{aligned} x^{(k+1)} &= (I + \alpha A^\top A)^{-1}(\alpha A^\top y + R_\theta(z^{(k)} - u^{(k)}) + u^{(k)}) \\ u^{(k+1)} &= u^{(k)} + R_\theta(z^{(k)} - u^{(k)}) - x^{(k+1)}. \end{aligned} \tag{4.10}$$

Note that the updates for $x^{(k+1)}$ and $u^{(k+1)}$ depend only on the previous iterates $x^{(k)}$ and $u^{(k)}$. Therefore, the above updates can be interpreted as fixed-point iterations on the joint variable $q = (x, u)$, where the iteration map $f_\theta(q; y)$ is implicitly defined as the map that satisfies

$$q^{(k+1)} = f_\theta(q^{(k)}, y) \text{ with } q^{(k)} := (z^{(k)}, u^{(k)}). \tag{4.11}$$

Here we take the estimated image to be $\mathbf{x}^\infty$, where $q^{(\infty)} = (x^{(\infty)}, u^{(\infty)})$ is a fixed-point of $f_\theta(\cdot; y)$. See Figure 4.5 for a block diagram illustrating this choice of f_θ.

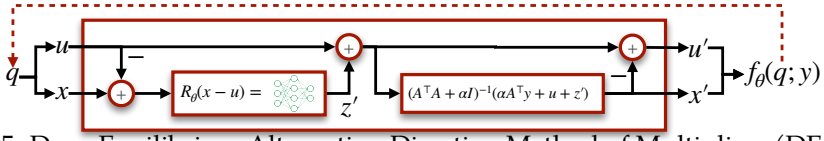


Figure 4.5: Deep Equilibrium Alternating Direction Method of Multipliers (DE-ADMM)

4.4 Calculating forward passes and gradient updates

Given a choice of iteration map $f_\theta(\cdot; y)$ defining a DEM, we confronted the following obstacles. (1) Forward calculation: given an observation y and network weights θ, we need to be able to compute a fixed point of $f_\theta(\cdot; y)$ *efficiently*. (2) Training: given a collection of training samples $\{x_i^\star\}_{i=1}^n$, we need to find the optimal network weights θ.

Calculating Fixed-Points

Both training and inference in a DEM require calculating a fixed point of the iteration map $f_\theta(\cdot; y)$ given some initial point y. The most straightforward approach is to use fixed-point iterations given in (4.4). Convergence of this scheme for specific f_θ designs is discussed in Section 4.5.

However, fixed-point iterations may not converge quickly. By viewing unrolled deep networks as fixed-point iterations, we inherit the ability to accelerate inference with standard fixed-point accelerators. To our knowledge, this work is the first time iterative inversion methods incorporating deep networks have been accelerated using fixed-point accelerators.

Anderson Acceleration: Anderson acceleration [37][1] utilizes past iterates to identify promising directions to move during the iterations. This takes the form of identifying a vector $\alpha^{(k)} \in \mathcal{R}^m$ and setting, for $\beta > 0$

$$x^{(k+1)} = (1 - \beta) \sum_{i=0}^{m-1} \alpha_i^{(k)} x^{(k-i)} + \beta \sum_{i=0}^{m-1} \alpha_i^{(k)} f_\theta(x^{(k-i)}; y).$$

[1] Anderson acceleration for Deep Equilibrium models was introduced in a NeurIPS tutorial by [18].

We find $\alpha^{(k)}$ by solving the optimization problem:

$$\arg\min_{\alpha} \|G\alpha\|_2^2, \quad \text{s.t.} \quad \mathbf{1}^\top \alpha = 1 \tag{4.12}$$

with G a matrix with m columns, where the i^{th} column is the (vectorized) residual $f_\theta(x^{(k-i)}; y) - x^{(k-i)}$. The optimization problem in (4.12) admits a least-squares solution, adding negligible computational overhead when m is small (*e.g.*, $m = 5$).

In Section 4.6 we compare the performance and time characteristics of solving using Anderson acceleration with standard fixed-point iterations, as well as a technique which uses Broyden's method (a quasi-Newton algorithm) to find fixed points, as proposed in [4].

An important practical consideration is that accelerating fixed-point iterations arising from optimization algorithms with auxiliary variables (like ADMM) is non-trivial. Our implementation of DE-ADMM accelerates ADMM using the results of [40]. However, in general acceleration is not required to learn to solve inverse problems, and for other algorithms or settings standard fixed-point iterations may be attractive for their simplicity of implementation.

Gradient Calculation

In this section, we provide a brief overview of the training procedure used to train all networks in Section 4.6. We use stochastic gradient descent to find network parameters θ that (locally) minimize a cost function of the form $\frac{1}{n}\sum_{i=1}^{n} \ell(x^{(\infty)}(y_i;\theta), x_i^*)$ where $\ell(\cdot,\cdot)$ is a given loss function, x_i^* is the ith training image with paired measurements y_i, and $x^{(\infty)}(y_i;\theta)$ denotes the reconstructed image given as the fixed-point of $f_\theta(\cdot\,; y_i)$. For our image reconstruction experiments, we use the mean-squared error (MSE) loss:

$$\ell(x, x^\star) = \frac{1}{2}\|x - x^\star\|_2^2. \tag{4.13}$$

To simplify the calculations below, we consider gradients of the cost function with respect to a single training measurement/image pair, which we denote (y, x^*). Following [4], we leverage the fact that $x^{(\infty)} := x^{(\infty)}(y;\theta)$ is a fixed-point of $f_\theta(\cdot; y)$ to find the gradient of the loss with respect to the network parameters θ without backpropagating through an arbitrarily-large number of fixed-point iterations. We summarize this approach below.

First, abbreviating $\ell(x^{(\infty)}, x^\star)$ by ℓ, then by the chain rule the gradient of ℓ with respect to the network parameters is given by

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial x^{(\infty)}}{\partial \theta}^\top \frac{\partial \ell}{\partial x^{(\infty)}}. \tag{4.14}$$

where $\frac{\partial x^{(\infty)}}{\partial \theta}$ is the Jacobian of $x^{(\infty)}$ with respect to θ, and $\frac{\partial \ell}{\partial x^{(\infty)}}$ is the gradient of ℓ with respect to its first argument evaluated at $x^{(\infty)}$. Since we assume ℓ is the MSE loss, the gradient $\frac{\partial \ell}{\partial x^{(\infty)}}$ is simply the residual between $x^\star$ and the equilibrium point: $\frac{\partial \ell}{\partial x^{(\infty)}} = x^{(\infty)} - x^\star$.

Now, in order to compute the Jacobian $\frac{\partial x^{(\infty)}}{\partial \theta}$ we start with the fixed point equation: $x^{(\infty)} = f_\theta(x^{(\infty)}; y)$. Differentiating both sides of this equation, and solving for $\frac{\partial x^{(\infty)}}{\partial \theta}$ gives

$$\frac{\partial x^{(\infty)}}{\partial \theta} = \left(I - \left.\frac{\partial f_\theta(x; y)}{\partial x}\right|_{x = x^{(\infty)}}\right)^{-1} \frac{\partial f_\theta(x^{(\infty)}; y)}{\partial \theta}. \tag{4.15}$$

Plugging this expression into (4.14) gives

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial f_\theta(x^{(\infty)}; y)}{\partial \theta}^\top \left(I - \frac{\partial f_\theta(x; y)}{\partial x}\bigg|_{x=x^{(\infty)}} \right)^{-\top} \frac{\partial \ell}{\partial x^{(\infty)}}$$

This converts the memory-intensive task of backpropagating through many iterations of $f_\theta(\cdot\,; y)$ to the problem of calculating an inverse Jacobian-vector product. To approximate the inverse Jacobian-vector product, first we define the vector $\beta^{(\infty)}$ by

$$\beta^{(\infty)} = \left(I - \frac{\partial f_\theta(x; y)}{\partial x}\bigg|_{x=x^{(\infty)}} \right)^{-\top} (x^{(\infty)} - x^\star).$$

Following [18], we note that $\beta^{(\infty)}$ is a fixed point of the equation

$$\beta^{(k+1)} = \left(\frac{\partial f_\theta(x; y)}{\partial x}\bigg|_{x=x^{(\infty)}} \right)^\top \beta^{(k)} + (x^{(\infty)} - x^\star), \tag{4.16}$$

and the same machinery used to calculate the fixed point $x^{(\infty)}$ may be used to calculate $\beta^{(\infty)}$. For analysis purposes, we note if $\beta^{(0)} = \mathbf{0}$, simple fixed-point iterations (4.16) may be represented by the Neumann series:

$$\beta^{(\infty)} = \sum_{n=0}^{\infty} \left[\left(\frac{\partial f_\theta(x; y)}{\partial x}\bigg|_{x=x^{(\infty)}} \right)^\top \right]^n (x^{(\infty)} - x^\star). \tag{4.17}$$

Convergence of the above Neumann series is discussed in Section 4.5.

Conventional autodifferentiation tools permit quickly computing the vector-Jacobian products in (4.16) and (4.17). Once an accurate approximation to $\beta^{(\infty)}$ is calculated, the gradient in (4.14) is given by

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial f(x^{(\infty)}; y)}{\partial \theta}^\top \beta^{(\infty)}. \tag{4.18}$$

The gradient calculation process is summarized in the following steps, assuming a fixed point $x^{(\infty)}$ of f_θ is known:

1. Compute the residual $r = x^{\infty} - x^{*}$.
2. Compute an approximate fixed-point $\beta^{(\infty)}$ of the equation $\beta = \left(\frac{\partial f_\theta(x; y)}{\partial x}\big|_{x=x^{(\infty)}} \right)^\top \beta + r$.
3. Compute $\frac{\partial \ell}{\partial \theta} = \frac{\partial f_\theta(x^{(\infty)}; y)}{\partial \theta}^\top \beta^{(\infty)}$.

4.5 Convergence Theory

Here we study convergence of the proposed deep equilibrium models to a fixed-point at inference time, *i.e.*, given the iteration map $f_\theta(\cdot\,; y) : \mathcal{R}^d \to \mathcal{R}^d$ we give conditions that guarantee the convergence of the iterates $x^{(k+1)} = f_\theta(x^{(k)}; y)$ to a fixed-point $x^{(\infty)}$ as $k \to \infty$.

Classical fixed-point theory ensures that the iterates converge to a unique fixed-point if the iteration map $f_\theta(\cdot; y)$ is *contractive*, *i.e.*, if there exists a constant $0 \leqslant c < 1$ such that $\|f_\theta(x; y) - f_\theta(x'; y)\| \leqslant c\|x - x'\|$. Below we give conditions on the regularization network $R_\theta : \mathcal{R}^d \to \mathcal{R}^d$ (replacing the gradient or proximal mapping of a regularizer) used in the DE-Grad, DE-Prox and DE-ADMM models

that that ensure the resulting iteration map is contractive and thus the fixed-point iterations for these models converge.

In particular, following [32], we assume that the regularization network R_θ satisfies the following condition: there exists an $\epsilon > 0$ such that for all $x, x' \in \mathcal{R}^d$ we have

$$\|(R_\theta - I)(x) - (R_\theta - I)(x')\| \leqslant \epsilon \|x - x'\| \tag{4.19}$$

where $(R_\theta - I)(x) := R_\theta(x) - x$. In other words, we assume the map $R_\theta - I$ is ϵ-Lipschitz.

If we interpret R_θ as a denoising or de-artifacting network, then $R_\theta - I$ is the map that outputs the noise or artifacts present in a degraded image. In practice, often R_θ is implemented with a residual "skip-connection", such that $R_\theta = I + N_\theta$, where N_θ is, e.g., a deep U-net. Therefore, in this case, (4.19) is equivalent to assuming the trained network N_θ is ϵ-Lipschitz.

First, we have the following convergence result for DE-Grad:

Theorem 4.1 (Convergence of DE-Grad)**.** *Assume that $R_\theta - I$ is ϵ-Lipschitz* (4.19)*, and let $L = \lambda_{\max}(A^\top A)$ and $\mu = \lambda_{\min}(A^\top A)$, where $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ denote the maximum and minimum eigenvalue, respectively. If the step-size parameter $\eta > 0$ is such that $\eta < 1/(L+1)$, then the* DE-Grad *iteration map $f_\theta(\cdot; y)$ defined in* (4.6) *satisfies*

$$\|f_\theta(x; y) - f_\theta(x'; y)\| \leqslant \underbrace{(1 - \eta(1+\mu) + \eta\epsilon)}_{=:\gamma} \|x - x'\|$$

for all $x, x' \in \mathcal{R}^d$. The coefficient γ is less than 1 if $\epsilon < 1 + \mu$, in which case the the iterates of DE-Grad *converge.*

Proof. Let $f_\theta(x; y)$ be the iteration map for DE-Grad. The Jacobian of $f_\theta(x; y)$ with respect to $x \in \mathcal{R}^d$, denoted by $\partial_x f_\theta(x; y)$, is given by

$$\partial_x f_\theta(x; y) = (I - \eta A^\top A) - \eta \partial_x R_\theta(x) \in \mathcal{R}^{d \times d}$$

where $\partial_x R_\theta(x) \in \mathcal{R}^{d \times d}$ is the Jacobian of $R_\theta : \mathcal{R}^d \to \mathcal{R}^d$ with respect to $x \in \mathcal{R}^d$. To prove $f_\theta(\cdot\,; y)$ is contractive it suffices to show $\|\partial_x f_\theta(x; y)\| < 1$ for all $x \in \mathcal{R}^d$ where $\|\cdot\|$ denotes the spectral norm. Towards this end, we have

$$\begin{aligned}
\|\partial_x f_\theta(x; y)\| &= \|(I - \eta A^\top A) - \eta \partial_x R_\theta(x)\| \\
&= \|\eta I + (1-\eta) I - \eta A^\top A - \eta \partial_x R_\theta(x)\| \\
&= \|(1-\eta) I - \eta A^\top A - \eta(\partial_x R_\theta(x) - I)\| \\
&\leqslant \|(1-\eta) I - \eta A^\top A\| + \eta \|\partial_x R_\theta(x) - I\| \\
&\leqslant \max_i |(1-\eta) - \eta\lambda_i| + \eta\epsilon
\end{aligned} \tag{4.20}$$

where λ_i denotes the ith eigenvalue of $A^\top A$, and in the final inequality (4.20) we used our assumption that the map $(R_\theta - I)(x) := R_\theta(x) - x$ is ϵ-Lipschitz, and therefore the spectral norm of its Jacobian $\partial_x R_\theta(x) - I$ is bounded by ϵ.

Finally, by our assumption $\eta < \frac{1}{1+L}$ where $L := \max_i \lambda_1$, we have $\eta < \frac{1}{1+\lambda_i}$ for all i, which implies $(1-\eta) - \eta\lambda_i > 0$ for all i. Therefore, the maximum in (4.20) is obtained at $\mu := \min_i \lambda_i$, which gives

$$\|\partial_x f_\theta(x; y)\| \leqslant 1 - \eta(1+\mu) + \eta\epsilon.$$

This shows f_θ is γ-Lipschitz with $\gamma = 1 - \eta(1+\mu) + \eta\epsilon$, proving the claim. □

Convergence of PnP approaches PnP-Prox and PnP-ADMM is studied in [32]. At inference time, the proposed DE-Prox and DE-ADMM methods are equivalent to the corresponding PnP method but with a retrained denoising network R_θ. Therefore, the convergence results in [32] apply directly to DE-Prox and DE-ADMM. To keep the paper self-contained, we restate these results below, specialized to the case of the quadratic data-fidelity term assumed in (4.2).

Theorem 4.2 (Convergence of DE-Prox)**.** *Assume that* $R_\theta - I$ *is* ϵ*-Lipschitz* (4.19)*, and let* $L = \lambda_{\max}(A^\top A)$ *and* $\mu = \lambda_{\min}(A^\top A) > 0$*, where* $\lambda_{\max}(\cdot)$ *and* $\lambda_{\min}(\cdot)$ *denote the maximum and minimum eigenvalue, respectively. Then the* DE-Prox *iteraion map* $f_\theta(\cdot, y)$ *defined in* (4.8) *is contractive if the step-size parameter* η *satisfies*

$$\frac{1}{\mu(1+1/\varepsilon)} < \eta < \frac{2}{L} - \frac{1}{L(1+1/\varepsilon)}.$$

Such an η *exists if* $\varepsilon < 2\mu/(L-\mu)$.

See Theorem 1 of [32].

Theorem 4.3 (Convergence of DE-ADMM)**.** *Assume that* $R_\theta - I$ *is* ϵ*-Lipschitz* (4.19)*, and let* $L = \lambda_{\max}(A^\top A)$ *and* $\mu = \lambda_{\min}(A^\top A) > 0$*, where* $\lambda_{\max}(\cdot)$ *and* $\lambda_{\min}(\cdot)$ *denote the maximum and minimum eigenvalue, respectively. Then the iteration map* $f_\theta(\cdot; y)$ *for* DE-ADMM *defined in* (4.11) *is contractive if the ADMM step-size parameter* α *parameter satisfies*

$$\frac{\varepsilon}{(1+\varepsilon-2\varepsilon^2)\mu} < \alpha.$$

See Corollary 1 of [32].

Unlike the convergence result for DE-Grad given in Theorem 1, the convergence results for DE-Prox and DE-ADMM in Theorem 2 and Theorem 3 make the assumption that $\lambda_{min}(A^\top A) > 0$, i.e., A has a trivial nullspace. This is condition is satisfied for certain inverse problems, such as denoising or deblurring, but violated in many others, including compressed sensing and undersampled MRI. However, in practice we observe that the iterates of DE-Prox and DE-ADMM still appear to converge even in situations where A has a nontrivial nullspace, indicating this assumption may be stronger than necessary.

Finally, an important practical concern when training deep equilibrium models is whether the fixed-point iterates used to compute gradients (as detailed in Section 4.4) will converge. Specifically, the gradient of the loss at the training pair (y, x^*) involves computing the truncated Neumann series in (4.17). This series converges if the Jacobian $\partial_x f_\theta(x; y)$ has spectral norm strictly less than 1 when evaluated at any $x \in \mathcal{R}^d$, which is true if and only if the iteration map $f_\theta(\cdot\,; y)$ is contractive. Therefore, the same conditions in Theorems 1-3 that ensure the iteration map is contractive also ensure that the Neumann series in (4.17) used to compute gradients converges.

4.6 Experimental Results

Comparison Methods and Inverse Problems

Our numerical experiments include comparisons with a variety of models and methods. Total-variation Regularized Least Squares (TV) is an important baseline that does not use any training data but rather

Table 4.1: Mean Test PSNR and SSIM comparison across reconstruction approaches and problems; for each setting, the best PSNR and SSIMs are in bold.

