

# Neumann Networks for Linear Inverse Problems in Imaging

Davis Gilton<sup>1</sup>, Greg Ongie, and Rebecca Willett<sup>2</sup>

**Abstract**—Many challenging image processing tasks can be described by an ill-posed linear inverse problem: deblurring, deconvolution, inpainting, compressed sensing, and superresolution all lie in this framework. Traditional inverse problem solvers minimize a cost function consisting of a data-fit term, which measures how well an image matches the observations, and a regularizer, which reflects prior knowledge and promotes images with desirable properties like smoothness. Recent advances in machine learning and image processing have illustrated that it is often possible to learn a regularizer from training data that can outperform more traditional regularizers. We present an end-to-end, data-driven method of solving inverse problems inspired by the Neumann series, which we call a Neumann network. Rather than unroll an iterative optimization algorithm, we truncate a Neumann series which directly solves the linear inverse problem with a data-driven nonlinear regularizer. The Neumann network architecture outperforms traditional inverse problem solution methods, model-free deep learning approaches, and state-of-the-art unrolled iterative methods on standard datasets. Finally, when the images belong to a union of subspaces and under appropriate assumptions on the forward model, we prove there exists a Neumann network configuration that well-approximates the optimal oracle estimator for the inverse problem and demonstrate empirically that the trained Neumann network has the form predicted by theory.

**Index Terms**—Image reconstruction, inverse problems, convergence, iterative algorithms, deconvolution, machine learning, estimation.

## I. LEARNING TO REGULARIZE

**I**N THIS paper we consider solving linear inverse problems in imaging in which a  $p$ -pixel image,  $\beta^* \in \mathbb{R}^p$  (in vectorized form), is observed via  $m$  noisy linear projections as  $y = X\beta^* + \epsilon$ , where  $y, \epsilon \in \mathbb{R}^m$  and  $X \in \mathbb{R}^{m \times p}$ . This general model is used throughout computational imaging, from basic image restoration tasks like deblurring, super-resolution, and image inpainting [1], to a wide variety of tomographic imaging applications, including common types of magnetic resonance imaging [2], X-ray computed tomography [3], radar imaging [4], among

others [5]. The task of estimating  $\beta^*$  from  $y$  is often referred to as *image reconstruction*. Classical image reconstruction methods assume some prior knowledge about  $\beta^*$  such as smoothness [6], sparsity in some dictionary or basis [7]–[9], or other geometric properties [10]–[13], and attempt to estimate a  $\hat{\beta}$  that is both a good fit to the observation  $y$  and that also conforms to this prior knowledge. In general, a *regularization function*  $r(\beta)$  measures the lack of conformity of  $\beta$  to this prior knowledge and  $\hat{\beta}$  is selected so that  $r(\hat{\beta})$  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 [14]–[16], 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  $\hat{\beta}$  to lie on this manifold [17] or learning a denoising autoencoder that can be treated as a regularization step (*i.e.*, proximal operator) within an iterative reconstruction scheme [18].

In this paper, we propose a novel neural network architecture based on the Neumann series expansion [19], [20] that we call a *Neumann network*, 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  $X$  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* [14] 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

Manuscript received January 16, 2019; revised May 30, 2019; accepted July 2, 2019. Date of publication October 21, 2019; date of current version January 13, 2020. This work was supported by the National Science Foundation under Grants 1740707, 1447449, and 0353079, as well as AFOSR FA9550-18-1-0166. The associate editor coordinating the review of this manuscript and approving it for publication was J. Portilla. (*Corresponding author: Davis Gilton.*)

D. Gilton is with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706 USA (e-mail: gilton@wisc.edu).

G. Ongie is with the Department of Statistics, University of Chicago, Chicago, IL 60637 USA (e-mail: gongie@uchicago.edu).

R. Willett is with the Department of Statistics and Computer Science, University of Chicago, Chicago, IL 60637 USA (e-mail: willett@uchicago.edu).

This article has supplementary downloadable material available at <http://ieeexplore.ieee.org>, provided by the authors.

Digital Object Identifier 10.1109/TCL.2019.2948732

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.
- The empirical performance of the Neumann network on superresolution, deblurring, compressed sensing, and inpainting problems exceeds that of competing methods.

## II. PREVIOUS WORK

There are several general categories of methods used to learn to solve inverse problems, which are reviewed below. Throughout, we assume we have training samples of the form  $(\beta_i, y_i)$  for  $i = 1, \dots, N$ , where  $y_i = X\beta_i + \epsilon_i$ ,  $X \in \mathbb{R}^{m \times p}$  is known and the noise  $\epsilon_i$  is treated as unknown.

### A. Agnostic

An agnostic learner uses the training data to learn a mapping from  $y$  to  $\beta$  without any knowledge of  $X$  at any point in the training or testing process [21]. The general principle is that, given enough training data, we should be able to learn everything we need to know about  $X$  to successfully estimate  $\beta$ . Empirically, the success of this approach appears to be highly dependent on the forward operator  $X$ . This straightforward approach has been demonstrated on superresolution [22], [23], blind deconvolution [21], and motion deblurring [24], among others. In general, this approach requires large quantities of training data because it is required to not only learn the geometry of the image space containing the  $\beta$ 's, but also aspects of  $X$ . A particularly successful approach to solving inverse problems with neural networks has been through residual learning [14].

### B. Decoupled

A decoupled approach operates in two stages. In the first stage, a collection of training images  $\beta_i$  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  $(y, X)$  to  $\hat{\beta}$ . 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.

First, we might learn a generative model  $G$  for  $\beta$ 's that takes as input a low-dimensional vector  $z \in \mathbb{R}^d$  for  $d < p$  and outputs  $\beta = G(z)$ . The basic idea is that the images of interest lie on a low-dimensional submanifold that can be indexed by  $z$ . Given the learned  $G$ , we can compute  $\hat{\beta}$  from  $y$  via

$$\hat{\beta} = \arg \min_{\beta = G(z), z \in \mathbb{R}^d} \|y - X\beta\|_2^2. \quad (1)$$

This approach was described in [17] with compelling empirical performance for compressed sensing.

Alternatively, we might learn a denoising autoencoder that could be used as a proximal operator in an iterative reconstruction method. Specifically, imagine we had a fixed regularizer  $r(\cdot)$  and want to set

$$\hat{\beta} = \arg \min_{\beta} \frac{1}{2} \|y - X\beta\|_2^2 + r(\beta). \quad (2)$$

A proximal gradient algorithm [25], [26] starts with an initial estimate  $\beta^{(0)}$  and step size  $\eta > 0$  and then iterates between computing a gradient descent step that pushes the current estimate  $\beta^{(k)}$  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 is well-regularized (as measured by  $r(\cdot)$ ). This second step is often thought of as a denoising step. One approach to learning to solve inverse problems is to implicitly learn  $r(\cdot)$  by explicitly learning a proximal operator in the form of a denoising autoencoder [18], [27], [28].

The key feature in both of these approaches is that all training takes place independently of  $X$ —*i.e.*, we either learn a generative model or a proximal operator using the training data, neither of which require knowledge of  $X$ . 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, 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  $\phi(\beta)$ . 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  $X$  at training time, then we only need to learn the *conditional* distribution  $\phi(\beta|X\beta)$  or  $\phi(\beta|y)$  [29]. For example, imagine an image inpainting scenario in which we only observe a subset of pixels in the image  $\beta$ . 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,  $\phi(\beta|X\beta)$ . Of course,  $\phi(\beta|X\beta)$  can be calculated from  $\phi(\beta)$  and  $X$  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  $\phi(\beta)$  has a minimax rate that scales as  $\mathcal{O}(N^{-\frac{2\alpha}{2\alpha+p}})$  with  $N$  the number of training samples,  $p$  the dimension of  $\phi(\beta)$ , and  $\alpha$  a smoothness term [30]–[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  $\mathcal{O}(N^{-\frac{2\alpha}{2\alpha+p'}})$  [29].

