An inner-loop free solution to inverse problems using deep neural networks

Qi Wei*
Duke University
Durham, NC 27710
qi.wei@duke.edu

Kai Fai*
Duke University
Durham, NC 27710
kai.fan@stat.duke.edu

Lawrence Carin
Duke University
Durham, NC 27710
lcarin@duke.edu

Katherine A. Heller
Duke University
Durham, NC 27710
khelle r@stat.duke.edu

Abstract

We propose a new method that uses deep learning techniques to accelerate the popular alternating direction method of multipliers (ADMM) solution for inverse problems. The ADMM updates consist of a proximity operator, a least squares regression that includes a big matrix inversion, and an explicit solution for updating the dual variables. Typically, inner loops are required to solve the first two sub-minimization problems due to the intractability of the prior and the matrix inversion. To avoid such drawbacks or limitations, we propose an *inner-loop free* update rule with two pre-trained deep convolutional architectures. More specifically, we learn a conditional denoising auto-encoder which imposes an implicit data-dependent prior/regularization on ground-truth in the first sub-minimization problem. This design follows an empirical Bayesian strategy, leading to so-called amortized inference. For matrix inversion in the second sub-problem, we learn a convolutional neural network to approximate the matrix inversion, i.e., the inverse mapping is learned by feeding the input through the learned forward network. Note that training this neural network does not require ground-truth or measurements, i.e., it is data-independent. Extensive experiments on both synthetic data and real datasets demonstrate the efficiency and accuracy of the proposed method compared with the conventional ADMM solution using inner loops for solving inverse problems.

1 Introduction

Most of the inverse problems are formulated directly to the setting of an optimization problem related to the a forward model [23]. The forward model maps unknown signals, i.e., the ground-truth, to acquired information about them, which we call data or measurements. This mapping, or forward problem, generally depends on a physical theory that links the ground-truth to the measurements. Solving inverse problems involves learning the inverse mapping from the measurements to the ground-truth. Specifically, it recovers a signal from a small number of degraded or noisy measurements. This is usually ill-posed [24, 23]. Recently, deep learning techniques have emerged as excellent models and gained great popularity for their widespread success in allowing for efficient inference techniques on applications include pattern analysis (unsupervised), classification (supervised), computer vision, image processing, etc [6]. Exploiting deep neural networks to help solve inverse problems has been explored recently [22, 1] and deep learning based methods have achieved state-of-the-art performance in many challenging inverse problems like super-resolution [3, 22], image reconstruction [18],

*The authors contributed equally to this work.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

automatic colorization [12]. More specifically, massive datasets currently enables learning end-to-end mappings from the measurement domain to the target image/signal/data domain to help deal with these challenging problems instead of solving the inverse problem by inference. The pairs $\{\mathbf{x}, \mathbf{y}\}$ are used to learn the mapping function from $\mathbf{y}$ to $\mathbf{x}$, where $\mathbf{x}$ is the ground-truth and $\mathbf{y}$ is its corresponding measurement. This mapping function has recently been characterized by using sophisticated networks, e.g., deep neural networks. A strong motivation to use neural networks stems from the universal approximation theorem [5], which states that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of $\mathbb{R}^n$, under mild assumptions on the activation function.

More specifically, in recent work [3, 22, 12, 18], an end-to-end mapping from measurements $\mathbf{y}$ to ground-truth $\mathbf{x}$ was learned from the training data and then applied to the testing data. Thus, the complicated inference scheme needed in the conventional inverse problem solver was replaced by feeding a new measurement through the pre-trained network, which is much more efficient. One main problem for this strategy is that it requires task-specific training of the networks, i.e., different problems require different networks. Thus, it is very expensive to solve diverse sets of problems. To improve the scope of deep neural network models, more recently, in [4], a splitting strategy was proposed to decompose an inverse problem into two optimization problems, where one sub-problem, related to regularization, can be solved efficiently using trained deep neural networks, leading to an alternating direction method of multipliers (ADMM) framework [2]. This method involves training a deep convolutional auto-encoder network for low-level image modeling, which explicitly imposes regularization that spans the subspace that the ground-truth images live in. For the sub-problem that requires inverting a big matrix, a conventional gradient descent algorithm was used, leading to an alternating update, iterating between feed-forward propagation through a network and iterative gradient descent. Thus, an inner loop for gradient descent is still necessary in this framework.