		Plug-n-Play (DnCNN denoiser)		RED (DnCNN denoiser)	Deep Unrolled Methods Trained End-to-End				Deep Equilibrium (Ours)		
	TV	Prox	ADMM	ADMM	Grad	Prox	ADMM	Neumann	Grad	Prox	ADMM
Deblur (1)											
PSNR	26.79	29.77	29.95	29.78	32.23	31.64	31.45	32.39	**32.43**	31.87	32.30
SSIM	0.86	0.88	0.89	0.89	0.93	0.93	0.93	0.94	**0.94**	0.93	0.94
Deblur (2)											
PSNR	31.31	35.25	35.61	35.22	36.10	36.92	36.14	36.24	**37.99**	37.84	37.95
SSIM	0.90	0.96	0.96	0.96	0.97	0.97	0.97	0.97	**0.98**	0.97	0.98
CS (4x)											
PSNR	26.04	27.79	27.85	27.80	29.32	29.65	29.09	29.59	31.46	31.51	**31.64**
SSIM	0.83	0.87	0.87	0.86	0.88	0.89	0.88	0.89	0.93	0.93	**0.93**
MRI (8x)											
PSNR	26.64	28.45	28.39	29.89	31.13	30.97	31.82	31.64	**32.01**	31.02	31.38
SSIM	0.78	0.85	0.85	0.88	0.89	0.88	0.88	0.89	**0.89**	0.88	0.89
MRI (4x)											
PSNR	31.22	31.56	31.92	32.37	32.44	32.49	32.56	32.62	33.41	33.66	**33.72**
SSIM	0.88	0.89	0.89	0.90	0.91	0.91	0.91	0.92	0.91	0.92	**0.92**

leverages geometric models of image structure [31, 33, 6]. The PnP and RED methods are described in section 4.2; we consider both the original ADMM variant of [36] PnP-ADMM and a proximal gradient PnP-Prox method as described in [32]. We utilize the ADMM formulation of RED. Deep Unrolled methods (DU) are described in section 4.2; we consider DU using gradient descent, proximal gradient, and ADMM. The preconditioned Neumann network [15] does not have simple Deep Equilibrium or Plug-and-Play analogues, and is included as an alternative deep unrolled method.

All deep unrolled methods have *tied weights*, i.e. the network used at each iteration is the same. This is done to ensure that the number of parameters in the deep unrolled, deep equilibrium, and Plug-and-Play/RED methods are the same. Moreover, on modestly-sized datasets recent work has shown that tied weights result in better reconstructions [2].

We compare the above approaches across three inverse problems: Image deblurring (Deblur), compressed sensing (CS), and accelerated MRI reconstruction (MRI). In the Deblur setting we simulate blurry images using 9×9 pixel Gaussian blur kernel with variance 5 and consider two noise levels: Deblur (1) refers to the setting of additive white Gaussian noise with variance $\sigma = 0.01$ (high noise), and Deblur (2) refers the setting of additive white Gaussian noise with standard deviation $\sigma = 0.0001$ (low noise). In the CS setting, we take linear measurements of the image by forming inner products with random Gaussian vectors whose entries i.i.d. standard normals, and use an undersampling factor of $4\times$, *i.e.*, the corresponding forward model A has $4\times$ is a random Gaussian matrix fewer rows than columns. The measurements are then corrupted with additive white Gaussian noise with standard deviation $\sigma = 0.01$. In the MRI setting, we investigate recovery at $4\times$ and $8\times$ acceleration, where "$\rho\times$ acceleration" indicates an undersampling factor of ρ in k-space (not to be confused with the acceleration techniques used in finding fixed points). Our MRI experiments focus on the case of (virtual) single-coil MR data acquired on a Cartesian grid in k-space, where corresponding forward model A is a subsampling of rows of the discrete Fourier transform matrix. We additionally add complex white Gaussian noise to the undersampled k-space measurements with standard deviation $\sigma = 0.01$.

For the Deblur and CS problems, we utilize a subset of the Celebrity Faces with Attributes (CelebA) dataset [20], which consists of centered human faces. We train on a subset of 10000 of the training

images. All images are resized to 128×128. For the MRI problem, we use a random subset of size 2000 of the fastMRI single-coil knee dataset [39] for training. We trim the ground truth MR images to a 320×320 pixel region-of-interest.

Architecture Specifics

For our learned network, we utilize a DnCNN architecture as in [32]. (We also experimented with U-Nets, but found that DnCNN yielded superior performance for both our proposed deep eq methods and the comparison methods.) For both the CelebA and fastMRI datasets, we train six DnCNN denoisers with noise variances $\sigma^2 = 0.1, 0.05, 0.02, 0.01, 0.005, 0.001$ on the training split. Training follows the methodology of [32]. Specifically, to ensure contractivity of the learned component, we add spectral normalization to all layers, ensuring that each layer has a Lipschitz constant bounded above by 1. This normalization is enforced during pretraining as well as during the Deep Equilibrium training phrase.

During training, we utilize Anderson acceleration for both the forward and backward pass fixed-point iterations. In backward passes, the number of fixed-point iterations was limited to 50 due to memory constraints, but fixed-point iterations in forward passes were run until convergence was observed (defined to be when the relative norm difference between iterations is less than 10^{-3}). Test time results were produced using the same method as the forward pass during training, but over the test set. We compare different fixed-point calculation methods for the forward and backward passes in Section 4.6.

Further details on settings, parameter choices, and data may be found in the supplementary materials and in our publicly-available code.[2]

Parameter Tuning and Pretraining

Each of the iterative optimization algorithms we test has its own set of hyperparameters to choose, *e.g.*, the step size η in DE-Grad, plus any parameters used to calculate the initial estimate $x^{(0)}$. Tuning hyperparameters requires choosing a particular regularization network R_θ during tuning. We choose from a collection of R_θ that have been pretrained for Gaussian denoising at different noise levels. Pretraining can be done on the training dataset (*e.g.*, training on MRI images directly) or using an independent dataset (*e.g.*, the BSD500 image dataset [22]). We use the former approach in our experiments.

To tune hyperparameters, we first choose parameters to optimize the performance of PnP on a validation set via a grid search over pretrained R_θ as well as algorithm-specific hyperparameters (such as the η step-size parameter in gradient descent approaches). Then, we use the PnP hyperparameter settings as initial hyperparameter values when training Deep Equilibrium or Deep Unrolling methods.

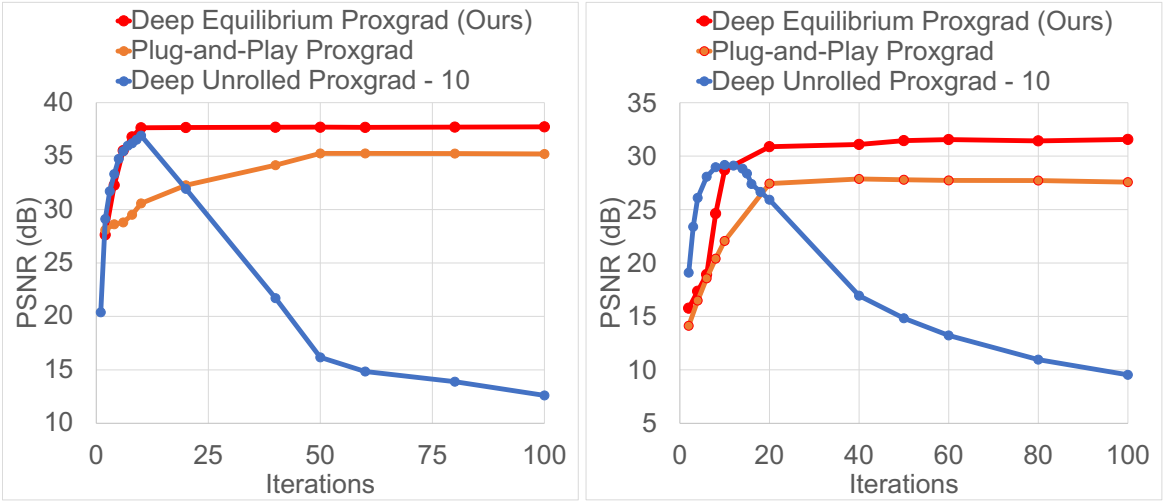
Main Results

We present the main reconstruction accuracy comparison in Table 4.1. Each entry for Deep Equilibrium (DE), Regularization by Denoising (RED), and Plug-and-Play (PnP) approaches is the result of running fixed-point iterations until the relative change between iterations is less than 10^{-3}. During training, all DE models were limited to a maximum 100 forward updates, but terminate iterations on the relative

[2] Available at: `https://github.com/dgilton/deep_equilibrium_inverse`

norm difference between iterations falling below 10^{-3}. The DU models are tested at the number of iterations for which they were trained and all parameters for TV reconstructions (including number of TV iterations) are cross-validated to maximize PSNR. Performance as a function of iteration is shown in Figs. 4.2, **??**, and section 4.6, with example reconstructions in Fig. 4.7. Further example reconstructions are available for qualitative evaluation in the supplementary materials.

We observe our Deep Equilibrium-based approaches consistently outperform Deep Unrolled approaches across different choices of base algorithm (*i.e.*, GRAD,PROX,ADMM). Among choices of iterative reconstruction architectures for f_θ, there does not appear to be an obvious winner, suggesting the optimal choice may be problem- or setting-dependent.



(a) CS. (b) Deblurring.

Figure 4.6: Iterations vs. reconstruction PSNR for DE-PROX and competing methods for (a) deblurring and (b) compressed sensing. MRI results are in Fig. 4.2. The deep unrolled prox grad was trained for 10 iterations. In all examples, deep unrolling is only effective at the number of iterations for which it is trained, whereas deep equilibrium achieves higher PSNR across a broad range of iterations, allowing a user to trade off computation time and accuracy.

(a) Ground truth (b) IFFT ($A^\top y$), PSNR = 24.53 dB. (c) DU-PROX, PSNR = 31.02 dB (d) DE-PROX, PSNR = 32.09 dB

Figure 4.7: 8× accelerated MRI reconstruction example. Best viewed digitally.

As seen in Figs. 4.2, **??**, and **??**, the Deep Equilibrium approach generally outperforms Deep Unrolled solvers. Figs. 4.2, **??**, and **??** show that our approach requires *no more computation* than Deep Unrolled networks to achieve the same performance level and has an increasing advantage with further

	Total Time (s)	Time/Iteration (s)	PSNR (dB)
Plug & Play	1.24	0.025	31.56
DU-Prox	0.25	0.025	32.49
DE-Prox	1.22	0.025	31.86
DE-Prox (Anderson)	1.17	0.046	33.66
DE-Prox (Broyden)	1.85	0.039	33.61

Table 4.2: Comparison of computation time required to reach convergence and resulting reconstruction PSNR in 4× accelerated MRI reconstruction.

computation.

Effect of Acceleration

Here we demonstrate the effect of using different fixed-point solvers during both the training and inference procedures. Leveraging acceleration can decrease computational costs during both training and inference and result in better empirical performance at inference. Table **??** compares Anderson accelerated Deep Equilibrium approaches with Deep Unrolling, Plug and Play, Deep Equilibrium utilizing Broyden's Method (as was used in [4]), and non-accelerated Deep Equilibrium. All results were determined using PyTorch utilizing an NVidia RTX 2080 Ti GPU. As mentioned previously, non-accelerated Deep Equilibrium at inference time has identical per-iteration cost and memory requirements as Plug and Play and Deep Unrolling.

We observe that while finding the fixed-points using Broyden's method and Anderson acceleration requires more time per iterate, convergence occurs faster than a standard fixed-point iteration so the net time spent at inference is less. Since the Broyden solution was slightly worse in terms of PSNR, Anderson acceleration was used for all other experiments.

For practical matters, the memory cost of each of the compared methods may also be an important factor to consider. At train time, Deep Unrolled methods require memory scaling linearly with the number of iterations used, while Deep Equilibrium methods require only the memory necessary to compute the gradient in (4.16), which is what permits training at convergence. Plug and Play methods are the least memory-intensive of all to train. At inference time, each method only needs to store at most a constant number of iterations, so all methods are cheap in terms of memory to evaluate.

Effect of Pre-Training

Here we compare the effect of initializing the learned component R_θ in our deep equilibrium models with a pretrained denoiser versus initializing with random weights. We use the same hyperparameter tuning scheme described in Section 4.6.

We present our results on Deep Equilibrium Proximal Gradient Descent (DE-Prox) in Figure 4.8(a). We observe an improvement in reconstruction quality when utilizing our pretraining method compared to a random initialization. We also note that pretraining enables a simple choice of algorithm-specific hyperparameters. For example, with random initialization, choosing the proper internal step size for DE-Prox would require training several different DE-Prox instances, and choosing the correct step size based on validation set, in addition to any other parameters, such as the learning rate used during training.

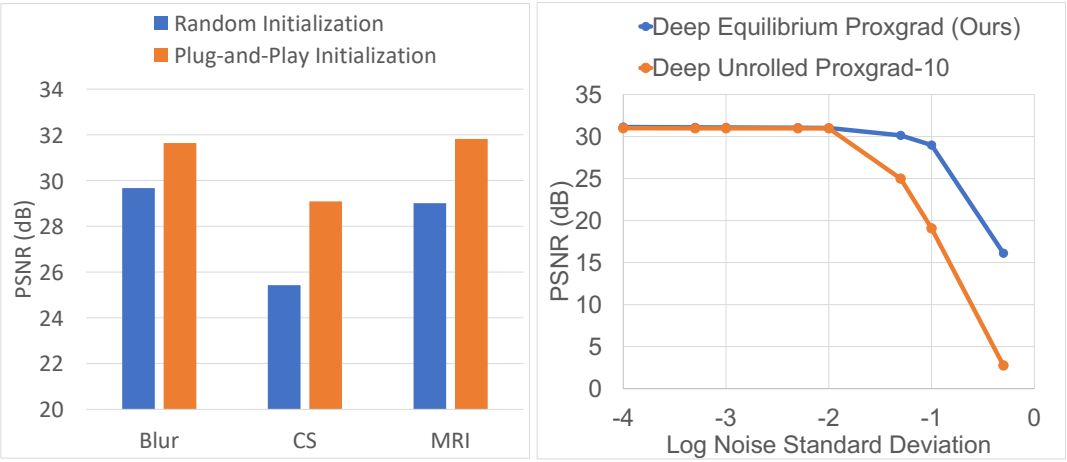


Figure 4.8: (a) Comparison of learned DE-Prox reconstruction quality across three different inverse problems: Deblurring (Blur), compressed sensing (CS), and undersampled MRI reconstruction (MRI). In our experiements, initializing with a pretrained denoiser routinely offered as good or better reconstruction quality (in terms of PSNR) than a random initialization. (b) Noise sensitivity comparison between DU-Prox and DE-Prox. The forward model used is $8\times$ MRI reconstruction, and σ here corresponds to the level of Gaussian noise added to observations.

Noise Sensitivity

We observe empirically that the Deep Equilibrium approach to training achieves competitive reconstruction quality and increased flexibility with respect to allocating computation budget at test time. Recent work in deep inversion has questioned these methods' robustness to noise and unexpected inputs [3, 28, 14].

To examine whether the Deep Equilibrium approach is brittle to simple changes in the noise distribution, we varied the level of Gaussian noise added to the observations at test time and observed the effect on reconstruction quality in a setting where DE-Prox and DU-Prox perform similarly. Fig. 4.8(b) demonstrates that the Deep Equilibrium model DE-Prox is more robust to variation in the noise level than the analogous Deep Unrolled approach DU-Prox. The forward model used in Fig. 4.8(b) is $8\times$ MRI reconstruction.

4.7 Notes for Practitioners

In this section, we will address certain issues that may arise when implementing the ideas presented in this chapter. In particular, this section will focus on some of the choices made during design and implementation of these algorithms.

Tips for Training

In the work done in this chapter, the same fixed-point solver was used to find both the gradients used for training and for finding the fixed-point. More concisely, the same solver was used for forward and backward passes. It is not clear that this is the optimal way to do things, and there is quite a bit of room in how best to train deep equilibrium models. Recent follow-on work [17] has proposed discarding the

entire inverse-Jacobian calculation, which is equivalent to just backpropagating on the final fixed-point step.

When implementing Anderson acceleration, it is possible to regularize the linear inverse problem by replacing G with $G + \lambda I$. Empirically, this stabilizes training for small λ, like 0.01 or 0.001, but does not seem to improve final reconstruction quality once the network is trained. However, this was not used for the experiments in this paper, for simplicity's sake. The β in Anderson acceleration is almost always set to 1.5 or 2.5, the "over-relaxed" setting.

Denoisers and Plug-and-Play

The role of the denoiser is central to Plug-and-Play algorithms. In particular, if the denoiser is not sufficiently expressive and not trained with sufficiently many images, the results will be poor. A core challenge for plug and play systems is that they are using Gaussian denoisers to correct what is usually highly structured noise. For a concrete example: the aliasing artifacts that arise in subsampled MRI are not spatially invariant, and usually have a great deal of structure that depends on the underlying image. This can present challenges depending on the quality of the denoiser.

Remarkably, it does seem that training a Gaussian denoiser with the right level of noise can (somewhat) sidestep these issues. However, a priori determining the optimal level of noise depends on the forward model and the architecture in question. While for several of the forward models used in this chapter the same denoiser was used for DE-Grad, DE-Prox, and DE-ADMM, that was not universally true.

The takeaways are: for severely corrupted image reconstruction problems, be prepared to spend a lot of time tuning Plug and Play approaches; train several denoisers at a range of noise levels to get the best performance; and choose the right denoiser for the setting and architecture.

Spectral Bounds

The technique used to bound the spectral norm in this chapter is only an approximate upper bound, and appears to limit the capacity of the network used somewhat. An earlier draft utilized normalized U-nets, which ran into trouble on the more complex MRI data, but the current draft is able to use normalized DnCNNs for all settings. Given that the current pretraining followed by tuning procedure is time-consuming, it is an open question whether the techniques used in this paper to train Deep Equilibrium networks for image reconstruction are optimal.

4.8 Conclusions

This chapter illustrates non-trivial quantitative benefits to using implicitly-defined infinite-depth networks for solving linear inverse problems in imaging. Other recent work has focused on such *implicit networks* akin to the deep equilibrium models considered here (e.g. [13]). Whether these models could lead to additional advances in image reconstruction remains an open question for future work. Furthermore, while the exposition in this work focused on *linear* inverse problems, nonlinear inverse problems may be solved with iterative approaches just as well. The conditions under which deep equilibrium methods proposed here may be used on such iterative approaches are an active area of investigation.

