Deep Decomposition Learning for Inverse Imaging Problems

Deep Decomposition Learning for Inverse Imaging Problems

Dongdong Chen, Mike E. Davies
School of Engineering, University of Edinburgh, UK
{d.chen, mike.davies}@ed.ac.uk

Abstract

Deep learning is emerging as a new paradigm for solving inverse imaging problems. However, the deep learning methods often lack the assurance of traditional physics-based methods due to the lack of physical information considerations in neural network training and deploying. The appropriate supervision and explicit calibration by the information of the physic model can enhance the neural network learning and its practical performance. In this paper, inspired by the geometry that data can be decomposed by two components from the null-space of the forward operator and the range space of its pseudo-inverse, we train neural networks to learn the two components and therefore learn the decomposition, i.e. *we explicitly reformulate the neural network layers as learning range-nullspace decomposition functions with reference to the layer inputs, instead of learning unreferenced functions. We show that the decomposition networks not only produce superior results, but also enjoy good interpretability and generalization. We demonstrate the advantages of decomposition learning on different inverse problems including compressive sensing and image super-resolution as examples.*

1. Introduction

We consider a linear inverse problem of the form :

$$\mathbf{y}_\epsilon = \mathbf{H}\mathbf{x} + \epsilon, \tag{1}$$

where the goal is to recover the unknown signal $\mathbf{x} \in \mathbb{R}^D$ from the noisy measurement $\mathbf{y}_\epsilon \in \mathbb{R}^d$ with typical dimension $D \gg d$, and $\mathbf{H} : \mathbb{R}^D \to \mathbb{R}^d$ is the forward operator which models the response of the acquisition device or reconstruction system, while $\epsilon \in \mathbb{R}^d$ represents the measurement noise intrinsic to the acquisition process.

Inverse problems have wide applications in computer vision, medical imaging, optics, radar, and many other fields. The forward operator $\mathbf{H}$ in (1) could represent various inverse problems, from *e.g.* an identity operator for image denoising, to convolution operators for image deblurring, random sensing matrices for compressive sensing(CS), filtered subsampling operators for super-resolution (SR), (undersampled) Fourier transform for magnetic resonance imaging (MRI) and the (subsampled) Radon transform in computed tomography (CT). The inverse problems in (1) are often noisy and ill-posed since the operator $\mathbf{H}$ has a non-trivial null space. Such under-determined systems are extremely difficult to solve and the solutions are very sensitive to the input data. The classical approach for solving them have traditionally been model-based [3, 8], which typically aim to regularize the solutions by constraining them to be consistent with prior knowledge about the signal and usually can only be solved iteratively.

More recently, due to the powerful representation learning and transformation ability, deep learning [19] or deep neural networks (DNN) have emerged as a new paradigm for inverse problems. The community has already taken significant steps in this direction, with deep neural networks being successfully applied to a wide variety of inverse problems [27, 22, 31]. For example, [4, 41] use a fully connected feedfoward neural network for image denoisng and inpainting. [10, 17] learn end-to-end mappings between $\mathbf{y}_\epsilon$ and $\mathbf{x}$ with vanilla convolutional neural networks (CNN). [44, 20] further use CNNs with residual blocks [13] and skip connections to improve the neural network performance. [24, 29, 15] learn downsampling and upsampling feature maps with encoder-decoder CNNs. [7, 43] use autoencoders for learning new representations for $\mathbf{x}$ and $\mathbf{y}_\epsilon$ to solve the inverse problems. [40] use CNN as a prior and train with early stopping criteria to recover a single image from its observation. [39, 26] unfold the model-based optimizations with DNN. [28, 2] use generative models for natural images to recover images from Gaussian measurements.

However, the DNN itself in the above deep learning-based approaches often lack the guarantees of traditional physics-based methods as they are purely data-driven and learning-based. In addition, the designing of DNNs is usually complicated and has poor intuitive interpretation when they are decoupled from the inverse problem of interest. Furthermore, it is a commonly held belief in the inverse problems community that using the physics is preferable to relying solely on data [35, 25]. This raises a number of

questions: is a purely data-driven neural network the best way to solve an inverse problem? Does physical information facilitate neural networks to find a better inverse problem solution? How should one best make use of the prior physical (acquisition) information?

All of these above questions inspire us to think about whether the introduction of physical information in neural networks would be beneficial to the training of deep learning methods to better solve inverse problems. To that end, in this paper, relying on the range-nullspace decomposition of data and the recently proposed null-space learning [35, 25], we construct a physics-engaged and end-to-end trainable architecture: Deep Decomposition Network (DDN). The contributions of this paper are summarized as follows.

- First, we propose to use two convolutional neural networks to separately capture the residuals lying on the range of $\mathbf{H}^\dagger$ and the nullspace of $\mathbf{H}$. By incorporating the two learned residuals with pseudo-inverse input, the proposed scheme, which we call a deep decomposition network (DDN), is able to recover both the components of the image and preserve data fidelity with respect to the measurements, $\mathbf{y}_\epsilon$. Thanks to the explicit deep decomposition learning and the introduction of physical information, the neural networks also enjoy intuitive interpretability.
- Second, to the best of our knowledge, this is the first method of using such a decomposition in neural networks to directly solve inverse problems. Although previous work has incorporated the physical model as well, these have either neglected the noise component [25, 35] or retained the undesirable iterative nature of model-based systems, *e.g.* [11, 12, 32, 45]. We evaluate DDNs on compressive sensing and image super-resolution tasks. It is shown that the DDN not only achieves superior reconstruction performances but also facilitates the generalization of the deep neural network for inverse problems. Our DDN code is available at: $\star\star\star$.

Notations. In this paper, except in some specified cases, lower-case bold letters $\mathbf{a}$ represent column vectors and upper-case bold ones $\mathbf{A}$ represent matrices. Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, let $\mathcal{R}(\mathbf{A})$ be the range of $\mathbf{A}$, *i.e.*, the subspace of $\mathbb{R}^m$ spanned by the columns of $\mathbf{A}$; the null space (or kernel) of $\mathbf{A}$, denoted by $\mathcal{N}(\mathbf{A})$, is the solution space (a subspace of $\mathbb{R}^n$) of the linear system $\mathbf{Ax} = 0$. $\mathbf{A}^\top$ and $\mathbf{A}^\dagger$ represent the transpose and pseudo-inverse of $\mathbf{A}$, respectively.

2. Background

2.1. Deep learning for the inverse problem

Depending on whether the physical acquisition information with respect to $\mathbf{H}$ is used during DNN training and testing, we divide the deep learning approaches into two categories: *Physics-free* and *Physics-engaged*.