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

### C. Unrolled Optimization

Another approach treats a learned component of a network as the gradient of a prior over the data or a proximal operator

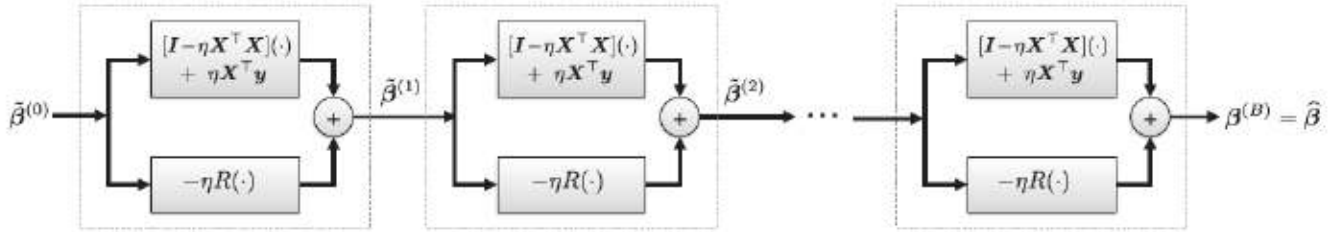


Fig. 1. Unrolled gradient descent network. The result of  $B$  iterations of gradient descent with a fixed step size  $\eta$  and regularizer with gradient  $R$ , as in (3) is equivalent to the output of the above network, with each block corresponding to a single iteration. The network maps a linear function of the measurements,  $\beta^{(0)} = \eta X^T y$ , to a reconstruction  $\hat{\beta}$  by successive application of an operator of the form  $[I - \eta X^T X](\cdot) - \eta R(\cdot)$  and addition of  $\eta X^T y$ . Here  $R$  is a trained neural network, and the scale parameter  $\eta$  is also trained.

for a regularizer [33]. Suppose the desired optimal point  $\beta^*$  satisfies the optimality condition in (2). Now we assume  $r(\cdot)$  is differentiable and let  $R(\beta) := \nabla r(\beta)$  denote the gradient of the regularizer. Then solving (2) can be accomplished using iterative optimization; for instance, gradient descent would result in the iterates

$$\beta^{(k+1)} = \beta^{(k)} - \eta \left[ X^T (X \beta^{(k)} - y) + R(\beta^{(k)}) \right] \quad (3)$$

for a step size  $\eta > 0$ . Imagine computing these iterates for a fixed number of iterations, which we will denote  $B$  (for Blocks, as will become clear shortly).

“Unrolling” an optimization method refers to taking an iterative optimization method and, instead of iterating until convergence, thinking of a series of  $B$  iterates as a single operation to be applied to an input. This idea as applied to gradient descent is represented pictorially in Fig. 1. 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  $X$  and training observations  $y_i$  by minimizing the disparity between the true  $\beta_i$  and  $\hat{\beta}(y_i)$ , the output of the full network (see Fig. 1). This end-to-end training sidesteps the sample complexity challenges described in Section II-B.

The unrolling approach can be applied to a variety of optimization algorithms beyond gradient descent. The earliest proposed unrolled inverse problem solver was [34], in which the authors proposed unrolling the Iterative Shrinkage and Thresholding Algorithm (ISTA) [35] and the coordinate descent algorithm; further refinements of this approach were proposed in [36], [37]. More recent work has illustrated the efficacy of unrolled optimization as applied to (proximal) gradient descent [33], [38], [39], alternating directions method of multipliers [40], primal-dual methods [41], half-quadratic splitting [42], [43], block coordinate descent [44]–[46], alternating minimization [47], iterative reweighted least squares [48], [49], and approximate message passing [50]. In proximal gradient settings, the learned neural network is interpreted as a learned proximal operator, whereas in the gradient descent network, the learned neural network is interpreted as the gradient of the regularizer at the input. In other words, for different unrolled optimization methods the learned neural network can play different roles.

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, [34] 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 [38] achieve promising performance with  $B = 8$  proximal gradient descent iterations. Another strategy to enable deeper unrollings is to perform block-by-block training as in [45]; 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 [47] has illustrated that in the low-sample setting, different learned weights in each block can be suboptimal.

### III. 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). 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 IV. 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 II-C.

#### A. Proposed Network Architecture

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

$$(X^T X + R) \beta^* = X^T y. \quad (4)$$

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

$$\beta^* = (X^T X + R)^{-1} X^T y. \quad (5)$$

In order to approximate the matrix inverse in (5) we consider a Neumann series expansion of linear operators [19], [20], which we now recall. Let  $A$  be any  $p \times p$  matrix and let  $I$  denote the  $p \times$

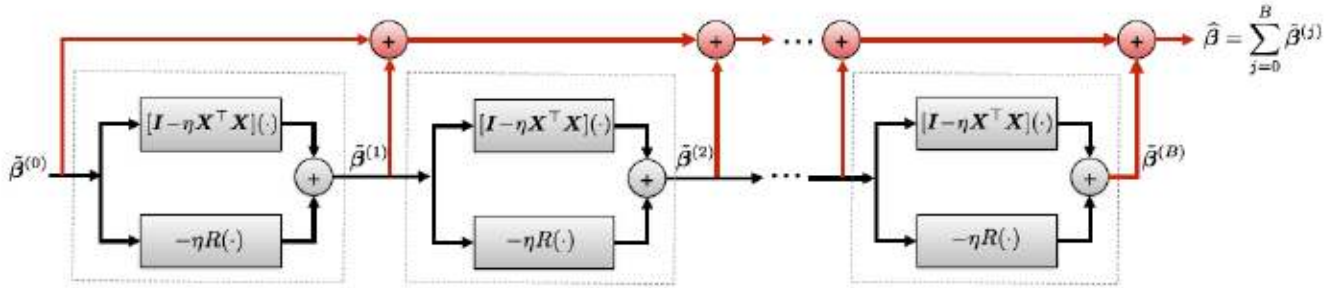


Fig. 2. Proposed Neumann network architecture. Inspired by the Neumann series expansion for computing the inverse of an operator, a Neumann network maps a linear function of the measurements,  $\tilde{\beta}^{(0)} = \eta X^T y$  to a reconstruction  $\hat{\beta}$  by successive application of an operator the form  $[I - \eta X^T X](\cdot) - \eta R(\cdot)$  while summing the intermediate outputs of each block. Here  $R$  is a trained neural network, and the scale parameter  $\eta$  is also trained. Unlike other networks based on unrolling of iterative optimization algorithms, the series structure of Neumann networks lead naturally to skip connections [14] (highlighted in red) that route the output of each dashed block to directly to the output layer.

$p$  identity matrix. If the Neumann series  $\sum_{k=0}^{\infty} A^k$  converges then  $I - A$  is invertible and we have

$$(I - A)^{-1} = \sum_{k=0}^{\infty} A^k = I + A + A^2 + A^3 \dots \quad (6)$$

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

$$B^{-1} = \eta \sum_{k=0}^{\infty} (I - \eta B)^k, \quad (7)$$

which is obtained through a change of variables.

Applying the Neumann series expansion (7) to the matrix inverse appearing in (5), we have<sup>1</sup>

$$\beta^* = \sum_{j=0}^{\infty} (I - \eta X^T X - \eta R)^j (\eta X^T y). \quad (8)$$

Truncating the series in (8) to  $B + 1$  terms, and replacing multiplication by the matrix  $R$  with a general mapping  $R : \mathbb{R}^p \rightarrow \mathbb{R}^p$ , motivates an estimator  $\hat{\beta}$  of the form

$$\hat{\beta}(y) := \sum_{j=0}^B ([I - \eta X^T X](\cdot) - \eta R(\cdot))^j (\eta X^T y). \quad (9)$$

We turn (9) 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  $R_{\theta}$  is a neural network, where  $\theta$  is a vectorized set of weights and biases that define the network. We also treat the step-size choice  $\eta$  as a trainable parameter. The class of estimators  $\hat{\beta}(y) = \hat{\beta}(y; \theta, \eta)$  specified (9) with trainable network  $R = R_{\theta}$  we call *Neumann networks*.

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

<sup>1</sup>The series in (7) is guaranteed to converge if  $\|I - \eta B\| < 1$ . Hence, the expansion in (8) is valid provided  $\|I - \eta(X^T X + R)\| < 1$ , which holds if and only if  $X^T X + R$  is positive definite and  $\eta < \|X^T X + R\|^{-1}$ .

$\hat{\beta}(y)$  in (9) may not be a good solution to the optimization problem (2). This is because the Neumann series identity (7) 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 (9) can be formulated as a network, observe that the terms in (9) have the following recursive form: let the input to the network be  $\tilde{\beta}^{(0)} := \eta X^T y$  and define

$$\tilde{\beta}^{(j)} := (I - \eta X^T X) \tilde{\beta}^{(j-1)} - \eta R(\tilde{\beta}^{(j-1)}) \quad (10)$$

for all  $j = 1, \dots, B$ . Then we have  $\hat{\beta}(y) = \sum_{j=0}^B \tilde{\beta}^{(j)}$ .

Fig. 2 shows a block diagram for implementing the Neumann network using the recursion (10). Each block with a dashed boundary in Fig. 2 represents an application of the operator  $[I - \eta X^T X](\cdot) - \eta R(\cdot)$ . 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  $\tilde{\beta}^{(j)}$ 's) to the output layer, similar to those found in residual networks [14] and related architectures [15]. These skip connections are a distinguishing feature of Neumann networks compared to networks derived from unrolled optimization approaches, such as unrolled gradient descent (see Fig. 1). 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 V-G for empirical evidence and more discussion on this point.

## B. Preconditioning

Efficiently finding a solution to the linear system (4) using an iterative method is challenging when the matrix  $X^T X + R$  is ill-conditioned. This suggests that our Neumann network approach, which is derived from a Neumann series expansion of the system in (4), may benefit from preconditioning. Here we derive a variant of Neumann networks inspired by a preconditioning of (4).