Bibliography

[1] Jonas Adler and Ozan Öktem. "Learned primal-dual reconstruction". In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1322–1332.

[2] Hemant K Aggarwal, Merry P Mani, and Mathews Jacob. "MoDL: Model Based Deep Learning Architecture for Inverse Problems". In: *IEEE Transactions on Medical Imaging* (2018).

[3] Vegard Antun et al. "On instabilities of deep learning in image reconstruction and the potential costs of AI". In: *Proceedings of the National Academy of Sciences* (2020).

[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "Deep equilibrium models". In: *Advances in Neural Information Processing Systems*. 2019, pp. 690–701.

[5] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "Trellis networks for sequence modeling". In: *arXiv preprint arXiv:1810.06682* (2018).

[6] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[7] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[8] Stanley H Chan, Xiran Wang, and Omar A Elgendy. "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications". In: *IEEE Transactions on Computational Imaging* 3.1 (2016), pp. 84–98.

[9] Ricky TQ Chen et al. "Neural ordinary differential equations". In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.

[10] Il Yong Chun et al. "Momentum-Net: Fast and convergent iterative neural network for inverse problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[11] Kostadin Dabov et al. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.

[12] Raj Dabre and Atsushi Fujita. "Recurrent stacking of layers for compact neural machine translation models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 6292–6299.

[13] Laurent El Ghaoui et al. "Implicit deep learning". In: *arXiv preprint arXiv:1908.06315* (2019).

[14] Martin Genzel, Jan Macdonald, and Maximilian März. "Solving Inverse Problems With Deep Neural Networks–Robustness Included?" In: *arXiv preprint arXiv:2011.04268* (2020).

[15] Davis Gilton, Greg Ongie, and Rebecca Willett. "Neumann Networks for Linear Inverse Problems in Imaging". In: *IEEE Transactions on Computational Imaging* (2019).

[16] Eldad Haber and Lars Ruthotto. "Stable architectures for deep neural networks". In: *Inverse Problems* 34.1 (2017), p. 014004.

[17] Howard Heaton et al. "Feasibility-based Fixed Point Networks". In: *arXiv preprint arXiv:2104.14090* (2021).

[18] Zico Kolter, David Duvenaud, and Matt Johnson. *Deep Implicit Layers - Neural ODEs, Deep Equilibirum Models, and Beyond*. 2020. URL: http://implicit-layers-tutorial.org/ (visited on 02/01/2021).

[19] Jiaming Liu et al. "Rare: Image reconstruction using deep priors learned without groundtruth". In: *IEEE Journal of Selected Topics in Signal Processing* 14.6 (2020), pp. 1088–1099.

[20] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[21] Morteza Mardani et al. "Neural proximal gradient descent for compressive imaging". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9573–9583.

[22] David Martin et al. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 2. IEEE. 2001, pp. 416–423.

[23] Abolfazl Mehranian and Andrew J Reader. "Model-based deep learning PET image reconstruction using forward-backward splitting expectation maximisation". In: *IEEE Transactions on Radiation and Plasma Medical Sciences* (2020).

[24] Tim Meinhardt et al. "Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1799–1808.

[25] Matthew J Muckley et al. "State-of-the-art Machine Learning MRI Reconstruction in 2020: Results of the Second fastMRI Challenge". In: *arXiv preprint arXiv:2012.06318* (2020).

[26] Gregory Ongie et al. "Deep Learning Techniques for Inverse Problems in Imaging". In: *arXiv preprint arXiv:2005.06001* (2020).

[27] Neal Parikh and Stephen Boyd. "Proximal algorithms". In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[28] Ankit Raj, Yoram Bresler, and Bo Li. "Improving Robustness of Deep-Learning-Based Image Reconstruction". In: *arXiv preprint arXiv:2002.11821* (2020).

[29] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The little engine that could: Regularization by denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[30] 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 (MICCAI)*. Springer. 2015, pp. 234–241.

[31] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[32] Ernest Ryu et al. "Plug-and-Play Methods Provably Converge with Properly Trained Denoisers". In: *International Conference on Machine Learning*. 2019, pp. 5546–5557.

[33] David Strong and Tony Chan. “Edge-preserving and scale-dependent properties of total variation regularization”. In: *Inverse problems* 19.6 (2003), S165.

[34] Andrey Nikolayevich Tikhonov. “On the stability of inverse problems”. In: *Dokl. Akad. Nauk SSSR*. Vol. 39. 1943, pp. 195–198.

[35] Tom Tirer and Raja Giryes. “Super-resolution via image-adapted denoising CNNs: Incorporating external and internal learning”. In: *IEEE Signal Processing Letters* 26.7 (2019), pp. 1080–1084.

[36] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. “Plug-and-play priors for model based reconstruction”. In: *2013 IEEE Global Conference on Signal and Information Processing*. IEEE. 2013, pp. 945–948.

[37] Homer F Walker and Peng Ni. “Anderson acceleration for fixed-point iterations”. In: *SIAM Journal on Numerical Analysis* 49.4 (2011), pp. 1715–1735.

[38] Dufan Wu, Kyungsang Kim, and Quanzheng Li. “Computationally efficient deep neural network for computed tomography image reconstruction”. In: *Medical physics* 46.11 (2019), pp. 4763–4776.

[39] Jure Zbontar et al. “fastMRI: An Open Dataset and Benchmarks for Accelerated MRI”. In: 2018. arXiv: 1811.08839.

[40] Juyong Zhang et al. “Accelerating ADMM for efficient simulation and optimization”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–21.

[41] Kai Zhang et al. “Learning deep CNN denoiser prior for image restoration”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2017.

Chapter 5

Learned Regularizers Using the Relevance Vector Machine for Sparse Linear Bandits

5.1 Learned Regularizers for Online Sparse Regression

Classical multi-armed Bandit (MAB) algorithms are often described in the context of a gambler facing an array of slot machines. Each time one of the machine's arms is pulled, it gives the gambler a reward; the rewards are typically stochastic and identically distributed for each arm, and each arm has its own unknown average payoff. The gambler must devise a strategy for pulling slot machine arms to maximize his/her total rewards from the machines, balancing between exploration (*i.e.*, estimating the average payoff of each arm) and exploitation (*i.e.*, pulling arms known to have large average payoffs). *Linear* MABs (LMABs) assume that each slot machine is associated with a known feature vector, and the expected payoff of the machine will be the inner product of this feature vector and an unknown weight vector. In this setting, a pull of any arm provides some information about the unknown weight vector and hence insight into the average payoff of all the other arms. Classical and linear MABs are applicable in a variety of settings, notably online advertising.

In many contexts, the unknown weight vector is *sparse* because only a few features of the arms are relevant to the expected rewards. However, standard linear bandit strategies do not exploit this sparsity and yield far smaller rewards than oracle LMAB algorithms with full knowledge of the support of the weight vector. Relatively few studies have focused on sparse LMAB. This paper describes a new approach to LMABs that leverages ideas from Linear Thompson Sampling [5] and Relevance Vector Machines [10]. We present our proposed algorithm, theoretical regret bounds, and simulation results illustrating how the proposed approach adapts to sparse weight vectors and outperforms competing algorithms.

Problem formulation

Consider a collection $\mathcal{X}$ of N arms that are represented by d-dimensional feature vectors denoted $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)} \in \mathbb{R}^d$. Our interest is in settings with large d. Without loss of generality, assume the features, $\boldsymbol{x} \in \mathcal{X}$ have ℓ_2 norm at most one, and that also $\|\theta^*\|_2 \leqslant 1$. These assumptions alter any theoretical bounds on performance only by constants. At each time $t = 1, 2, \ldots$, the user selects an arm $\boldsymbol{x}_t \in \mathcal{X}$ and receives a reward $y_t \in \mathbb{R}$. We model the reward as an unknown, noisy linear function of $\boldsymbol{x}_t$; specifically,

$$y_t = \langle \theta^*, \boldsymbol{x}_t \rangle + \eta_t \,, \tag{5.1}$$

where η_t is a noise term that is drawn from a zero-mean Gaussian distribution with variance σ^2 conditioned on $\boldsymbol{x}_{1:t}$ and $\eta_{1:(t-1)}$, and $\theta^* \in \mathbb{R}^d$ is an unknown weight vector reflecting how much different features of the arms contribute to the reward. If we knew θ^*, then we would choose

$$\boldsymbol{x}^* := \arg\max_{\boldsymbol{x} \in \mathcal{X}} \langle \theta^*, \boldsymbol{x} \rangle$$

at each round to maximize expected rewards. However, θ^* is unknown, and estimating θ^* is non-trivial due to the high dimensionality of the arms. This paper assumes that θ^* is s-sparse for some $s \ll d$, meaning that θ^* is supported on at most s unknown indices.

We also define $\mathbf{X}_t := \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_t \end{bmatrix}^\top \in \mathbb{R}^{t \times d}$, $\mathbf{y}_t := \begin{bmatrix} y_1 & \cdots & y_t \end{bmatrix}^\top \in \mathbb{R}^t$, and let

$$\hat{\boldsymbol{\theta}}_t := \left(\mathbf{X}_t^\top \mathbf{X}_t + \lambda \mathbf{I}\right)^{-1} \mathbf{X}_t^\top \mathbf{y}_t := \boldsymbol{\Sigma}_t^{-1} \mathbf{X}_t^\top \mathbf{y}_t \tag{5.2}$$

be the ridge regression estimator of $\boldsymbol{\theta}^*$ given the arms and rewards up to time t, where $\boldsymbol{\Sigma}_t \in \mathbb{R}^{d \times d}$ is defined as above, and $\lambda \in \mathbb{R}$ is a regularizing constant.

Existing linear MAB algorithms provide analysis on their cumulative rewards up to time step T. Define $[T] := \{1, \ldots, T\}$. Instead of guaranteeing an absolute quantity on the cumulative rewards, these algorithms guarantee a relative quantity called *cumulative pseudo-regret*, which measures the extra expected cumulative reward one could have received by pulling $\mathbf{x}^*$ at each round:

$$\mathcal{R}(T) := \sum_{t=1}^{T} \langle \boldsymbol{\theta}^*, \mathbf{x}^* \rangle - \langle \boldsymbol{\theta}^*, \mathbf{x}_t \rangle \,, \tag{5.3}$$

which we refer to as *regret*.

5.2 Related work on linear bandits

The first formal linear stochastic MAB analysis appears in [6] but is under a more general family of problems. Dani *et al.* [8] focus on the linear stochastic MAB case and propose an algorithm based on confidence bounds with a regret bound that is later shown to be optimal up to logarithmic factors by [9].

Optimism in the Face of Uncertainty for Linear Rewards (OFUL)

Abbasi-Yadkori *et al.* [1] propose a significantly tighter confidence bound, which slightly improves the regret bound and improves the practical performance by orders of magnitude. This method is coined "Optimism in the Face of Uncertainty for Linear Rewards (OFUL)" [1]. estimator of $\boldsymbol{\theta}^*$ given the arms and rewards up to time t.

OFUL chooses which arm to pull by solving the following optimization problem:

$$(\tilde{\boldsymbol{\theta}}_t, \mathbf{x}_t^{\text{OFUL}}) := \underset{\boldsymbol{\theta} \in C_{t-1}, \mathbf{x} \in \mathcal{X}}{\arg\max} \langle \boldsymbol{\theta}, \mathbf{x} \rangle \,. \tag{5.4}$$

where $C_{t-1} \subseteq \mathbb{R}^d$ is a confidence set that contains $\boldsymbol{\theta}^*$ with high probability. This approach is "optimistic" in the following sense: we know $\boldsymbol{\theta}^* \in C_{t-1}$ with high probability, but nothing more, so we choose $\tilde{\boldsymbol{\theta}}_t$ to be the weight vector in C_{t-1} that would yield the maximum reward given the best arm pull, and we optimistically use that weight vector to choose $\mathbf{x}_t$.

Define $\|\mathbf{x}\|_A := \sqrt{\mathbf{x}^\top A \mathbf{x}}$. One can fix $\mathbf{x}$ and find the maximizer $\tilde{\boldsymbol{\theta}}(\mathbf{x}) := \max_{\boldsymbol{\theta} \in \mathbf{C}_{t-1}} \langle \boldsymbol{\theta}, \mathbf{x} \rangle$ in a closed form using the Lagrangian method, which simplifies (5.4) to

$$\mathbf{x}_t^{\text{OFUL}} = \underset{\mathbf{x} \in \mathcal{X}}{\arg\max} \langle \hat{\boldsymbol{\theta}}_{t-1}, \mathbf{x} \rangle + \sqrt{\beta_{t-1}} \cdot \|\mathbf{x}\|_{\boldsymbol{\Sigma}_{t-1}^{-1}}. \tag{5.5}$$

for $\beta_{t-1} \in \mathbb{R}$. Details can be found in [1]. The optimization problem (5.5) has a natural interpretation of balancing between the first "exploitation" term that encourages arms aligned with the current ridge

regression estimator $\hat{\theta}_t$ and the second "exploration" term that encourages arms aligned with the directions that are probed less so far.

OFUL achieves the near-optimal regret $\tilde{O}(d\sqrt{T})$ [9], which implies that in the long-term, OFUL will retrieve as large a cumulative reward as one could hope.

Linear Thompson Sampling

Linear Thompson Sampling (LTS) is a linear extension of the Thompson sampling algorithm. Thompson sampling is empirically and theoretically attractive, being simple to implement and near optimal in practice and theory [3]. Departing from OFUL-based algorithms, LTS samples a vector $\tilde{\theta}_t$ from a multivariate normal distribution $\mathcal{N}(\hat{\theta}_t, v^2\Sigma_t^{-1})$ for $v \propto \sqrt{9d\sigma^2 \log(T)} \in \mathbb{R}$ and then chooses the arm

$$x_t^{LTS} = \arg\max_{x \in \mathcal{X}} \langle \tilde{\theta}_t, x \rangle.$$

The regret bound of LTS is $\tilde{O}(d^{3/2}\sqrt{T})$ [4].

Algorithm 4: Linear Thompson Sampling (LTS) [4]

1: Initialize $\hat{\theta}_1 = 0$ and $\Sigma_1 = I_d$
2: Let $\delta \in (0,1)$ and set $v = \sqrt{9d\sigma^2 \log(T/\delta)}$ **for** $t = 1, 2, \ldots, T$ **do**
3:
Sample $\tilde{\theta}_t$ from the distribution $\mathcal{N}(\hat{\theta}_t, v^2\Sigma_t^{-1})$
4: Play arm $x_t := \arg\max_{x \in \mathcal{X}} \langle x, \tilde{\theta}_t \rangle$
5: Receive reward y_t
6: Set $\Sigma_{t+1} = \Sigma_t + x_t x_t^\top = I_d + X_t^\top X_t$
7: Set $\hat{\theta}_{t+1} = \Sigma_{t+1}^{-1} X_t^\top y_t$

LTS and OFUL share some important features. Specifically, both algorithms begin by computing the ridge regression estimate $\hat{\theta}_t$, and encouraging exploration in order to avoid the potential pitfalls of being greedy. OFUL chooses the most optimistic estimate from $\tilde{\theta}_t \in C_{t-1}$, and LTS draws $\tilde{\theta}_t$ from a Gaussian centered at $\hat{\theta}_t$, which has C_{t-1} as a level set. Both algorithms choose the next arm x_t to maximize the inner product between the arm feature vector and their "proxy" weight vector. For large collections of arms, LTS is more computationally efficient, while OFUL has the lower regret bound.

5.3 Sparse linear bandits

The above approaches do not leverage sparsity in the weight vector θ^*. While there is increasing interest in sparse models, there is no consensus on how to best incorporate them within bandit algorithms. Naïvely, we might consider computing a sparse estimate $\hat{\theta}_t$ using LASSO or elastic net regression instead of the ridge regression estimate in (5.2). This approach can yield some empirical advantages, but it is difficult to characterize the confidence set C_t associated with the new sparsity-promoting estimate $\hat{\theta}_t$. Without this confidence set, we cannot improve the scaling of the regret bounds.

Bounding techniques used in standard LASSO analysis rely upon the matrix X_t satisfying certain properties (*e.g.*, the restricted isometry property or restricted eigenvalue condition), but in the bandit setting X_t corresponds to the sequence of arms selected and is not guaranteed to have this property. Two approaches have been proposed to address this challenge.

The first approach [2] assumes the existence of an online learning algorithm that can be used to estimate $\hat{\theta}_t$ (*e.g.*, online LASSO). This algorithm is used to predict y_t and it is assumed that there is a known upper bound on the prediction loss. The predictions and the bound on the prediction error is used to determine a confidence set analogous to C_t in the OFUL algorithm. However, known bounds on online LASSO algorithms are too loose for this method to perform well in practice.

The second approach [7] operates in two epochs. For the first $O(\sqrt{T})$ rounds, random non-adaptive arms are pulled in a "support exploration phase"; based on the rewards from these rounds, the support of θ^* is estimated, and for the remaining rounds the OFUL algorithm is run on the estimated support of θ^*. However, the support exploration phase is non-adaptive and hence this approach can yield smaller cumulative rewards than alternative methods, particularly with the support size s of θ^* is small relative to $\sqrt{T}$. This is illustrated in the simulations below.

5.4 Relevance vector machines

The relevance vector machine (RVM) is a Bayesian framework that is particularly useful in sparse regression and classification tasks, introduced in [10]. The RVM setup considered here assumes we observe

$$y = X\theta^* + \eta,$$

where $\eta \in \mathbb{R}^t$ is a Gaussian noise vector with distribution $\mathcal{N}(0, \sigma^2 I_t)$. The RVM assumes a prior on θ^* of the form

$$\theta^* \sim \mathcal{N}(0, A^{-1}),$$

where $A \in \mathbb{R}^{d\times d}$ is a diagonal matrix of hyperparameters with $A_{i,i} = \alpha_i$; we define $\boldsymbol{\alpha} := \begin{bmatrix}\alpha_1, & \alpha_2, & \ldots, & \alpha_d\end{bmatrix}$ to be the diagonal elements of A. That is, the i^{th} element of θ^* is a zero-mean Gaussian with variance α_i^{-1}.