Physics-free. The DNN aims to learn a direct mapping from $\mathbf{y}_\epsilon$ (or its projection, e.g. $\mathbf{H}^\dagger\mathbf{y}_\epsilon$) to $\mathbf{x}$ without exploiting the knowledge of $\mathbf{H}$ at any point in the training or testing process (with the exception of the input). The general principle is that, given enough training data, we should be able to design a proper neural network to learn everything we need to know about $\mathbf{H}$ to successfully estimate $\mathbf{x}$ directly. The success of this approach is likely to depend on the complexity of the forward operator $\mathbf{H}$. However, it has been observed to work well for numerous computer vision tasks, such as denoising and inpainting [41], superresoluion [10] and deblurring [42, 18]. The DNN can be trained by a sole least squares loss [41, 24, 15] or a combination of least squares loss and auxiliary losses such as adversarial loss [20, 18, 30, 37]. 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 $\mathbf{x}$, but also aspects of $\mathbf{H}$. Hence, an interesting question is how hard (relatively) are each of these components to learn and how important is it to incorporate $\mathbf{H}$ into the learning process. When the forward problem is too complex such that it can not be incorporated into the neural network model it will always be necessary to go Physics-free. Finally, since direct estimation using a DNN for solving inverse problems is essentially a form of regression, there is a potential generalization issue with such physic-free DNN approaches.

Physics-engaged. The most widely used strategy considering physics of $\mathbf{H}$ in deep learning approaches is through a model-based approach, in which one or more pretrained DNNs are used within a more traditional iterative physics-engaged model-based framework such as [3, 8]. As mentioned before, the inverse problem (1) typically boils down to solving an optimisation problem broadly of the following form:

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) + \lambda\phi(\mathbf{x}), \tag{2}$$

where the first term $f(\mathbf{x})$ aims to enforce data fidelity, *e.g.* f could be the MSE between $\mathbf{Hx}$ and $\mathbf{y}_\epsilon$, while the regularizer ϕ allows us to insert knowledge onto the solution $\mathbf{x}$, and $\lambda \in \mathbb{R}^+$ controls the strength of the regularization. Typically there is no closed-form solution to (2) and it usually needs to be solved iteratively. This has led to the following proposed uses for pretrained DNNs: (i) use DNN to replace the proximal operator associated with $\phi(\mathbf{x})$ in a proximal gradient algorithm [11, 12, 32, 44, 45], (ii) use DNN to replace the gradient $\nabla\phi$ in an unrolled optimization method [5, 9, 26, 39], (iii) directly replace the regularizer ϕ with DNN [23, 33], (iv) use DNN as a generative model to generate $\mathbf{x}$ from a latent code that needs to be estimated [2, 36]. These iterative methods are Physics-engaged, as they actually use the regularizer along with the forward model and

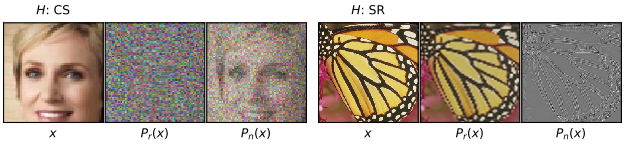


Figure 1: Examples of $\mathcal{R}$-$\mathcal{N}$ decomposition for clean signal $\mathbf{x}$. Left: Compressive sensing ($d/D = 0.1$). Right: Super-resolution (25% downscaled).

observation by minimizing the disparity between the oracle and its reconstruction.

As an exception to the above physics-engaged deep learning approaches, there have been some recent studies aimed at explicitly using $\mathbf{H}$-related information during the DNN training process in an end-to-end manner. For example, [35, 25] explicitly learn the nullspace component of $\mathbf{x}$ with respect to $\mathbf{H}$. However, this separate nullspace learning does not deal with the presence of noise in the input nor with situations where no nullspace exists. Another interesting direction presented in [9] considers a Neumann series expansion of linear operators to approximate the inverse mapping of (2). However, this requires the network to precondition and store the results of each iteration in order to accumulate the approximate Neumann series to solve the inverse problem.

In this paper, inspired by the nullspace method of [35, 25], we explore the possibility of a more flexible end-to-end neural network structure that is capable of exploiting both the range and null space structures of the inverse problem. Before discussing the proposed method, let us briefly recall the Range-Nullspace decomposition of data.

2.2. Range-Nullspace ($\mathcal{R}$-$\mathcal{N}$) Decomposition

Given a linear forward operator $\mathbf{H} \in \mathbb{R}^{d\times D}$ and its right pseudo inverse $\mathbf{H}^\dagger \in \mathbb{R}^{D\times d}$, which satisfies $\mathbf{H}\mathbf{H}^\dagger = \mathbf{I}_d$, it holds that $\mathbb{R}^D = \mathcal{R}(\mathbf{H}^\dagger) \oplus \mathcal{N}(\mathbf{H})$, which implies that for any sample $\forall \mathbf{x} \in \mathbb{R}^D$ there exists two unique elements $\mathbf{x}^+ \in \mathcal{R}(\mathbf{H}^\dagger)$ and $\mathbf{x}^\perp \in \mathcal{N}(\mathbf{H})$ such that $\mathbf{x} = \mathbf{x}^+ + \mathbf{x}^\perp$. Therefore we define the following range-nullspace ($\mathcal{R}$-$\mathcal{N}$) decomposition,

***Definition* 1** *$\mathcal{R}$-$\mathcal{N}$ **Decomposition:** Let $\mathcal{P}_r \triangleq \boldsymbol{H}^\dagger\boldsymbol{H}$ be the operator that projects the sample $\boldsymbol{x}$ from sample domain to the range of $\boldsymbol{H}^\dagger$, and denote by $\mathcal{P}_n \triangleq (\boldsymbol{I}_D - \boldsymbol{H}^\dagger\boldsymbol{H})$ the operator that projects $\boldsymbol{x}$ to the null space of $\boldsymbol{H}$. Then $\forall \boldsymbol{x} \in \mathbb{R}^D$, there exists the unique decomposition:*

$$\boldsymbol{x} = \mathcal{P}_r(\boldsymbol{x}) + \mathcal{P}_n(\boldsymbol{x}), \tag{3}$$

*where we will call $\mathcal{P}_r(\boldsymbol{x})$ and $\mathcal{P}_n(\boldsymbol{x})$ the r-*component *and n-*component *of $\boldsymbol{x}$, respectively.*

***Remark* 1** *In this paper we will only focus on the above decomposition. However, we comment that in principle the pseudo-inverse, $\boldsymbol{H}^\dagger$, could be replaced by any general right inverse of $\boldsymbol{H}$ in the above decomposition which might provide added flexibility in certain inverse problems.*

An illustration of $\mathcal{R}$-$\mathcal{N}$ Decomposition is shown in Figure 1. Thus, the task of solving an inverse problem is to find these two components $\mathcal{P}_r(\mathbf{x})$ and $\mathcal{P}_n(\mathbf{x})$ based on the observed data, $\mathbf{y}_\epsilon$. The simple linear estimator to solve this problem is to use the approximation:

$$\mathbf{x}^* = \mathbf{H}^\dagger\mathbf{y}_\epsilon. \tag{4}$$

This estimator enjoys global and exact data-consistency, *i.e.* $\mathbf{H}\mathbf{x}^* \equiv \mathbf{y}_\epsilon$, which is an important consideration when solving inverse problems [25]. However, comparing (4) with (3) we can see that this is achieved by simply setting the nullspace component to zero: $\mathcal{P}_n(\mathbf{H}^\dagger\mathbf{y}_\epsilon) = 0$. In general this provides a poor solution for ill-posed problems. Thus it is necessary to further estimate the missing component $\mathcal{P}_n(\mathbf{x})$. Such an estimator is necessarily nonlinear.