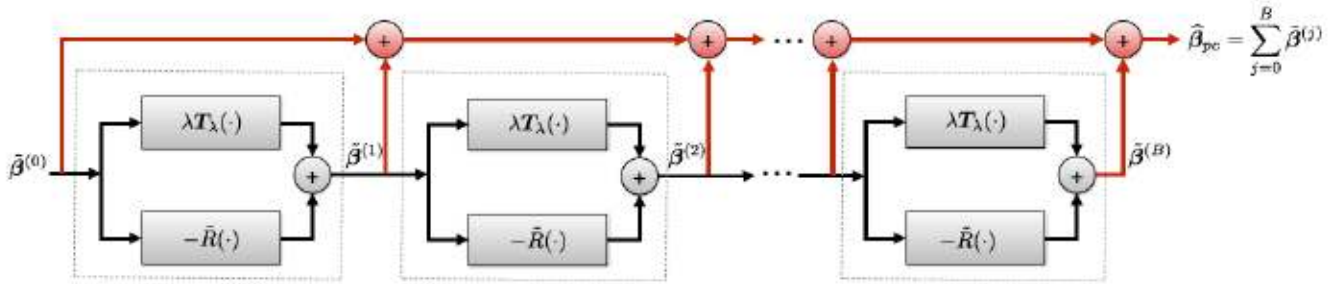


Fig. 3. Proposed preconditioned Neumann network architecture. The network has the same basic architecture as a Neumann network, but uses a different linear component given by  $T_\lambda = (X^T X + \lambda I)^{-1}$  where  $\lambda > 0$  and a different initialization  $\tilde{\beta}^{(0)} = T_\lambda X^T y$ . When the matrix inverse in  $T_\lambda$  is computationally prohibitive to apply, we replace all instances of  $T_\lambda$  with an unrolling of a fixed number of iterations of the conjugate gradient algorithm, similar to [47]. Here  $\tilde{R}$  is a trained neural network, and the scale parameter  $\lambda$  is also trained when feasible.

Starting from (4), for any  $\lambda > 0$  we have

$$(X^T X + \lambda I)\beta^* + (R - \lambda I)\beta^* = X^T y. \quad (11)$$

Applying  $T_\lambda := (X^T X + \lambda I)^{-1}$  to both sides and rearranging terms gives

$$(I - \lambda T_\lambda + \tilde{R})\beta^* = T_\lambda X^T y. \quad (12)$$

where we have set  $\tilde{R} = T_\lambda R$ . Following the same steps used to derive the Neumann network, we arrive at the modified estimator

$$\hat{\beta}_{pc}(y) = \sum_{j=0}^B (\lambda T_\lambda(\cdot) - \tilde{R}(\cdot))^j T_\lambda X^T y \quad (13)$$

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

As shown in Fig. 3, a preconditioned Neumann network has the same basic network architecture as the standard Neumann network, except the linear component  $[I - \eta X^T X](\cdot)$  is replaced with  $[\lambda T_\lambda](\cdot)$  and the learned component  $[-\eta R](\cdot)$  is replaced with  $[-\tilde{R}](\cdot)$ . The preconditioned Neumann network also has a different initialization,  $\tilde{\beta}^{(0)} = T_\lambda X^T y$ , which is the solution to the Tikhonov regularized least squares problem  $\min_{\beta} \|X\beta - y\|^2 + \lambda \|\beta\|^2$ . For many inverse problems in imaging, such as deblurring, this is much more accurate approximation to the ideal solution than the matrix transpose initialization  $\tilde{\beta}^{(0)} = \eta X^T y$  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 Fig. 9(b)) support this observation.

Applying  $T_\lambda(\cdot)$  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 [47] and replace all instances of  $T_\lambda(\cdot)$  in the preconditioned Neumann network by an unrolling of a fixed number of iterations of the conjugate gradient (CG) algorithm [51], which approximates the application of  $T_\lambda(\cdot)$ . Unrolling CG does not require any additional trainable parameters, and backpropagation 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 [48], [49]. Incorporating a trainable  $\lambda$  parameter into this approach is simple since the derivatives of the end-to-end network  $\hat{\beta}_{pc}$  with respect to  $\lambda$  are also easily computed by automatic differentiation. In particular, we do not need  $T_\lambda$  to have an analytic expression in terms of  $\lambda$  in order to compute derivatives.

Finally, we note other preconditioned Neumann networks could be derived by replacing  $I$  with a general matrix  $S$  such that  $X^T X + \lambda S$  is positive definite, e.g.,  $S = D^T D$  where  $D$  is a discrete approximation of the image gradient. For simplicity, we restrict ourselves to the choice  $S = I$  in this work.

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

Suppose the learned component  $R$  is linear, i.e.,  $R(\beta) = R\beta$  for some matrix  $R \in \mathbb{R}^{p \times p}$ . The  $B$ th iteration  $\beta^{(B)}$  of unrolled gradient descent (3) with step size  $\eta > 0$  and initialization  $\beta^{(0)} = \eta X^T y$  can be expanded to obtain

$$\beta^{(B)} = \eta \sum_{j=0}^B (I - \eta X^T X - \eta R)^j X^T y,$$

which is precisely the form of the Neumann network estimator (9). Therefore, if  $R$  is linear the estimator obtained using an unrolling of gradient descent and the Neumann network estimator are the same. When  $R$  is nonlinear we no longer have this equivalence.

## IV. 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)) only if  $R$  is linear. However, restricting  $R$  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. Our reasons for focusing on a UoS model are two-fold: First, UoS models can be found in a wide range of modern signal processing applications, either as a deterministic model [52]–[54] or as a statistical model in the form of a Gaussian mixture model with low-rank covariances [55]–[57]. Second, we believe UoS models represent a reasonable trade-off between model complexity/expressiveness and analytic tractability, and allow us to 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*  $\hat{\beta}$  given in (9) that are specified by a (potentially nonlinear) mapping<sup>2</sup>  $R: \mathbb{R}^p \rightarrow \mathbb{R}^p$ , step size  $\eta$ , 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) 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.

#### A. Images Belonging to a Single Subspace

Suppose the ground truth images belong to an  $r$ -dimensional subspace  $\mathcal{S} \subset \mathbb{R}^p$ . Let  $U \in \mathbb{R}^{p \times r}$  be a matrix whose columns form an orthonormal basis for  $\mathcal{S}$ . Assume  $m \geq r$  and  $XU \in \mathbb{R}^{m \times r}$  is full rank. In other words, we assume there is no image in the subspace also in the nullspace of  $X$  besides the zero image.<sup>3</sup> Then given noise-free linear measurements of the form  $y = X\beta^*$  of any data point  $\beta^* = Uw^* \in \mathcal{S}$  we can always recover  $\beta^*$  by applying the linear estimator

$$\hat{\beta}_o(y) = U(U^\top X^\top XU)^{-1}U^\top X^\top y \quad (14)$$

since it is easy to check that  $\hat{\beta}_o(y) = \beta^*$ . 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 (9) (*i.e.*, a linear choice of  $R$  in (9)) such that for all points in the subspace the reconstruction

<sup>2</sup>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  $\mathbb{R}^p$ .

<sup>3</sup>This assumption is often met in 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.

error can be made arbitrarily small by choosing the step size  $\eta$  and block size  $B$  appropriately. For simplicity, we restrict ourselves to the case of noise-free measurements and where  $X$  has orthonormal rows.

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

$$R = -c_{\eta,B}(I - X^\top X)U(U^\top X^\top XU)^{-1}U^\top X^\top X \quad (15)$$

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

$$\|\hat{\beta}(X\beta^*) - \beta^*\| \leq (1 - \eta)^{B+1} \|X\beta^*\|. \quad (16)$$

for all  $\beta^* \in \mathcal{S}$ .

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

$$\hat{\beta}^{(j)} = a_j X^\top X\beta^* + b_j (I - X^\top X)\beta^* \quad (17)$$

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  $\hat{\beta}(y) = \sum_{j=0}^B \hat{\beta}^{(j)} \approx X^\top X\beta^* + (I - X^\top X)\beta^* = \beta^*$ .

#### B. 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  $k = 1, \dots, K$  we let  $U_k \in \mathbb{R}^{p \times r}$  denote a matrix whose columns form an orthonormal basis for  $\mathcal{S}_k$ . Again, we assume  $m \geq r$  and  $XU_k \in \mathbb{R}^{m \times r}$  is full rank for every  $k = 1, \dots, K$ . In other words, we assume there is no image in the UoS also in the nullspace of  $X$  besides the zero image.