The RVM further assigns a hyperprior on $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha} \sim \prod_{i=1}^{d} \text{Gamma}(0,0).$$

Using Bayes' rule, the posterior distribution of θ^* is

$$\theta^*|y, \boldsymbol{\alpha} \sim \mathcal{N}(\hat{\theta}, \Sigma^{-1}) \tag{5.6a}$$

where

$$\Sigma = A + \sigma^{-2}X^\top X \tag{5.6b}$$

$$\hat{\theta} = \sigma^{-2}\Sigma^{-1}X^\top y \tag{5.6c}$$

If all the α_i's are equal to λ, then this estimate is equivalent to (5.2). Since $\boldsymbol{\alpha}$ is unknown and computing the full posterior is computationally intensive, [10] proposes choosing $\boldsymbol{\alpha}$ to maximize $p(y|\boldsymbol{\alpha})$ and describes an iterative approach to solving this optimization problem, with computational speedups detailed in [11].

The Relevance Vector Machine is appealing for our task for two reasons. First, as noted in [10], the

RVM tends to produce estimates of $\boldsymbol{\alpha}$ which are very sparse. Secondly, the form of $\hat{\theta}$ in (5.6) is very similar to the ridge regression estimator of (5.2) that is used in LTS and OFUL, so some regret bounds might be feasible to propose.

5.5 Relevance vector machine linear Thompson sampling (RVM-LTS)

In the stochastic linear bandit regime, the RVM is interesting because the posterior distribution in (5.6) can be used within an LTS framework to draw $\tilde{\theta}_t$ (instead of the original $\mathcal{N}(\hat{\theta}_t, v^* \boldsymbol{\Sigma}_t^{-1})$ described in Section 5.2). By using the RVM posterior this way, we essentially form a sampling distribution (with a level set corresponding to a confidence set C_t) that adapts to the unknown sparse support of θ^*.

Algorithm

Our proposed approach is in Algorithm 5; the overall structure is similar to the LTS algorithm described in Section 5.2, with two key differences. First, in line 7, the covariance matrix is offset by diagonal matrix $\mathbf{A}$ instead of the conventional $\lambda \mathbf{I}_d$. Second, in line 6, $\mathbf{A}$ is updated adaptively based on the data acquired so far. It is this adaptation that allows the sparsity of θ^* to be exploited. Note that $\mathbf{A}$ can be updated quickly by using the algorithm outlined by Tipping and Faul [11] [1].

Algorithm 5: Relevance Vector Machine Linear Thompson Sampling (RVM-LTS)

1: Initialize $\hat{\theta}_1 = 0$ and $\mathbf{A} = \boldsymbol{\Sigma}_1 = \mathbf{I}_d$
2: Let $\delta \in (0,1)$ and set $v = \sqrt{9d\sigma^2 \log(T/\delta)}$ **for** $t = 1, 2, \ldots, T$ **do**
3:
Sample $\tilde{\theta}_t$ from the distribution $\mathcal{N}(\hat{\theta}_t, v^2 \boldsymbol{\Sigma}_t^{-1})$
4: Play arm $x_t := \arg\max_{x \in \mathcal{X}} \langle x, \tilde{\theta}_t \rangle$
5: Receive reward y_t
6: Update $\mathbf{A}$ using [11]
7: Set $\boldsymbol{\Sigma}_{t+1} = \boldsymbol{\Sigma}_t + x_t x_t^\top = v^2 \mathbf{A} + \mathbf{X}_t^\top \mathbf{X}_t$
8: Set $\hat{\theta}_{t+1} = \boldsymbol{\Sigma}_{t+1}^{-1} \mathbf{X}_t^\top y_t$

To better understand the sparse case, consider an oracle with perfect knowledge of the support of θ^*, denoted $S := \text{supp}(\theta^*)$. Then if $\alpha_i = \infty$ for $i \notin S$ and $\alpha_i = \lambda$ for $i \in S$, and if we omit updating $\mathbf{A}$ in line 6, then RVM-LTS amounts to LTS on the true support of θ^* and ignoring all other coordinate axes. Of course, in a practical problem we do not have such oracle knowledge. To adapt to this, our method updates its estimate of $\boldsymbol{\alpha}$ using [11].

Regret bounds for fixed A

The above algorithm is, on the surface, very similar to standard Linear Thompson Sampling. However, the addition of the $\mathbf{A}$ matrix instead of $\mathbf{I}_d$ has some significant theoretical consequences.

First, note that within the RVM updating scheme, for $i \notin \text{supp}(\theta^*)$, $\alpha_i \to \infty$. Let

$$S_{\boldsymbol{\alpha}} := \{i : 1/\alpha_i = 0\}$$

[1] A note for practical implementation is added here. The authors of [11] allow α_i to go to infinity, which allows for the omission of that particular dimension from calculations of $\hat{\theta}_t$. Our implementation, for numerical stability, instead sets all α_i that have been set to ∞ by the $\boldsymbol{\alpha}$-updating algorithm to some suitably large $\alpha_{\max}$.

denote the support of the diagonal of $\mathbf{A}^{-1}$ and let $s_{\boldsymbol{\alpha}} := |S_{\boldsymbol{\alpha}}|$ be the size of that support; in practice, we often see $s_{\boldsymbol{\alpha}}$ is close to the true support size of θ^*. Further, let $S = \mathrm{supp}(\theta^*)$ and $\boldsymbol{\alpha}_S$ be $\boldsymbol{\alpha}$ on S and zero elsewhere.

We then have the following regret bound for Algorithm 5 for fixed $\mathbf{A}$ (*i.e.*, when we omit line 6).

Theorem 5.1. *Assume supp*$(\theta^*) \subseteq S_{\boldsymbol{\alpha}}$. *For* $\delta' > 0$, *with probability at least* $1 - \delta'$, *the pseudo-regret of Algorithm 5 for fixed* $\mathbf{A}$ *at time* T *is bounded by*

$$\mathcal{R}(T) = O\Bigg(\min\{\sqrt{s_{\boldsymbol{\alpha}}}, \sqrt{\log N}\} \times \left[\sigma \sqrt{s_{\boldsymbol{\alpha}} \log \frac{T^2 \bar{\boldsymbol{\alpha}} + T^3}{\delta' \sqrt[s]{\boldsymbol{\alpha}}}} + \|\boldsymbol{\alpha}_S\|_2 \right] \Bigg),$$

where $\bar{\boldsymbol{\alpha}} := \frac{1}{s_{\boldsymbol{\alpha}}} \sum_{i \in S_{\boldsymbol{\alpha}}} \alpha_i$ *is the arithmetic mean of* $\boldsymbol{\alpha}$ *and* $\sqrt[s]{\boldsymbol{\alpha}} = (\prod_{i \in S_{\boldsymbol{\alpha}}} \alpha_i)^{1/s_{\boldsymbol{\alpha}}}$ *is the geometric mean of* $\boldsymbol{\alpha}$, *both on the nonzero support.*

Sketch proof: The proof of this theorem follows the main steps of the LTS regret bound in [4, Theorem 1], but accounts for $\mathbf{A}$ by leveraging [1, Lemma 9]. This results in an altered [4, Lemma 8] giving:

$$\|\xi\|_{M_t^{-1}} \leqslant R \sqrt{d \log \frac{\bar{\boldsymbol{\alpha}} + t}{\delta' \sqrt[s]{\boldsymbol{\alpha}}}}.$$

Furthermore, Lemma 1 of [4] bounds $|x_i^\top \hat{\theta}_t - x_i^\top \theta^*|$ for $i = 1, \ldots, N$; this bound must be altered to account for arbitrary $\mathbf{A}$, which results in the $\|\boldsymbol{\alpha}_S\|_2$ term in the final bound. This can be done by noticing that in the proof of [4, Lemma 1] we have now that $\hat{\mu} - \mu = M_{t-1}^{-1}(\xi_{t-1} - \mathbf{A}\mu)$, and propagating this through gives that

$$\|\xi_{t-1} - \mathbf{A}\mu\|_{M_{t-1}^{-1}} \leqslant R \sqrt{d \log \frac{\bar{\boldsymbol{\alpha}} + t}{\delta' \sqrt[s]{\boldsymbol{\alpha}}}} + \|\boldsymbol{\alpha}_S\|_2.$$

The remainder of the proof does require changing the constants l_t and v_t in the proof to fit the above results, but such redefinition does not substantially affect the structure of the proof.

5.6 Simulations

In this section we present several simulations to demonstrate regret performance for the algorithm. First, plots of regret *vs.* number of nonzero elements in $\theta^*(s)$ are provided. A comparison is done with state-of-the-art linear bandit algorithms and another sparse linear bandit algorithm. The effect of noise on the above algorithms is also investigated.

The four main algorithms that we compare are OFUL [1], as outlined in Section 5.2, Linear Thompson Sampling ([4]), as outlined in Section 5.2, our proposed RVM-Linear Thompson Sampling (Algorithm 5, and the sparse linear bandit algorithm of [7], outlined in 5.3, which we refer to as *Epoch SOFUL* (Epoch Sparse OFUL).

In Figure 5.1 we illustrate the per-round regret of these algorithms. The experimental setup consists of $N = 1000$ possible arms with unit-norm feature vectors in $\mathbb{R}^{100}$, which are 5-sparse. The bandit runs for 200 rounds, and the results are averaged over 100 trials. The RVM-LTS algorithm achieves favorable

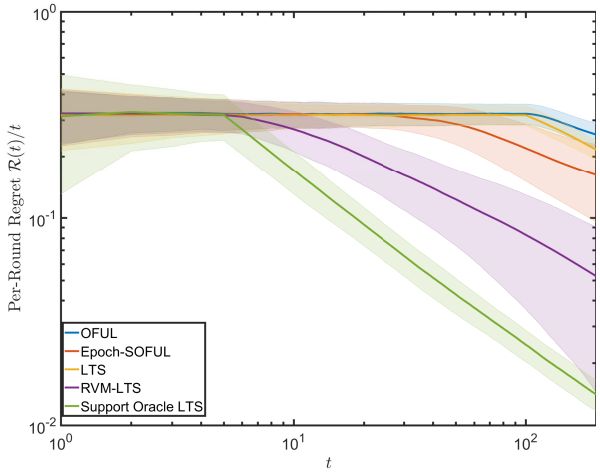


Figure 5.1: The per-round regret for several different bandit algorithms. Here, $s = 5$, $d = 100$, and the number of arms is $N = 1000$. Colorbars represent the standard deviation of the per-round regret, and the center lines are the mean over the 100 trials.

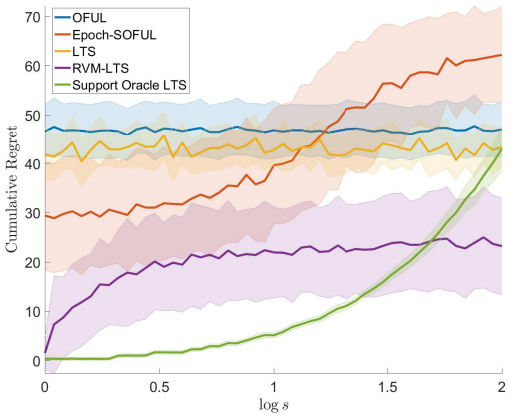


Figure 5.2: The total regret *vs.* s at $T = 200$ for several different bandit algorithms. Here, $d = 100$, the number of arms is $N = 1000$. Each algorithm was run 300 times for each value of s. The shaded regions represent the standard deviation of the cumulative regret, and the center lines are the mean over the 300 trials.

per-round regret, notably beginning its exploitation phase at the same time as the Oracle Thompson Sampling. Because they do not make use of the sparsity of θ^*, standard Thompson Sampling and OFUL require roughly d arm pulls to begin reducing the per-round regret, while Epoch-OFUL pulls arms for a much shorter time before beginning its exploitation phase.

In Figure 5.2, the total regret is plotted *vs.* s, where θ^* is s-sparse. The algorithms compared include the previously identified, plus an "Oracle Thompson Sampling" algorithm. In the oracle algorithm, the bandit is given the true support of θ^*, and runs Linear Thompson Sampling over that true support. As can be seen in Figure 5.2, the oracle should outperform any other algorithm for low s, but is identical to Linear Thompson Sampling when $s = d$. The RVM-LTS regret is closer to the Oracle Thompson Sampling bound for low s, implying that especially for low S, RVM-LTS is performing nearly as well as we could with full prior knowledge of the support.

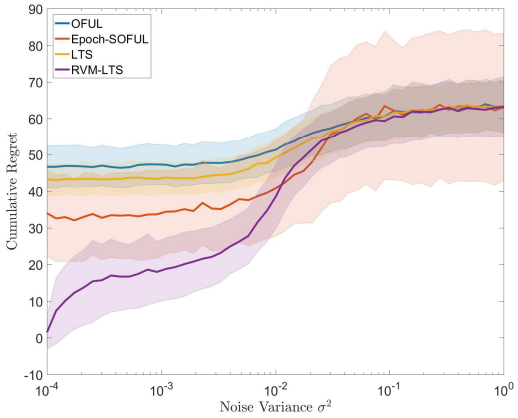


Figure 5.3: The total regret at $T = 200$ *vs.* noise power for several different bandit algorithms. Here, $d = 100$, the number of arms is $N = 1000$. Each algorithm was run 500 times for each value of σ^2. Colorbars represent the standard deviation of the cumulative regret, and the center lines are the mean over the 500 trials.

As s grows, the Epoch SOFUL regret goes up significantly, which is expected of a sparse linear bandit algorithm. Interestingly, the RVM-LTS algorithm achieves lower regret than the oracle as s approaches d, indicating that the RVM-LTS algorithm may have some advantages even in the classical (non-sparse) linear bandit setting.

Finally, Figure 5.3 illustrates the regret response of the different linear bandit algorithms to increasing the variance of the zero-mean Gaussian noise in the rewards. Past $\sigma^2 = 1$, all the algorithms have roughly similar cumulative regret, so no algorithm under consideration appears to have an advantage in the presence of high-variance noise. However, in the low-noise regime, the RVM-LTS algorithm achieves very good regret.

5.7 Wrap-Up

This paper outlines an algorithm that combines the Relevance Vector Machine with Linear Thompson Sampling to yield a sparse linear bandits algorithm that empirically outperforms competing algorithms and is nearly as good as a support oracle. The RVM allows the sparse support of θ^* to be estimated quickly and accurately, so that after relatively few pulls the algorithm performs as well as one operating in a lower-dimensional feature space. As a result, the algorithm is able to home in on the best arms more quickly than conventional algorithms. In addition, we have a preliminary regret bound for fixed $\mathbf{A}$ case.

The empirical results demonstrate that this algorithm achieves better regret than the state-of-the-art linear bandit algorithms, and performs well compared to another sparse bandit algorithm. Perhaps more impressively, the RVM-LTS algorithm achieves regret only slightly worse than the regret obtained by Thompson Sampling over the true sparse support of θ^* for relatively small s. The algorithm responds to noise relatively well, and in the worst noise case appears to be no worse than standard Thompson Sampling in terms of regret.

From a purely practical standpoint, one disadvantage of the RVM-LTS method is the time complexity. Because $\mathbf{A}$ changes with time, rank-one updates to $\mathbf{\Sigma}_t^{-1}$ are no longer possible. Especially in the case

where d is very large, this can lead to long computation times. Further work on the updating process which takes advantage of the structure of the RVM updates could yield computational benefits.

Finally, note that the RVM-LTS algorithm has better regret and higher rewards than other methods even when s is close to d or θ^* was dense. The behavior of the $\mathbf{A}$ updates and of the algorithm as a whole are fairly straightforward in the sparse case, at least in a general sense. However, as θ^* becomes less sparse, the behavior of the RVM updates is less well understood. The fact that the algorithm, at least in simulations, outperforms standard linear bandit methods indicates that further investigation is warranted.

Bibliography

[1] Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. "Improved Algorithms for Linear Stochastic Bandits". In: *Advances in Neural Information Processing Systems (NIPS)* (2011), pp. 1–19.

[2] Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. "Online-to-Confidence-Set Conversions and Application to Sparse Stochastic Bandits." In: *AISTATS*. Vol. 22. 2012, pp. 1–9.

[3] Shipra Agrawal and Navin Goyal. "Analysis of Thompson Sampling for the Multi-armed Bandit Problem." In: *COLT*. 2012, pp. 39–1.

[4] Shipra Agrawal and Navin Goyal. "Thompson Sampling for Contextual Bandits with Linear Payoffs". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2013, pp. 127–135. URL: http://jmlr.org/proceedings/papers/v28/agrawal13.html.

[5] Shipra Agrawal and Navin Goyal. "Thompson Sampling for Contextual Bandits with Linear Payoffs." In: *ICML (3)*. 2013, pp. 127–135.

[6] Peter Auer and M Long. "Using Confidence Bounds for Exploitation-Exploration Trade-offs". In: *Journal of Machine Learning Research* 3 (2002), p. 2002.

[7] Alexandra Carpentier and Rémi Munos. "Bandit Theory meets Compressed Sensing for high dimensional Stochastic Linear Bandit". In: *AISTATS*. 2012, pp. 190–198.

[8] Varsha Dani, Thomas P Hayes, and Sham M Kakade. "Stochastic Linear Optimization under Bandit Feedback." In: *Proceedings of the Conference on Learning Theory (COLT)*. 2008, pp. 355–366.

[9] Paat Rusmevichientong and John N Tsitsiklis. "Linearly Parameterized Bandits". In: *Math. Oper. Res.* 35.2 (2010), pp. 395–411.

[10] Michael E Tipping. "Sparse Bayesian learning and the relevance vector machine". In: *Journal of machine learning research* 1.Jun (2001), pp. 211–244.

[11] Michael E Tipping, Anita C Faul, et al. "Fast marginal likelihood maximisation for sparse Bayesian models." In: *AISTATS*. 2003.

Chapter 6

Learned Change Captioning using Image Sequences

Figure 6.1: A subsequence from "Street Change," a proposed dataset for change detection and description for visual streams. The dotted line denotes the location of a change point, where at least one non-distractor element of the scene has changed. Some human descriptions for the change across the change point are: "The truck is gone." "The construction vehicle is no longer there."

6.1 Introduction

This paper poses and addresses a new challenge in computer vision: detection of meaningful changes in sequences of images. Classical change detection methods model samples before and after a change as drawn from different probability distributions and typically focus on low-dimensional data or known models of the changes of interest. Such tools are inapplicable here. Rather, our approach leverages the probability distributions (implicitly) represented by a learned model to detect changes.

We focus on *visual streams*: time series of images of a scene captured with variable time-lapses between images, under variable camera angles, lighting conditions, and in the midst of other natural changes, such as seasonal changes. This setting is quite different from a video analysis setting, in which we might expect significant consistency between subsequent frames. For instance, a sequence of street-level data (Figure C.2) reflects both changes in the status of road repairs and changes in the weather, where the former is considered meaningful and the latter considered a nuisance to a human observer. Similar challenges arise in diverse applications such as remote sensing, evaluating insurance claims, urban planning, and disaster relief.