Nullspace network. Recently, Schwab et al. [35] study the use of a neural network $\mathcal{G}$ to feed a refined backprojection $\mathcal{G}(\mathbf{H}^\dagger\mathbf{y}_\epsilon)$ to the null-space projection operator $\mathcal{P}_n$, then the reconstruction in (4) is reformulated as

$$\mathbf{x}^* = \mathbf{H}^\dagger\mathbf{y}_\epsilon + \mathcal{P}_n(\mathcal{G}(\mathbf{H}^\dagger\mathbf{y}_\epsilon)), \tag{5}$$

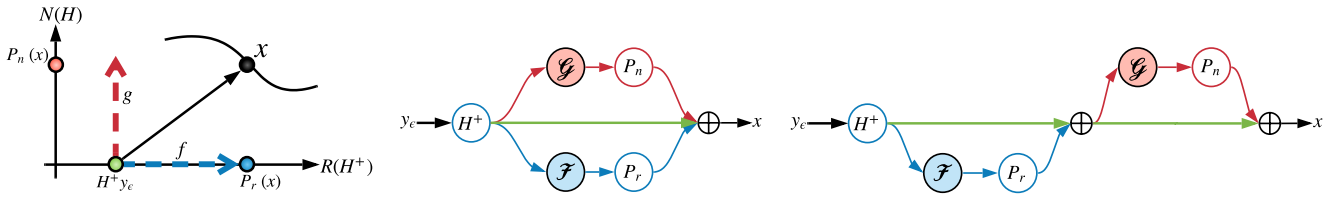
where the network $\mathcal{G}$ is suggested to be trained by minimizing the MSE between $\mathbf{x}$ and $\mathbf{x}^*$. Note the solution (5) enjoys global data consistency, *i.e.* $\mathbf{H}\mathbf{x}^* \equiv \mathbf{y}_\epsilon$. However, the solution (5) unfortunately, only works for the noise-free situation, and does not allow any denoising in the range $\mathcal{R}(\mathbf{H}^\dagger)$. Indeed, (5) can only denoise in the nullspace and the denoising ability is therefore worst-case bounded by $\|\epsilon\|/\|\mathbf{H}\|$ since $\|\mathbf{H}(\mathbf{x} - \mathbf{x}^*)\| = \|\epsilon\|$. The noise may further limit the ability to predict the null space component from the noisy measurements. Although it is reminiscent of decoupling the neural network denoiser from the inverse problem, it does not benefit from this since the training needs to be tailored to the task [32], which will be confirmed in our experiments.

3. Deep Decomposition Learning

Inspired by nullspace learning [35, 25] we aim to remove the range space denoising deficiency while still exploiting the nullspace property. For convenience, we rewrite the inverse problem (1) as $\mathbf{y}_\epsilon = \mathbf{H}\mathbf{x} - \epsilon$, let us consider the case $\epsilon \neq 0$ which is more piratical. By the $\mathcal{R}$-$\mathcal{N}$ decomposition, it holds that $\mathbf{x}$ can be exactly recovered by,

$$\begin{aligned}\mathbf{x} &= \mathcal{P}_r(\mathbf{x}) + \mathcal{P}_n(\mathbf{x}) \\ &= \mathbf{H}^\dagger\mathbf{y}_\epsilon + \mathbf{H}^\dagger\epsilon + \mathcal{P}_n(\mathbf{x}).\end{aligned} \tag{6}$$

However, as mentioned before, in the scenario of the inverse problem, both $\mathbf{x}$ and ϵ in (6) are still unknown and need to be recovered.



(a) $\mathcal{R}$-$\mathcal{N}$ decomposition of $\mathbf{x}$ (b) Independent (c) Cascade

Figure 2: The illustration of deep decomposition learning. In all sub-figures, $\mathcal{P}_n$ and $\mathcal{P}_r$ are the operators that project data to the nullspace of $\mathbf{H}$ and the range of $\mathbf{H}^\dagger$, respectively, $\mathcal{G}$ and $\mathcal{F}$ are two neural networks that need to be trained. In (a), $\mathbf{x}$ admits the range-nullspace ($\mathcal{R}$-$\mathcal{N}$) decomposition $\mathbf{x} = \mathbf{H}^\dagger\mathbf{y}_\epsilon + f + g$. The component $f = \mathcal{P}_r(\mathcal{F}(\mathbf{H}^\dagger\mathbf{y}_\epsilon))$ in both (b) and (c). The component $g = \mathcal{P}_n(\mathcal{G}(\mathbf{H}^\dagger\mathbf{y}_\epsilon))$ in (b) and $g = \mathcal{P}_n(\mathcal{G}(\mathbf{H}^\dagger\mathbf{y}_\epsilon + \mathcal{F}(\mathbf{H}^\dagger\mathbf{y}_\epsilon)))$ in (c).

We address this problem by using two neural networks and introducing the decomposition learning framework. Instead of hoping a single neural network will directly fit a desired underlying mapping between $\mathbf{H}^\dagger\mathbf{y}_\epsilon$ and $\mathbf{x} = \mathcal{P}_r(\mathbf{x}) + \mathcal{P}_n(\mathbf{x})$, we explicitly let two networks, denoted by $\mathcal{F}$ and $\mathcal{G}$ fit the two mappings from $\mathbf{H}^\dagger\mathbf{y}_\epsilon$ to the residual r-component $\mathcal{P}_r(\mathbf{x}) - \mathbf{H}^\dagger\mathbf{y}_\epsilon$ and the n-component $\mathcal{P}_n(\mathbf{x})$, respectively. In particular, the output of $\mathcal{F}$ should be bounded by the magnitude of noise ϵ, while $\mathcal{G}$ should be a smooth, *i.e.* a Lipschitz continuous neural network, since $\mathcal{G}$ is essentially a nullspace network, which does not need to be strongly bounded but should be regularized in order to get reasonable generalization. Therefore, the oracle $\mathbf{x}$ is decomposed as the sum of a linear component $\mathbf{H}^\dagger\mathbf{y}_\epsilon$ (the input), a bounded residual component $\mathcal{P}_r \circ \mathcal{F} \in \mathcal{R}(\mathbf{H}^\dagger)$ and a smooth n-component $\mathcal{P}_n \circ \mathcal{G} \in \mathcal{N}(\mathbf{H})$.