Let  $y = X\beta^*$  be the measurements of any point  $\beta^*$  belonging to the UoS. If we know  $\beta^*$  belongs to the  $k$ th subspace, *i.e.*,  $\beta^* = U_k w^*$  for some  $w^* \in \mathbb{R}^r$ , then similar to the single subspace case, we can apply the estimator

$$\hat{\beta}_o(y; k) := U_k(U_k^\top X^\top XU_k)^{-1}U_k^\top X^\top y \quad (18)$$

since it is easy to see that  $\beta^* = \hat{\beta}_o(y; k)$ . We call  $\hat{\beta}_o(y; k)$  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 (9) 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

$$R^*(\beta) = \begin{cases} R_1\beta & \text{if } \beta \in \mathcal{C}_1 \\ \vdots & \vdots \\ R_K\beta & \text{if } \beta \in \mathcal{C}_K \end{cases} \quad (19)$$

where each  $R_k$  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  $\beta^*$  belongs to the  $k$ th subspace, then we prove that the Neumann series summands  $\tilde{\beta}^{(0)}, \tilde{\beta}^{(1)}, \dots, \tilde{\beta}^{(B)}$  all lie in the same region  $\mathcal{C}_k$ . This means that the same  $R_k$  is used in computing each summand, so we can write

$$\tilde{\beta}(\mathbf{y}) = \sum_{j=0}^B \tilde{\beta}^{(j)} = \sum_{j=0}^B (\mathbf{I} - \eta \mathbf{X}^\top \mathbf{X} - \eta R_k)^j (\eta \mathbf{X}^\top \mathbf{y}).$$

Choosing  $R_k$  to have the same form as in the single subspace case (see Lemma 1), we then will have  $\beta^* = \tilde{\beta}_o(\mathbf{y}, k) \approx \tilde{\beta}(\mathbf{y})$ .

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

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

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

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

where  $d_{\mathbf{X}, k}(\beta) := \|(\mathbf{I} - \mathbf{X} \mathbf{U}_k (\mathbf{X} \mathbf{U}_k)^\dagger) \mathbf{X} \beta\|$  is the distance between the vector  $\mathbf{X} \beta$  and the subspace  $\text{span}(\mathbf{X} \mathbf{U}_k)$ . In other words,  $\mathcal{C}_k$  is the set of all points whose distance to the  $k$ th subspace is smaller than the distance to all other subspaces, as measured by the functions  $d_{\mathbf{X}, \ell}$  for all  $\ell = 1, \dots, K$ .

We now state our main theorem:

**Theorem 1:** Let  $\mathbf{X} \in \mathbb{R}^{m \times p}$  be any measurement matrix with orthonormal rows, and for all  $k = 1, \dots, K$  let  $\mathbf{U}_k \in \mathbb{R}^{p \times r}$  be an orthonormal basis for the  $k$ th subspace  $\mathcal{S}_k$  with  $\dim \text{span}(\mathbf{X} \mathbf{U}_k) = r$ . Suppose  $\text{span}(\mathbf{X} \mathbf{U}_k) \cap \text{span}(\mathbf{X} \mathbf{U}_\ell) = \{0\}$  for all  $k \neq \ell$ . Then the Neumann network estimator  $\tilde{\beta}$  with step size  $\eta \in (0, 1)$  and piecewise linear  $R = R^*$  as defined in (19) satisfies

$$\|\tilde{\beta}(\mathbf{X} \beta^*) - \beta^*\| \leq (1 - \eta)^{B+1} \|\mathbf{X} \beta^*\|, \quad (20)$$

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

The condition  $\text{span}(\mathbf{X} \mathbf{U}_k) \cap \text{span}(\mathbf{X} \mathbf{U}_\ell) = \{0\}$  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  $m \geq 2r$ , regardless of the number of subspaces in the union.<sup>4</sup> Moreover, based on results in compressive sensing using

<sup>4</sup>This is because if  $\text{span}(\mathbf{U}_k)$ ,  $k = 1, \dots, K$ , are generic  $r$ -dimensional subspaces in  $\mathbb{R}^p$ , then  $\mathcal{V}_k = \text{span}(\mathbf{X} \mathbf{U}_k)$ ,  $k = 1, \dots, K$ , 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.

low-rank Gaussian mixture models [55], [58], we conjecture this condition can be weakened under appropriate assumptions on  $\mathbf{X}$  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 [59]. 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. III-C), we show that an unrolled gradient descent network as defined in (3) with the same piecewise linear  $R = R^*$  as defined in (19) satisfies the error bounds as in Theorem 1:

**Corollary 1:** Under the same assumptions as Theorem 1, the unrolled gradient descent estimator  $\tilde{\beta}'(\mathbf{y}) = \beta^{(B)}$  with step size  $\eta \in (0, 1)$  and  $R = R^*$  as defined in (19) satisfies

$$\|\tilde{\beta}'(\mathbf{X} \beta^*) - \beta^*\| \leq (1 - \eta)^{B+1} \|\mathbf{X} \beta^*\|, \quad (21)$$

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

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.

### C. 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.,  $\mathbf{X} \in \mathbb{R}^{5 \times 10}$  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).

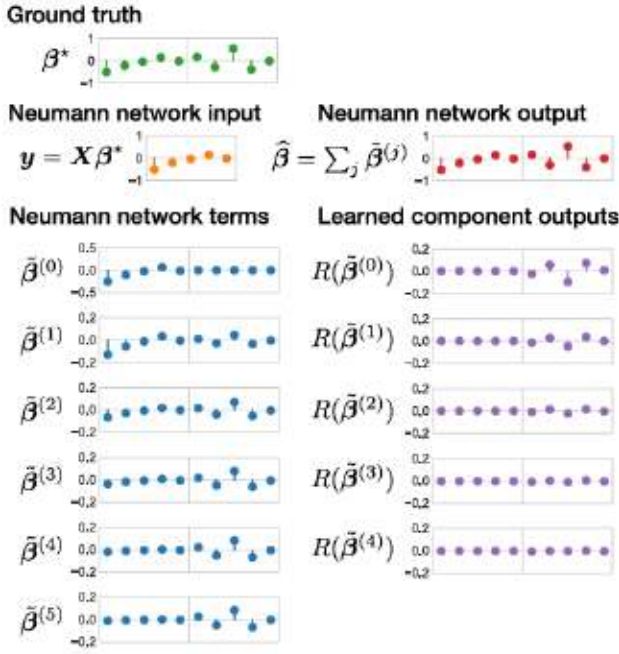


Fig. 4. Example output of Neumann network trained on synthetic union of subspaces data for a 1-D inpainting task. Here a vector  $\beta^* \in \mathbb{R}^{10}$  is drawn from one of the subspaces, and its measurements  $y = X\beta^*$  (restriction to first five coordinates) are input into the Neumann network, which faithfully restores the missing coordinates. The output  $\hat{\beta}$  of the Neumann network is a sum of terms  $\tilde{\beta}^{(j)}$  (shown in bottom left). As predicted by Theorem 1, the terms  $\tilde{\beta}^{(j)}$  are weighted linear combinations the projections of  $\beta^*$  onto the observed and unobserved coordinates. Also as predicted by Theorem 1, the outputs of the learned component  $R(\tilde{\beta}^{(j)})$  (shown in bottom right) are zero in the observed coordinates and scaled projections of  $\beta^*$  in the unobserved coordinates.

Fig. 4 illustrates the output of the trained Neumann network for one specific input, including the outputs from the intermediate Neumann network terms  $\tilde{\beta}^{(j)}$  and the learned component outputs  $R(\tilde{\beta}^{(j)})$ . As predicted by Theorem 1, the Neumann network terms  $\tilde{\beta}^{(j)}$  have the form  $a_j X^\top X \beta^* + b_j (I - X^\top X) \beta^*$  for some constants  $a_j$  and  $b_j$ . Also, the outputs of the learned component  $R(\tilde{\beta}^{(j)})$  all lie in the null space of  $X$ , *i.e.*, are vectors supported on coordinates 6 – 10.

Fig. 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  $R(\beta_1^* + \beta_2^*) \approx R(\beta_1^*) + R(\beta_2^*)$ , for all  $\beta_1^*, \beta_2^*$  belonging to the same subspace. As baselines we compare to the case where  $\beta_1^*$  and  $\beta_2^*$  belong to different subspaces, and the case where  $\beta_1^*$  and  $\beta_2^*$  are Gaussian random vectors. In Fig. 5 we display a boxplot of the relative error  $\|R(\beta_1^* + \beta_2^*) - R(\beta_1^*) - R(\beta_2^*)\|/\gamma$  of 1024 randomly generated  $\beta_1^*, \beta_2^*$ , which are normalized such that  $\|\beta_1^*\| = \|\beta_2^*\| = \gamma$ . 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 error concentrates near zero in the case where  $\beta_1^*, \beta_2^*$  belong to the same subspace, and is otherwise large, indicating

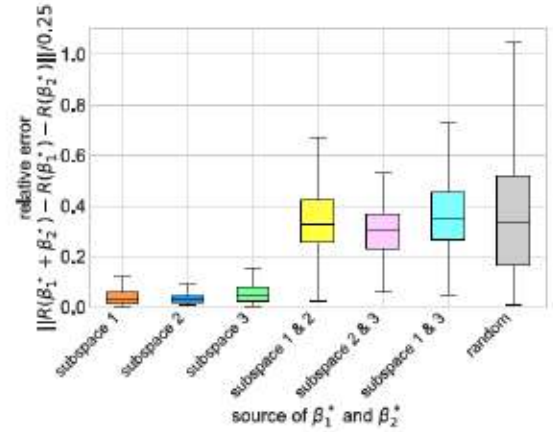


Fig. 5. Piecewise linearity test. We measure how linear the learned  $R$  is when evaluated at two vectors drawn from the same subspace, from two different subspaces, or from two random Gaussian vectors. The plot illustrates that the learned  $R$  only behaves like a linear operator when the vectors 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.