We place no *a priori* assumptions on what constitutes a meaningful change. Instead, we aim to learn the relevance of visual change from supervision provided by *natural language descriptions of relevant change*. We show that such language-informed learning improves our ability to localize change in a visual stream. In addition, it provides us with the tools to generate change descriptions, making the detections more interpretable and accessible to human users.

Past work [**shi2020finding**, 33] has explored describing the change between pairs of images. However, when presented with a stream of images, using the outputs of change description methods to localize change in time is nontrivial. We go beyond this setup and develop a novel change detection framework that uses representations learned from both language and visual supervision. These representations are used to (a) construct a graph in which edge weights reflect the likelihood of a change between each pair of images in the stream and (b) quantify the consistency of change descriptions among pairs of graph edges. This leads to a graph-cut algorithm to localize the time of the change.

A prevailing challenge in this approach is the difficulty and expense of collecting visual streams annotated with natural language descriptions of change. On the other hand, unannotated visual streams are plentiful and reflect both meaningful and nuisance changes. To address this challenge, we develop a semi-supervised extension of our method, that uses both a small collection of annotated visual streams and a separate collection of unannotated visual streams to train a system for automatic detection and description of changes.

In summary, our contributions include:

- A new approach to change detection in visual streams, based on a graph cut measure incorporating consistency information.
- A framework that incorporates natural language change annotations into joint training for change detection and change description.
- Two new datasets, one synthetic and one real, of visual streams, annotated with known change points and change descriptions in natural language.

In our experiments, we show that the graph cut change detection mechanism, and the language-informed representation learning in our framework, improve upon baselines for both change detection and change description that do not use these innovations. This is a new, previously unexplored way to leverage the interaction between vision and language.

As an additional contribution, we propose a semi-supervised framework for learning to describe complex human-defined visual changes that may be of independent interest, leveraging unlabeled data to improve performance – an important feature given the cost of labeling visual streams by humans.

6.2 Related work

Change point detection and localization

Change point detection and localization is a classical problem in time series analysis. Typical approaches aim to determine whether and at what time(s) the underlying generative model associated with the signal being recorded has changed [29, 10, 27]. Such techniques are also used in video indexing, to detect a scene cut, a shot boundary, etc. [34, 37, 16, 36, 15, 48].

While more modern methods for visual change detection can deal with certain distractor changes [4], in general such approaches are designed to perform pixel-level detection. These networks require expensive handcrafted ground truth maps for supervised training and are typically not robust to complex combinations of distractors like seasonal variation, lighting changes, and long between-frame times. Much current work on high-dimensional change point detection (*e.g.,* [5, 42]) assumes that the time series has a mean that is piecewise-constant over time; this is violated by real-world data, such as shown in Fig C.2.

Automatic description of the visual world

While automatic image description (or "captioning") has received growing attention in the computer vision community in the past several years [46, 18, 8, 25], such tools are inadequate for the detection of changes in a visual stream. Describing *changes* in a scene is fundamentally different from recognizing objects in an image. Unlike in classical automatic image captioning, a visual stream may contain no relevant changes and it may contain multiple objects that do not change. Furthermore, video captioning methods [40, 30, 43] are inapplicable because in many visual streams there are relatively large periods of time between subsequent images; for instance, a satellite may pass over a location only once per day. Finally, the images of interest may be corrupted by blur, shifts, and varying environmental conditions that make identifying changes challenging.

Visual descriptions. The task of describing the content of an image or video in natural language, often referred to as *image/video captioning*, has emerged as one of the central tasks in computer vision in

recent years [41, 7, 18, 40, 35, 49, 20]. Typically, captioning is approached as a translation task, mapping an input image to a sequence of words. In most existing work, learning is fully supervised, using existing data sets [47, 24, 21, 45] of "natural images/videos" with associated descriptions. However, when faced with drastically different visual content (*e.g.*, remote sensing or surveillance videos) and a different type of descriptions (in our case describing change), models trained on natural data lack both the visual "understanding" of the input and an appropriate vocabulary for the output. We address this limitation by leveraging novel semi-supervised learning methods, requiring only a small amount of labeled data.

Automatic description of change. Change is implicitly handled in methods for captioning *video* [30, 43, 31, 3]. However, when describing changes in the videos, these methods tend to focus on describing the "updated scene" rather than the nature of the change between the scenes.

The most related work to ours are [**shi2020finding**, 17, 33, 28], which also attempt to generate descriptions for human-defined changes in images. [17] proposes the Spot-the-Diff dataset, where pairs of images are extracted from surveillance videos. The viewpoints are fixed and there is always some change. [33] proposes CLEVR-Change, a dataset where pairs of synthesized images are provided with synthesized change descriptions. Lighting and viewpoint can vary between image pairs. [**shi2020finding**] improves on [33] by introducing an architecture that explicitly disentangles viewpoint changes from other changes between two images. We extend [33] in two ways. First, we consider a series of images instead of two, where a change can possibly happen at any timestep. Secondly, we consider a semi-supervised setting where only a subset of image series has associated change descriptions annotated. Moreover, we report results not only on CLEVR-change and Spot-the-Diff, but also a newly collected street-view dataset.

Semi- and unsupervised captioning. The goal of unsupervised or unpaired captioning is to learn to caption from a set of images and a corpus, with no mapping between the two. Usually, some pivoting is needed: [11, 12] tackle unpaired captioning with a third modality: Chinese captions and scene graphs respectively. Similarly, [9, 22] use visual concept detectors trained on external data to bridge the unpaired images and captions.

In the semi-supervised setting, extra unlabeled data supplement paired image caption data. [**liu2018show**] uses self-retrieval to mine negative samples from unlabeled images to improve captioning performance. [19] uses a GAN to generate pseudo labels, making extra unpaired images and captions paired. [2, 39] exploit additional unpaired text and image for novel object captioning.

Our work is similar to [**liu2018show**] but there are several important differences. First, we use a real-fake discriminator instead of a retrieval-based discriminator. Second, the datasets/tasks are different. Our datasets are more under-annotated and out-of-domain than COCO[24], a large natural image dataset that can benefit easily from pretrained vision networks. Note that, despite the similar technique, our focus in this paper is not to propose a new semi-supervised technique, but a more general setting of change descriptions in an annotation-restrained situation.

6.3 Change detection as a graph cut

Assume we are given a visual stream: an ordered sequence of images $\{I_t\}_{t=0}^{T}$. Nuisance changes, such as viewpoint, lighting, or weather variation, may be observed between any pairs of frames of the stream. There is some unknown $\tau \in \{0, \ldots, T-1\}$ such that between $t = \tau$ and $t' = \tau + 1$ a relevant (non-nuisance) change occurs. Our goal in change detection is to estimate τ.

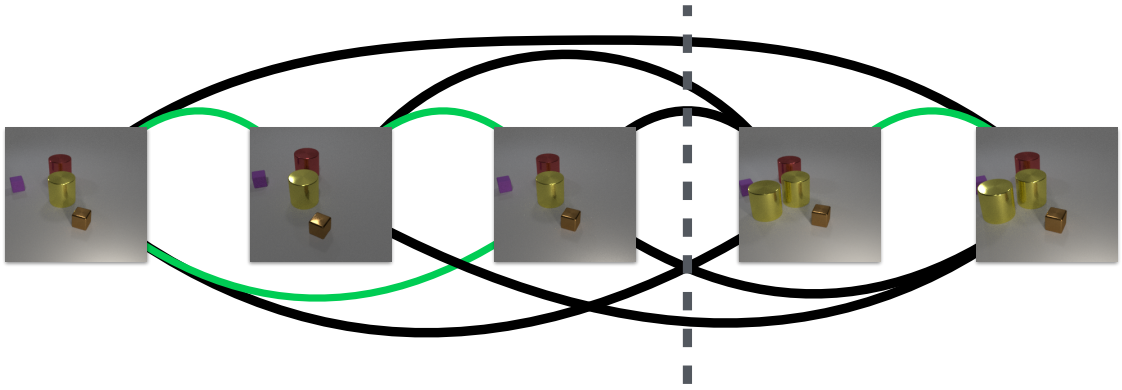
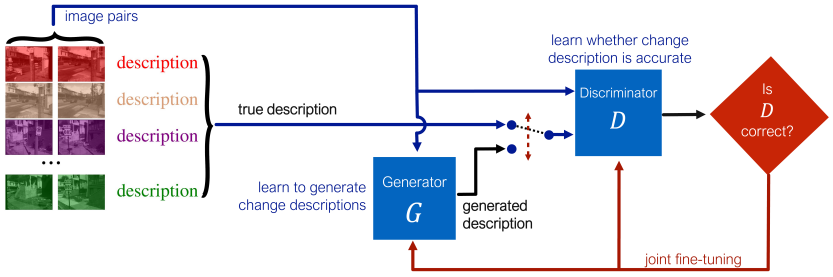


Figure 6.2: Visual representation of graph given by a time series. Selecting a changepoint τ (the dotted line), partitions the edges into E_τ, in black, and E^c_τ, in green.

Figure 6.3: System diagram. G generates descriptions for an input image pair $(I_t, I_{t'})$, and D predicts whether a candidate text string is an accurate description of the change between I_t and $I_{t'}$. The generator G and discriminator D are trained jointly to account for limited quantities of annotated training data.



All the approaches we will discuss rely on two "building blocks", which operate on an image pair $(I_t, I_{t'})$:

- a change score $P(I_t, I_{t'})$: a statistic which is high for pairs $(I_t, I_{t'})$ where a change has occurred between t and t', and is low otherwise; and
- a latent representation of the image pair, denoted $h(I_t, I_{t'})$.

The change score $P(I_t, I_{t'})$ could, for instance, be derived from the distance between feature vectors extracted from the two images (perhaps modulated by an attention mechanism), or, as we will see later, from the likelihood that the description "no change" is accurate for the image pair, as computed by visual change description systems like [**liu2018show**]. A simple choice for $h(I_t, I_{t'})$ could be the concatenation of a representation of I_t and $I_{t'}$. We propose an $h(I_t, I_{t'})$ that depends on the internal representation of a neural network that has learned to describe the change between $h(I_t, I_{t'})$.

We can use P and h to define different *score functions* for various candidate values of τ in a visual stream. We will compute an estimate $\hat{\tau}$ by maximizing these score functions: for any method which generates a score $S_X(\tau)$, the detected changepoint is $\tau_X = \arg\max_\tau S_X(\tau)$. To make this concrete, we describe these change measures in the context of language-informed representations in Section 6.4.

Step-wise scores. This approach finds the *consecutive* pair of frames in the stream which is most likely to contain a change between that pair, according to P. We define the step-wise score as $S_{\text{step}}(\tau) :=$

$P(I_{\tau-1}, I_\tau)$ and the associated change point estimate as $\tau_{\text{step}} = \arg\max_\tau S_{\text{step}}(\tau)$. Unfortunately, if P is noisy or imperfect, the step-wise approach may be prone to errors and spurious detections.

Graph cut scores. To address the noise sensitivity of the step-wise approach, we propose a strategy based on graph cuts. Consider the complete graph with vertices for all frames, $t \in \{0, \ldots, T-1\}$. A candidate change point τ induces a graph cut, where the edges that are cut connect some $t \leqslant \tau$ and $t' > \tau$; we denote the set of these edges as E_τ. A visualization is provided in Figure 6.2, where a candidate τ results in an edge partitioning into E_τ, in black, and its complement E_τ^c, in green.

We assign an edge weight given by P to each pair of nodes t and t', and define a score function for the graph cut induced by τ:

$$S_{\text{GC}}(\tau) = \frac{1}{|E_\tau|} \sum_{(t,t') \in E_\tau} P(I_t, I_{t'}) - \frac{1}{|E_\tau^c|} \sum_{(t,t') \in E_\tau^c} P(I_t, I_{t'}). \tag{6.1}$$

The value of S_{GC} is high if the change score is high for frame pairs straddling the hypothesized change point, compared to the score for pairs that are on the same side of the change. While graph cut problems are often computationally prohibitive, in this setting, there are only $T-1$ cuts (corresponding to $T-1$ candidate values of τ) that are considered.

Consistency scores. The scores above depend only on the pairwise relation $P(I_t, I_{t'})$. However, learned hidden representations $h(I_t, I_{t'})$ may contain more information relevant to change between t and t'. In particular, we may want to ensure that all the pairs $(I_t, I_{t'})$ with $t \leqslant \tau$ and $t' > \tau$ have similar representations (since they should correspond to the same change). Informally, if τ is the true change point, then we may expect that all pairs $(I_t, I_{t'})$ which straddle τ will result in a similar joint representation, if that representation is the result of training to be used for detecting or describing changes.

We measure the similarity between joint representations of two frame pairs, $h(I_t, I_{t'})$ and $h(I_s, I_{s'})$ using cosine similarity. This leads to a "consistency score":

$$S_{\text{CS}}(\tau) = \sum_{\substack{(t,t') \in E_\tau \\ (s,s') \in E_\tau}} \cos\left[h(I_t, I_{t'}), h(I_s, I_{s'})\right]. \tag{6.2}$$

The consistency score S_{CS} sums over all *pairs* of black edges in Figure 6.2, finding the average similarity of the associated hidden representations.

Consistency-regularized graph cut scores. Ideas from both the graph cut score and the consistency score can be combined to form a consistency-regularized graph cut score or "regularized cut" (RC):

$$S_{\text{RC}}^{(\beta)}(\tau) = \beta S_{\text{GC}}(\tau) + S_{\text{CS}}(\tau) \tag{6.3}$$

for a user-specified tuning parameter $\beta \geqslant 0$. The consistency score (6.2) is equivalent to S_{RC} (6.3) with $\beta = 0$.

6.4 Language-informed change representation

Until now we have used the ground truth indicating the change point as the only source of supervision regarding meaningful changes in training data. We now introduce an additional source: Suppose we have L image pairs *labeled* with change descriptions, $\left(I^{\ell}_{t'}, I^{\ell}_{t}, w^{\ell}\right), \ell = 1, \ldots, L$. The natural language expression w^{ℓ} describes the relevant change between $I^{\ell}_{t'}$ and I^{ℓ}_{t}, or indicates "no change".This suggests a new task: learning to *describe the change*. We next explain our approach to this task and then discuss how language supervision may allow us to learn better frame pair representations to use in graph-cut-based change detection.

Learning to generate change description

We train a conditional generative model G that allows us to calculate the likelihood $p(w|I_t, I_{t'})$ of a change description w given an image pair, as well as sample (estimate) a description $\hat{w}_G$ for a given image pair. As part of its operation, $G(I_{t'}, I_t)$ computes a hidden state $h(I_{t'}, I_t)$, which we can use for change detection (see Section 6.3).

We also train (jointly with the generator G) a *discriminator* D. For two images I_t and $I_{t'}$ and description w, this discriminator computes a "compatibility score" $D(w, I_t, I_{t'})$, which is an estimate of the binary variable $\mathbb{P}(w, I_t, I_{t'})$ indicating if w is a valid description of the change between I_t and $I_{t'}$. We introduce D for two reasons: it provides an adversarial loss, in addition to the traditional cross-entropy loss, which improves the learning of G and allows for a semi-supervised extension. It also provides us with a language-informed pairwise change score: two frames are likely to convey a change if D assigns low probability for "no change" as their description.

We first pretrain G and D, then fine-tune them jointly. Pretraining of the change description generator G proceeds by maximum likelihood as outlined in [33]. We diverge from their formulation by describing pairs of images with no relevant change with a special "nochange" token, rather than full sentences like "Nothing happened," or "No change." For all other pretraining details, see [33].

To train the discriminator, we minimize the loss:

$$L_D := -\sum_{\ell=1}^{L} \log D(w^{\ell}, I^{\ell}_{t}, I^{\ell}_{t'}) \tag{6.4}$$

$$- \sum_{\ell \neq \ell'} \log(1 - D(w^{\ell}, I^{\ell'}_{t}, I^{\ell'}_{t'})). \tag{6.5}$$

The sum in (6.4) is over labeled pairs with known change annotations; these are the positive examples for D. Note that these may include pairs with no change, annotated by $w =$"nochange". The second sum in (6.5) is over examples obtained by matching one pair with the annotation for another pair. Here we assume that, with high probability, such a randomly matched annotation will not be valid for the given pair, and so these are negative examples for D.

During the joint fine-tuning, the objective for D is to attempt to distinguish descriptions generated by G and true natural language annotations. The objective for G is to generate change descriptions that the discriminator labels as "valid," while still staying close to the ground truth data. To this end, we simultaneously maximize the likelihood of the ground truth and the outputs of the discriminator for descriptions generated by G. We use the following loss, with L_G being the loss used during pretraining

G:

$$L_G^{\text{joint}} := \sum_{\ell=1}^{L} L_G(w^\ell, I_t^\ell, I_{t'}^\ell) \tag{6.6}$$

$$- \lambda_G \sum_{\ell=1}^{L} \log D(G(I_t, I_{t'}), I_t, I_{t'}). \tag{6.7}$$

In (6.7) we pass to D descriptions sampled from G, $G(I_t, I_{t'})$. Since this sampling is not differentiable, we treat training G using D as a reinforcement learning problem, which we solve using REINFORCE [44]. This approach has been used elsewhere for learning in sequence generation [26, 14].

Here the policy is given by G, parametrized by the network parameters θ_G, which predicts a distribution $p_G(w|I_t, I_{t'})$ over descriptions. We wish to maximize some reward, $R(w, I_t, I_{t'})$, which should be high for valid change descriptions and their associated images. The gradient of the expected reward for G is approximated by:

$$\nabla_G \mathbb{E}_w \left[R(\hat{w}_G, I_t, I_{t'})\right] \approx R(\hat{w}_G, I_t, I_{t'}) \nabla_{\theta_G} \log p_G(\hat{w}_G | I_t, I_{t'}),$$

where $\hat{w}_G$ is the result of sampling from $p_G(w|I_t, I_{t'})$, and $p_G(\hat{w}_G|I_t, I_{t'})$ is defined by the output of G. In our setting, $R(\hat{w}_G, I_t, I_{t'})$ is given by $\log D(\hat{w}_G, I_t, I_{t'})$. This permits the full gradient of L_G^{joint} to be approximated at training points. The loss for D during joint training becomes

$$L_D^{\text{joint}} := L_D + \lambda_D \sum_{\ell=1}^{L} \log D(G(I_t^\ell, I_{t'}^\ell), I_t^\ell, I_{t'}^\ell) \tag{6.8}$$

where we retain the pretraining loss L_D to ensure that D continues to perform well on real descriptions. Reinforcement learning is not needed during this phase, since the sampled descriptions from G can be treated as constant inputs and are not backpropagated through.