We consider two versions of DDN estimators. First, we define an independent connection architecture DDN estimator $\mathcal{A}_i$ using the $\mathcal{R}$-$\mathcal{N}$ decomposition,

$$\mathcal{A}_i(\mathbf{y}_\epsilon) \triangleq \mathbf{H}^\dagger\mathbf{y}_\epsilon + \mathcal{P}_r(\mathcal{F}(\mathbf{H}^\dagger\mathbf{y}_\epsilon)) + \mathcal{P}_n(\mathcal{G}(\mathbf{H}^\dagger\mathbf{y}_\epsilon)). \quad (7)$$

where there are no interactions between $\mathcal{F}$ and $\mathcal{G}$.

An alternative (but essentially equivalent) mapping $\mathcal{A}_c$ from $\mathbf{y}_\epsilon$ to $\mathbf{x}$ which is related to (7) uses a cascade of networks, *i.e.* first denoising with $\mathcal{F}$, then feeding the denoised $\mathbf{H}^\dagger\mathbf{y}_\epsilon + \mathcal{P}_r(\mathcal{F}(\mathbf{H}^\dagger\mathbf{y}_\epsilon))$ into $\mathcal{G}$ such that:

$$\begin{aligned}\mathcal{A}_c(\mathbf{y}_\epsilon) \triangleq\ & \mathbf{H}^\dagger\mathbf{y}_\epsilon + \mathcal{P}_r\left(\mathcal{F}\left(\mathbf{H}^\dagger\mathbf{y}_\epsilon\right)\right) \\ & + \mathcal{P}_n\left(\mathcal{G}\left(\mathbf{H}^\dagger\mathbf{y}_\epsilon + \mathcal{P}_r\left(\mathcal{F}(\mathbf{H}^\dagger\mathbf{y}_\epsilon)\right)\right)\right).\end{aligned} \quad (8)$$

Intuitively (8) is preferable since it is no more complex than the independent network topology and provides the nullspace network $\mathcal{G}$ with a range denoised input, thereby reducing the learning burden needed to be done by $\mathcal{G}$. Our experiments also verified the cascade connections typically perform better than the independent one.

Figure 2 gives an illustration of the above deep decomposition learning, as well as the corresponding independent architecture and the cascade architecture for $\mathcal{F}$ and $\mathcal{G}$ in (7) and (8), respectively. Note the DDN $\mathcal{A}(\mathbf{y}_\epsilon)$ defined in (7) and (8) offers the ability to denoise both the r-component $\mathcal{P}_r(\mathbf{x})$ and the n-component $\mathcal{P}_n(\mathbf{x})$, and if $\|\mathbf{H}\mathcal{F}\| \leq \|\epsilon\|$, the solution enjoys a relaxed notion of data-consistency (in the spirit of the discrepancy principle popular in inverse problems), which convergences to exact data consistency when $\|\mathbf{H}\mathcal{F} - \epsilon\| \to 0$.

Note that the independent model is by its nature a shallower but wider network than the cascade model which by construction is deeper, but given the decomposition both networks have essentially the same complexity.

3.1. Training strategy

Let $\mathbf{X} = \{(\mathbf{y}_\epsilon^{(i)}, \mathbf{x}^{(i)})\}_{i=1}^N$ denote a training set of N samples, where $\mathbf{x}^{(i)}$ and $\mathbf{y}_\epsilon^{(i)}$ are the clean oracle signal and its noisy measurement given by (1). Denote by $\ell(x, y)$ the loss function, which measures the discrepancy between x and y. Given an estimator $\mathcal{A}$, its empirical loss associated with the training set $\mathbf{X}$ is defined as:

$$\ell_{\text{emp}}(\mathcal{A}) \triangleq \frac{1}{N} \sum_{\mathbf{y}_\epsilon^{(i)}, \mathbf{x}^{(i)} \in \mathbf{X}} \ell(\mathcal{A}(\mathbf{y}_\epsilon^{(i)}), \mathbf{x}^{(i)}). \quad (9)$$

In this paper we will consider the case where ℓ is the least-squares loss.

According to the two connection types for $\mathcal{F}$ and $\mathcal{G}$, we consider the following two training strategies.

Joint training. We first consider jointly training $\mathcal{F}$ and $\mathcal{G}$ by solving a single optimization program,

$$\min_{\mathcal{F},\mathcal{G}} \ell_{\text{emp}}(\mathcal{A}) + \lambda_1\phi_1(\mathcal{F}) + \lambda_2\phi_2(\mathcal{G}), \quad (10)$$

where the first term serves the data-fidelity, and $\mathcal{A}(\mathbf{y}_\epsilon)$ take the form either in (7) or in (8). ϕ_1 and ϕ_2 are two regularization terms that are used in training $\mathcal{F}$ and $\mathcal{G}$ to impose the desired boundedness and smoothness conditions. Accordingly, we define different regularizations for $\mathcal{F}$ and $\mathcal{G}$ to tune the networks to their specific tasks.

For $\mathcal{F}$, we set ϕ_1 as

$$\phi_1(\mathcal{F}) = \sum_{i=1}^N \ell(\mathbf{H}\mathcal{F}(\mathbf{H}^\dagger\mathbf{y}_\epsilon^{(i)}), \epsilon^{(i)}), \quad (11)$$

in order to encourage the data discrepancy term to be small.

For $\mathcal{G}$ let $\{\mathbf{w}^{(j)}\}_{j=1}^{L}$ denote networks connection weights for layers $1 \ldots, L$. We then set ϕ_2 as the weight decay term to control the Lipschitz of the network [35] and to encourage good generalization [16], *i.e.*

$$\phi_2(\mathcal{G}) = \sum_j \|\mathbf{w}^{(j)}\|_2^2. \tag{12}$$

Decoupled training. Due to the decomposition $\mathbf{x} = \mathcal{P}_r(\mathbf{x}) + \mathcal{P}_n(\mathbf{x})$ in the independent architecture there are no interactions between the respective targets of $\mathcal{F}$ and $\mathcal{G}$, therefore one can decouple (10) into two independent sub-optimizations to train $\mathcal{F}$ and $\mathcal{G}$ separately. For example, $\mathcal{F}$ can be trained by,

$$\arg\min_{\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{H}^\dagger \mathbf{y}_\epsilon^{(i)} + \mathcal{P}_r(\mathcal{F}(\mathbf{H}^\dagger \mathbf{y}_\epsilon^{(i)})), \mathcal{P}_r(\mathbf{x}^{(i)})) + \lambda_1 \phi_1(\mathcal{F}), \tag{13}$$

and $\mathcal{G}$ can be trained by,

$$\arg\min_{\mathcal{G}} \frac{1}{N}\sum_{i=1}^{N} \ell(\mathcal{P}_n(\mathcal{G}(\mathbf{H}^\dagger \mathbf{y}_\epsilon^{(i)})), \mathcal{P}_n(\mathbf{x}^{(i)})) + \lambda_2 \phi_2(\mathcal{G}). \tag{14}$$