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  $X$ . 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  $X$  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.

## V. 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.

### A. Datasets and Comparison Methods

In our experiments, we consider three different small-scale training sets: CIFAR10 [60], CelebA [61], and STL10 [62], 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 [60]. The dataset has been resized to be  $32 \times 32$  pixels. We use a subset of the aligned Celebrity Faces With Attributes (CelebA) dataset [61]. 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 [62] 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 CelebA and STL10 images to be  $64 \times 64$  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  $\epsilon$  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 IV, 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) [47], 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) [33] 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 [63], is an *agnostic* method. In Section II, we discussed agnostic methods that learn a mapping from  $y$  to  $\beta$ , but in these experiments, we consider a variant of an agnostic learner that learns a mapping from  $X^T y$  to  $\hat{\beta}$  but does not otherwise use  $X$ . 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) [17] 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 [17], 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 [64], with hyperparameters chosen via cross-validation over a held-out validation set for each dataset and inverse problem.

### B. Training and Implementation

Given training pairs  $\{(\beta_i, y_i)\}_{i=1}^N$ , and assuming the learned component  $R$  inside the Neumann network depends smoothly on a set of parameters  $\theta$ , *i.e.*, the partial derivatives  $\partial_{\theta} R(\beta; \theta)$  exist, we train a Neumann network  $\hat{\beta}$  by minimizing the empirical risk  $\mathcal{L}(\theta) = \sum_{i=1}^N \|\hat{\beta}(y_i; \theta) - \beta_i\|^2$ . In the Supplemental Materials we derive the backpropagation gradients  $\partial_{\theta} \mathcal{L}(\theta)$  in the case where  $\hat{\beta}$  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 [65], inspired by architectural choices in [63], [66], [67].

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 V-E. 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.

TABLE I  
PSNR COMPARISON FOR THE CIFAR10, CELEBA, AND STL10 DATASETS  
RESPECTIVELY. VALUES REPORTED ARE THE MEDIAN ACROSS A  
TEST SET OF SIZE 256

	Inpaint	Deblur	Deblur+e	CS2	CS8	SR4	SR10	
CIFAR10	NN	28.20	36.55	29.43	33.83	25.15	24.48	23.09
	PNN	28.40	<b>37.83</b>	<b>30.47</b>	33.75	23.43	<b>26.06</b>	21.79
	GDN	27.76	31.25	29.02	<b>34.99</b>	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	<b>29.05</b>	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	<b>31.06</b>	31.01	30.43	<b>35.12</b>	<b>28.38</b>	27.31	23.57
	PNN	30.45	<b>33.79</b>	<b>30.89</b>	32.61	26.41	<b>28.70</b>	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	<b>24.92</b>
	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	<b>31.98</b>	<b>26.65</b>	24.88	21.80
	PNN	28.00	<b>30.66</b>	27.21	31.40	23.43	<b>25.95</b>	<b>22.19</b>
	GDN	<b>28.07</b>	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

The ResAuto architecture imitates the architecture of [63], an approach that highly resembles the U-Net [68], 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. Further implementation details can be found in supplementary materials.

### C. 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 I.

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 Fig. 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.

The residual autoencoder in particular has excellent performance on certain problems like inpainting and superresolution,

Original and  
 $X^T y$  NN GDN ResAuto



Fig. 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.

but is not competitive for compressed sensing and deblurring. Recall the motivation for the residual autoencoder: the closer  $X^T y$  is to the ground truth  $\beta^*$ , the simpler the residual  $\beta^* - X^T y$  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  $X^T y$  is likely to be a poor approximation of  $\beta^*$ .

The difference in performance between PNN and NN in Table I 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.

In addition, we observe some variance in results across datasets. CIFAR10 in particular seems to be an outlier: while the best reconstructions on CelebA are uniformly more accurate than on STL10, the apparent "difficulty" of reconstruction in CIFAR10 is more task-specific.

Fig. 6 and 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  $\beta^* - \hat{\beta}$  where  $\beta^*$  is the true image, and  $\hat{\beta}$  the estimate, scaled by a factor of 6. Magnitudes are clipped to be less than or equal to 1.

### D. Effect of Sample Size

In Section II-B we hypothesized that incorporating information about the forward operator would have implications for the sample sizes required to achieve particular error rates.

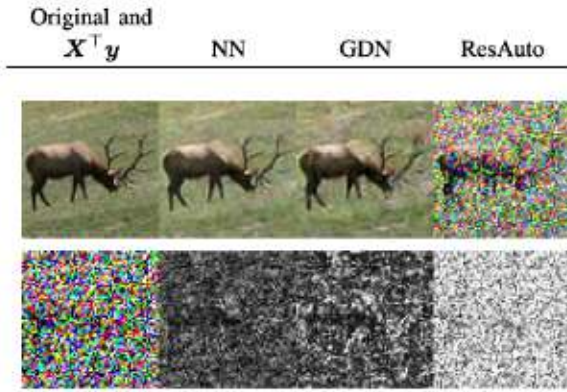


Fig. 7. 8x compressed sensing reconstruction comparison on STL10. ResAuto fails to invert the compressed sensing problem adequately. The Gradient Descent network (GDN) reconstructs accurately but generates more artifacts than the Neumann network (NN).

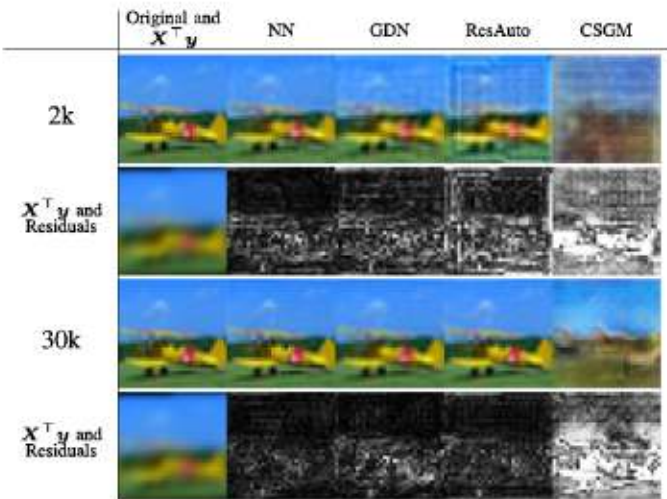


Fig. 8. A qualitative comparison of the reconstructions produced for the deblurring problem on a single image at two different training set sizes, along with the associated residual images. Residual images are scaled by a factor of 6.

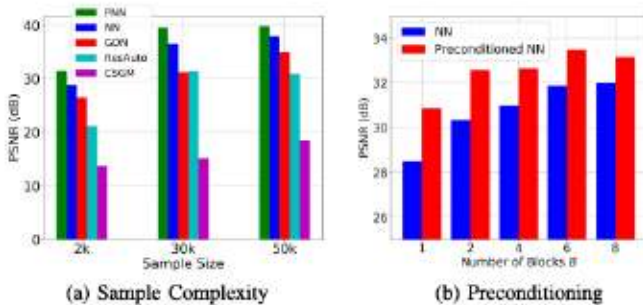


Fig. 9. Performance comparisons. (a) Median PSNR of methods trained with different sample sizes of 2,000, 30,000, and 50,000. Neumann networks (NN) and Preconditioned Neumann networks (PNN) scale very well with training set size, with smaller marginal gains as training sizes increase. All PSNR values 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  $5 \times 5$  and variance  $\sigma = 5.0$ .

A coarse comparison of the presented learning-based methods at different sample sizes is provided in Fig. 9(a). 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 network performs well even at very low sample sizes, but artifacts are present in reconstructions at low sample sizes, visible in Fig. 8.

Methods that do not incorporate the forward model, like ResAuto and CSGM, perform poorly in the low-sample regime, as discussed in Section II-B. While ResAuto performs competitively at 30 k 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.

### E. Effect of Preconditioning

Fig. 9(b) 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  $5 \times 5$ . The corresponding  $X$  is very poorly conditioned, and a  $\lambda$  of 0.01 is used in the preconditioning matrix  $(X^T X + \lambda I)^{-1}$ .

