One Network to Solve Them All — Solving Linear Inverse Problems using Deep Projection Models

J. H. Rick Chang, Chun-Liang Li, Barnabás Póczos, B. V. K. Vijaya Kumar, and Aswin C. Sankaranarayanan
Carnegie Mellon University, Pittsburgh, PA

Abstract—While deep learning methods have achieved state-of-the-art performance in many challenging inverse problems like image inpainting and super-resolution, they invariably involve problem-specific training of the networks. Under this approach, different problems require different networks. In scenarios where we need to solve a wide variety of problems, *e.g.*, on a mobile camera, it is inefficient and costly to use these specially-trained networks. On the other hand, traditional methods using signal priors can be used in all linear inverse problems but often have worse performance on challenging tasks. In this work, we provide a middle ground between the two kinds of methods — we propose a general framework to train a single deep neural network that solves arbitrary linear inverse problems. The proposed network acts as a proximal operator for an optimization algorithm and projects non-image signals onto the set of natural images defined by the decision boundary of a classifier. In our experiments, the proposed framework demonstrates superior performance over traditional methods using a wavelet sparsity prior and achieves comparable performance of specially-trained networks on tasks including compressive sensing and pixel-wise inpainting.

Index Terms—Linear inverse problems, Image processing, Adversarial learning, Proximal operator

1 Introduction

At the heart of many image processing tasks is a linear inverse problem, where the goal is to reconstruct an image $\mathbf{x} \in \mathbb{R}^d$ from a set of measurements $\mathbf{y} \in \mathbb{R}^m$ of the form $\mathbf{y} = A\mathbf{x} + \mathbf{n}$, where $A \in \mathbb{R}^{m \times d}$ is the measurement operator and $\mathbf{n} \in \mathbb{R}^m$ is the noise. For example, in image inpainting, $\mathbf{y}$ is an image with masked regions and A is the linear operation applying a pixelwise mask to the original image $\mathbf{x}$; in super-resolution, $\mathbf{y}$ is a low-resolution image and the operation A downsamples high resolution images; in compressive sensing, $\mathbf{y}$ denotes compressive measurements and A is the measurement matrix, *e.g.*, a random Gaussian matrix. Linear inverse problems, like those described above, are often underdetermined, *i.e.*, they involve fewer measurements than unknowns. Such underdetermined systems are extremely difficult to solve since the operator A has a non-trivial null space and there are an infinite number of feasilbe solutions but only a few of them are natural images.

Solving linear inverse problems. There are two broad classes of methods for solving linear underdetermined problems. At one end, we have techniques that use signal priors to regularize the inverse problems. Signal priors enable identification of the true solution from the infinite set of feasible solutions by enforcing image-specific features to the solution. Thereby, designing a signal prior plays a key role when solving linear inverse problems. Traditionally, signal priors are hand-designed based on empirical observations of images. For example, since natural images are usually sparse after wavelet transformation and are generally piecewise smooth, signal priors that constrain the sparsity of wavelet coefficients or spatial gradients are widely used [1]–[5]. Even though these signal priors can be used in any linear inverse problems related to images and usually have efficient solvers, these signal priors are often too generic, in that many non-image signals can also satisfy the constraints. Thereby, these hand-designed signal priors cannot easily deal with challenging problems like image inpainting or super-resolution.

Instead of using a universal signal prior, a second class of method learn a mapping from the linear measurement domain of $\mathbf{y}$ to the image space of $\mathbf{x}$, with the help of large datasets and deep neural nets [6]–[8]. For example, to solve image super-resolution, low-resolution images are generated from a high-resolution image dataset, and the mapping between the corresponding image pairs are learned with a neural net [6]. Similarly, a network can be trained to solve compressive sensing problem [9]–[11] or image debluring [12], *etc*. These methods have achieved state-of-the-art performance in many challenging problems.

Despite their superior performance, these specially-trained solvers are designed for specific problems and usually cannot solve other problems without retraining the mapping function — even when the problems are similar. For example, a $2\times$-super-resolution network cannot be easily readapted to solve $4\times$ or $8\times$ super-resolution problems; a compressive sensing network for Gaussian random measurements is not applicable to sub-sampled Hadamard measurements. Training a new network for every single instance of an inverse problem is a wasteful proposition. In comparison, traditional methods using hand-designed signal priors can solve any linear inverse problems but have poorer performance on an individual problem. Clearly, a middle ground between these two classes of methods is needed.

One network to solve them all. In this paper, we ask the following question: *if we have a large image dataset, can we learn from the dataset a signal prior that can deal with* ***any*** *linear inverse problems of images?* Such a signal prior can significantly lower the cost to incorporate inverse algorithms into consumer products, for example, via the form of specialized hardware design.

Chang, Bhagavatula and Sankaranarayanan were supported, in part, by the ARO Grant W911NF-15-1-0126. Chang was also partially supported by the CIT Bertucci Fellowship.

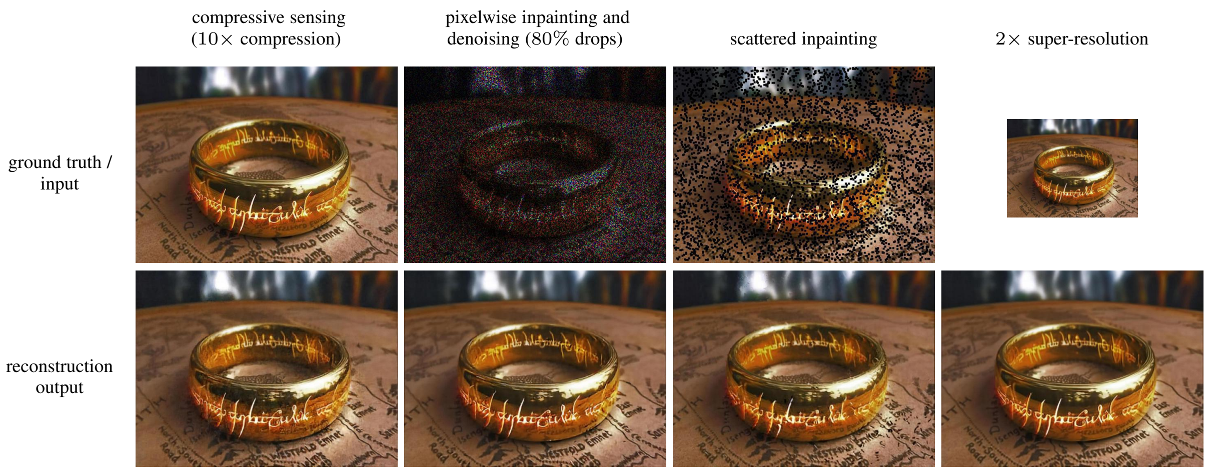

compressive sensing (10× compression)
pixelwise inpainting and denoising (80% drops)
scattered inpainting
2× super-resolution
ground truth / input
reconstruction output

Fig. 1: The same proposed network is used to solve the following tasks: compressive sensing problem with $10\times$ compression, pixelwise random inpainting with 80% dropping rate, scattered inpainting, and $2\times$-super-resolution. Note that even though the nature and input dimensions of the problems are very different, the proposed framework is able to use a single network to solve them all without retraining.

To answer this question, we observe that in optimization algorithms for solving linear inverse problems, signal priors usually appears in the form of proximal operators. Geometrically, the proximal operator *projects* the current estimate closer to the feasible sets (natural images) constrained by the signal prior. Thus, we propose to learn the proximal operator with a *deep projection model*, which can be integrated into many standard optimization frameworks for solving arbitrary linear inverse problems.

Contributions. We make the following contributions.

- We propose a general framework that implicitly learns a signal prior and a projection operator from large image datasets. When integrated into an alternating direction method of multipliers (ADMM) algorithm, the same proposed projection operator can solve challenging linear inverse problems related to images.
- We identify sufficient conditions for the convergence of the nonconvex ADMM with the proposed projection operator, and we use these conditions as guidelines to design the proposed projection network.
- We show that it is inefficient to solve generic linear inverse problems with state-of-the-art methods using specially-trained networks. Our experiment results also show that they are prone to be affected by changes in the linear operators and noise in the linear measurements. In contrast, the proposed method is more robust to these factors.

2 Related Work

Given noisy linear measurements $\mathbf{y}$ and the corresponding linear operator A, which is usually underdetermined, the goal of linear inverse problems is to find a solution $\mathbf{x}$, such that $\mathbf{y} \approx A\mathbf{x}$ and $\mathbf{x}$ be a signal of interest, in our case, an image. Based on their strategies to deal with the underdetermined nature of the problem, algorithms for linear inverse problems can be roughly categorized into those using hand-designed signal priors and those learning from datasets. We briefly review some of these methods.