In the cascade architecture we have the option of decoupling the optimization in a cascaded manner. First training $\mathcal{F}$ as in (13), and then, with $\mathcal{F}$ fixed, training $\mathcal{G}$ using,

$$\arg\min_{\mathcal{G}} \frac{1}{N}\sum_{i=1}^{N} \ell(\mathcal{P}_n(\mathcal{G}(\mathbf{H}^\dagger \mathbf{y}_\epsilon^{(i)} + \mathcal{P}_r(\mathcal{F}(\mathbf{H}^\dagger \mathbf{y}_\epsilon^{(i)}))), \mathcal{P}_n(\mathbf{x}^{(i)})) + \lambda_2 \phi_2(\mathcal{G}). \tag{15}$$

While joint training and decoupled training are theoretically equivalent, in practice, the decoupled training enjoys more intuitive interpretability, and it is easier to control $\mathcal{F}$ and $\mathcal{G}$ each to achieve better convergence results. However, the joint training is slightly more efficient than the decoupled because the networks can be trained simultaneously.

3.2. The relationship to other work

In the noise-free case ($\epsilon = 0$), the decomposition learning (6) reduces naturally to vanilla nullspace learning (5). Thus a DDN can be regarded as a generalized nullspace network. While in the noisy case, one might be tempted to consider adopting a separate generic denoiser to preprocess the measurements $\mathbf{y}_\epsilon$. However, such denoisers are typically built in a manner decoupled from the inverse problem of interest, therefore to train a nullspace network with such denoised measurements could amplify the reconstruction error, causing more inconsistent results. This will be demonstrated in the experiments later. In contrast, the denoising process in the DDN is not decoupled from the inverse problem but integrated into a unified learning model. The experiments show this not only improves the quality of the results but in addition, helps the model's generalization.

Our decomposition learning can also be regarded as a special gated neural network. To be specific, if we rewrite (7) as

$$\mathcal{A} = \mathbf{T}\mathcal{F}(\mathbf{z}) + (\mathbf{I} - \mathbf{T})\mathcal{G}(\mathbf{z}) + \mathbf{z}, \tag{16}$$

where $\mathbf{z} = \mathbf{H}^\dagger \mathbf{y}_\epsilon$, $\mathbf{T} = \mathbf{H}^\dagger \mathbf{H}$, it can be seen the output of $\mathcal{F}$ and $\mathcal{G}$ are gated in terms of $\mathbf{T}$ and $\mathbf{I} - \mathbf{T}$. The importance of the two components is determined by the physics in terms of $\mathbf{H}^\dagger \mathbf{H}$. This is different from previous gated networks such as [14, 6, 38] in which the model is gated by some bounded numerical function such as a sigmoid or hardlim which are not typically related to the physics of the forward model or its inverse. Our method can also be regarded as a generalized residual learning [13], *i.e.* we decompose the residual $\mathcal{A} - \mathbf{z}$ into two components in $\mathcal{R}(\mathbf{H}^\dagger)$ and $\mathcal{N}(\mathbf{H})$ with more explicit interpretability. In particular, in the absence of the nullspace $\mathcal{N}(\mathbf{H})$ or in the case $\mathbf{H} = \mathbf{I}$ such that $\mathcal{P}_r = \mathbf{I}$ and $\mathcal{P}_n = 0$, *i.e.* there is no nullspace learning, $\mathcal{G}$ will be irrelevant and only $\mathcal{F}$ will be learnable, and the decomposition learning will be reduced to a non-gated neural network and equivalent to the standard residual learning.

3.3. The implementation

Architecture. Our goal here is not to explore the potential designs of the neural networks, $\mathcal{F}$ and $\mathcal{G}$, but the usage of physics in the neural network. Therefore we directly apply the corresponding state-of-the-art neural network architectures to build $\mathcal{F}$ and $\mathcal{G}$, respectively. As an example, we use the denoising CNN (DnCNN) [44], a fully convolutional network, as the architecture of $\mathcal{F}$, and we use the U-net (encoder-decoder architecture) [15, 34] to build $\mathcal{G}$. Our network can process both gray and color images, so the number of channels for the input layer and output layer can be 1 or 3 appropriately.

Operator. A key aspect of the DDN framework is access to the projection operators $\mathcal{P}_n$ and $\mathcal{P}_r$. If the inverse problem is relatively small, then $\mathbf{H}^\dagger$ can be calculated directly or approximated using truncated SVD. However, in larger scale problems this will not be feasible and alternative approximations must be sought. For some problems, such as the ones considered in this paper - super-resolution and compressed sensing - the pseudo-inverse is readily available or easy to approximate. When this is not the case one may have to resort to an iterative approximation of the projectors, e.g. using a preconditioned conjugate gradient solver. In such a scenario the DDN will require more computation (both in training and implementation) and it is not clear how this cost will compare to that of iterative DNN solutions such as those discussed in section 2.

4. Experiments

We designed our experiments to address the following questions: Which training strategy, *e.g.* jointly or decoupled, is best for training the DDN? Which connection type, *e.g.* the independent or cascade one, is better? Does the proposed deep decomposition learning help the neural network produce superior results on different inverse problem tasks? Does the DDN enjoy better generalization?

We conduct super-resolution (SR) and compressed sensing (CS) tasks as examples. We describe the details of the experiments and results in the following.

4.1. Experimental setup

Datasets. We train the models on two public natural image datasets: the CelebA [21] and BSDS300 [1]. The former is used in CS and the latter used for the SR task.

The CelebA dataset contains more than 200K celebrity images, each with 40 binary attributes. We pick the attribute *"smile"* to evaluate the proposed method. The center part of the aligned images in the CelebA dataset are cropped and scaled to 48×48. We divide the selected images equally into three subsets for training, validation and testing. There are $32,557$ images in each subset and the training batch size is 100. The forward operator $\mathbf{H}$ is thus a random sensing matrix with a predefined compressive ratio and the operator $\mathbf{H}^\dagger$ is calculated directly by $\mathbf{H}^\dagger = \mathbf{H}^\top(\mathbf{H}\mathbf{H}^\top)^{-1}$.