Depending on the structure of  $X$  and how easily  $(X^T X + \lambda I)^{-1}$  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.

### F. 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 [47], 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 [47].

All experiments are for  $4 \times$  undersampling, although we differ from [47] 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 Fig. 10.

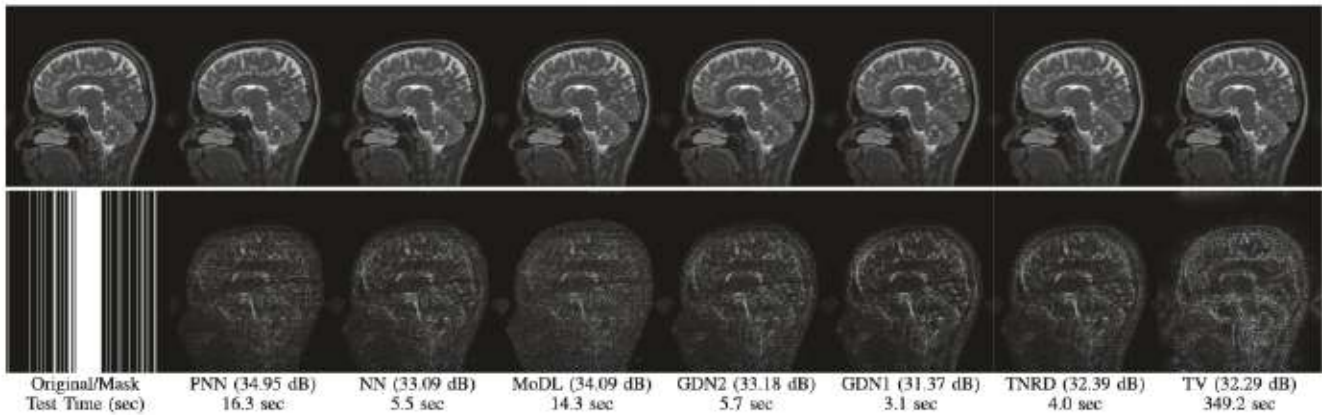


Fig. 10. A comparison of MRI reconstruction quality for a variety of trainable and non-trainable image reconstruction methods. The 12-coil data is undersampled by a factor of  $4\times$  and Gaussian noise with  $\sigma = 0.01$  is added in k-space. The reconstructions are displayed in the first row, 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 the MRI experiments we follow the precedent set by [47] in our choice of learned component, using only a simple five-layer convolutional network with 64 filters per layer and ReLU nonlinearities for all architectures other than TNRD. The TNRD architecture follows the architecture proposed in [33]. 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 [47]. 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 Fig. 10. 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.

### G. 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 [14] (the inclusion of the identity  $I$  in the linear part

$[I - \eta X^T X](\cdot)$  of each block of NN and GDN is a residual connection). Other architectures based on unrolled iterative methods such as MoDL and TNRD have similar residual connections. However, the main difference is that NN contain additional “skip” connections that connect each block with the final layer, similar to architectures like DenseNets [15]. Recent work [69] 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 minima tend to be present, and those minima tend to be wide, as opposed to sharp. In addition, the authors of [69] 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 Fig. 11 we illustrate the optimization landscapes using the method of [69], 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  $\hat{\theta} \in \mathbb{R}^K$ . We draw two independent standard Gaussian vectors  $v_1, v_2$  with dimension  $K$ , and normalize them in the manner described in [69] that accounts for the scaling ambiguity of ReLU networks. Then we compute the test and training set error for parameters given by  $\hat{\theta} + \tau(iv_1 + jv_2)$  for step size  $\tau > 0$  and integers  $i, j$ . The plots above are generated for  $i, j \in \{-125, \dots, 125\}$  and  $\tau = 0.01$ . We demonstrate plots for three different forward models: deblurring, compressed sensing with  $8\times$  compression, and  $10\times$  superresolution.

Fig. 11 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

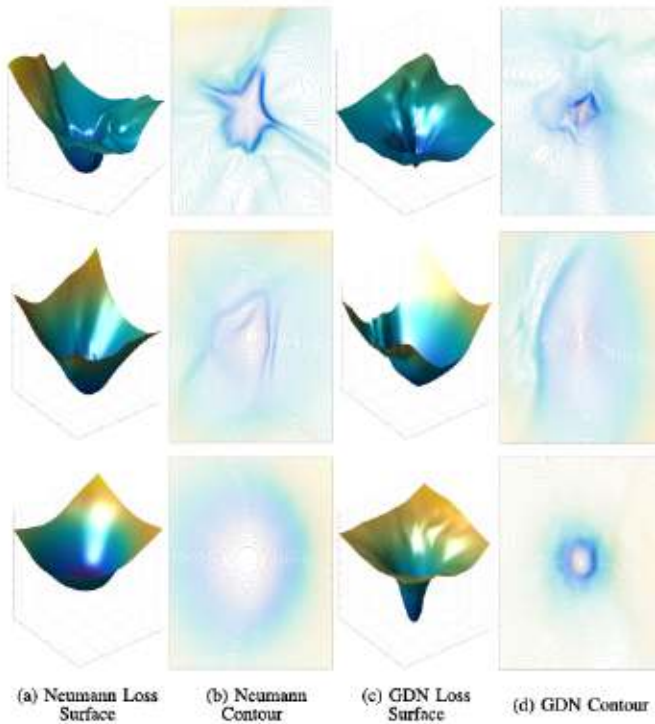


Fig. 11. Optimization landscapes and contour plots. (a) The optimization landscape associated with the training loss of the Neumann network around the center optimal point. (b) The associated contour plot. (c) The optimization landscape associated with the training loss of the gradient descent network. (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.

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 [70]. 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.

## VI. CONCLUSION

This paper describes a novel network architecture that departs from the currently-popular unrolled optimization framework described in Section II-C. Our approach is based on the Neumann series expansion for inverting linear operators and has several key features. First, Neumann networks naturally contain “skip connections” [14], [15] that appear to yield optimization landscapes that facilitate more efficient training, but are absent from previously proposed network architectures.

Our theoretical analysis reveals that when the training data lie in a union of subspaces, the optimal *oracle* estimator that has prior knowledge of both the subspaces in the union and the identity of the subspace to which true image belongs is

piecewise linear. We show this piecewise linear oracle estimator can be approximated arbitrarily well by a Neumann network whose learned component coincides with a specific piecewise linear map, which in principle can always be realized by a neural network using ReLU activations. Furthermore, we observe empirically on simulated union-of-subspaces data that the nonlinear learned component in the trained Neumann network well-approximates the specific piecewise linear map predicted by theory. We are unaware of past work on using neural networks to solve inverse problems demonstrating such properties.

Third, we describe a simple preconditioning step that, when combined with the Neumann network architecture, provides an additional increase in reconstruction PSNR and can reduce the number of blocks  $B$  needed for accurate reconstruction, which in turn decreases reconstruction computational complexity when the preconditioning can be computed efficiently. As a result, using a truncated series expansion with only  $B$  blocks results in a small, bounded approximation error. Finally, we explore the proposed Neumann network’s empirical performance on a variety of inverse problems relative to the performance of representative agnostic, decoupled, and unrolled optimization methods described in Section II.

While this paper has focused on solving linear inverse problems in imaging using training data to train a neural network, *more generally we can think of this paper as a case study in leveraging physical models to guide neural network architecture design*. More specifically, we can think of networks such as the Neumann network as a single large neural network in which a subset of edge weights (*i.e.*, those corresponding to the operation  $I - \eta X^\top X$  and other zero-valued “edges” that define the general architecture of Fig. 2) are determined by the physical forward model that specifies the inverse problem at hand and are held fixed during training, while the remaining edges (*i.e.*, those that correspond to the operation  $R(\cdot)$ ) can be learned during training. In other words, *we use knowledge of the inverse problem structure to define the neural network architecture*.

This perspective leads to interesting potential avenues for future work. Specifically, our proposed Neumann network is inspired by series expansions for inverting linear operators, but there are alternative methods for inverting nonlinear operators that may yield new challenges and opportunities. For instance, Adomian decompositions and polynomial expansions have been successfully used to solve differential equations with both linear and nonlinear components [71], [72], and so designing future networks inspired by this framework could lead to new theoretical insights beyond what we present above.

In addition, the reader might note that neural networks based on the Neumann series expansion or iterative optimization methods have several repeated blocks, leading to the question of whether standard stochastic gradient descent is the most efficient training regimen. For instance, recent work on “Neural Ordinary Differential Equations” [73] has considered an ODE representation of the operation of a neural network instead of a series of discrete layers and used this perspective to devise training methods that leverage ODE solvers for more efficient training. Such techniques might be leveraged to improve training of Neumann networks.