Hand-designed signal priors. Linear inverse problems are usually regularized by signal priors in a penalty form:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda\phi(\mathbf{x}), \tag{1}$$

where $\phi : \mathbb{R}^d \to \mathbb{R}$ is the signal prior and λ is the non-negative weighting term. Signal priors constraining the sparsity of $\mathbf{x}$ in some transformation domain have been widely used in literatures. For example, since images are usually sparse after wavelet transformation or after taking gradient operations, a signal prior ϕ can be formulated as $\phi(\mathbf{x}) = \|W\mathbf{x}\|_1$, where W is a operator representing either wavelet transformation, taking image gradient, or other hand-designed linear operation that produces sparse features from images [13]. Using signal priors of ℓ_1-norms enjoys two advantages. First, it forms a convex optimization problem and provides global optimality. The optimization problem can be solved efficiently with a variety of algorithms for convex optimization. Second, ℓ_1 priors enjoy many theoretical guarantees, thanks to results in compressive sensing [14]. For example, if the linear operator A satisfies conditions like the restricted isometry property and $W\mathbf{x}$ is sufficiently sparse, the optimization problem (1) provides the sparsest solution.

Despite their algorithmic and theoretical benefits, hand-designed priors are often too generic to constrain the solution set of the inverse problem (1) to be images — we can easily generate noise signals that have sparse wavelet coefficients or gradients.

Learning-based methods. The ever-growing number of images on the Internet enables state-of-the-art algorithms to deal with challenging problems that traditional methods are incapable of solving. For example, image inpainting and restoration can be performed by pasting image patches or transforming statistics of pixel values of similar images in a large dataset [15], [16]. Image denoising and super-resolution can be performed with dictionary learning methods that reconstruct image patches with sparse linear combinations of dictionary entries learned from datasets [17], [18]. Large datasets can also help learn end-to-end mappings from the linear measurement domain to the image domain. Given a linear operator A and a dataset $\mathcal{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the pairs $\{(\mathbf{x}_i, A\mathbf{x}_i)\}_{i=1}^n$ can be used to learn an inverse mapping $f \approx A^{-1}$ by minimizing the distance between $\mathbf{x}_i$ and $f(A\mathbf{x}_i)$, even when A is underdetermined. State-of-the-art methods usually parametrize the mapping functions with deep neural nets. For example, stacked auto-encoders and convolutional neural nets have been used to solve compressive sensing and image deblurring problems [9]–

[12]. Recently, adversarial learning [19] has been demonstrated its ability to solve many challenging image problems, such as image inpainting [8] and super-resolution [7], [20].

Despite their ability to solve challenging problems, solving linear inverse problems with end-to-end mappings have a major disadvantage — the number of mapping functions scales linearly with the number of problems. Since the datasets are generated based on specific operators As, these end-to-end mappings can only solve the given problems A. Even if the problems change slightly, the mapping functions (nerual nets) need to be retrained. For example, a mapping to solve $2\times$-super-resolution cannot be used directly to solve $4\times$-super-resolution with satisfactory performance; it is even more difficult to re-purpose a mapping for image inpainting to solve image super-resolution. This specificity of end-to-end mappings makes it costly to incorporate them into consumer products that need to deal with a variety of image processing applications.

Deep generative models. Another thread of research learns generative models from image datasets. Suppose we have a dataset containing samples of a distribution $P(\mathbf{x})$. We can estimate $P(\mathbf{x})$ and sample from the model [21]–[23], or directly generate new samples from $P(\mathbf{x})$ without explicitly estimating the distribution [19], [24]. Dave *et al.* [25] use a spatial long-short-term memory network to learn the distribution $P(\mathbf{x})$; to solve linear inverse problems, they solve a maximum a posteriori estimation — maximizing $P(\mathbf{x})$ for $\mathbf{x}$ such that $\mathbf{y} = A\mathbf{x}$. Nguyen *et al.* [26] use a discriminative network and denoising autoencoders to implicitly learn the joint distribution between the image and its label $P(\mathbf{x}, y)$, and they generate new samples by sampling the joint distribution $P(\mathbf{x}, y)$, *i.e.*, the network, with an approximated Metropolis-adjusted Langevin algorithm. To solve image inpainting, they replace the values of known pixels in sampled images and repeat the sampling process. Similar to the proposed framework, these methods can be used to solve a wide variety of inverse problems. They use a probability framework and thus can be considered orthogonal to the proposed framework, which is motivated by a geometric perspective.

3 One Network to Solve Them All

Signal priors play an important role in regularizing underdetermined inverse problems. As mentioned in the introduction, traditional priors constraining the sparsity of signals in gradient or wavelet bases are often too generic, in that we can easily create non-image signals satisfying these priors. Instead of using traditional signal priors, we propose to learn a prior from a large image dataset. Since the prior is learned directly from the dataset, it is tailored to the statistics of images in the dataset and, in principle, provide stronger regularization to the inverse problem. In addition, similar to traditional signal priors, the learned signal prior can be used to solve any linear inverse problems pertaining to images.

3.1 Problem formulation

The proposed framework is motivated by the optimization technique, alternating direction method of multipliers method (ADMM) [27], that is widely used to solve linear inverse problems as defined in (1). A typical first step in ADMM is to separate a complicated objective into several simpler ones by variable splitting,

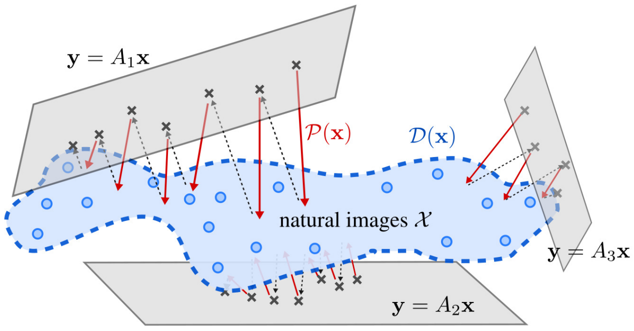

$\mathbf{y} = A_1\mathbf{x}$
$\mathcal{P}(\mathbf{x})$
$\mathcal{D}(\mathbf{x})$
natural images $\mathcal{X}$
$\mathbf{y} = A_3\mathbf{x}$
$\mathbf{y} = A_2\mathbf{x}$

Fig. 2: Given a large image dataset, the proposed framework learns a classifier $\mathcal{D}$ that fits a decision boundary of the natural image set. Based on $\mathcal{D}$, a projection network $\mathcal{P}(\mathbf{x})$:$\mathbb{R}^d \to \mathbb{R}^d$ is trained to fit the proximal operator of $\mathcal{D}$, which enables to solve a variety of linear inverse problems related to images using ADMM.

i.e., introducing an additional variable $\mathbf{z}$ that is constrained to be equal to $\mathbf{x}$. This gives us the following optimization problem:

$$\min_{\mathbf{x},\mathbf{z}} \ \frac{1}{2}\|\mathbf{y} - A\mathbf{z}\|_2^2 + \lambda\,\phi(\mathbf{x}) \tag{2}$$

$$\text{s.t.} \ \ \mathbf{x} = \mathbf{z}, \tag{3}$$

which is equivalent to the original problem (1). The scaled form of the augmented Lagrangian of (3) can be written as

$$\mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{u}) = \frac{1}{2}\|\mathbf{y} - A\mathbf{z}\|_2^2 + \lambda\,\phi(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z} + \mathbf{u}\|_2^2, \tag{4}$$

where $\rho > 0$ is the penalty parameter of the constraint $\mathbf{x} = \mathbf{z}$, and $\mathbf{u}$ is the dual variables divided by ρ. By alternatively optimizing $\mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{u})$ over $\mathbf{x}$, $\mathbf{z}$, and $\mathbf{u}$, ADMM is composed of the following procedures:

$$\mathbf{x}^{(k+1)} \leftarrow \arg\min_{\mathbf{x}} \frac{\rho}{2}\left\|\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}\right\|_2^2 + \lambda\,\phi(\mathbf{x}) \tag{5}$$

$$\mathbf{z}^{(k+1)} \leftarrow \arg\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{y} - A\mathbf{z}\|_2^2 + \frac{\rho}{2}\left\|\mathbf{x}^{(k+1)} - \mathbf{z} + \mathbf{u}^{(k)}\right\|_2^2 \tag{6}$$

$$\mathbf{u}^{(k+1)} \leftarrow \mathbf{u}^{(k)} + \mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)}. \tag{7}$$

The update of $\mathbf{z}$ (6) is a least squares problem and can be solved efficiently via algorithms like conjugate gradient descent. The update of $\mathbf{x}$ (5) is the proximal operator of the signal prior ϕ with penalty $\frac{\rho}{\lambda}$, denoted as $\mathbf{prox}_{\phi,\frac{\rho}{\lambda}}(\mathbf{v})$, where $\mathbf{v}{=}\mathbf{z}^{(k)}{-}\mathbf{u}^{(k)}$. When the signal prior uses ℓ_1-norm, the proximal operator is simply a soft-thresholding on $\mathbf{v}$. Notice that the ADMM algorithm separates the signal prior ϕ from the linear operator A. This enables us to learn a signal prior that can be used with any linear operator.

3.2 Learning a proximal operator

Since the signal prior only appears in the form of a proximal operator in ADMM, instead of explicitly learning a signal prior ϕ and solving the proximal operator in each step of ADMM, we propose to directly learn the proximal operator.