In this work, we propose an inner-loop free framework, in the sense that no iterative algorithm is required to solve sub-problems, using a splitting strategy for inverse problems. The alternating updates for the two sub-problems were derived by feeding through two pre-trained deep neural networks, i.e., one using an amortized inference based denoising convolutional auto-encoder network for the proximity operation and one using structured convolutional neural networks for the huge matrix inversion related to the forward model. Thus, the computational complexity of each iteration in ADMM is linear with respect to (w.r.t.) the dimensionality of the signals. The network for the proximity operation imposes an implicit prior learned from the training data, including the measurements as well as the ground-truth, leading to amortized inference. The network for matrix inversion is independent from the training data and can be trained from noise, i.e., a random noise image and its output from the forward model. This independence from training data allows the proposed framework to be used to accelerate almost all the existing training data/example free solutions for inverse problems based on a splitting strategy. To make training the networks for the proximity operation easier, three tricks have been employed: the first one is to use a pixel shuffling technique to equalize the dimensionality of the measurements and ground-truth; the second one is to optionally add an adversarial loss borrowed from the GAN (Generative Adversarial Nets) framework [10] for sharp image generation; the last one is to introduce a perceptual measurement loss derived from pre-trained networks, such as AlexNet [11] or VGG-16 Model [21]. Arguably, the speed of the proposed algorithm, which we term Inf-ADMM-ADNN (*Inner-loop free ADMM with Auxiliary Deep Neural Network*), comes from the fact that it uses two auxiliary pre-trained networks to accelerate the updates of ADMM.

Contribution The main contribution of this paper is comprised of **i**) learning an implicit prior/regularizer using a denoising auto-encoder neural network, based on amortized inference; **ii**) learning the inverse of a big matrix using structured convolutional neural networks, without using training data; **iii**) each of the above networks can be exploited to accelerate the existing ADMM solver for inverse problems.

2 Linear Inverse Problem

Notation: trainable networks by calligraphic font, e.g., $\mathcal{A}$, fixed networks by italic font e.g., A.

As mentioned in the last section, the low dimensional measurement is denoted as $\mathbf{y} \in \mathbb{R}^m$, which is reduced from high dimensional ground truth $\mathbf{x} \in \mathbb{R}^n$ by a linear operator A such that $\mathbf{y} = A\mathbf{x}$. Note that usually $n \geq m$, which makes the number of parameters to estimate no smaller than the number of data points in hand. This imposes an ill-posed problem for finding solution $\mathbf{x}$ on new observation

$\mathbf{y}$, since A is an underdetermined measurement matrix. For example, in a super-resolution set-up, the matrix A might not be invertible, such as the strided Gaussian convolution in [19, 22]. To overcome this difficulty, several computational strategies, including Markov chain Monte Carlo (MCMC) and tailored variable splitting under the ADMM framework, have been proposed and applied to different kinds of priors, e.g., the empirical Gaussian prior [27], the Total Variation prior [20], etc. In this paper, we focus on the popular ADMM framework due to its low computational complexity and recent success in solving large scale optimization problems. More specifically, the optimization problem is formulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x},\mathbf{z}} \|\mathbf{y} - A\mathbf{z}\|^2 + \lambda\mathcal{R}(\mathbf{x}), \quad s.t. \quad \mathbf{z} = \mathbf{x} \tag{1}$$

where the introduced auxiliary variable $\mathbf{z}$ is constrained to be equal to $\mathbf{x}$, and $\mathcal{R}(\mathbf{x})$ captures the structure promoted by the prior/regularization. If we design the regularization in an empirical Bayesian way, by imposing an implicit data dependent prior on $\mathbf{x}$, i.e., $\mathcal{R}(\mathbf{x};\mathbf{y})$ for amortized inference [22], the augmented Lagrangian for (1) is

$$\mathcal{L}(\mathbf{x},\mathbf{z},\mathbf{u}) = \|\mathbf{y} - A\mathbf{z}\|^2 + \lambda\mathcal{R}(\mathbf{x};\mathbf{y}) + \langle\mathbf{u}, \mathbf{x} - \mathbf{z}\rangle + \beta\|\mathbf{x} - \mathbf{z}\|^2 \tag{2}$$

where $\mathbf{u}$ is the Lagrange multiplier, and $\beta > 0$ is the penalty parameter. The usual augmented Lagrange multiplier method is to minimize $\mathcal{L}$ w.r.t. $\mathbf{x}$ and $\mathbf{z}$ simultaneously. This is difficult and does not exploit the fact that the objective function is separable. To remedy this issue, ADMM decomposes the minimization into two subproblems that are minimizations w.r.t. $\mathbf{x}$ and $\mathbf{z}$, respectively. More specifically, the iterations are as follows:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \beta\|\mathbf{x} - \mathbf{z}^k + \mathbf{u}^k/2\beta\|^2 + \lambda\mathcal{R}(\mathbf{x};\mathbf{y}) \tag{3}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \|\mathbf{y} - A\mathbf{z}\|^2 + \beta\|\mathbf{x}^{k+1} - \mathbf{z} + \mathbf{u}^k/2\beta\|^2 \tag{4}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + 2\beta(\mathbf{x}^{k+1} - \mathbf{z}^{k+1}). \tag{5}$$