## APPENDIX

Here we let  $P_X$  and  $P_{X_\perp}$  denote the projectors onto the row space of  $X$  and the null space of  $X$ , respectively. In particular,  $P_X = X^T X$  and  $P_{X_\perp} = I - X^T X$ , since we assume  $X$  has orthonormal rows in Lemma 1, Theorem 1, and Corollary 1.

## A. Proof of Lemma 1

We have  $\hat{\beta}(y) = \sum_{j=0}^B \tilde{\beta}^{(j)}$  where  $\tilde{\beta}^{(0)} = \eta X^T y$ ,  $y = X\beta^*$ , and

$$\tilde{\beta}^{(j)} = (I - \eta X^T X - \eta R) \tilde{\beta}^{(j-1)} \quad (22)$$

$$= (P_{X_\perp} + (1 - \eta)P_X - \eta R) \tilde{\beta}^{(j-1)} \quad (23)$$

for all  $j = 1, \dots, B$ , and where in the last line we used the identity  $I = P_X + P_{X_\perp}$ .

We show that  $R(\beta) = R\beta$  with  $R$  as specified in Lemma 1 satisfies the desired error bounds. Define  $Q = P_{X_\perp} U(U^T X^T X U)^{-1} U^T X^T$  so that  $R = -c_{\eta, B} Q X$ . With this choice of  $R$  we have  $P_X R = 0$ , and an easy induction shows

$$\tilde{\beta}^{(j)} = \eta(1 - \eta)^j X^T y - \eta \sum_{k=0}^{j-1} R \tilde{\beta}^{(k)} \quad (24)$$

for all  $j \geq 1$ . Summing this over  $j = 0, 1, \dots, B$  gives

$$\begin{aligned} \hat{\beta}(y) &= \sum_{j=0}^B \eta(1 - \eta)^j X^T y - \eta \sum_{j=0}^B \sum_{k=0}^{j-1} R \tilde{\beta}^{(k)} \\ &= \sum_{j=0}^B \eta(1 - \eta)^j X^T y - \eta \sum_{j=0}^{B-1} (B - j) R \tilde{\beta}^{(j)} \end{aligned} \quad (25)$$

Next, we show we can choose the constant  $c_{\eta, B}$  so that the second term above simplifies to

$$-\eta \sum_{j=0}^{B-1} (B - j) R \tilde{\beta}^{(j)} = Q y. \quad (26)$$

Observe that  $R^2 \tilde{\beta}^{(k)} = 0$  for all  $k = 0, \dots, j - 1$ , and so from (24) we have  $R \tilde{\beta}^{(j)} = \eta(1 - \eta)^j R X^T y$ , which gives

$$\sum_{j=0}^{B-1} (B - j) R \tilde{\beta}^{(j)} = \eta \sum_{j=0}^{B-1} (B - j)(1 - \eta)^j R X^T y.$$

Letting  $c_{\eta, B} = (\eta^2 \sum_{j=0}^{B-1} (B - j)(1 - \eta)^j)^{-1}$  we obtain (26). Therefore, combining (25) and (26) we have

$$\hat{\beta}(y) = \eta \sum_{j=0}^B (1 - \eta)^j X^T y + Q y. \quad (27)$$

Finally, since we assume  $y = X\beta^*$ , we see that  $X^T y = P_X \beta^*$  and  $Q y = P_{X_\perp} \beta^*$ , and using the fact that  $\eta \sum_{j=0}^B (1 - \eta)^j = 1 - (1 - \eta)^{B+1}$  from (27) we have  $\hat{\beta}(y) = \beta^* - (1 - \eta)^{B+1} P_X \beta^*$  which gives the desired error bound.

## B. Proof of Theorem 1 and Corollary 1

To prove Theorem 1 we show that if  $\beta^*$  belongs to the  $k$ th subspace then  $R^*$  acts as the linear map  $R_k$  when applied to each Neumann network term  $\tilde{\beta}^{(j)}$ . That is, we show  $\tilde{\beta}^{(j)} \in \mathcal{C}_k$  for all  $j = 0, \dots, B$ , where  $\tilde{\beta}^{(0)} = \eta X^T y$  and  $\tilde{\beta}^{(j)} = (I - \eta X^T X) \tilde{\beta}^{(j-1)} - \eta R^* (\tilde{\beta}^{(j-1)})$  for  $j = 1, \dots, B$ . The desired error bounds then follow by direct application of Lemma 1.

First, an easy induction shows that

$$\tilde{\beta}^{(j)} = \eta(1 - \eta)^j X^T y - \eta \sum_{i=0}^{j-1} R^* (\tilde{\beta}^{(i)}). \quad (28)$$

Using the fact that  $P_X R^* (\beta) = 0$  for all  $\beta \in \mathbb{R}^p$ , and  $y = X\beta^*$ , we have  $P_X \tilde{\beta}^{(j)} = \eta(1 - \eta)^j P_X \beta^*$  for all  $j = 0, \dots, B$ . Since the region  $\mathcal{C}_k$  is a cone, in order to prove  $\tilde{\beta}^{(j)} \in \mathcal{C}_k$  for all  $j = 0, \dots, B$  it suffices to show  $P_X \beta^* \in \mathcal{C}_k$ . This means we need to show  $d_{X, k}(\beta^*) < d_{X, \ell}(\beta^*)$  for all  $\ell \neq k$ , or equivalently,

$$\begin{aligned} &\|(I - X U_k (X U_k)^+) X \beta^*\| \\ &< \|(I - X U_\ell (X U_\ell)^+) X \beta^*\| \end{aligned}$$

for all  $\ell \neq k$ . Since  $X U_k (X U_k)^+$  is projection onto  $\text{span}(X U_k)$ , and  $X \beta^* \in \text{span}(X U_k)$ , we have  $(I - X U_k (X U_k)^+) X \beta^* = 0$  and so  $\|(I - X U_k (X U_k)^+) X \beta^*\| = 0$ . Furthermore, since by assumption  $X \beta^* \notin \text{span}(X U_\ell)$  for all  $\ell \neq k$ , we have  $(I - X U_\ell (X U_\ell)^+) X \beta^* \neq 0$ , which means  $\|(I - X U_\ell (X U_\ell)^+) X \beta^*\| > 0$ , proving the claim.

Similarly, to prove Corollary 1 we need to show  $\beta^{(j)} \in \mathcal{C}_k$  for all  $j = 0, \dots, B$ , where  $\beta^{(0)} = \eta X^T y \in \mathcal{C}_k$  and  $\beta^{(j)} = (I - \eta X^T X) \beta^{(j-1)} - \eta R^* (\beta^{(j-1)}) + \beta^{(0)}$ . Since  $P_X R^* (\beta) = 0$  for all  $\beta \in \mathbb{R}^p$ , an easy induction shows that

$$P_X \beta^{(j)} = \eta \sum_{i=0}^j (1 - \eta)^i P_X \beta^*. \quad (29)$$

Since each  $\beta^{(j)}$  is a scalar multiple of  $P_X \beta^*$ , by the same argument as above we have  $\beta^{(j)} \in \mathcal{C}_k$  for all  $j = 0, \dots, B$ , which proves the claim.

## REFERENCES

- [1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. London, U.K.: Pearson, 2007.
- [2] J. A. Fessler, "Model-based image reconstruction for MRI," *IEEE Signal Process. Mag.*, vol. 27, no. 4, pp. 81–89, Jul. 2010.
- [3] I. Elbakri and J. Fessler, "Statistical image reconstruction for polyenergetic X-ray computed tomography," *IEEE Trans. Med. Imag.*, vol. 21, no. 2, pp. 89–99, Feb. 2002.
- [4] R. E. Blahut, *Theory of Remote Image Formation*. New York, NY, USA: Cambridge University Press, 2004.
- [5] H. H. Barrett and K. J. Myers, *Foundations of Image Science*. Hoboken, NJ, USA: John Wiley & Sons, 2013.
- [6] A. N. Tychonoff and V. Arsenin, *Solution of Ill-Posed Problems*. Washington, DC, USA: Winston & Sons, 1977.
- [7] M. A. Figueiredo and R. D. Nowak, "A bound optimization approach to wavelet-based image deconvolution," in *Proc. IEEE Int. Conf. Image Process.*, 2005, pp. 782–785.
- [8] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. Int. Conf. Mach. Learn. ACM*, 2009, pp. 689–696.