Let $\mathcal{X}$ represent the set of all natural images. The best signal prior is the indicator function of $\mathcal{X}$, denoted as $\mathcal{I}_{\mathcal{X}}(\cdot)$, and its corresponding proximal operator $\mathbf{prox}_{\mathcal{I}_{\mathcal{X}},\rho}(\mathbf{v})$ is a projection operator that projects $\mathbf{v}$ onto $\mathcal{X}$ from the geometric perspective— or equivalently, finding a $\mathbf{x} \in \mathcal{X}$ such that $\|\mathbf{x} - \mathbf{v}\|$ is minimized. However, we do not have the oracle indicator function $\mathcal{I}_{\mathcal{X}}(\cdot)$ in practice, so we cannot evaluate $\mathbf{prox}_{\mathcal{I}_{\mathcal{X}},\rho}(\mathbf{v})$ to solve the projection operation. Instead, we propose to train a classifier $\mathcal{D}$ with a large dataset whose classification cost function approximates $\mathcal{I}_{\mathcal{X}}$. Based on the learned classifier $\mathcal{D}$, we can learn a projection function $\mathcal{P}$ that maps a signal $\mathbf{v}$ to the set defined by the classifier. The learned projection function $\mathcal{P}$ can then replace the proximal operator (5), and we simply update $\mathbf{x}$ via

$$\mathbf{x}^{(k+1)} \leftarrow \mathcal{P}(\mathbf{z}^{(k)} - \mathbf{u}^{(k)}). \tag{8}$$

An illustration of the idea is shown in Figure 2.

There are some caveats for this approach. First, when the classification cost function of the classifier $\mathcal{D}$ is nonconvex, the overall optimization becomes nonconvex. For solving general nonconvex optimization problems, the convergence result is not guaranteed. Based on the theorems for the convergence of nonconvex ADMM [28], we provide the following theorem to the proposed ADMM framework.

Theorem 1. *Assume the function $\mathcal{P}$ solves the proximal operator (5). If the gradient of $\phi(x)$ is Lipschitz continuous and with large enough ρ, the ADMM algorithm is guaranteed to attain a stationary point.*

The proof follows directly from [28] and we omit the details here. Although Theorem 1 only guarantees convergence to stationary points instead of the optimal solution as other nonconvex formulations, it ensures that the algorithm will not diverge after several iterations. Second, we initialize the scaled dual variables $\mathbf{u}$ with zeros and $\mathbf{z}^{(0)}$ with the pseudo-inverse of the least-square term. Since we initialize $\mathbf{u}^0 = \mathbf{0}$, the input to the proximal operator $\mathbf{v}^{(k)}=\mathbf{z}^{(k)}-\mathbf{u}^{(k)} = \mathbf{z}^{(k)} + \sum_{i=1}^{k} \mathbf{x}^{(i)} - \mathbf{z}^{(i)} \approx \mathbf{z}^{(k)}$ resembles an image. Thereby, even though it is in general difficult to fit a projection function from any signal in $\mathbb{R}^d$ to the natural image space, we expect that the projection function only needs to deal with inputs that are close to images, and we train the projection function with slightly perturbed images from the dataset. Third, techniques like denoising autoencoders learn projection-like operators and, in principle, can be used in place of a proximal operator; however, our empirical findings suggest that ignoring the projection cost $\|\mathbf{v} - \mathcal{P}(\mathbf{v})\|^2$ and simply minimizing the reconstruction loss $\|\mathbf{x}_0 - \mathcal{P}(\mathbf{v})\|^2$, where $\mathbf{v}$ is a perturbed image from $\mathbf{x}_0$, leads to instability in the ADMM iterations.

3.3 Implementation details

We use two deep neural nets as the classifier $\mathcal{D}$ and the projection operator $\mathcal{P}$, respectively. Based on Theorem 1, we require the gradient of ϕ to be Lipschitz continuous. Since we choose to use cross entropy loss, we have $\phi(\mathbf{x}) = \log(\sigma(\mathcal{D}(\mathbf{x})))$ and in order to satisfy Theorem 1, we need $\mathcal{D}$ to be differentiable. Thus, we use the smooth exponential linear unit [29] as activation function, instead of rectified linear units. To bound the gradients of $\mathcal{D}$ w.r.t. $\mathbf{x}$, we truncate the weights of the network after each iteration.

We show an overview of the proposed method in Figure 3 and leave the details in Appendix A. The projector shares the same architecture of a typical convolutional autoencoder, and the classifier is a residual net [30]. One way to train the classifier $\mathcal{D}$ is to feed $\mathcal{D}$ natural images from a dataset and their perturbed counterparts. Nevertheless, we expect the projected images produced by the projector $\mathcal{P}$ be closer to the dataset $\mathcal{M}$ (natural images) than those perturbed images. Therefore, we jointly train two networks using adversarial learning: The projector $\mathcal{P}$ is trained to minimize (5), that is, confusing the classifier $\mathcal{D}$ by projecting $\mathbf{v}$ to the natural image set defined by the decision boundary of $\mathcal{D}$. When the projector improves and generates outputs that are within or closer to the boundary, the classifier can be updated to tighten its decision boundary. Although we start from a different perspective from [19], the above joint training procedure can also be understood as a two player game in adversarial learning, where the projector $\mathcal{P}$ tries to confuse the classifier $\mathcal{D}$.

Specifically, we optimize the projection network with the following objective function:

$$\min_{\boldsymbol{\theta}_{\mathcal{P}}} \sum_{\mathbf{x}\in\mathcal{M}, \mathbf{v}\sim f(\mathbf{x})} \lambda_1 \|\mathbf{x} - \mathcal{P}(\mathbf{x})\|^2 + \lambda_2 \|\mathbf{x} - \mathcal{P}(\mathbf{v})\|^2 \tag{9}$$

$$+ \lambda_3 \|\mathbf{v} - \mathcal{P}(\mathbf{v})\|^2 - \lambda_4 \log\left(\sigma(\mathcal{D}_\ell \circ \mathcal{E}(\mathbf{v}))\right) - \lambda_5 \log\left(\sigma(\mathcal{D} \circ \mathcal{P}(\mathbf{v}))\right), \tag{10}$$

where f is the function we used to generate perturbed images, and the first two terms in (9) are similar to (denoising) autoencoders and are

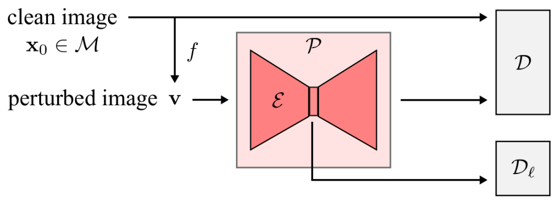


Fig. 3: Training the projection network $\mathcal{P}$. The adversarial learning is conducted on both image and latent spaces of $\mathcal{P}$.

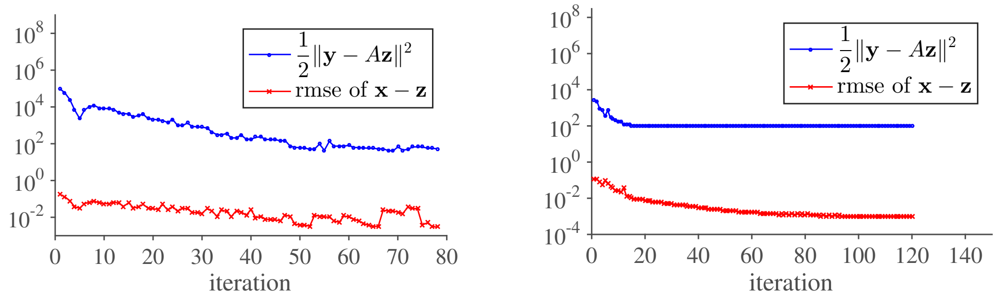
added to help the training procedure. The remaining terms in (10) form the projection loss as we need in (5). We use two classifiers $\mathcal{D}$ and $\mathcal{D}_\ell$, for the output (image) space and the latent spaces of the projector ($\mathcal{E}(\mathbf{v})$ in Figure 3), respectively. The latent classifier $\mathcal{D}_\ell$ is added to further help the training procedure [31]. We find that adding $\mathcal{D}_\ell$ also helps the projector to avoid overfitting. In all of our experiments, we set $\lambda_1 = 0.01, \lambda_3 = 0.005, \lambda_2 = 1.0, \lambda_4 = 0.0001$, and $\lambda_5 = 0.001$. The diagram of the training objective is shown in 3.

We briefly discuss the architecture of the networks. More details can be seen in Appendix A. As we have discussed above, to improve the convergence of ADMM, we use exponential linear unit and truncate the weights of $\mathcal{D}$ and $\mathcal{D}_\ell$ after each training iteration. Following the guidelines in [24], [32], we use a 10-layer convolution neural net that uses strides to downsample/upsample the images, and we use virtual batch normalization. However, we do not truncate the outputs of $\mathcal{P}$ with a tanh or a sigmoid function, in order to authentically compute the ℓ_2 projection loss. We find that using linear output helps the convergence of the ADMM procedure. We also use residual nets with six residual blocks as $\mathcal{D}$ and $\mathcal{D}_\ell$ instead of typical convolutional neural nets. We found that the stronger gradients provided by the shortcuts usually helps speed up the training process. Besides, we add a channel-wise fully connected layer followed by a 2×2 convolution to enable the projector to learn the context in the image, as in [8]. The classifiers $\mathcal{D}$ and $\mathcal{D}_\ell$ are trained to minimize the negative cross entropy loss. We use early stopping in order to alleviate overfitting. The complete architecture information is shown in the supplemental material.