Language-informed change scores

Computing change statistics The change detection scores described in Section 6.3 relied on two key elements: $P(I_t, I_{t'})$, which reflects the likelihood that a change occurred between times t and t′, and $h(I_t, I_{t'})$, a latent representation of the pair. Here, we describe how the framework depicted in Figure 6.3 and described at the beginning of Section 6.4 can be used to compute these quantities, as well as a standalone, "image-only" network trained specially for the task that we use as a baseline for comparisons. First, we note that by fixing the test description to the "nochange" token, we can define:

$$P(I_t, I_{t'}) := -D(\text{“nochange”}, I_t, I_{t'}) \tag{6.9}$$

which uses the discriminator's predicted label as a measure of how poorly the "nochange" description fits the pair of images I_t and $I_{t'}$. Furthermore, we let the representation of a pair, $h(I_t, I_{t'})$, be the final hidden state of the description generator $G(I_t, I_{t'})$.

Now a natural interpretation of the scores in Section 6.3 arises: we seek a change point $\hat{\tau}$ such that for all pairs $(I_t, I_{t'})$, where $t \leqslant \tau$ but $t' > \tau$, D("no change", $I_t, I_{t'}$) is low. In addition, the representation

consistency term encourages us to choose a change point where the generator G will output similar descriptions of the change that has occurred across τ for all possible pairs. In this formulation, the language model becomes an integral part of the change point detection method.

Language-free variant for comparison The above methods rely only on a statistic $P(I_t, I_{t'})$ that quantifies the likelihood of there being a meaningful change between I_t and $I_{t'}$ and a latent representation of the pair, $h(I_t, I_{t'})$. We described above how these quantities could be computed using a change description framework. However, P and h might also be computed by an analogous system trained only to predict whether $(I_t, I_{t'})$ reflects a meaningful change *without natural language annotations on the samples*. We construct such a system to serve as a baseline method, allowing us to assess the change detection performance impact of using natural language annotations during training.

Specifically, we define $\tilde{D}(I_t, I_{t'})$ as an "image-only" change detector, trained to classify whether pairs contain a change or not using a binary cross-entropy loss, where a positive label indicates no change. In our implementation, $\tilde{D}(I_t, I_{t'})$ learns to produce visual features using the same architecture to extract visual features as the generator G. The visual features are passed through several fully-connected layers, and the output of $\tilde{D}(I_t, I_{t'})$ is a single label. Now we can define an image-only change score:

$$\tilde{P}(I_t, I_{t'}) := -\tilde{D}(I_t, I_{t'}), \tag{6.10}$$

where $\tilde{P}(I_t, I_{t'})$ is an estimate of the probability that a change has occurred between t and t' that has been trained *without using language information*. This yields an "image-only" version of S_{step} and S_{GC} (6.1) where $\tilde{P}$ is used instead of P. We can also define an "image-only representation consistency" used in S_{CS} (6.2) and S_{RC} (6.3), but using the penultimate layer of the network used to compute $\tilde{D}(I_t, I_{t'})$ as the latent $h(I_t, I_{t'})$.

Semi-supervised learning

Since obtaining natural language annotations for change is much more expensive than collecting just the raw data with a known change point, we consider a semi-supervised extension. In addition to the labeled data, a set of U *unlabeled* pairs $(I^u_{t'}, I^u_t), u = 1, \ldots, U$ is presumed to only contain pairs where a change does occur between $I^u_{t'}$ and I^u_t, although no description of this change is available.

During pretraining, we add a term to the discriminator loss in (6.4) and (6.5):

$$L_D \;=\; (6.4) + (6.5) - \sum_{\ell,u} \log(1 - D(w^\ell, I^u_t, I^u_{t'})) \tag{6.11}$$

This expression matches annotations from labeled data to unlabeled pairs, which we assume is unlikely to create valid descriptions. Since the labeled pairs may be pairs with no relevant change, we ensure the unlabeled pairs contain some relevant change so "nochange" is not a valid description. This procedure uses un-annotated data to create more negative examples for D. Finally, during joint fine-tuning of G and D, we modify the sums in (6.7) and in (6.8) to be over both labeled and unlabeled examples. Thus, the un-annotated data are incorporated into training both G and D.

6.5 Experiments

Datasets: change description CLEVR-Change (introduced in [33]) is a synthetic dataset, with a variety of distractor changes and a curated set of "relevant" changes. Each image depicts colored geometric primitives on a flat, infinite ground plane (see Fig. 6.2 for an example). CLEVR-Change includes, for each initial frame, a second frame in which only nuisance changes have occurred, as well as a second frame in which both nuisance and relevant changes have occurred. We augmented the roughly 39000 change/distractor pairs in the CLEVR-change dataset with 10000 unlabeled pairs of images, using the original generation procedure. These additional unlabeled pairs always contain a change.

Spot-the-Diff [17] is a video surveillance change description dataset that consists of pairs of aligned surveillance images and associated human-generated captions for the visual changes in each pair. Each pair is assumed to contain a change, with similar lighting conditions between the "before" and "after" frames, so this dataset does not contain the nuisance changes in contrast to CLEVR-Change.

Datasets: visual streams Due to the lack of standard datasets for our task, we propose two datasets for evaluation of captioning and changepoint detection on visual streams.

The first dataset we introduce is a modification of CLEVR-Change. We adapt the generation procedure in [33] to generate a visual stream instead of a pair of images. Over the course of a sequence, the camera follows a random walk, while the light intensity also follows a random walk, to encourage robustness to nuisance changes. At some time in the sequence, one of the non-nuisance change types from the original CLEVR-Change occurs. Each sequence contains 10 images, with changepoints uniformly distributed over the dataset. See the supplementary materials for further details.

We also propose the "Street Change" dataset, an extension of the VL-CMU dataset [**badino2012real**], which consists of 151 image sequences captured in an urban environment during different seasons and times of day. Each sequence contains multiple views of a scene that are captured at an initial date, and then a second subsequence that is captured at a later date. The original change detection dataset was curated by [1], and we have engaged human labelers from Amazon's Mechanical Turk to provide change annotations for training. There are at least three captions for both the original *and time-reversed* sequences, which were curated after collection to ensure high-quality labels.

The "Street Change" dataset contains both minor and major nuisance changes across the dataset. Frames are captured from many different viewpoints, and there are often lighting changes across the sequence. Some sequences span seasons, introducing changes in vegetation, weather, etc., which a model must learn to ignore. An example sequence is in Fig. C.2, along with associated human annotations. More examples are included in the supplementary materials.

The goals of our experiments are an assessment of the proposed system of change description, of the role language supervision may play in improving change detection, and of the effect of unlabeled data on the performance. We find that the RC approach, which leverages both D and the generative language model G, outperforms other proposed changepoint detection methods and provides superior change descriptions, with both the representation consistency term and the use of language in training contributing to its performance.

Implementation details Before presenting our main results, we outline the architecture choices for G and D. G uses the DUDA architecture introduced in [35], which can be trained to be robust to lighting and pose changes. Briefly, visual features from I_t and $I_{t'}$ are extracted by a pretrained ResNet

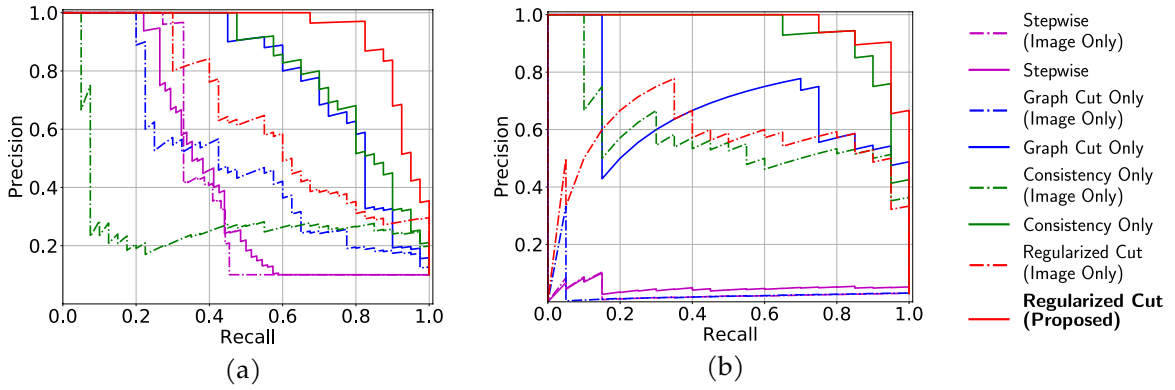


Figure 6.4: Precision-recall curves for (a) CLEVR-Sequence, and (b) Street Change. Methods compared are change detection via Stepwise Comparisons, change detection with Consistency Only, change detection with Graph Cut Only, and change detection via the **Regularized Cut (Proposed)**. Methods denoted with dotted lines do not leverage language annotations during training (Image Only). Adding language to training and leveraging the internal language representations generated by a change description system improve change detection performance in visual streams significantly on both datasets.

[13]. These features are added together based on a temporal attention mechanism, which permits the network to attend to I_t and $I_{t'}$ in a dynamic fashion as the caption-generating LSTM evolves. The LSTM outputs a hidden state, which is used to estimate a probability distribution over words at each time step $1, \ldots, K$. G is pretrained by standard maximum-likelihood methods and teacher forcing.

The discriminator D shares the architecture of DUDA with G. However, instead of using the LSTM model to estimate the word sequence $w = (w_1, \ldots, w_K)$ given $(I_t, I_{t'})$, in D the LSTM model takes w as an input. The final LSTM state h_K is fed to a multi-layer fully connected classifier. The output $D(w, I_t, I_{t'})$ attempts to classify if the input w is a valid description of change from I_t to $I_{t'}$. D is pretrained with a standard cross-entropy loss.

The three generators differ architecturally only in the dimension of the final output of G, $p_G(w|I_t, I_{t'})$, which is 64, 2404, and 420 for CLEVR, Spot-the-diff, and Street Change respectively. At test time, descriptions are sampled greedily. See the supplementary material for further details.

Both G and D are pretrained for 40 epochs using Adam [**adam**]. Joint training proceeds for 20 epochs. The learning rate begins at 0.001 and decays by a factor of 0.9 every 5 epochs. λ in L_3 is set to 0.001. The image-only discriminator $\tilde{D}$ is trained for 40 epochs using Adam with a learning rate of 0.001. To choose β in S_{RC} (Eq. 6.3), we used a held-out set of 20 sequences from CLEVR Sequence. We use $\beta = 0.1$ for both CLEVR Sequence and Street Change.

Evaluation To evaluate the quality of change detection methods, we report precision-recall curves in Figure 6.4. For description tasks, we report BLEU-4 [32], CIDEr [38], ROUGE-L [23], and METEOR [6]. BLEU-4 and CIDER scores across different labeled and unlabeled training set sizes are displayed in Figure 6.5.

Change detection and ablation study We begin by reporting changepoint detection results on CLEVR Sequence and Street Change in Figures 6.4. In addition, we compare to several ablations of the proposed method.

First, we compare our proposed method S_{RC} (6.3) to baselines that use only one of its components, either the Consistency score S_{CS} (6.2) or the Graph Cut score (6.1), and to the more naive Stepwise score S_{step}.

We also assess the importance of language-informed representations, by constructing an "image-only" counterpart for each of the baselines, as outlined in Section 6.4. These methods use $\tilde{P}$ in (6.10) to calculate scores, and the penultimate layer of $\tilde{D}(I_t, I_{t'})$ for hidden representations $\tilde{h}(I_t, I_{t'})$. This is in contrast to our language-informed methods that rely on the machinery learned in the context of change descriptions to derive P and h.

We observe that traditional statistical methods for change detection, based on density estimation, are not applicable here, due to the high dimensionality of image features, scarce data, and the need to separate nuisance change from meaningful change. Our "image only" ablations can be seen as an adaptation of such traditional methods using modern learned representations.

We observe a consistent improvement from image-only methods to their counterparts that use a language-informed D. The results indicate that incorporating language into the model may provide a more direct signal for focusing representation learning on meaningful changes. In addition, the Graph Cut and Consistency scores both improve detection compared to stepwise approaches; combining the two leads to further improvements.

Adapting ST&D to change description. An alternative approach to recognizing valid descriptions called Show, Tell, and Discriminate (ST&D) is proposed in [**liu2018show**] for single-image captioning. It embeds the descriptions w and images I in a common space and attempts to increase the cosine similarity of the embeddings $s(w, I)$. The embedding module is trained with a hard triplet loss on labeled data: $L_{ST\&D} = \sum_{i=1}^{L} \max_{i \neq j} \left(\alpha - s(w^i, I^i) + s(w^i, I^j)\right)_+$ for margin α. Joint training proceeds similarly to our approach, with $L_{ST\&D}$ providing the discriminative reward to G.

We have modified ST&D to perform change description, so the embedding module operates on pairs of images rather than single images. Since our change detection method does not require any particular architecture to function, we note that this adapted approach can be used for change detection. In this case, the pairwise statistic $P(I_t, I_{t'})$ is the cosine similarity between the embedding of the pair $(I_t, I_{t'})$ and the "nochange" token. The generator's hidden vector is used for the Description Consistency term. Further implementation details are available in the supplement.

Change description performance comparison We also evaluate the change description performance of our generator-discriminator architecture. We compare with DUDA [33], Finding it at Another Side (FAS) [**shi2020finding**], and our modified ST&D [**liu2018show**] on the CLEVR-Change (Table 6.1) and Spot-the-Diff (Table 6.2) datasets. We observe that our approach consistently performs favorably to all other methods across datasets and metrics.

For Street Change, we compare our proposed approach to the captions produced by the modified ST&D method (Table 6.3). We again observe competitive results to the existing method. Since Street Change is a set of visual streams rather than pairs, we generate descriptions from random pairs of images across the changepoints.

Semi-supervised learning of change description Finally, we explore the performance of the proposed semi-supervised change captioning scheme. We compare our semisupervised approach to an adapted version of ST&D, which learns a joint embedding of image pairs and captions, rather than

Metric	BLEU-4	METEOR	ROUGE	CIDER
DUDA [33]	47.3	33.9	70.8	112.3
ST&D [**liu2018show**]	56.4	40.2	**73.3**	106.4
FAS [**shi2020finding**]	51.3	37.8	70.4	**115.8**
Ours	**56.8**	**40.7**	72.9	109.5

Table 6.1: Comparisons of performance on the change captioning subtask on the CLEVR-Change [33] dataset, using a variety of standard language metrics.

Metric	BLEU-4	METEOR	ROUGE	CIDER
DUDA [33]	8.1	11.5	28.3	34.0
ST&D [**liu2018show**]	13.0	13.5	30.8	47.7
FAS [**shi2020finding**]	11.1	12.9	33.2	42.5
Ours	**15.8**	**14.0**	**33.3**	**51.4**

Table 6.2: Comparisons of performance on the change captioning subtask on the Spot-the-Diff [17] dataset.

Metric	BLEU-4	METEOR	ROUGE	CIDER
ST&D	39.1	39.2	74.2	119.3
Ours	**42.9**	**39.9**	**74.6**	**124.1**

Table 6.3: Comparisons of performance on the change captioning subtask on the introduced Street Change dataset.

learning a discriminator. Figure 6.5 explores a range of training set sizes, varying the number of unlabeled image pairs for a set of 10000 labeled image pairs. We see consistent improvements from unlabeled data; our approach appears to use additional unlabeled data more effectively, outperforming the adapted ST&D approach.

We provide further plots across a range of labeled training set sizes in the Supplementary Material, along with similar plots for Spot-the-Diff and Street Change.

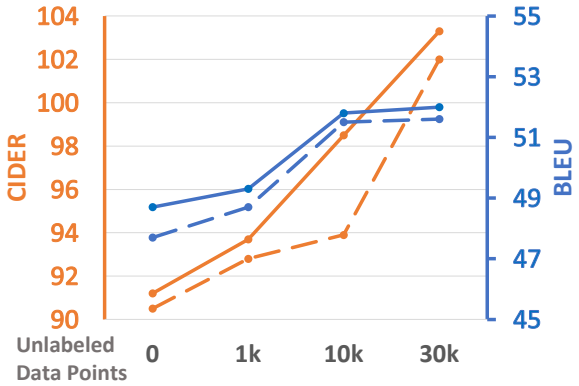


Figure 6.5: Illustrating the effect of additional unlabeled samples for the CLEVR-Change change captioning subtask. We observe that additional unlabeled examples improve change captioning performance significantly. The number of pairs with known captions is 10000. Dashed lines represent the performance of ST&D adapted to image pairs, whereas solid lines are our proposed method. Further plots are available in the Supplementary Materials.

6.6 Conclusions

This paper explores learned change detection in visual streams using training data that has been annotated with natural language descriptions of the interesting changes. Our novel change detection framework is based on a graph cut algorithm that uses a learned model of the likelihood of a change between two images and a consistency score that leverages learned latent representations of image pairs. While natural language labels have a strong positive impact on change detection performance, we recognize that labeled training data is often difficult and expensive to obtain. With this challenge in mind, we develop a semi-supervised training regimen that fully leverages unlabeled data. A broad array of experiments on newly developed datasets consisting of visual streams with natural language labels help quantify the performance gains associated with different amounts of unlabeled data on both captioning and detection. A core foundational challenge moving forward is to develop a broader understanding of the role of language annotations and models in facilitating change detection.

Bibliography

[1] Pablo F Alcantarilla et al. "Street-view change detection with deconvolutional networks". In: *Autonomous Robots* 42.7 (2018), pp. 1301–1322.

[2] Lisa Anne Hendricks et al. "Deep compositional captioning: Describing novel object categories without paired training data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[3] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. "Hierarchical boundary-aware neural encoder for video captioning". In: 2017.

[4] Pierre-Luc St-Charles, Guillaume-Alexandre Bilodeau, and Robert Bergevin. "Subsense: A universal change detection method with local adaptive sensitivity". In: *IEEE Transactions on Image Processing* 24.1 (2014), pp. 359–373.

[5] Haeran Cho and Piotr Fryzlewicz. "Multiple-change-point detection for high dimensional time series via sparsified binary segmentation". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 77.2 (2015), pp. 475–507.

[6] Michael Denkowski and Alon Lavie. "Meteor universal: Language specific translation evaluation for any target language". In: *Proceedings of the ninth workshop on statistical machine translation*. 2014, pp. 376–380.

[7] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[8] Hao Fang et al. "From captions to visual concepts and back". In: 2015.