If the prior $\mathcal{R}$ is appropriately chosen, such as $\|\mathbf{x}\|_1$, a closed-form solution for (3), i.e., a soft thresholding solution is naturally desirable. However, for some more complicated regularizations, e.g., a patch based prior [8], solving (3) is nontrivial, and may require iterative methods. To solve (4), a matrix inversion is necessary, for which conjugate gradient descent (CG) is usually applied to update $\mathbf{z}$ [4]. Thus, solving (3) and (4) is in general cumbersome. Inner loops are required to solve these two sub-minimization problems due to the intractability of the prior and the inversion, resulting in large computational complexity. To avoid such drawbacks or limitations, we propose an *inner loop-free* update rule with two pretrained deep convolutional architectures.

3 Inner-loop free ADMM

3.1 Amortized inference for $\mathbf{x}$ using a conditional proximity operator

Solving sub-problem (3) is equivalent to finding the solution of the proximity operator

$$\mathcal{P}_{\mathcal{R}}(\mathbf{v};\mathbf{y}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2 + \mathcal{R}(\mathbf{x};\mathbf{y}) \tag{6}$$

where we incorporate the constant $\frac{\lambda}{2\beta}$ into $\mathcal{R}$ without loss of generality. If we impose the first order necessary conditions [16], we have

$$\mathbf{x} = \mathcal{P}_{\mathcal{R}}(\mathbf{v};\mathbf{y}) \Leftrightarrow 0 \in \partial\mathcal{R}(\cdot;\mathbf{y})(\mathbf{x}) + \mathbf{x} - \mathbf{v} \Leftrightarrow \mathbf{v} - \mathbf{x} \in \partial\mathcal{R}(\cdot;\mathbf{y})(\mathbf{x}) \tag{7}$$

where $\partial\mathcal{R}(\cdot;\mathbf{y})$ is a partial derivative operator. For notational simplicity, we define another operator $\mathcal{F} =: \mathcal{I} + \partial\mathcal{R}(\cdot;\mathbf{y})$. Thus, the last condition in (7) indicates that $\mathbf{x}^{k+1} = \mathcal{F}^{-1}(\mathbf{v})$. Note that the inverse here represents the inverse of an operator, i.e., the inverse function of $\mathcal{F}$. Thus our objective is to learn such an inverse operator which projects $\mathbf{v}$ into the prior subspace. For simple priors like $\|\cdot\|_1$ or $\|\cdot\|_2^2$, the projection can be efficiently computed. In this work, we propose an implicit example-based prior, which does not have a truly Bayesian interpretation, but aids in model optimization. In line with this prior, we define the implicit proximity operator $\mathcal{G}_{\boldsymbol{\theta}}(\mathbf{x};\mathbf{v},\mathbf{y})$ parameterized by $\boldsymbol{\theta}$ to approximate unknown $\mathcal{F}^{-1}$. More specifically, we propose a neural network architecture referred to as conditional Pixel Shuffling Denoising Auto-Encoders (cPSDAE) as the operator $\mathcal{G}$, where pixel shuffling [19] means periodically reordering the pixels in each channel mapping a high resolution image to a low resolution image with scale r and increase the number of channels to r^2 (see [19] for more details). This allows us to transform $\mathbf{v}$ so that it is the same scale as $\mathbf{y}$, and concatenate it with $\mathbf{y}$ as the input of cPSDAE easily. The architecture of cPSDAE is shown in Fig. 1 (d).