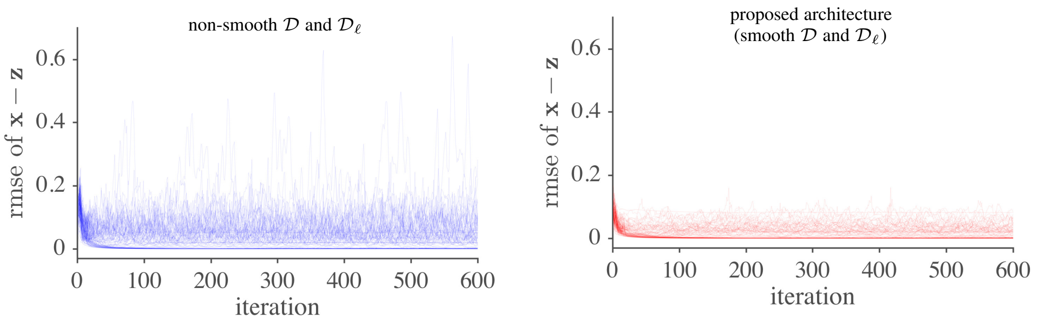
Image perturbation. While adding Gaussian noise may be the simplest method to perturb an image, we found that the projection network will easily overfit the Gaussian noise and become a dedicated Gaussian denoiser. Since during the ADMM process, the inputs to the projection network, $\mathbf{z}^{(k)} - \mathbf{u}^{(k)}$, do not usually follow a Gaussian distribution, the overfitted projection network may fail to project the general signals produced by the ADMM process. To avoid overfitting, we generate perturbed images with two methods — adding Gaussian noise with spatially varying standard deviations and smoothing the input images. We generate the noise by multiplying a randomly sampled standard Gaussian noise with a weighted mask upsampled from a low-dimensional mask with bicubic algorithm. The weighted mask is randomly sampled from a uniform distribution ranging from $[0.05, 0.5]$. Note that the images are ranging from $[-1, 1]$. To smooth the input images, we first downsample the input images and then use nearest-neighbor method to upsample the results. The ratio to the downsample is uniformly sampled from $[0.2, 0.95]$. After smoothing the images, we add the noise described above. We only use the smoothed images on ImageNet and MS-Cele-1M datasets.

3.4 Relationship to other algorithms

The proposed framework, which is composed of a classifier network and a projection network, is very similar to the works involving both adversarial learning and denoising auto-encoder, *e.g.*, context encoder [8] and mode-regularized GAN [33]. Compared to the probabilistic perspective typically used in adversarial learning [19] that matches the distributions of the dataset and the generated images, the proposed framework is based on the geometric perspective and the ADMM framework. We approximate the oracle indicator function



(a) Iterations of ADMM for compressive sensing (left) and scattered inpainting (right) of results shown in Figure 1.



(b) Comparison of ADMM convergence between a projection network trained with indifferentiable $\mathcal{D}$ and $\mathcal{D}_\ell$ (left) and the proposed architecture (right), in which the gradient of $\mathcal{D}$ and $\mathcal{D}_\ell$ are Lipschitz continuous. We perform scattered inpainting with box size equal to 6 and a total of 10 boxes on 100 random images in ImageNet dataset. For both cases, we set $\rho = 0.05$. We use transparent lines ($\alpha = 0.1$) in order to show densities.

Fig. 4: Convergence of ADMM

$\mathcal{I}_\mathcal{X}$ by $\mathcal{D}$ and its proximal operator by the projection operator $\mathcal{P}$. Our use of the adversarial training is simply for learning a tighter decision boundary, based on the hypothesis that images generated by $\mathcal{P}$ should be closer in ℓ_2 distance to $\mathcal{X}$ than the arbitrarily perturbed images.

The projection network $\mathcal{P}$ can be thought as a denoising autoencoder with a regularization term. Compared to the typical denoising auto-encoder, which always projects a perturbed image $\mathbf{x}_0 + \mathbf{n}$ back to the origin image $\mathbf{x}_0$, the proposed $\mathcal{P}$ may project $\mathbf{x}_0 + \mathbf{n}$ to the closest $\mathbf{x}$ in $\mathcal{X}$. In our empirical experience, the additional projection ℓ_2 loss help stabilize the ADMM process.

A recent work by Dave *et al*. can also be used to solve generic linear inverse problems. They use a spatial recurrent generative network to model the distribution of natural images $P(\mathbf{x})$and solve linear inverse problems by performing maximum a posteriori inference (maximizing $P(\mathbf{x})$ given $\mathbf{y} = A\mathbf{x}$). During the optimization, their method needs to compute the gradient of the network w.r.t. the input in each iteration, which can be computationally expensive when the network is very deep and complex. In contrast, the proposed method directly provides the solution to the x-update (8) and is thus computationally efficient.

The proposed method is also very similar to the denoising-based approximate message passing algorithm (D-AMP) [34], which has achieved state-of-the-art performance in solving compressive sensing problems. D-AMP is also motivated by geometric perspective of linear inverse problems and solve the compressive sensing problem with a variant of proximal gradient algorithm. However, instead of designing the proximal operator as the proposed method, D-AMP uses existing Gaussian denoising algorithms and relies on a Onsager correction term to ensure the noise resemble Gaussian noise. Thereby, D-AMP can only deal with linear operator A formed by random Gaussian matrices. In contrast, the proposed method directly learns a proximal operator that projects a signal to the image domain, and therefore, has fewer assumptions to the linear operator A.

3.5 Limitations

Unlike traditional signal priors whose weights λ can be adjusted at the time of solving the optimization problem (1), the prior weight of the proposed framework is fixed once the projection network is trained. While an ideal projection operator should not be affected by the value of the prior weights, sometimes, it may be preferable to control the effect of the signal prior to the solution. In our experiments, we find that adjusting ρ sometimes has similar effects as adjusting λ.

The convergence analysis of ADMM in Theorem 1 is based on the assumption that the projection network can provide global optimum of (5). However, in practice the optimality is not guaranteed. While there are convergence analyses with inexact proximal operators, the general properties are too complicated to analyze for deep neural nets. In practice, we find that for problems like pixelwise inpainting, compressive sensing, $2\times$ super-resolution and scattered inpainting the proposed framework converges gracefully, as shown in Figure 4a, but for more challenging problems like image inpainting with large blocks and $4\times$-super-resolution on ImageNet dataset, we sometimes need to stop the ADMM procedure early.

4 Experiments

We evaluate the proposed framework on three datasets:

MNIST dataset [35] contains randomly deformed images of MNIST dataset. The images are 28×28 and grayscale. We train the projector and the classifier networks on the training set and test the results on the test set. Since the dataset is relatively simpler, we remove the upper three layers from both $\mathcal{D}$ and $\mathcal{D}_\ell$. We use batches with 32 instances and train the networks for 80000 iterations.

MS-Celeb-1M dataset [36] contains a total of 8 million aligned and cropped face images of 10 thousand people from different viewing angles. We randomly select images of 73678 people as the training set and those of 25923 people as the test set. We resize the images into 64×64. We use batches with 25 instances and train the network for 10000 iterations.

ImageNet dataset [37] contains 1.2 million training images and 100 thousand test images on the Internet. We resize the images into 64×64. We use batches with 25 instances and train the network for 68000 iterations.