[9] Yang Feng et al. "Unsupervised image captioning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[10] Damien Garreau, Sylvain Arlot, et al. "Consistent change-point detection with kernels". In: *Electronic Journal of Statistics* 12.2 (2018), pp. 4440–4486.

[11] Jiuxiang Gu et al. "Unpaired image captioning by language pivoting". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[12] Jiuxiang Gu et al. "Unpaired image captioning via scene graph alignments". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[13] Kaiming He et al. "Deep residual learning for image recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[14] Lisa Anne Hendricks et al. "Generating visual explanations". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016.

[15] Weiming Hu et al. “A survey on visual content-based video indexing and retrieval”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41.6 (2011), pp. 797–819.

[16] Michal Irani and Prabu Anandan. “Video indexing based on mosaic representations”. In: *Proceedings of the IEEE* 86.5 (1998), pp. 905–921.

[17] Harsh Jhamtani and Taylor Berg-Kirkpatrick. “Learning to Describe Differences Between Pairs of Similar Images”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 4024–4034.

[18] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. “Densecap: Fully convolutional localization networks for dense captioning”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[19] Dong-Jin Kim et al. “Image Captioning with Very Scarce Supervised Data: Adversarial Semi-Supervised Learning Approach”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 2012–2023.

[20] Ranjay Krishna et al. “Dense-Captioning Events in Videos”. In: *International Conference on Computer Vision (ICCV)*. 2017.

[21] Ranjay Krishna et al. “Visual genome: Connecting language and vision using crowdsourced dense image annotations”. In: *International Journal of Computer Vision* 123.1 (2017).

[22] Iro Laina, Christian Rupprecht, and Nassir Navab. “Towards Unsupervised Image Captioning with Shared Multimodal Embeddings”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[23] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out: Proceedings of the ACL-04 workshop*. Vol. 8. Barcelona, Spain. 2004.

[24] Tsung-Yi Lin et al. “Microsoft COCO: Common objects in context”. In: 2014.

[25] Huan Ling and Sanja Fidler. “Teaching machines to describe images via natural language feedback”. In: 2017.

[26] Ruotian Luo et al. “Discriminability objective for training descriptive captions”. In: 2018.

[27] David S Matteson and Nicholas A James. “A nonparametric approach for multiple change point analysis of multivariate data”. In: *Journal of the American Statistical Association* 109.505 (2014), pp. 334–345.

[28] Ariyo Oluwasanmi et al. “Fully Convolutional CaptionNet: Siamese Difference Captioning Attention Model”. In: *IEEE Access* 7 (2019), pp. 175929–175939.

[29] Oscar Hernan Madrid Padilla et al. “Optimal nonparametric multivariate change point detection and localization”. In: *arXiv preprint arXiv:1910.13289* (2019).

[30] Pingbo Pan et al. “Hierarchical recurrent neural encoder for video representation with application to captioning”. In: 2016.

[31] Yingwei Pan et al. “Video captioning with transferred semantic attributes”. In: 2017, pp. 6504–6512.

[32] Kishore Papineni et al. “BLEU: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 2002.

[33] Dong Huk Park, Trevor Darrell, and Anna Rohrbach. “Robust change captioning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4624–4633.

[34] GG Lakshmi Priya and S Domnic. “Video cut detection using block based histogram differences in RGB color space”. In: *2010 International Conference on Signal and Image Processing*. IEEE. 2010, pp. 29–33.

[35] Rakshith Shetty and Jorma Laaksonen. “Frame-and segment-level features and candidate pool evaluation for video caption generation”. In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, pp. 1073–1076.

[36] Stephen W Smoliar and HongJiang Zhang. “Content based video indexing and retrieval”. In: *IEEE multimedia* 1.2 (1994), pp. 62–72.

[37] Kin-Wai Sze, Kin-Man Lam, and Guoping Qiu. “Scene cut detection using the colored pattern appearance model”. In: *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*. Vol. 2. IEEE. 2003, pp. II–1017.

[38] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. “Cider: Consensus-based image description evaluation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4566–4575.

[39] Subhashini Venugopalan et al. “Captioning images with diverse objects”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5753–5761.

[40] Subhashini Venugopalan et al. “Sequence to sequence-video to text”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015.

[41] Oriol Vinyals et al. “Show and tell: A neural image caption generator”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3156–3164.

[42] Tengyao Wang and Richard J Samworth. “High dimensional change point estimation via sparse projection”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 80.1 (2018), pp. 57–83.

[43] Xin Wang et al. “Video Captioning via Hierarchical Reinforcement Learning”. In: *arXiv preprint arXiv:1711.11135* (2017).

[44] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[45] Jun Xu et al. “Msr-vtt: A large video description dataset for bridging video and language”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5288–5296.

[46] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International Conference on Machine Learning (ICML)*. 2015.

[47] Peter Young et al. “From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions”. In: *Transactions of the Association for Computational Linguistics* 2 (2014), pp. 67–78.

[48] Jun Yu and Mandyam D Srinath. "An efficient method for scene cut detection". In: *Pattern Recognition Letters* 22.13 (2001), pp. 1379–1391.

[49] Luowei Zhou et al. "End-to-end dense video captioning with masked transformer". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8739–8748.

Appendix A

Chapter 3: Additional Material

		Baselines					
		TV	RED		Train w/A_0 Test w/A_0	Train w/A_0 Test w/A_1	Train w/A_1 Test w/A_1
Blur		27.61 ± 2.57	30.23 ± 2.98	U-Net	34.15 ± 2.33	25.42 ± 1.74	33.98 ± 2.15
				MoDL	36.25 ± 2.25	23.91 ± 2.02	36.13 ± 2.19
SR		28.33 ± 2.42	28.59 ± 2.09	U-Net	30.74 ± 2.59	26.3 ± 1.65	31.22 ± 2.71
				MoDL	31.32 ± 2.65	22.27 ± 2.04	31.98 ± 2.61
MRI		25.09 ± 2.50	27.76 ± 3.37	U-Net	31.51 ± 2.83	27.47 ± 2.47	32.33 ± 2.64
				MoDL	31.88 ± 2.85	22.82 ± 2.75	31.79 ± 2.81
		Proposed Model Adaptation Methods					
		Known A_1			Unknown A_1		
		P&P (Alg. 1)	R&R (Alg. 2)	R&R+ (Alg. 3)	P&P (Alg. 1)	R&R (Alg. 2)	R&R+ (Alg. 3)
Blur	U-Net	33.01 ± 1.85	32.11 ± 2.64	33.50 ± 2.47	29.18 ± 1.81	27.67 ± 2.23	30.05 ± 2.73
	MoDL	30.08 ± 1.59	33.82 ± 1.73	34.73 ± 1.82	29.89 ± 1.66	27.81 ± 2.3	27.94 ± 2.4
SR	U-Net	28.00 ± 1.83	29.95 ± 2.49	29.99 ± 2.48	27.77 ± 2.15	26.98 ± 2.39	29.35 ± 2.36
	MoDL	24.59 ± 2.31	28.18 ± 1.64	29.83 ± 2.00	23.14 ± 2.01	24.93 ± 2.04	25.29 ± 2.33
MRI	U-Net	29.07 ± 2.72	29.71 ± 2.75	31.43 ± 2.99	28.92 ± 3.04	28.06 ± 2.63	29.54 ± 2.53
	MoDL	30.63 ± 2.85	30.25 ± 3.10	31.44 ± 2.75	26.64 ± 2.60	23.46 ± 2.54	27.67 ± 2.62

Table A.1: Comparison of performance of various baseline methods for inverse problems across a variety of datasets and forward models. The metric presented is the mean PSNR $\pm$ the standard deviation.

		Baselines					
		TV	RED		Train w/A_0 Test w/A_0	Train w/A_0 Test w/A_1	Train w/A_1 Test w/A_1
Blur		0.94 ± 0.06	0.96 ± 0.05	U-Net	0.98 ± 0.05	0.89 ± 0.09	0.98 ± 0.05
				MoDL	0.98 ± 0.03	0.84 ± 0.08	0.98 ± 0.04
SR		0.95 ± 0.06	0.96 ± 0.03	U-Net	0.97 ± 0.03	0.92 ± 0.09	0.97 ± 0.02
				MoDL	0.97 ± 0.04	0.89 ± 0.06	0.98 ± 0.04
MRI		0.79 ± 0.04	0.80 ± 0.05	U-Net	0.82 ± 0.06	0.74 ± 0.06	0.82 ± 0.06
				MoDL	0.83 ± 0.06	0.65 ± 0.08	0.83 ± 0.06
		Proposed Model Adaptation Methods					
		Known A_1			Unknown A_1		
		P&P (Alg. 1)	R&R (Alg. 2)	R&R+ (Alg. 3)	P&P (Alg. 1)	R&R (Alg. 2)	R&R+ (Alg. 3)
Blur	U-Net	0.81 ± 0.09	0.81 ± 0.07	0.82 ± 0.07	0.75 ± 0.05	0.79 ± 0.09	0.77 ± 0.08
	MoDL	0.82 ± 0.07	0.81 ± 0.09	0.83 ± 0.07	0.72 ± 0.09	0.68 ± 0.07	0.75 ± 0.10
SR	U-Net	0.94 ± 0.09	0.96 ± 0.04	0.96 ± 0.04	0.94 ± 0.09	0.94 ± 0.08	0.96 ± 0.06
	MoDL	0.92 ± 0.06	0.94 ± 0.02	0.95 ± 0.02	0.91 ± 0.04	0.92 ± 0.02	0.94 ± 0.01
MRI	U-Net	0.81 ± 0.09	0.81 ± 0.07	0.82 ± 0.07	0.75 ± 0.05	0.79 ± 0.09	0.77 ± 0.08
	MoDL	0.82 ± 0.07	0.81 ± 0.09	0.83 ± 0.07	0.72 ± 0.09	0.68 ± 0.07	0.75 ± 0.10

Table A.2: Comparison of performance of various baseline methods for inverse problems across a variety of datasets and forward models. The metric presented is the mean SSIM $\pm$ the standard deviation.

This Appendix presents companion tables to Table 3.1 which contain further information, including corresponding mean SSIM over the test set, as well as the standard deviations of PSNR and SSIM.

Supplemental Figures

Here we include visual figures to illustrate some sample reconstructions from the methods outlined in the main work. The figures illustrated are chosen to be close to the median performance while maintaining a range of subjects. The three test images used in the figures are within the inter-quartile range of reconstruction PSNR for all reconstruction methods.

We also include a brief ablation study indicating that the proximity term in (5) of the main text is

Figure A.1: Role of proximity term in model adaptation for motion deblurring with P&P. When ablating the term that promotes proximity to the initial network weights in the P&P approach (*i.e.*, setting $\mu = 0$), the retrained network outputs degenerate solutions that match the measurements (center row), but lack fidelity with the ground truth (top row). By including the proximity term ($\mu \neq 0$), we find a small corrective perturbation to the network weights that improves reconstruction accuracy significantly (bottom row).

necessary for high-quality reconstructions, as well as providing an additional row of Figure 10 for a U-Net trained on multiple forward models.

Motion Deblurring

In Figures A.2, A.3, and A.4 we present sample visualizations for the motion deblurring setting. Figure A.2 contains the original images, as well as the blurred images and a sample total-variation regularized reconstruction.

Figure A.3 and A.4 present reconstruction results for the images shown in Figure A.2. Digital viewing is recommended.

Superresolution

As in the previous subsection, Figures A.5, A.6, and A.7 demonstrate sample visualizations for the superresolution setting. Figure A.2 contains the original images, as well as the upsampled y and a sample total-variation regularized reconstruction.

Figure A.6 and A.7 present reconstruction results for the images shown in Figure A.5. Digital viewing is recommended.

Proximity loss study

In this section we examine the effect of training without the proximity term in the adaptation loss of Eq. 5, demonstrating that just refitting the data without staying close to the original model is not enough.

In Figure A.1 we demonstrate the effect of ablating the proximity term in Eq. 5, which we believe are necessary to avoid degenerate solutions. Optimizing Eq. 5 without proximity terms fails to leverage the network learned for A_0 *using ground truth images* x. Without the proximity term, our solution relies

Figure A.2: Ground truth, blurred, and a classical TV-regularized reconstruction for the motion deblurring inverse problem. Compare the reconstructions to the learned reconstructions in Fig. A.3 and A.4.

Figure A.3: Visual examples of reconstruction quality for the U-Net solver for the motion deblurring inverse problem.

Figure A.4: Visual examples of reconstruction quality for the MoDL solver for the motion deblurring inverse problem.

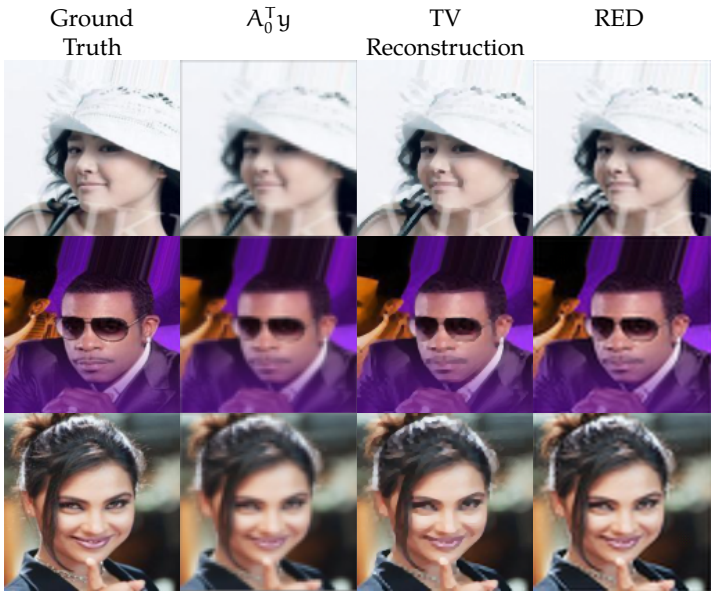


Figure A.5: Ground truth, $A_0^T y$, and a classical TV-regularized reconstruction for the superresolution inverse problem. Here, $A_0^T y$ is the input of the initial trained network, and corresponds to downsampling under A_0 and upsampling back to the original size with A_0^T.

Figure A.6: Visual examples of reconstruction quality for the U-Net network for the superresolution inverse problem.

Figure A.7: Visual examples of reconstruction quality for the MoDL network for the superresolution inverse problem.

too heavily on y, and not enough on the information encoded in f_0. We empirically find the proximity term is necessary to maintain good reconstructions.

Adapting to variable sampling rates in single-coil MRI: Expanded figure

In Fig. A.8 we present an expanded version of the original figure in the main chapter, including the reconstructions produced by the U-Net trained to reconstruct multiple forward models.

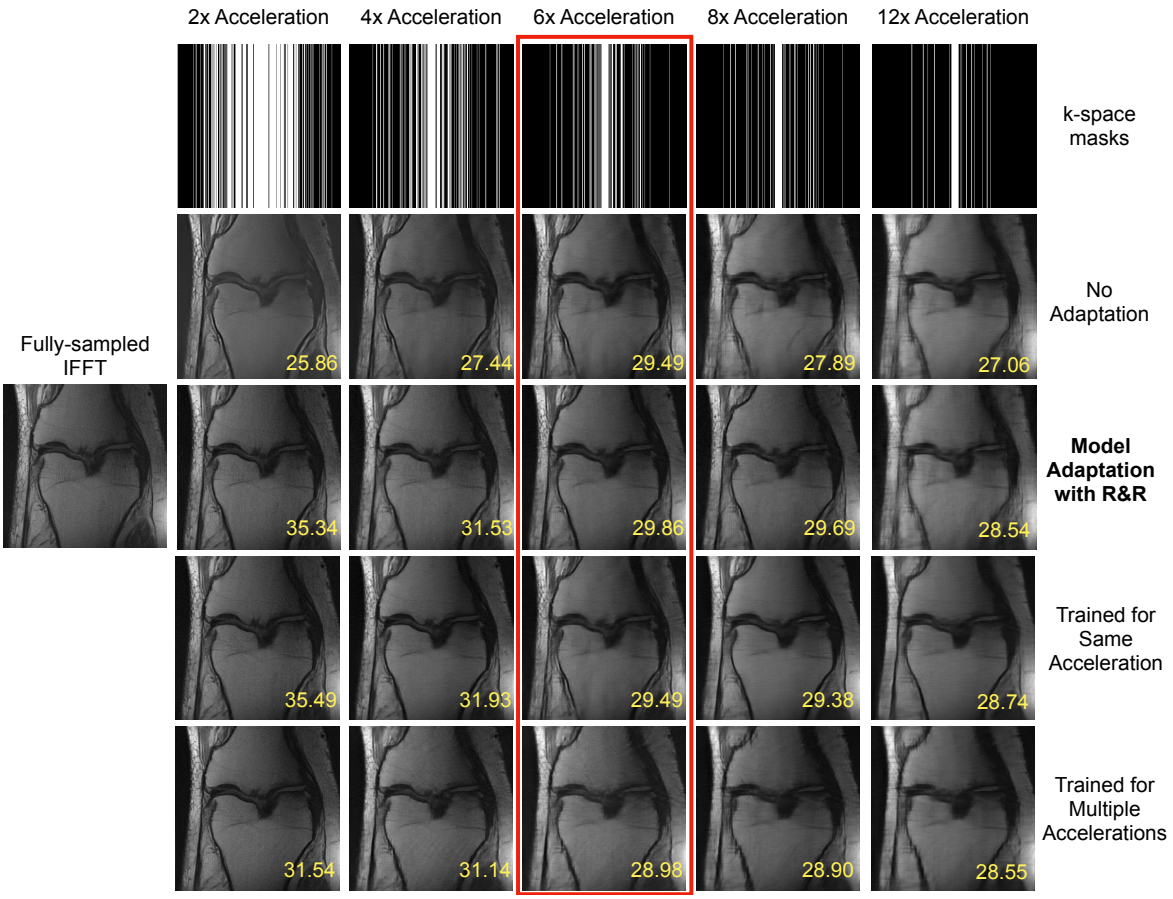


Figure A.8: Using the R&R model adaptation approach permits using a U-Net trained for 6× acceleration on MRI reconstruction across a range of acceleration parameters. The various k-space sampling patterns used in these experiments are shown in the top row. Without adaptation (second row), the reconstruction quality decreases when changing the acceleration factor, *even when more k-space measurements are taken*, as originally observed in [1]. The R&R reconstructions (third row) compare favorably to the performance of networks trained on each particular k-space sampling pattern (second-to-bottom row). Training for multiple accelerations (bottom row) appears to be inferior in terms of reconstruction quality to dedicated training or adapting with R&R. The PSNR of each image is presented in dB in yellow on each image.