The BSDS300 includes 300 high-resolution (HR) natural images. We use these images for training and evaluate the models on the benchmark datasets BSDS100 [1]. We super-resolve low-resolution (LR) images by a scale-factor of 2 and 4. The LR images are formed by downsampling with bi-cubic anti-aliasing the HR images [10], and then corrupting them with Gaussian noise with a standard deviation, σ_ϵ varying between 0 (no noise) and 25 (10% noise). The presence of anti-aliasing in the downsampling means that the bi-cubic upsampling operator is a reasonable approximation to $\mathbf{H}^\dagger$, and this is what we use in our DDN. Training is performed using 40×40 RGB image patches (LR) for both scale-factors 2 and 4, with a batch size set to 15. The quantitative results presented are then evaluated on the luminance (Y) channel.

Metric. We quantify the performance of the image reconstruction using Peak Signal to Noise Ratio (PSNR), and generalization error (GE). In particular, PSNR $= 10\log_{10}(255/\text{MSE})^2$ is used to measure the accuracy of the image reconstruction. The GE, the difference between the expected loss and the training loss, is evaluated to examine whether inclusion of the physics in the end-to-end training reduces the burden on training data [16]. Here approximating the expected loss with the testing loss: $\ell_{\text{exp}}(\mathcal{A}) \triangleq \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim P}[\ell(\mathcal{A}(\mathbf{y}),\mathbf{x})] \approx \ell(\mathcal{A}(\mathbf{y}),\mathcal{X}_{\text{test}})$ we measure GE by:

$$\text{GE}(\mathcal{A}) = |\ell(\mathcal{A},\mathcal{X}_{\text{test}}) - \ell(\mathcal{A},\mathcal{X}_{\text{train}})|, \qquad (17)$$

where $\ell(\mathcal{A},\mathcal{X})$ denotes the loss evaluated over the data $\mathcal{X}$.

Comparators. We provide evaluation of our decomposition learning for inverse problems against various other network configurations: residual learning, Unet with residual connection (ResUnet) [15]; nullspace learning, nullspace network (NSN) [35]; and pre-denoising+nullspace learning, denoising with a pre-trained DnCNN [44] followed by a NSN (DnNSN). For fairness in all the experiments, the backbone network architectures were the same for ResUnet, NSN and the $\mathcal{G}$ network in DDN. Similarly the architectures for the denoiser used in DnNSN was the same as that of $\mathcal{F}$ used in DDN. All networks used $\mathbf{H}^\dagger\mathbf{y}_\epsilon$ as the input, were implemented in Pytorch, and trained on NVIDIA 2080Ti GPUs.

4.2. Results and Analysis

We used ADAM to optimize the DDN with an initial learning rate of 10^{-3}, and we tuned the λ_1 and λ_2 for specific inverse problems, *i.e.* $\lambda_1 = 10^{-8}$, $\lambda_2 = 10^{-8}$ for the SR task and $\lambda_1 = 10^{-6}$ and $\lambda_2 = 10^{-7}$ in the CS problem.

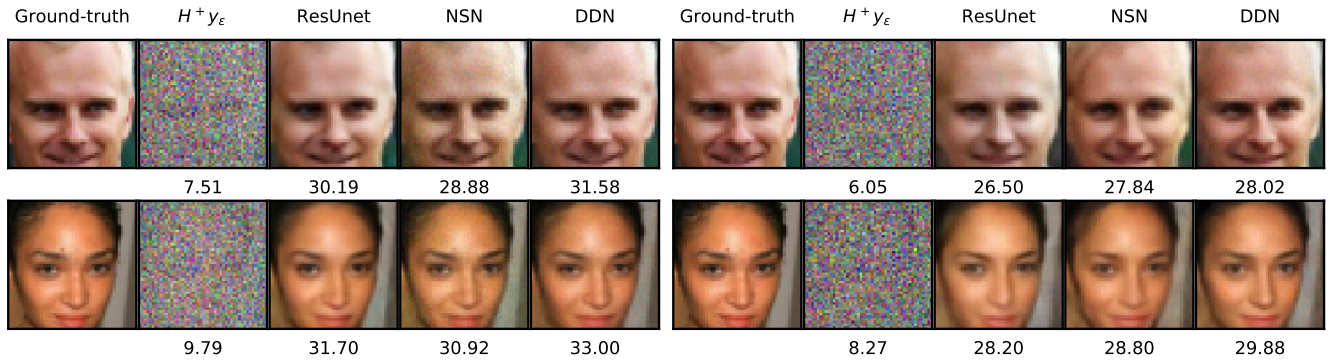
Study of training strategy: We first investigate the performance of DDN with the different training strategies (jointly/decoupled training) and different connection types (independent/cascade). Note that for the independent model the joint and decoupled training are exactly equivalent. We perform SR with a scale factor of 2 and noise level $\sigma_\epsilon = 25$ (10%) on the BSDS300 dataset. The results are reported in the Table 1 which demonstrates the cascade architecture always performs slightly better than the independent one in term of both PSNR and GE. We also find that $\mathcal{F}$ and $\mathcal{G}$ benefit from more iterations in the decoupled training, especially for the noisy case. In contrast, joint training was observed to be more efficient. In all subsequent experiments, we therefore use the cascade architecture with joint training for the DDNs.

Metric	Independent	Cascade	
	Joint/Decoupled	Joint	Decoupled
PSNR (dB)	25.85	26.71	**26.74**
GE (10^{-4})	**1.10**	1.27	1.25

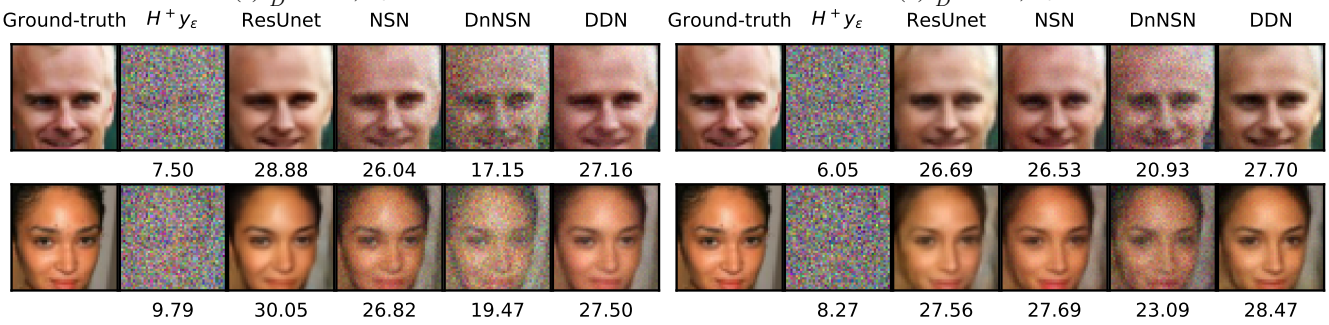
Table 1: Comparison of DDN training strategies v.s. connection types of $\mathcal{F}$ and $\mathcal{G}$ on SR ($2\times$, $\sigma_\epsilon = 25$) task.