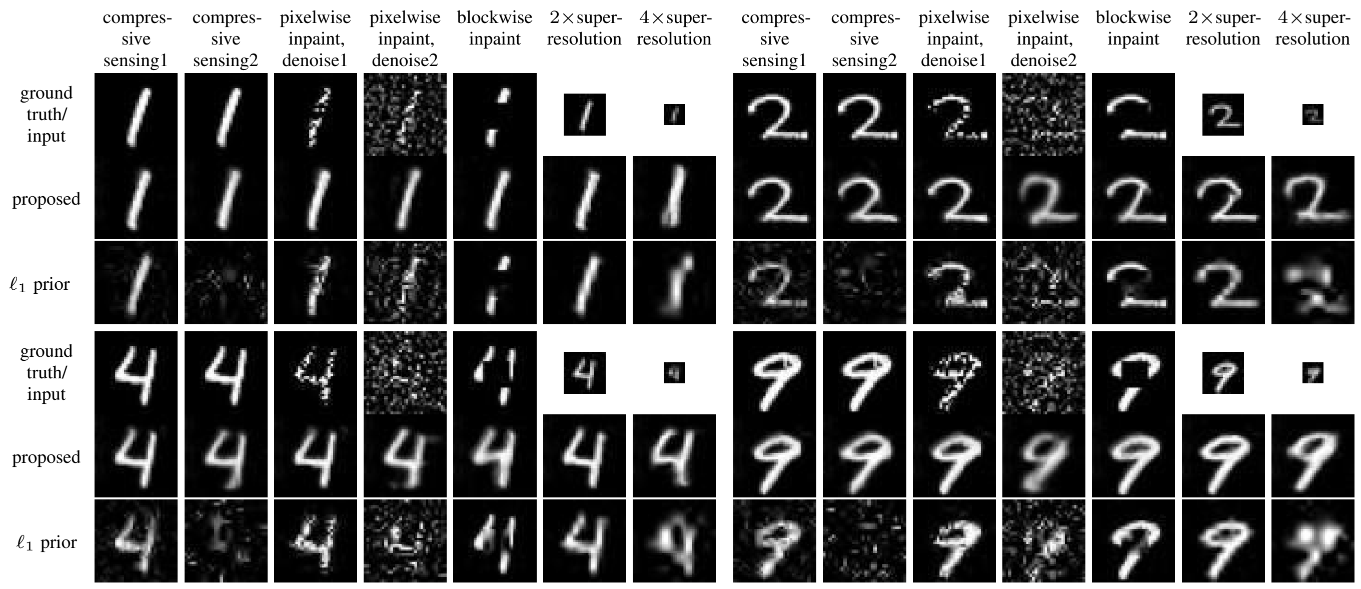


Fig. 5: Results on MNIST dataset. Since the input of compressive sensing cannot be visualized, we show the ground truth instead. Compressive sensing 1 uses $\frac{m}{d} = 0.3$ and compressive sensing 2 uses $\frac{m}{d} = 0.03$. Pixelwise inpaint 1 drops 50% of the pixels, and pixelwise inpaint 2 drops 70% of the pixels and add Gaussian noise with $\sigma = 0.3$. We use $\rho = 0.1$ for pixelwise inpainting and $\rho = 0.05$ for blockwise inpainting.

For each of the datasets, we perform the following tasks:

Compressive sensing. We use $m \times d$ random Gaussian matrices of different compression ($\frac{m}{d}$) as the linear operator A. The images are vectorized into d-dimensional vectors $\mathbf{x}$ and multiplied with the random Gaussian matrices to form $\mathbf{y}$.

Pixelwise inpainting and denoising. We randomly drop pixel values (independent of channels) by filling zeros and add Gaussian noise with different standard deviations.

Scattered inpainting. We randomly drop 10 small blocks by filling zeros. Each block is of 10% width and height of the input.

Blockwise inpainting. We fill the center 30% region of the input images with zeros.

Super resolution. We downsample the images into 50% and 25% of the original width and height using box-averaging algorithm.

Comparison to specially-trained networks. For each of the tasks, we train a specially-trained neural network using context encoder [8] with adversarial training. For compressive sensing, we design the network based on the work of [9], which applies $A^\top$ to the linear measurements and resize it into the image size to operate in image space. The measurement matrix A is a random Gaussian matrix and is fixed. For pixelwise inpainting and denoise, we randomly drop 50% of the pixels and add Gaussian noise with $\sigma = 0.5$ for each training instances. For blockwise inpainting, we drop a block with 30% size of the images at a random location in the images. For super resolution, we follow the work of Dong *et al.* [6] which first upsamples the low-resolution images to the target resolution using bicubic algorithm. We train a network for $2\times$-super resolution. Note that we do not train a network for $4\times$-super resolution and for scattered inpainting — to demonstrate that the specially-trained networks do not generalize well to similar tasks. Since the inputs to the $2\times$-super resolution network are bicubic-upsampled images, we also apply the upsampling to $\frac{1}{4}$-resolution images and feed them to the same network. We also feed scattered inpainting inputs to the blockwise inpainting network.

Comparison to hand-designed signal priors. We compare the proposed framework with the traditional signal prior using ℓ_1-norm of wavelet coefficients. We tune the weight of the ℓ_1 prior, λ, based on the dataset. For image denoising, we compare with the state-of-the-art algorithm, BM3D [38]. We add Gaussian random noise with different standard deviation σ (out of 255) to the test images, which were taken by the author with a cell phone camera. The value of σ of each image is provided to BM3D. For the proposed method, we let $A = I$, the identity matrix, and set $\rho = \frac{3}{255}\sigma$. We use the same projection network learned from ImageNet dataset and apply it to 64×64 patches. As shown in Figure 9, when σ is larger than 40, the proposed method consistently outperform BM3D. For image super-resolution, we compare with the work of Freeman and Fattal [39]. We perform $4\times$- and $8\times$-super resolution on the images from the result website of [39]. The super-resolution results are shown in Figure 10.

For each of the experiments, we use $\rho = 0.3$ if not mentioned. The results on MNIST, MS-Celeb-1M, and ImageNet dataset are shown in Figure 5, Figure 6, and Figure 7, respectively. In addition, we apply the projection network trained on ImageNet dataset on an image on the Internet [40]. To deal with the 384×512 image, when solving the projection operation (5), we apply the projection network on 64×64 patches and stitch the results directly. The reconstruction outputs are shown in Figure 1, and their statistics of each iteration of ADMM are shown in Figure 4a.

As can be seen from the results, using the proposed projection operator/network learning from datasets enables us to solve more challenging problems than using the traditional wavelet sparsity prior. In Figure 5 and Figure 6, while the traditional ℓ_1-prior of wavelet coefficients is able to reconstruct images from compressive measurements with $\frac{m}{d} = 0.3$, it fails to handle larger compression ratios like $\frac{m}{d} = 0.1$ and 0.03. Similar observations can be seen on pixelwise inpainting of different dropping probabilities and scattered and blockwise inpainting. In contrast, since the proposed projection network is tailored to the datasets, it enables the ADMM algorithm to solve challenging problems like compressive sensing with small $\frac{m}{d}$ and blockwise inpainting on MS-Celeb dataset.

Robustness to changes in linear operator and to noise. Even though the specially-trained networks are able to generate state-of-the-art results on their designing tasks, they are unable to deal with similar problems, even with a slight change of the linear operator A. For example, as shown in Figure 6, the blockwise inpainting network is able to deal with much larger vacant regions; however, it overfits the problem and fails to fill contents to smaller blocks in scatted inpainting problems. The $2\times$-super resolution network also fails to reconstruct higher resolution images for $4\times$-super resolution tasks, despite both

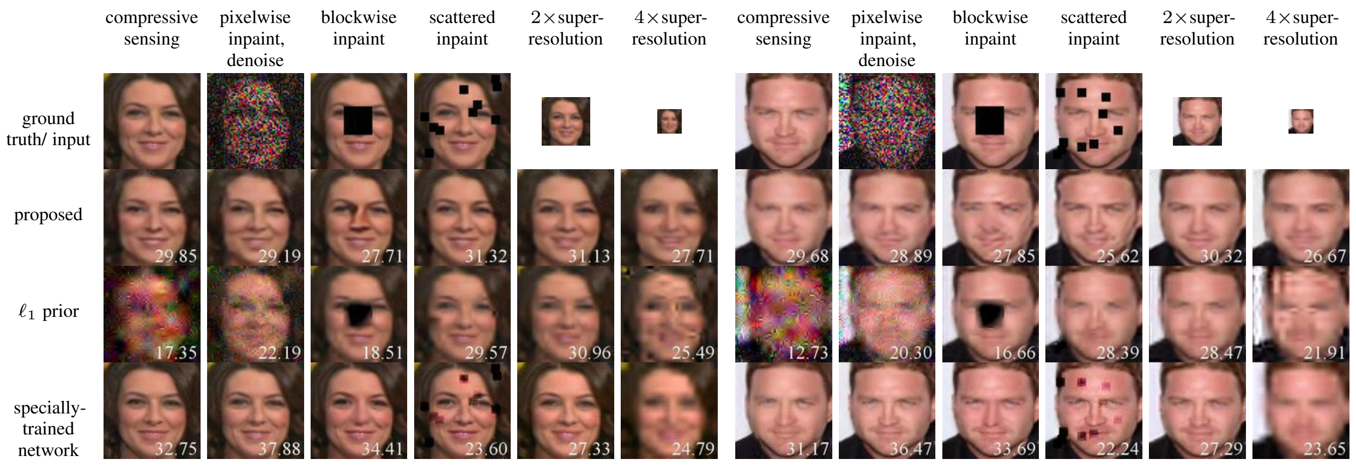


Fig. 6: Results on MS-Celeb-1M dataset. The PSNR values are shown in the lower-right corner of each image. For compressive sensing, we test on $\frac{m}{d} = 0.1$. For pixelwise inpainting, we drop 50% of the pixels and add Gaussian noise with $\sigma = 0.1$. We use $\rho = 1.0$ on both super resolution tasks.