Bibliography

[1] Vegard Antun et al. “On instabilities of deep learning in image reconstruction and the potential costs of AI”. In: *Proceedings of the National Academy of Sciences* (2020).

Appendix B

Chapter 4: Additional Material

B.1 Further Qualitative Results

In this section, we provide further visualizations of the reconstructions produced by Deep Equilibrium models and the corresponding Deep Unrolled approaches, beyond those shown in the main body. Figures B.3, B.2, and B.1 are best viewed electronically, and contain the ground-truth images, the measurements (projected back to image space in the case of MRI and compressed sensing), and reconstructions by DU-Prox and DE-Prox.

We also visualize the intermediate iterates in the fixed-point iterations, to further demonstrate the convergence properties of DEMs for image reconstruction. We find that DEMs converge quickly to reasonable reconstructions, and maintain high-quality reconstructions after more than one hundred iterations.

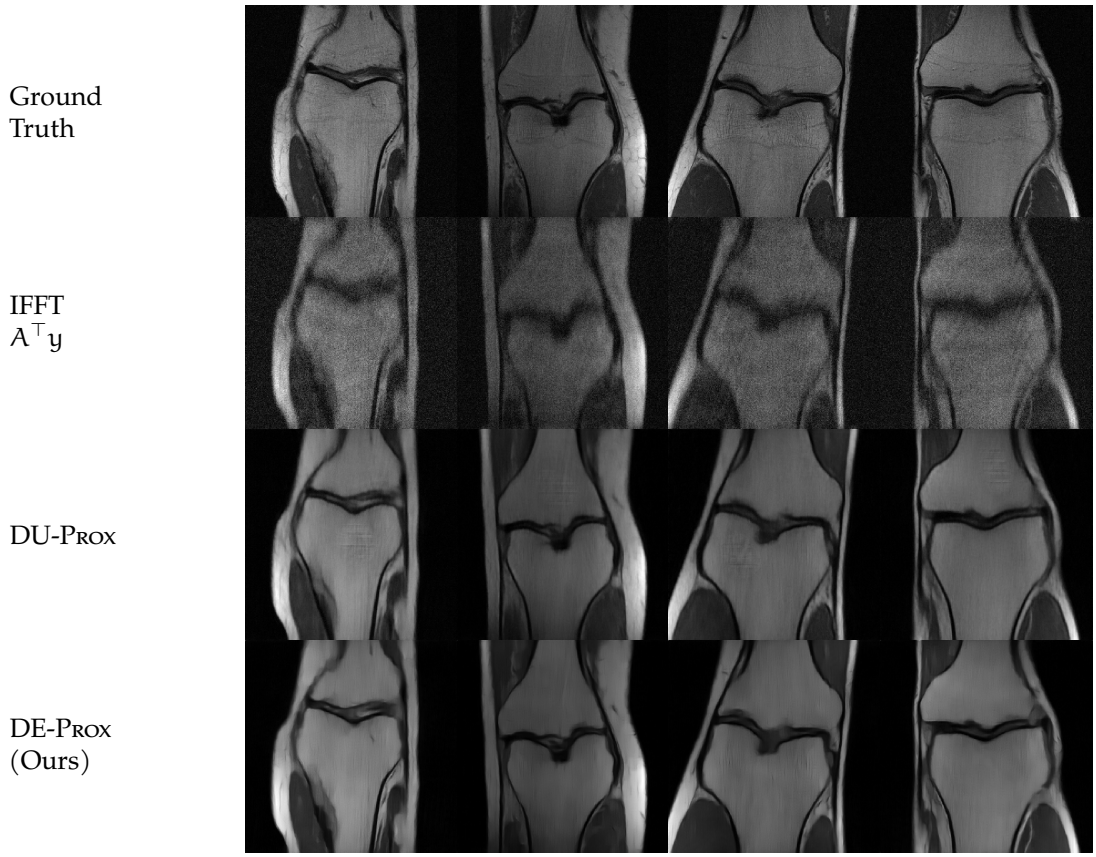


Figure B.1: Sample images and reconstructions with for $8\times$ accelerated MRI reconstruction with additive noise of $\sigma = 0.01$. Best viewed electronically.

Visualizing Iterates

In Figures B.5 and B.4 we visualize the outputs of the K'th iteration of the mapping f_θ in DE-Prox. We observe that across forward problems, the reconstructions converge to good reconstructions.

We illustrate 90 iterations for both compressed sensing and MRI reconstructions.

For further illustration we also demonstrate the qualitative effects of running DU-Prox for more iterations than it was trained for.

Figure B.2: Sample images and reconstructions for 8× Gaussian compressed sensing with additive noise of $\sigma = 0.01$. Best viewed electronically.

Further Experimental Details

In this section we provide further details related to the experimental setup.

The input to the deblurring algorithms is the preconditioned measurement $(A^\top A + \lambda I)^{-1} A^\top y$, where λ is set to be equal to the noise level σ. For MRI reconstruction and compressed sensing experiments, the input is instead simply $A^\top y$. The masks used in the MRI reconstruction experiments are based on a Cartesian sampling pattern, as in the standard fastMRI setting. For both 4× and 8×, the center 4% of frequencies are fully sampled, and further frequencies are sampled according to a Gaussian distribution centered at 0 frequency with $\sigma = 1$.

The compressed sensing design matrices have entries sampled and scaled so that each entry is drawn from a Gaussian distribution with variance $1/m$, where $A \in \mathcal{R}^{m \times n}$. The same design matrix is used for all learned methods.

Optimization algorithm parameters for RED, Plug-and-Play, and all Deep Equilibrium approaches are all chosen via a logarithmic grid search from 10^{-4} to 10^{1} with 20 elements in each dimension of the grid. All DU methods were trained for 10 iterations. All testing was done on an NVidia RTX 2080 Ti. All networks were trained on a cluster with a variety of computing resources [1]. Every experiment was run utilizing a single GPU-single CPU setup with less than 12 GB of GPU memory.

[1] See: `https://slurm.ttic.edu/`

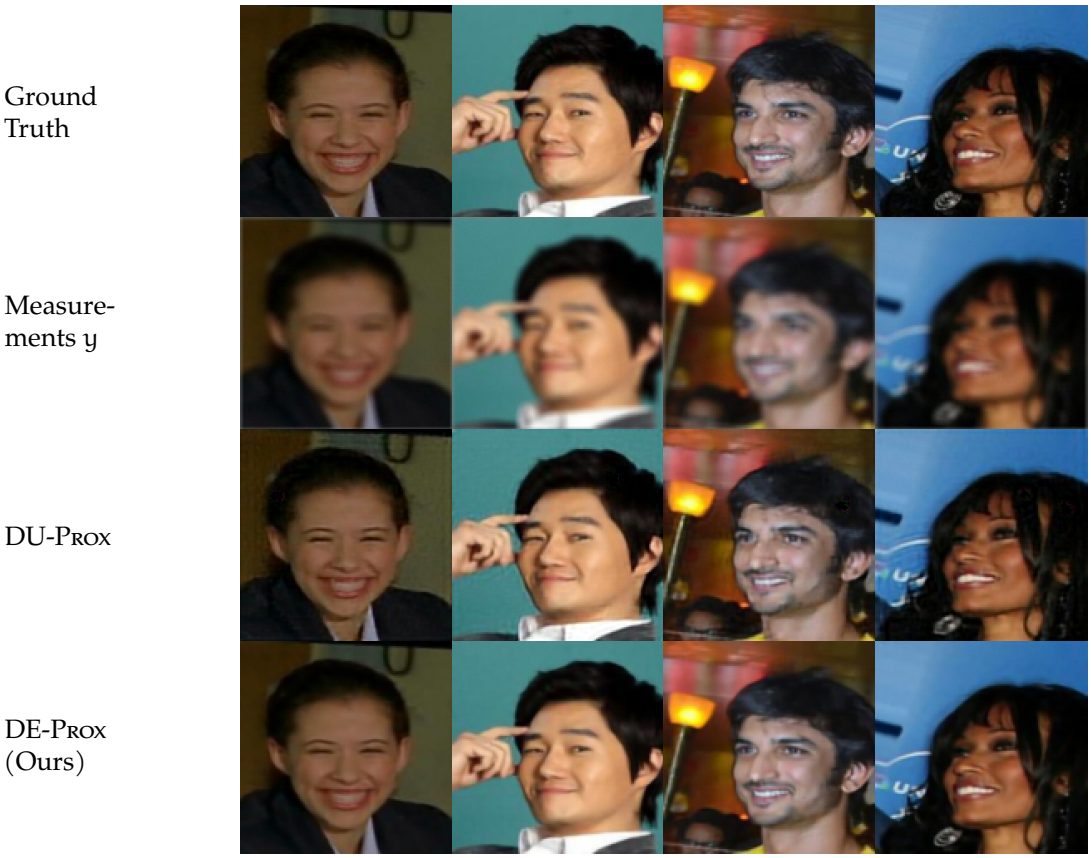


Figure B.3: Sample images and reconstructions for Gaussian deblurring with additive noise of $\sigma = 0.01$. Best viewed electronically.

Figure B.4: Sample images and reconstructions for MRI reconstruction with acceleration $4\times$ and additive Gaussian noise with $\sigma = 0.01$. Each image represents the output of iterate number K. Below each image is the residual between iterate K and the previously-visualized iterate, or in the case of $K = 0$, between the input to the network and the output of the initial iterate. In this case, the algorithm stops iterating (as the relative norm between iterations drops below 10^{-3}) at iteration 31. The ground truth may be viewed in the initial column of Figure B.1. Best viewed electronically.

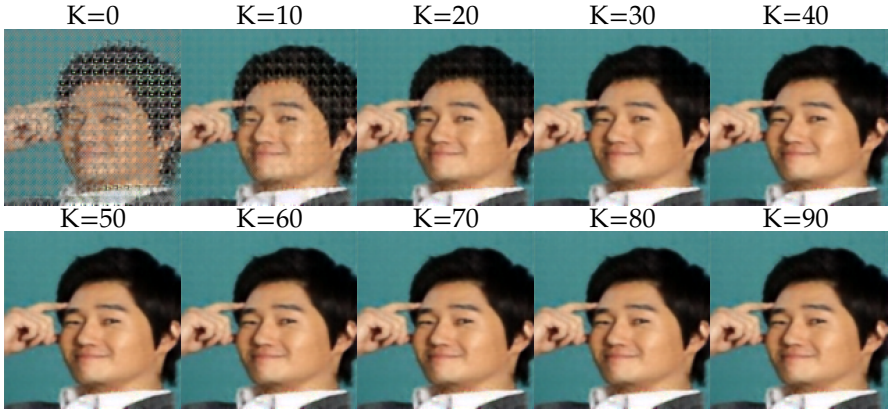


Figure B.5: Sample images and reconstructions from DE-Prox reconstructions, with the forward model $8\times$ Gaussian compressed sensing with Gaussian noise with $\sigma = 0.01$. Each image represents the output of iterate number K. The ground truth may be viewed in the final column of Figure B.2. Best viewed electronically.

Appendix C

Chapter 6: Additional Material

Figure C.1: A sample sequence from "Street Change." Ground Truth Annotations: "Construction signs are gone from the street," "the road sign is gone," "the construction sign is gone," "the construction signs are gone." Generated Annotations: "the street sign is missing," "the construction work is done," "the signs were placed." Note the visual distortions and occlusion in the initial half of the sequence, which is present in several other sequences in the dataset.

In the following appendix we provide further information regarding the "Street Change" and CLEVR-Sequence datasets. We provide several example sequences, explain the annotation gathering procedure for Street Change, and provide further information that was not contained in the main document.

C.1 CLEVR-Sequence

CLEVR-Sequence is an extension of the CLEVR-Change dataset of [2]. The CLEVR-Change dataset consists of triplets of images along with annotations: one "original" image, a "change" image, and a "distractor" image.

CLEVR-Sequence, by contrast, consists of sequences of ten images. The changes between all but one pair in the sequence are "distractor" changes, in which the camera moves and the lighting changes slightly. For one pair, there is, in addition to the distractor change, a non-distractor change, in which one of the possible changes from [2] occurs with uniform probability over the change types. For clarity, these are: Color (an object changes color), Texture (the material of a single object changes), Add (a new object appears), Drop (a previously-present object disappears), and Move (a single object moves in the scene).

We generate 400 sequences for each candidate changepoint in the range $\{1, 2, 3, 4, 5, 6, 7, 8\}$, assuming 0 indexing. We also generate 400 "distractor" sequences in which no change occurs at any point.

We attempt to imitate the characteristics of distractor changes as outlined by the authors of [2] as closely as possible. Annotations are produced automatically for each sequence using the same methodology as the CLEVR-Change dataset as outlined by the authors.

C.2 Street Change

The images in the Street Change dataset were curated by [1], and are grouped into 151 unique image sequences of variable lengths. The minimum sequence length is 2 (this length-2 sequence was omitted from training and testing), and the maximum is 42. As mentioned in the main body, the average sequence length is approximately 9.

Figure C.2: A sample sequence from "Street Change." Ground Truth Annotations: "There is no more sign on sidewalk," "the wooden barrier is gone," "the barricade is gone," "the wooden barricade on the sidewalk disappeared," "the construction barrier is gone," "there is no longer a wooden barrier on the sidewalk." Generated Annotations: "the saw horse is gone," "the construction barricade is gone," "the construction barrier on the sidewalk is no longer there." While visually finding the changepoint is straightforward in some sequences, there are many examples like this sequence, where the visual distinction between the first and second halves of the sequence is subtle.

Figure C.3: A sample sequence from "Street Change." Ground Truth Annotations: "The trash is gone," "the garbage can is gone," "the bush is no longer there," "the garbage can has been removed," "the yard now has grass." Generated Annotations: "The garbage can was removed," "the trash can is gone," "garbage is gone." Shorter sequences tend to contain larger viewpoint changes between frames.

Labeled		1000			2000			10000			20000		
Unlabeled		1k	10k	30k	1k	10k	30k	1k	10k	30k	1k	10k	30k
Ours	C	57.1	44.8	42.7	58.1	60.2	61.4	93.7	98.5	103.3	100.2	100.5	110.0
	B	30.8	28.5	26.9	34.2	34.6	34.7	49.3	51.5	51.6	54.5	54.8	56.4
	M	32.2	30.1	29.5	32.4	32.6	33.2	39.3	39.3	39.5	39.9	41.1	42.9
	R	69.2	65.6	65.1	70.2	70.3	70.9	78.8	79.8	80.8	81.2	82.4	83.5
ST&D	C	42.4	42.7	43.6	55.7	58.2	60.3	92.8	93.9	102.0	98.4	105.1	106.4
	B	27.0	26.3	27.1	32.4	46.9	47.2	48.7	51.8	52.0	55.3	55.3	57.3
	M	29.4	29.8	29.6	31.5	32.8	33.0	38.3	38.7	40.7	38.9	40.9	42.2
	R	65.4	65.8	65.6	65.8	69.8	70.0	78.9	79.1	81.5	79.3	82.0	83.3

Table C.1: Semi-supervised scaling with respect to labeled and unlabeled training set size for CLEVR-Change using our method and Show, Tell, and Discriminate. We report CIDEr (C), BLEU-4 (B), METEOR (M), and ROUGE-L (R) (scaled by 100×).

In Street Change, the change point is always the center frame of the sequence. For methods which learn by leveraging the entire sequence simultaneously, this may be a problematic bias, but since we learn and infer using only pairwise comparisons, this bias is less concerning.

Captions were gathered using Amazon's Mechanical Turk. Annotators were presented with a pair of images and asked to describe in a single brief sentence the major changes between them, ignoring nuisance changes. Annotators were given guidance on what to consider nuisance changes, like weather, lighting, time of day, and camera angle. Annotations for both the original sequences and time-reversed sequences were gathered, as a form of data augmentation.

The gathered annotations were curated by hand to remove descriptions of nuisance changes, ensure consistent spelling, ensure each annotation was a single sentence, and confirm accuracy of the annotations. We collected six annotations for each sequence (and six more for the time-reversed sequence), to ensure that there would be at least three acceptable annotations for each sequence.

Example Street Change Sequences

Here we illustrate some sample full sequences from the Street Change Dataset, along with the associated ground-truth annotations, and the annotations produced by our method with 100 labeled training sequences and 30 unlabeled training sequences. The displayed images are drawn from the test sequences. To generate multiple captions, different pairs in the sequence were sampled; the language model is sampled greedily for all pairs.

All images best viewed digitally.

C.3 Additional Language Results

In this section we provide additional metrics on performance of the semi-supervised change captioning subtask, exploring the sample complexity of these methods with respect to size of the labeled and unlabeled dataset size.

Labeled		2000				5000			8000		L9000	30	100	130
Unlabeled		1k	2k	5k	8k	1k	2k	5k	1k	2k	U 1k	100	30	0
Ours	C	11.1	11.4	12.5	13.2	21.7	21.8	23.1	30.4	30.8	33.6	86.6	116.0	119.8
	B	5.2	14.4	13.3	14.6	10.5	11.7	15.2	9.9	12.8	16.1	30.5	41.7	42.6
	M	19.6	22.1	22.7	23.1	22.2	22.1	23.5	22.9	23.0	24.1	33.8	40.5	39.9
	R	39.1	47.7	48.6	50.3	46.2	47.5	50.4	48.5	49.3	51.4	66.6	75.9	73.5
ST&D	C	10.2	9.9	10.5	11.4	18.5	19.6	21.3	29.0	30.1	30.6	64.4	105.2	112.7
	B	3.8	8.4	10.1	9.6	11.2	12.9	13.7	10.6	11.5	13.0	26.7	38.9	38.6
	M	16.9	16.6	17.1	17.1	20.4	21.2	22.3	22.5	22.9	23.5	32.0	39.3	38.8
	R	38.9	38.7	40.6	41.1	42.4	43.1	47.7	47.0	47.1	47.7	64.5	72.6	73.1

Table C.2: **Left**: Semi-supervised scaling with respect to labeled and unlabeled training set size for Spot-the-Diff, and **Right** Street Change on a variety of language metrics (scaled by $100\times$) using our method and Show, Tell, and Discriminate. Note that training set size for Street Change is in terms of number of labeled sequences, instead of image pairs.

Bibliography

[1] Pablo F Alcantarilla et al. “Street-view change detection with deconvolutional networks”. In: *Autonomous Robots* 42.7 (2018), pp. 1301–1322.

[2] Dong Huk Park, Trevor Darrell, and Anna Rohrbach. “Robust change captioning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4624–4633.