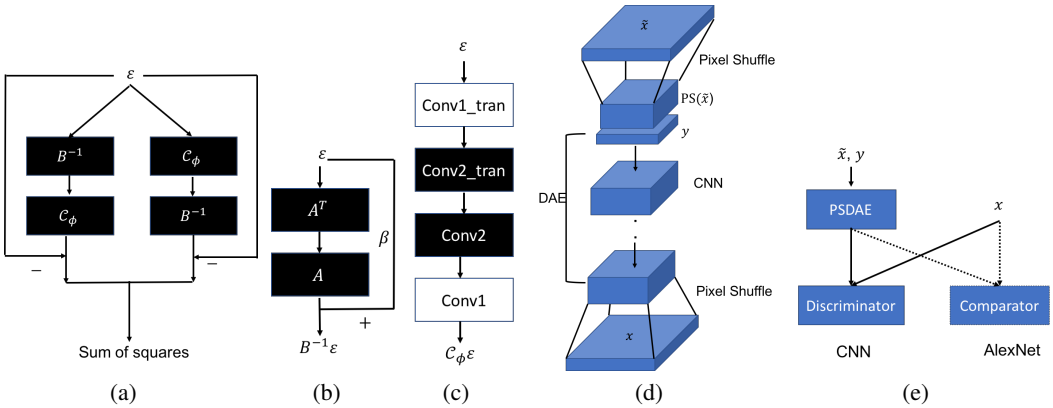


Figure 1: Network for updating $\mathbf{z}$ (in black): (a) loss function (10), (b) structure of B^{-1}, (c) struture of $\mathcal{C}_\phi$. Note that the input ϵ is random noise independent from the training data. Network for updating $\mathbf{z}$ (in blue): (d) structure of cPSDAE $\mathcal{G}_{\boldsymbol{\theta}}(\mathbf{x}; \tilde{\mathbf{x}}, \mathbf{y})$ ($\tilde{\mathbf{x}}$ plays the same role as $\mathbf{v}$ in training), (e) adversarial training for $\mathcal{R}(\mathbf{x}; \mathbf{y})$. Note again that (a)(b)(c) describes the network for inferring $\mathbf{z}$, which is data-independent and (d)(e) describes the network for inferring $\mathbf{x}$, which is data-dependent.

3.2 Inversion-free update of z

While it is straightforward to write down the closed-form solution for sub-problem (4) w.r.t. $\mathbf{z}$ as is shown in (8), explicitly computing this solution is nontrivial.

$$\mathbf{z}^{k+1} = K\left(A^\top \mathbf{y} + \beta \mathbf{x}^{k+1} + \mathbf{u}^k/2\right), \text{ where } K = \left(A^\top A + \beta \mathbf{I}\right)^{-1} \tag{8}$$

In (8), $A^\top$ is the transpose of the matrix A. As we mentioned, the term K in the right hand side involves an expensive matrix inversion with computational complexity $O(n^3)$. Under some specific assumptions, e.g., A is a circulant matrix, this matrix inversion can be accelerated with a Fast Fourier transformation, which has a complexity of order $\mathcal{O}(n \log n)$. Usually, the gradient based update has linear complexity in each iteration and thus has an overall complexity of order $\mathcal{O}(n_{\text{int}} \log n)$, where n_{int} is the number of iterations. In this work, we will learn this matrix inversion explicitly by designing a neural network. Note that K is only dependent on A, and thus can be computed in advance for future use. This problem can be reduced to a smaller scale matrix inversion by applying the Sherman-Morrison-Woodbury formula:

$$K = \beta^{-1}\left(\mathbf{I} - A^\top B A\right), \text{ where } B = \left(\beta \mathbf{I} + A A^\top\right)^{-1}. \tag{9}$$

Therefore, we only need to solve the matrix inversion in dimension $m \times m$, i.e., estimating B. We propose an approach to approximate it by a trainable deep convolutional neural network $\mathcal{C}_{\boldsymbol{\phi}} \approx B$ parameterized by $\boldsymbol{\phi}$. Note that $B^{-1} = \lambda \mathbf{I} + A A^\top$ can be considered as a two-layer fully-connected or convolutional network as well, but with a fixed kernel. This inspires us to design two auto-encoders with shared weights, and minimize the sum of two reconstruction losses to learn the inversion $\mathcal{C}_{\boldsymbol{\phi}}$:

$$\arg\min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\|\boldsymbol{\varepsilon} - \mathcal{C}_{\boldsymbol{\phi}} B^{-1}\boldsymbol{\varepsilon}\|_2^2 + \|\boldsymbol{\varepsilon} - B^{-1}\mathcal{C}_{\boldsymbol{\phi}}\boldsymbol{\varepsilon}\|_2^2\right] \tag{10}$$