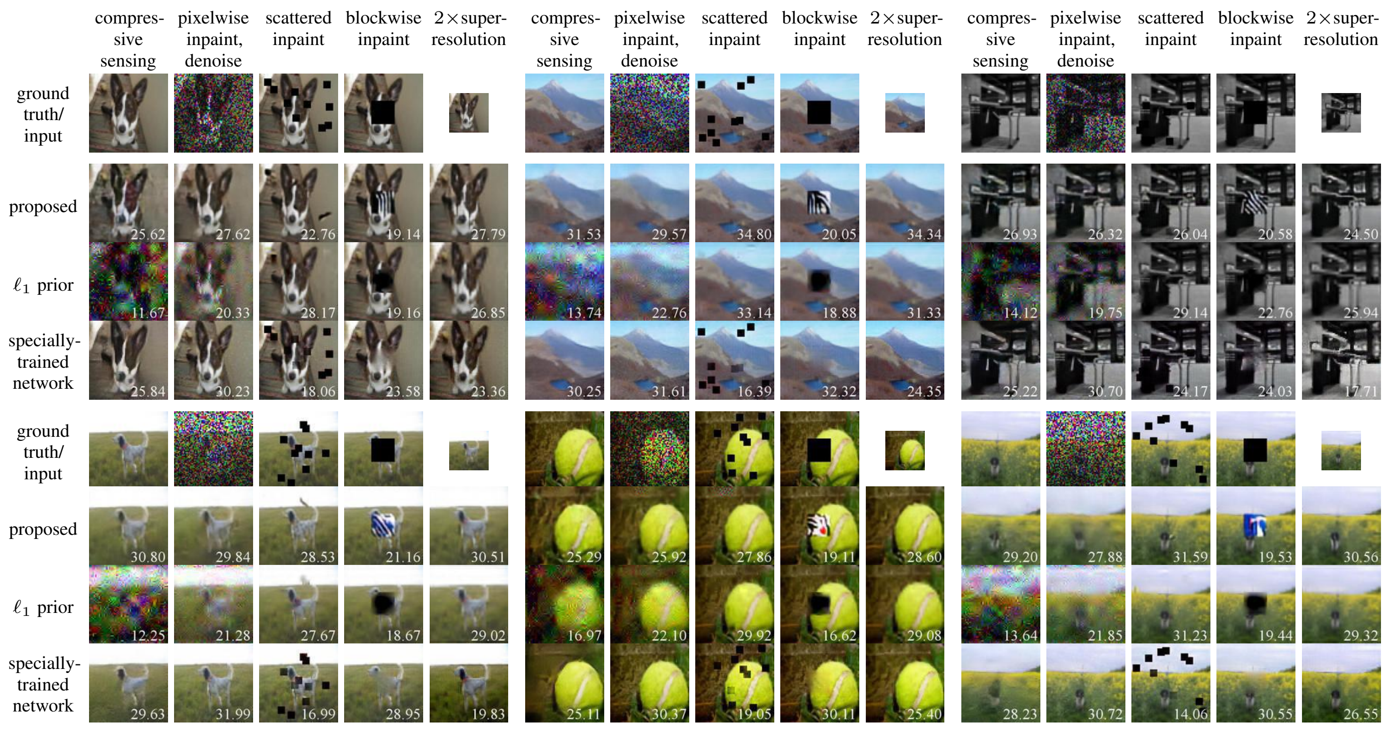


Fig. 7: Results on ImageNet dataset. The PSNR values are shown in the lower-right corner of each image. Compressive sensing uses $\frac{m}{d} = 0.1$. For pixelwise inpainting, we drop 50% of the pixels and add Gaussian noise with $\sigma = 0.1$. We use $\rho = 0.05$ on scattered inpainting and $\rho = 0.5$ on super resolution.

inputs are upsampled using bicubic algorithm beforehand. We extend this argument with a compressive sensing example. We start from the random Gaussian matrix A_0 used to train the compressive sensing network, and we progressively resample elements in A_0 from the same distribution constructing A_0. As shown in Figure 8, once the portion of resampled elements increases, the specially-trained network fails to reconstruct the inputs, even though the new matrices are still Gaussian. The network also shows lower tolerant to Gaussian noise added to the clean linear measurements $\mathbf{y} = A_0\mathbf{x}_0$. In comparison, the proposed projector network is robust to changes of linear operators and noise.

Convergence of ADMM. Theorem 1 provides a sufficient condition for the nonconvex ADMM to converge. As discussed in Section 3.3, based on Theorem 1, we use exponential linear units as the activation functions in $\mathcal{D}$ and $\mathcal{D}_\ell$ and truncate their weights after each training iteration, in order for the gradient of $\mathcal{D}$ and $\mathcal{D}_\ell$ to be Lipschitz continuous. Even though Theorem 1 is just a sufficient condition, in practice, we observe improvement in terms of convergence. We conduct experiments on scattered inpainting on ImageNet dataset using two projection networks — one trained with $\mathcal{D}$ and $\mathcal{D}_\ell$ using the smooth exponential linear units, and the other trained with $\mathcal{D}$ and $\mathcal{D}_\ell$ using the non-smooth leaky rectified linear units. Note that leaky rectified linear units are indifferentiable and thus violate the sufficient condition provided by Theorem 1. Figure 4b shows the root mean square error of $\mathbf{x} - \mathbf{z}$, which is a good indicator of the convergence of ADMM, of the two networks. As can be seen, using leaky rectified linear units results in higher and spikier root mean square error of $\mathbf{x} - \mathbf{z}$ than using exponential linear units. This indicates a less stable ADMM process. These results show that following Theorem 1 can benefit the convergence of ADMM.

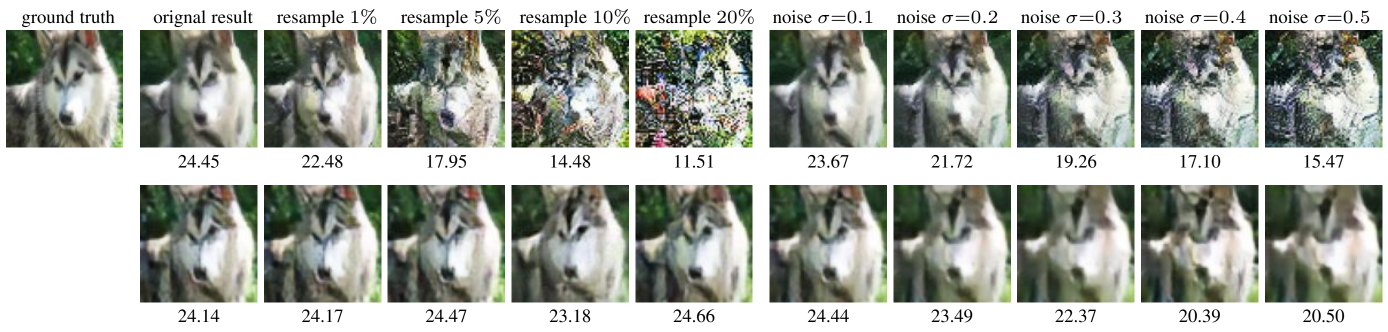


Fig. 8: Comparison on the robustness to the linear operator A and noise on compressive sensing. We start from the training measurement matrix A of the specialized network and randomly resample some elements in A. We also add Gaussian noise with different standard deviation σ to the linear measurements $\mathbf{y}$ of original A. The upper row shows the results of the specialized trained network, and the bottom one demonstrates results from the proposed framework. We use $\rho = 0.5$ for $\sigma = 0.2$, $\rho = 0.7$ for $\sigma = 0.3$, $\rho = 1.0$ for $\sigma = 0.4$, $\rho = 1.1$ for $\sigma = 0.5$, and $\rho = 0.3$ for all other cases. The PSNR values are shown in the below of each result.

Failure cases. The proposed projection network can fail to solve very challenging problems like the blockwise inpainting on ImageNet dataset, which has higher varieties in image contents than the other two datasets we test on. As shown in Figure 7, the proposed projection network tries to fill in random edges in the missing regions. In these cases, the projection network fails to project inputs to the natural image set, and thereby, violates our assumption in Theorem 1 and affects the overall ADMM framework. Even though increasing ρ can improve the convergence, it may produce low-quality, overly smoothed outputs.

5 Conclusion

In this paper, we propose a general framework to implicitly learn a signal prior — in the form of a projection operator — for solving generic linear inverse problems. The learned projection operator enjoys the high flexibility of deep neural nets and wide applicability of traditional signal priors. With the ability to solve generic linear inverse problems like denoising, inpainting, super-resolution and compressive sensing, the proposed framework resolves the scalability of specially-trained networks. This characteristic significantly lowers the cost to design specialized hardwares (ASIC for example) to solve image processing tasks. Thereby, we envision the projection network to be embedded into consumer devices like smart phones and autonomous vehicles to solve a variety of image processing problems.

Appendix

We now describe the architecture of the networks used in the paper. We use exponential linear unit (elu) [29] as activation function. We also use virtual batch normalization [32], where the reference batch size b_{ref} is equal to the batch size used for stochastic gradient descent. We weight the reference batch with $\frac{b_{\text{ref}}}{b_{\text{ref}}+1}$. We define some shorthands for the basic components used in the networks.