- [9] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity," *IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2481–2499, May 2012.
- [10] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992.
- [11] R. M. Willett and R. D. Nowak, "Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging," *IEEE Trans. Med. Imag.*, vol. 22, no. 3, pp. 332–350, Mar. 2003.
- [12] A. Danielyan, V. Katkovich, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1715–1728, Apr. 2012.
- [13] W. Marais and R. Willett, "Proximal-gradient methods for Poisson image reconstruction with BM3D-based regularization," in *Proc. IEEE 7th Int. Workshop Comput. Adv. Multi-Sensor Adapt. Process.*, 2017, pp. 1–5.
- [14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.
- [15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, vol. 1, no. 2, p. 3.
- [16] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," in *Proc. Int. Conf. Learn. Represent.*, 2017. [Online]. Available: <https://openreview.net/forum?id=S1VaB4cex&notId=S1VaB4cex>
- [17] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 537–546.
- [18] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems," in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 1799–1808.
- [19] P. Schafheitlin, *Die Theorie Der Besselschen Funktionen*, vol. 4. Leipzig, Germany: BG Teubner, 1908.
- [20] I. Gohberg and S. Goldberg, *Basic Operator Theory*. Basel, Switzerland: Birkhäuser, 2013.
- [21] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. Adv. Neural Inform. Process. Syst.*, 2014, pp. 1790–1798.
- [22] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vision*. Springer, 2014, pp. 184–199.
- [23] C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 4681–4690.
- [24] Y. Tan, D. Zhang, F. Xu, and D. Zhang, "Motion deblurring based on convolutional neural network," in *Proc. Int. Conf. Bio-Inspired Comput. Theories Appl.* Springer, 2017, pp. 623–635.
- [25] M. Schmidt, N. L. Roux, and F. R. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," in *Proc. Adv. Neural Inform. Process. Syst.*, 2011, pp. 1458–1466.
- [26] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.
- [27] J. R. Chang, C.-L. Li, B. Póczos, and B. V. Kumar, "One network to solve them all — Solving linear inverse problems using deep projection models," in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 5889–5898.
- [28] H. Gupta, K. H. Jin, H. Q. Nguyen, M. T. McCann, and M. Unser, "CNN-based projected gradient descent for consistent CT image reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1440–1453, Jun. 2018.
- [29] S. Efronovich, "Conditional density estimation in a regression setting," *Ann. Statist.*, vol. 35, no. 6, pp. 2504–2535, 2007.
- [30] D. L. Donoho, I. M. Johnstone, G. Kerkycharian, and D. Picard, "Density estimation by wavelet thresholding," *Ann. Statist.*, vol. 24, no. 2, pp. 508–539, 1996.
- [31] B. Delyon and A. Juditsky, "On minimax wavelet estimators," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 3, pp. 215–228, 1996.
- [32] J. Lafferty, H. Liu, and L. Wasserman, *Minimax theory*. (2008). Accessed Jan. 7, 2019. [Online]. Available: <http://www.stat.cmu.edu/~larryf/sml/Minimax.pdf>
- [33] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Jun. 2017.
- [34] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 399–406.
- [35] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
- [36] P. Sprechmann, A. Bronstein, and G. Sapiro, "Learning efficient structured sparse models," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 219–226.
- [37] U. S. Kamilov and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 747–751, May 2016.
- [38] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein, "Unrolled optimization with deep priors," *arXiv:1705.08041*, 2017.
- [39] M. Mardani et al., "Neural proximal gradient descent for compressive imaging," in *Proc. Adv. Neural Inform. Process. Syst.*, 2018, pp. 9573–9583.
- [40] J. Sun, H. Li, and Z. Xu, "Deep ADMM-Net for compressive sensing MRI," in *Proc. Adv. Neural Inform. Process. Syst.*, 2016, pp. 10–18.
- [41] J. Adler and O. Öktem, "Learned primal-dual reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1322–1332, Jun. 2018.
- [42] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2014, pp. 2774–2781.
- [43] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Honolulu, HI, 2017, vol. 2, pp. 2808–2817.
- [44] S. Ravishanker, I. Y. Chun, and J. A. Fessler, "Physics-driven deep training of dictionary-based algorithms for MR image reconstruction," in *Proc. Asilomar Conf. Signals Syst., Comput.*, 2017, pp. 1859–1863.
- [45] S. Ravishanker, A. Lahiri, C. Blocker, and J. A. Fessler, "Deep dictionary-transform learning for image reconstruction," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2018, pp. 1208–1212.
- [46] Y. Chun and J. A. Fessler, "Deep BCD-net using identical encoding-decoding CNN structures for iterative image recovery," in *Proc. IEEE Image, Video, Multidimens. Signal Process. Workshop*, 2018, pp. 1–5.
- [47] H. K. Aggarwal, M. P. Mani, and M. Jacob, "MoDL: Model based deep learning architecture for inverse problems," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 394–405, Feb. 2019.
- [48] H. K. Aggarwal, M. P. Mani, and M. Jacob, "MoDL-MUSSELS: model-based deep learning for multishot sensitivity-encoded diffusion MRI," *IEEE Trans. Med. Imag.*, to be published.
- [49] A. Pramanik, H. K. Aggarwal, and M. Jacob, "Off-the-grid model based deep learning (O-MoDL)," in *Proc. IEEE 16th Int. Symp. Biomed. Imag.*, 2019, pp. 1395–1398.
- [50] C. Metzler, A. Mousavi, and R. Baraniuk, "Learned D-AMP: Principled neural network based compressive image recovery," in *Proc. Adv. Neural Inform. Process. Syst.*, 2017, pp. 1772–1783.
- [51] M. R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, vol. 49. Washington, DC, USA: NBS, 1952.
- [52] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2009, pp. 2790–2797.
- [53] T. Blumensath, "Sampling and reconstructing signals from a union of linear subspaces," *IEEE Trans. Inform. Theory*, vol. 57, no. 7, pp. 4660–4671, Jul. 2011.
- [54] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.
- [55] F. Renna, R. Calderbank, L. Carin, and M. R. Rodrigues, "Reconstruction of signals drawn from a Gaussian mixture via noisy compressive measurements," *IEEE Trans. Signal Process.*, vol. 62, no. 9, pp. 2265–2277, May 2014.
- [56] J. Yang et al., "Compressive sensing by learning a Gaussian mixture model from measurements," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 106–119, Jan. 2015.
- [57] A. Houdard, C. Bouveyron, and J. Delon, "High-dimensional mixture models for unsupervised image denoising (HDMI)," *SIAM J. Imag. Sci.*, vol. 11, no. 4, pp. 2815–2846, 2018.
- [58] H. Reboredo, F. Renna, R. Calderbank, and M. R. Rodrigues, "Compressive classification," in *Proc. IEEE Int. Symp. Inform. Theory*, 2013, pp. 674–678.
- [59] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," in *Proc. Int. Conf. Learn. Represent.*, 2018. [Online]. Available: [https://openreview.net/forum?id=B1J\\_rgWRW](https://openreview.net/forum?id=B1J_rgWRW)
- [60] A. Krizhevsky, "Learning multiple layers of features from tiny images," CiteSeer, Tech. Rep., 2009.
- [61] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vision*, Santiago, Chile, Dec. 2015, pp. 3730–3738.
- [62] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

- [63] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. Adv. Neural Inform. Process. Syst.*, 2016, pp. 2802–2810.
- [64] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imag. Sci.*, vol. 1, no. 3, pp. 248–272, 2008.
- [65] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 2536–2544.
- [66] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.
- [67] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1646–1654.
- [68] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent.*, Springer, 2015, pp. 234–241.
- [69] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inform. Process. Syst.*, 2018, pp. 6389–6399.
- [70] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv:1609.04836*, 2016.
- [71] L. Gabet, "The theoretical foundation of the Adomian method," *Comput. Math. Appl.*, vol. 27, no. 12, pp. 41–52, 1994.
- [72] G. Adomian, *Solving Frontier Problems of Physics: The Decomposition Method*, vol. 60. New York, NY, USA: Springer Science & Business Media, 2013.
- [73] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv:1806.07366*, 2018.



**Davis Gilton** received the B.S. in electrical engineering from the Arizona State University, Tempe, AZ, USA, in 2016. He is a Graduate Student at the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI, USA.

His research interests include machine learning, computational imaging, and statistical signal processing.



**Greg Ongie** received the Ph.D. degree in applied mathematics and computational sciences from the University of Iowa, IA, USA, in 2016.

He is a Postdoctoral Scholar with the Department of Statistics, University of Chicago, IL, USA. He also was a Postdoctoral Scholar with the University of Michigan, Ann Arbor, MI, USA, from 2016 to 2018. His research interests include machine learning, compressed sensing, and image reconstruction problems in biomedical imaging.



**Rebecca Willett** received the Ph.D. degree in electrical and computer engineering from the Rice University, Houston, TX, USA, in 2005.

She was an Assistant then tenured Associate Professor with the Electrical and Computer Engineering Department, Duke University, from 2005 to 2013. She was an Associate Professor of Electrical and Computer Engineering and Harvey D. Spangler Faculty Scholar. She is a Professor of Statistics and Computer Science with the University of Chicago. Her research interests include machine learning, signal processing,

and large-scale data science.

Dr. Willett is a Fellow of the Wisconsin Institutes for Discovery, University of Wisconsin-Madison, USA, from 2013 to 2018. She received the National Science Foundation CAREER Award in 2007, was a member of the DARPA Computer Science Study Group, and received an Air Force Office of Scientific Research Young Investigator Program award in 2010.