Comparison results. The results on the CelebA dataset for CS and the BSDS100 dataset for SR are shown in Figures 3[1] and 4, respectively. We also list the statistics of PSNRs and GEs of the reconstruction outputs in Table 2 and 3 for the different noise levels $\sigma_\epsilon = 0$ and 25, respectively. From the results, we have the following conclusions.

[1]Note the compressed measurements $\{\mathbf{y}_\epsilon\}$ are not shown in Figure 3 since they are random compressed vectors without visual semantics.



(a) $\frac{d}{D} = 0.3, \sigma_\epsilon = 0$ (b) $\frac{d}{D} = 0.1, \sigma_\epsilon = 0$



(c) $\frac{d}{D} = 0.3, \sigma_\epsilon = 25$ (d) $\frac{d}{D} = 0.1, \sigma_\epsilon = 25$

Figure 3: Visual demonstration and PSNRs of CS ($d/D = 0.3, 0.1$) results on the CelebA dataset with noise scale $\sigma_\epsilon = 0, 25$. From left to right in each subfigure: oracle celebrity smile face $\mathbf{x}$, input $\mathbf{H}^\dagger\mathbf{y}_\epsilon$, results of ResUnet [15], NSN [35], DnNSN: DnCNN [44] + NSN (only for (c) and (d)), and results of the proposed method.

Task	ResUnet	NSN	DDN
CS (0.3)	29.75/0.84	29.10/**0.82**	**31.24/0.82**
CS (0.1)	26.18/1.86	27.31/1.86	**27.89/1.69**
SR (2×)	27.56/1.02	28.23/0.82	**31.55/0.80**
SR (4×)	26.20/1.34	26.38/1.32	**26.70/1.23**

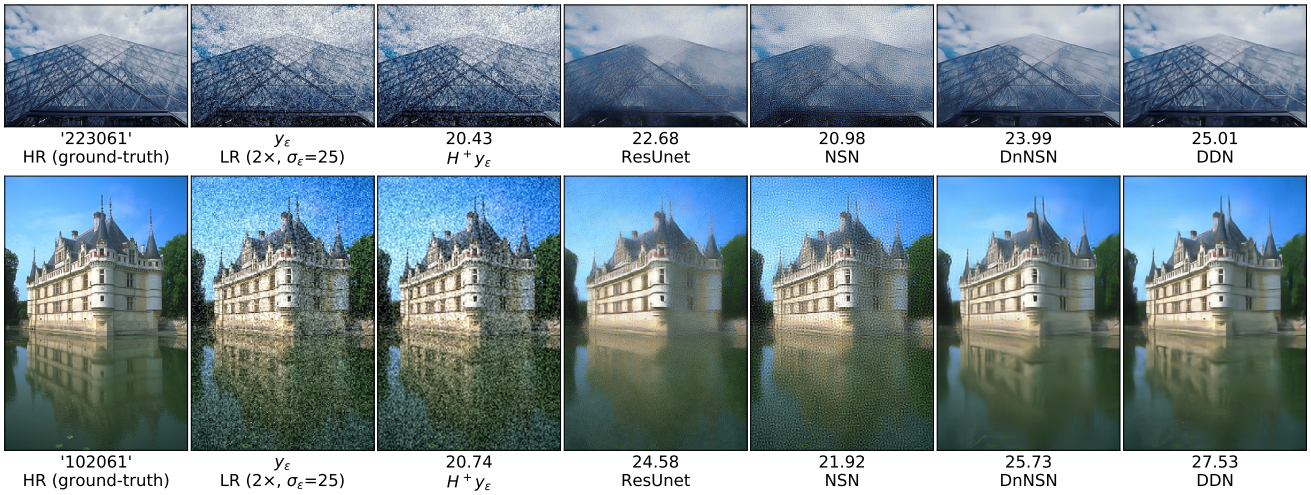
Table 2: Comparison results of PSNR/GE ($\times 10^{-4}$) on different inverse problems with noise level $\sigma_\epsilon = 0$.

Task	ResUnet	NSN	DnNSN	DDN
CS (0.3)	**28.44**/0.55	25.84/0.55	18.08/1.45	26.83/**0.54**
CS (0.1)	26.17/1.35	26.27/1.59	21.13/2.06	**27.00/1.30**
SR (2×)	24.31/1.75	21.85/1.41	25.92/1.53	**26.71/1.27**
SR (4×)	23.94/1.82	23.80/1.75	24.05/1.81	**24.19/1.60**

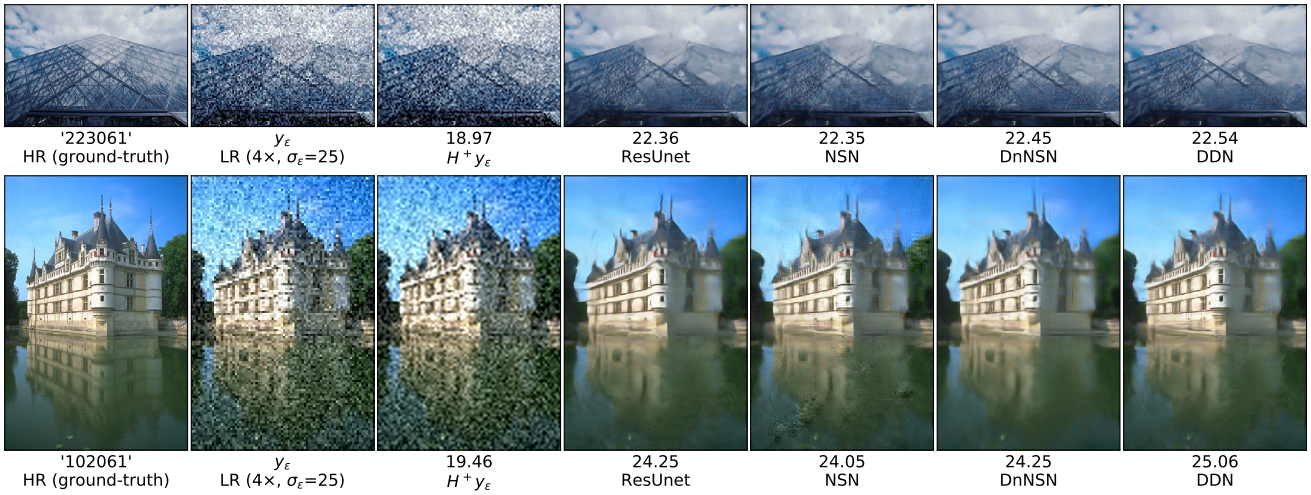
Table 3: Comparison results of PSNR/GE ($\times 10^{-4}$) on different inverse problems with noise level $\sigma_\epsilon = 25$.