- *conv(w, c, s)*: convolution with $w \times w$ window size, c output channels and s stride.
- *dconv(w, c, s)* deconvolution (transpose of the convolution operation) with $w \times w$ window size, c output channels and s stride.
- *vbn*: virtual batch normalization.
- *bottleneck(same/half/quarter)*: bottleneck residual units [41] having the same, half, or one-fourth of the dimensionality of the input. Their block diagrams are shown in Figure 11.
- *cfc*: a channel-wise fully connected layer, whose output dimension is with same size as the input dimension.
- *fc(s)*: a fully-connected layer with the output size s.

To simply the notation, we use the subscript *ve* on a component to indicate that it is followed by *vbn* and *elu*.

.0.0.1 Projection network $\mathcal{P}$.: The projection network $\mathcal{P}$ is composed of one encoder network $\mathcal{E}$ and one decoder network, like a typical autoencoder. The encoder $\mathcal{E}$ projects an input to a 1024-dimensional latent space, and the decoder projects the latent representation back to the image space. The architecture of $\mathcal{E}$ is as follows.

$$\begin{array}{lcccc} Input & \rightarrow & conv(4,64,1)_{ve} & \rightarrow & conv(4,128,1)_{ve} \\ & \rightarrow & conv(4,256,2)_{ve} & \rightarrow & conv(4,512,2)_{ve} \\ & \rightarrow & conv(4,1024,2)_{ve} & \rightarrow & cfc \\ & \rightarrow & conv(2,1024,1)_{ve} & & (latent) \end{array} \tag{11}$$

The decoder is a symmetric counter part of the encoder:

$$\begin{array}{lcccc} latent & \rightarrow & dconv(4,512,2)_{ve} & \rightarrow & dconv(4,256,2)_{ve} \\ & \rightarrow & dconv(4,128,1)_{ve} & \rightarrow & dconv(4,64,1)_{ve} \\ & \rightarrow & dconv(4,3,1) & & (Output) \end{array} \tag{12}$$

.0.0.2 Image-space classifier $\mathcal{D}$.: As shown in Figure 3 of the paper, we use two classifiers — one operates in the image space $\mathbb{R}^d$ and discriminates natural images from the projection outputs, the other operates in the latent space of $\mathcal{P}$ based on the hypothesis that after encoded by $\mathcal{E}$, a perturbed image and a natural image should already lie in the same set.

For the image-space classifier $\mathcal{D}$, we use the 50-layer architecture of [30] but use the bottleneck blocks suggested in [41]. The detailed architecture is as follows.

$$\begin{array}{lcccc} Input & \rightarrow & conv(4,64,1) & & \\ & \rightarrow & bottleneck(half) & \rightarrow & \{bottleneck(same)\}_{\times 3} \\ & \rightarrow & bottleneck(half) & \rightarrow & \{bottleneck(same)\}_{\times 4} \\ & \rightarrow & bottleneck(half) & \rightarrow & \{bottleneck(same)\}_{\times 6} \\ & \rightarrow & bottleneck(half) & \rightarrow & \{bottleneck(same)\}_{\times 3} \\ & \rightarrow & vbn\ \&\ elu & \rightarrow & fc(1)\ (output), \end{array} \tag{13}$$

where $\{\}_{\times n}$ means we repeat the building block n times.

.0.0.3 Latent-space classifier $\mathcal{D}_\ell$.: The latent space classifier $\mathcal{D}_\ell$ operates on the output of the encoder $\mathcal{E}$. Since the input dimension is smaller than that of $\mathcal{D}$, we use fewer *bottleneck* blocks than we did in $\mathcal{D}$.

$$\begin{array}{lcc} Input & \rightarrow & bottleneck(same)_{\times 3} \\ & \rightarrow & bottleneck(quarter) \\ & \rightarrow & \{bottleneck(same)\}_{\times 2} \\ & \rightarrow & vbn\ \&\ elu \\ & \rightarrow & fc(1)\ (output) \end{array} \tag{14}$$

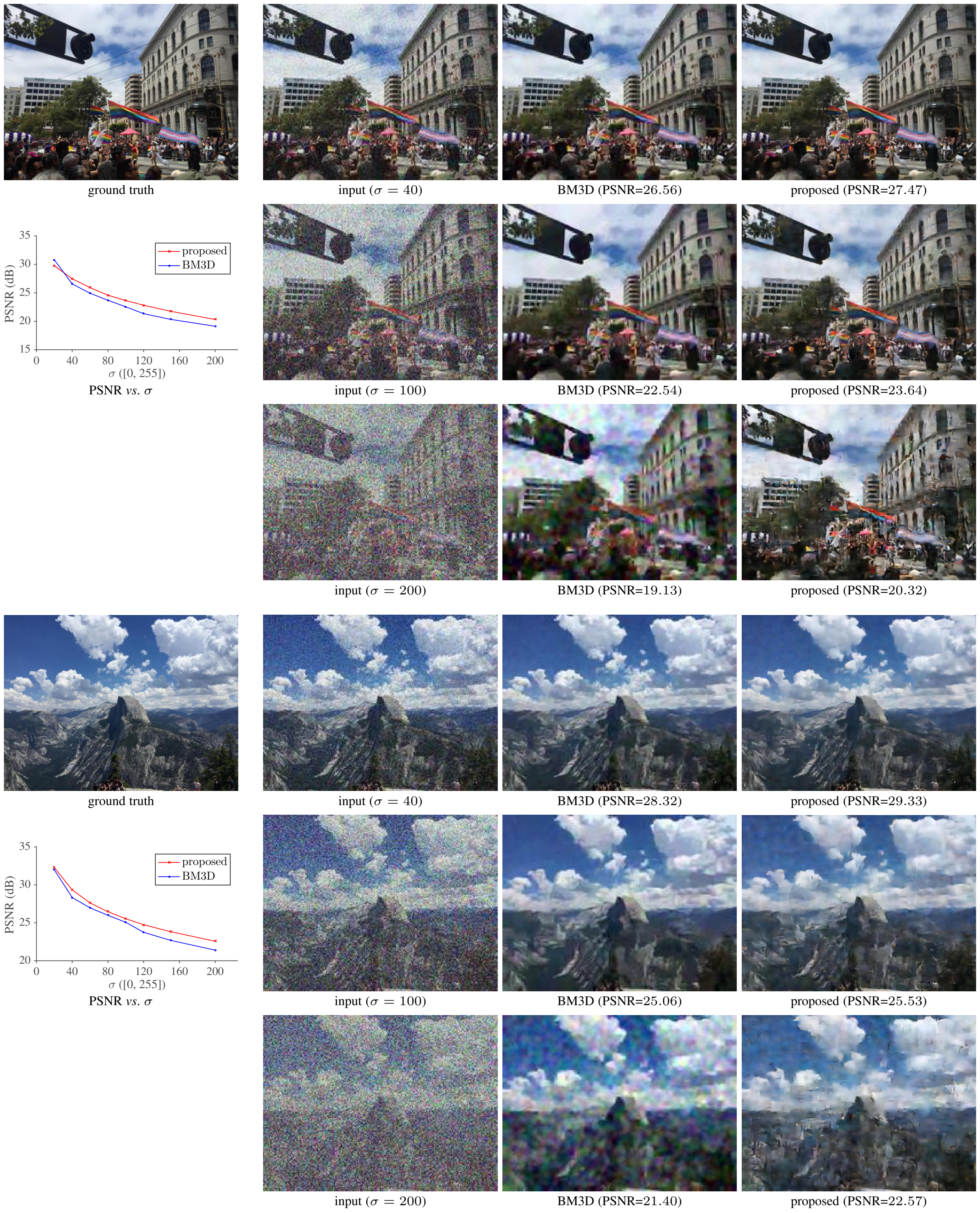


Fig. 9: Comparison to BM3D on image denoising

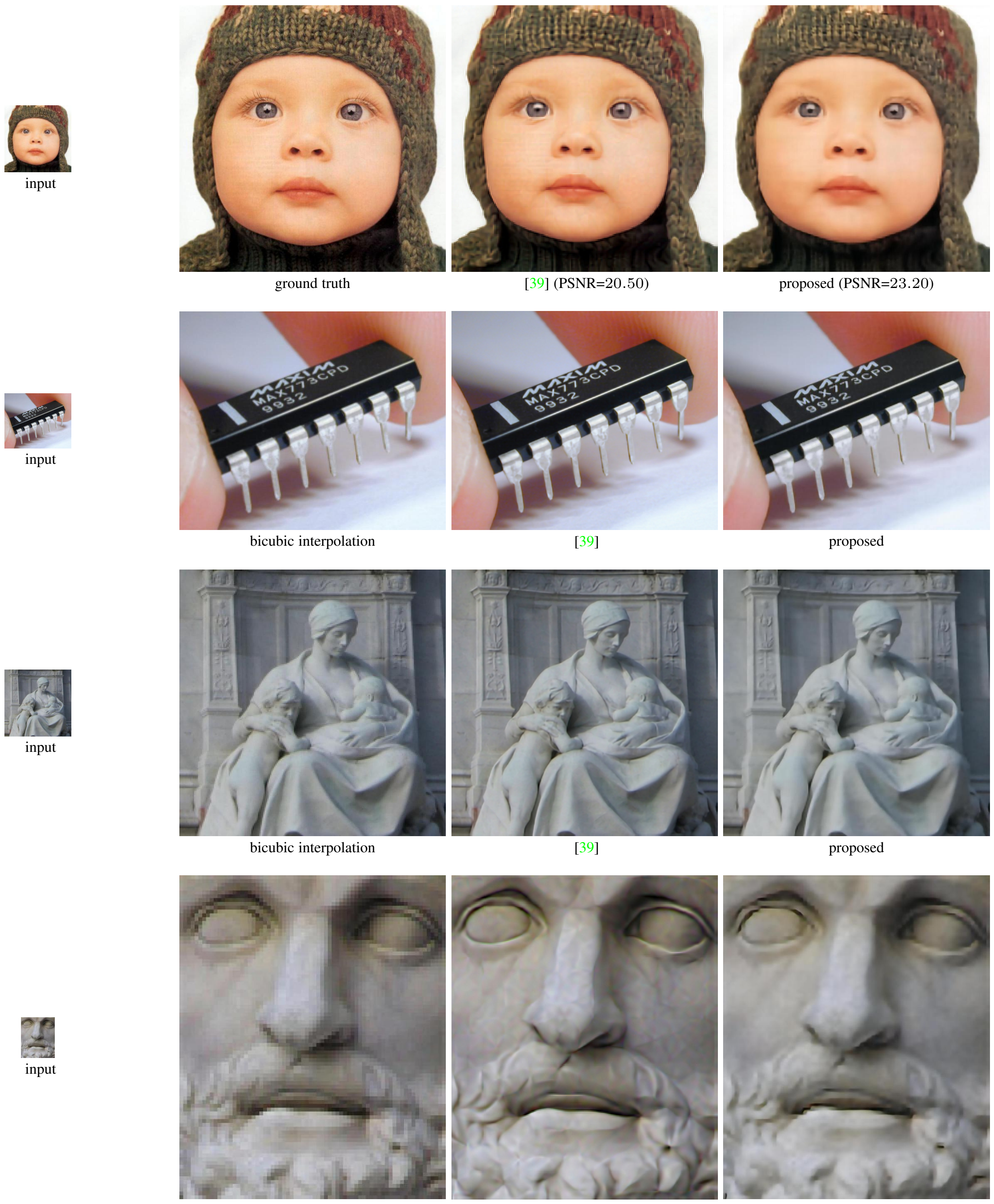
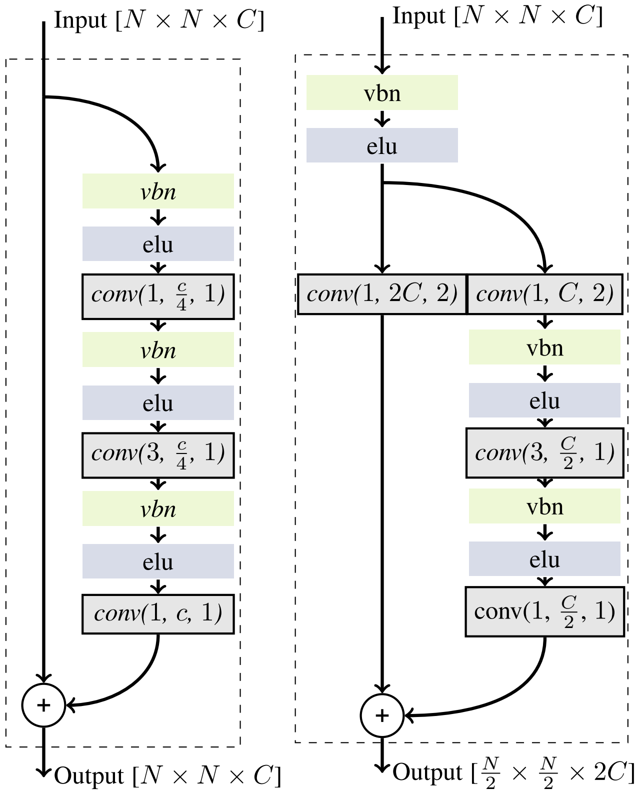


Fig. 10: Results of $4\times$ (on the first three rows) and $8\times$ (on the last row) super-resolution of Freeman and Fattal [39] (on the third column) and the proposed method (on the last column). All the input images are from [39]. Note that all the images, except for the one in the first row, do not have ground truth. For the proposed method, we use the projection network trained on ImageNet dataset and set $\rho = 1.0$.


Input $[N \times N \times C]$
vbn
elu
conv(1, $\frac{c}{4}$, 1)
vbn
elu
conv(3, $\frac{c}{4}$, 1)
vbn
elu
conv(1, c, 1)
+
Output $[N \times N \times C]$
Input $[N \times N \times C]$
vbn
elu
conv(1, $2C$, 2)
conv(1, C, 2)
vbn
elu
conv(3, $\frac{C}{2}$, 1)
vbn
elu
conv(1, $\frac{C}{2}$, 1)
+
Output $[\frac{N}{2} \times \frac{N}{2} \times 2C]$
Input $[N \times N \times C]$
vbn
elu
conv(1, C, 2)
conv(1, $\frac{C}{4}$, 2)
vbn
elu
conv(3, $\frac{C}{4}$, 1)
vbn
elu
conv(1, C, 1)
+
Output $[\frac{N}{2} \times \frac{N}{2} \times C]$

(a) bottleneck(same) (b) bottleneck(half) (c) bottleneck(quarter)

Fig. 11: Block diagrams of the bottleneck components used in the paper. (a) *bottleneck(same)* preserves the dimensionality of the input by maintaining the same output spatial dimension and the numger of channels. (b) *bottleneck(half)* reduces dimensionality by 2 via halving each spatial dimension and doubling the number of channels. (c) *bottleneck(quarter)* reduces dimensionality by 4 via halving each spatial dimension.

References

[1] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Info. Theory*, vol. 41, no. 3, pp. 613–627, 1995. 1

[2] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Trans. Image Proc.*, vol. 12, no. 11, pp. 1338–1351, 2003. 1

[3] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Modeling & Simulation*, vol. 7, no. 1, pp. 214–241, 2008. 1

[4] T. F. Chan, J. Shen, and H.-M. Zhou, "Total variation wavelet inpainting," *Journal of Mathematical imaging and Vision*, vol. 25, no. 1, pp. 107–125, 2006. 1

[5] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Trans. Image Proc.*, vol. 20, no. 7, pp. 1838–1857, 2011. 1

[6] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *ECCV*, 2014. 1, 6

[7] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016. 1, 3

[8] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016. 1, 3, 4, 6

[9] A. Mousavi and R. G. Baraniuk, "Learning to invert: Signal recovery via deep convolutional networks," *arXiv preprint arXiv:1701.03891*, 2017. 1, 2, 6

[10] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed measurements," in *CVPR*, 2016. 1, 2

[11] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *Allerton Conference on Communication, Control, and Computing*, 2015. 1, 2

[12] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *NIPS*, 2014. 1, 2

[13] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies, "Data compression and harmonic analysis," *IEEE Trans. Info. Theory*, vol. 44, no. 6, pp. 2435–2476, 1998. 2

[14] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006. 2

[15] J. Hays and A. A. Efros, "Scene completion using millions of photographs," *ACM TOG*, vol. 26, no. 3, p. 4, 2007. 2

[16] K. Dale, M. K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister, "Image restoration using online photo collections," in *ICCV*, 2009. 2

[17] M. Aharon, M. Elad, and A. Bruckstein, "k-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Proc.*, vol. 54, no. 11, pp. 4311–4322, 2006. 2

[18] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Proc.*, vol. 19, no. 11, pp. 2861–2873, 2010. 2

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014. 3, 4

[20] R. Dahl, M. Norouzi, and J. Shlens, "Pixel recursive super resolution," *arXiv preprint arXiv:1702.00783*, 2017. 3

[21] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *AISTATS*, 2009. 3

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014. 3

[23] L. Theis and M. Bethge, "Generative image modeling using spatial lstms," in *NIPS*, 2015. 3

[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015. 3, 4

[25] A. Dave, A. K. Vadathya, and K. Mitra, "Compressive image recovery using recurrent generative model," *arXiv preprint arXiv:1612.04229*, 2016. 3

[26] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space," *arXiv preprint arXiv:1612.00005*, 2016. 3

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011. 3

[28] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *arXiv preprint arXiv:1511.06324*, 2015. 4

[29] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015. 4, 8

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016. 4, 8

[31] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015. 4

[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016. 4, 8

[33] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016. 4

[34] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Trans. Info. Theory*, vol. 62, no. 9, pp. 5117–5144, 2016. 5

[35] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. MIT Press, 2007, pp. 301–320. 5

[36] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *ECCV*, 2016. 5

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015. 5

[38] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Bm3d image denoising with shape-adaptive principal component analysis," in *Signal Processing with Adaptive Sparse Structured Representations*, 2009. 6

[39] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM TOG*, vol. 28, no. 3, pp. 1–10, 2010. 6, 10

[40] "One ring to rule them all. http://i.onionstatic.com/avclub/5329/98/16x9/1600.jpg." 6

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016. 8