First, it can be seen that the proposed DDN outperforms all the counterparts on the SR task both in the noisy and noise-free cases. While the physics-free approach, ResUnet, achieves the best results on the CS with compression ratio $\frac{d}{D} = 0.3$, it fails to handle larger compression ratios, *e.g.* $\frac{d}{D} = 0.1$ where we believe NSN and DDN benefit more from the presence of explicit nullspace structure. NSN enjoys good performance only in the noise-free case, but it performed poorly in the presence of large noise ($\sigma_\epsilon = 25$), as in this scenario the denoising task plays a significant role in the inverse problem. Since the ResUnet is purely learning-based, it consistently provides stable restoration regardless of whether the measurements are clean or noisy, but in both cases it has higher GEs than DDN and NSN. This suggests that introducing physics into deep learning models can facilitate the neural network to enjoy better generalization for solving inverse problems. In addition the decomposition learning appears to enjoy similar GE to the NSN in the zero noise case and slightly better GE when noise is present. Thanks to the introduction of $\mathcal{F}$ and $\mathcal{G}$, regardless of whether the null space is large or small, or whether the denoising significant or not, the DDN offers good restoration.

Second, preprocessing the NSN with a denoiser, as in the DnNSN performed worse than the NSN in the noisy CS problem (Figure 3), although it performs better than NSN in the noisy SR task (Figure 4). Furthermore, in both tests the DnNSN performed substantialy worse than the proposed DNN. This demonstrates that in order to gain the full benefits of the denoiser it cannot be completely decoupled from the inverse problem. Similar observations can be found in



(a) $2\times$, $\sigma_\epsilon = 25$



(b) $4\times$, $\sigma_\epsilon = 25$

Figure 4: Visual demonstration and PSNRs of SR ($2\times$ and $4\times$) results on the BSDS100 dataset with noise scale $\sigma_\epsilon = 25$. From left to right: oracle image $\mathbf{x}$ with its index number in the dataset, LR noisy measurement $\mathbf{y}_\epsilon$, bi-cubic upsample, results of ResUnet [15], NSN [35], DnNSN: DnCNN [44] + NSN, and results of the proposed method.

[32]. In contrast, in the DDN, the denoising process is integrated into the proposed decomposition learning, which allows the DDN to remove structural noise and to simultaneously accurately approximate the r-component $\mathcal{P}_r$ and predict the n-component $\mathcal{P}_n$ from its noisy observations. From the above discussions, we conclude that decomposition learning is well-principled, structurally simple, and highly interpretable.

Finally, we emphasize that given a linear forward operator $\mathbf{H}$, the decomposition learning naturally exists and is easily to define. One can plug the decomposition learning, (7) or (8), into other existing specialized solvers for different problems, with which we believe one could increase the performance limit of the deep learning solvers. However, we leave this as our future work.

5. Conclusion

In this paper, we have proposed a deep decomposition learning method for building an end-to-end neural network solution for inverse problems. We have explicitly reformulated the DNN layers to learn range-nullspace decomposition functions with reference to the layer inputs, instead of learning unreferenced functions. We have shown that the decomposition networks not only produce superior results, but also enjoy good interpretability and generalization. We have demonstrated the advantages of decomposition learning on compressive sensing and image super-resolution examples. In future work, we will explore adapting the proposed deep decomposition learning to more challenging inverse problems such as tomographic imaging.

6. Acknowledgements

This work is funded by the ERC Advanced grant 694888, C-SENSE.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017.

[2] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 537–546. JMLR. org, 2017.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[4] Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE conference on computer vision and pattern recognition*, pages 2392–2399. IEEE, 2012.

[5] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1256–1272, 2016.

[6] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[7] Zhen Cui, Hong Chang, Shiguang Shan, Bineng Zhong, and Xilin Chen. Deep network cascade for image super-resolution. In *European Conference on Computer Vision*, pages 49–64. Springer, 2014.

[8] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[9] Gilton Davis, Ongie Greg, and Willett Rebecca. Neumann networks for linear inverse problems in imaging. *IEEE Transactions on Computational Imaging*, 2019.

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[11] Shuhang Gu, Radu Timofte, and Luc Van Gool. Integrating local and non-local denoiser priors for image restoration. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2923–2928. IEEE, 2018.

[12] Harshit Gupta, Kyong Hwan Jin, Ha Q Nguyen, Michael T McCann, and Michael Unser. Cnn-based projected gradient descent for consistent ct image reconstruction. *IEEE transactions on medical imaging*, 37(6):1440–1453, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[16] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[17] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458, 2016.

[18] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8183–8192, 2018.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[20] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[22] Alice Lucas, Michael Iliadis, Rafael Molina, and Aggelos K Katsaggelos. Using deep neural networks for inverse problems in imaging: beyond analytical methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018.

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

[24] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in neural information processing systems*, pages 2802–2810, 2016.

[25] Morteza Mardani, Enhao Gong, Joseph Y Cheng, Shreyas Vasanawala, Greg Zaharchuk, Lei Xing, and John M Pauly. Deep generative adversarial networks for compressed sensing automates mri. *IEEE Transactions on Medical Imaging*, 38(1):167–179, 2019.

[26] Morteza Mardani, Qingyun Sun, David Donoho, Vardan Papyan, Hatef Monajemi, Shreyas Vasanawala, and John Pauly. Neural proximal gradient descent for compressive imag-

ing. In *Advances in Neural Information Processing Systems*, pages 9573–9583, 2018.

[27] Michael T McCann, Kyong Hwan Jin, and Michael Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, 34(6):85–95, 2017.

[28] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1336–1343. IEEE, 2015.

[29] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[30] Tran Minh Quan, Thanh Nguyen-Duc, and Won-Ki Jeong. Compressed sensing mri reconstruction using a generative adversarial network with a cyclic loss. *IEEE transactions on medical imaging*, 37(6):1488–1497, 2018.

[31] Saiprasad Ravishankar, Jong Chul Ye, and Jeffrey A Fessler. Image reconstruction: From sparsity to data-adaptive methods and machine learning. *arXiv preprint arXiv:1904.02816*, 2019.

[32] JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5888–5897, 2017.

[33] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

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

[35] Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. Deep null space learning for inverse problems: Convergence analysis and rates. *Inverse Problems*, 2019.

[36] Viraj Shah and Chinmay Hegde. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4609–4613. IEEE, 2018.

[37] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580, 2019.

[38] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.

[39] Jian Sun, Huibin Li, Zongben Xu, et al. Deep admm-net for compressive sensing mri. In *Advances in neural information processing systems*, pages 10–18, 2016.

[40] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[41] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[42] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Advances in neural information processing systems*, pages 1790–1798, 2014.

[43] Kun Zeng, Jun Yu, Ruxin Wang, Cuihua Li, and Dacheng Tao. Coupled deep autoencoder for single image super-resolution. *IEEE transactions on cybernetics*, 47(1):27–37, 2015.

[44] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[45] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Deep plug-and-play super-resolution for arbitrary blur kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1671–1681, 2019.