where $\boldsymbol{\varepsilon}$ is sampled from a standard Gaussian distribution. The loss in (10) is clearly depicted in Fig. 1 (a) with the structure of B^{-1} in Fig. 1 (b) and the structure of $\mathcal{C}_{\boldsymbol{\phi}}$ in Fig. 1 (c). Since the matrix B is symmetric, we can reparameterize $\mathcal{C}_{\boldsymbol{\phi}}$ as $\mathcal{W}_{\boldsymbol{\phi}}\mathcal{W}_{\boldsymbol{\phi}}^\top$, where $\mathcal{W}_{\boldsymbol{\phi}}$ represents a multi-layer convolutional network and $\mathcal{W}_{\boldsymbol{\phi}}^\top$ is a symmetric convolution transpose architecture using shared kernels with $\mathcal{W}_{\boldsymbol{\phi}}$, as shown in Fig. 1 (c) (the blocks with the same colors share the same network parameters). By plugging the learned $\mathcal{C}_{\boldsymbol{\phi}}$ in (9) , we obtain a reusable deep neural network $\mathcal{K}_{\boldsymbol{\phi}} = \beta^{-1}\left(\mathbf{I} - A^\top \mathcal{C}_{\boldsymbol{\phi}} A\right)$ as a surrogate for the exact inverse matrix K. The update of $\mathbf{z}$ at each iteration can be done by applying the same $\mathcal{K}_{\boldsymbol{\phi}}$ as follows:

$$\mathbf{z}^{k+1} \leftarrow \beta^{-1}\left(\mathbf{I} - A^\top \mathcal{C}_{\boldsymbol{\phi}} A\right)\left(A^\top \mathbf{y} + \beta \mathbf{x}^{k+1} + \mathbf{u}^k/2\right). \tag{11}$$

3.3 Adversarial training of cPSDAE

In this section, we will describe the proposed adversarial training scheme for cPSDAE to update $\mathbf{x}$. Suppose that we have the paired training dataset $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$, a single cPSDAE with the input pair $(\tilde{\mathbf{x}}, \mathbf{y})$ is trying to minimize the reconstruction error $\mathcal{L}_r(\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \mathbf{y}), \mathbf{x})$, where $\tilde{\mathbf{x}}$ is a corrupted version of $\mathbf{x}$, i.e., $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{n}$ where $\mathbf{n}$ is random noise. Notice $\mathcal{L}_r$ in traditional DAE is commonly defined as ℓ_2 loss, however, ℓ_1 loss is an alternative in practice. Additionally, we follow the idea in [17, 7] by introducing a discriminator and a comparator to help train the cPSDAE, and find that it can produce sharper or higher quality images than merely optimizing $\mathcal{G}$. This will wrap our conditional generative model $\mathcal{G}_{\boldsymbol{\theta}}$ into the conditional GAN [10] framework with an extra feature matching network (comparator). Recent advances in representation learning problems have shown that the features extracted from well pre-trained neural networks on supervised classification problems can be successfully transferred to others tasks, such as zero-shot learning [14], style transfer learning [9]. Thus, we can simply use pre-trained AlexNet [11] or VGG-16 Model [21] on ImageNet as the comparator without fine-tuning in order to extract features that capture complex and perceptually important properties. The feature matching loss $\mathcal{L}_f(C(\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \mathbf{y})), C(\mathbf{x}))$ is usually the ℓ_2 distance of high level image features, where C represents the pre-trained network. Since C is fixed, the gradient of this loss can be back-propagated to $\boldsymbol{\theta}$.

For the adversarial training, the discriminator $\mathcal{D}_{\boldsymbol{\psi}}$ is a trainable convolutional network. We can keep the standard discriminator loss as in a traditional GAN, and add the generator loss of the GAN to the previously defined DAE loss and comparator loss. Thus, we can write down our two objectives as follows,

$$\mathcal{L}_D(\mathbf{x}, \mathbf{y}) = -\log \mathcal{D}_{\boldsymbol{\psi}}(\mathbf{x}) - \log\left(1 - \mathcal{D}_{\boldsymbol{\psi}}(\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \mathbf{y}))\right) \tag{12}$$

$$\mathcal{L}_G(\mathbf{x}, \mathbf{y}) = \lambda_r \|\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \mathbf{y}) - \mathbf{x}\|_2^2 + \lambda_f \|C(\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \mathbf{y})) - C(\mathbf{x})\|_2^2 - \lambda_a \log \mathcal{D}_{\boldsymbol{\psi}}(\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \mathbf{y})) \tag{13}$$

The optimization involves iteratively updating $\boldsymbol{\psi}$ by minimizing $\mathcal{L}_D$ keeping $\boldsymbol{\theta}$ fixed, and then updating $\boldsymbol{\theta}$ by minimizing $\mathcal{L}_G$ keeping $\boldsymbol{\psi}$ fixed. The proposed method, including training and inference has been summarized in Algorithm 1. Note that each update of $\mathbf{x}$ or $\mathbf{z}$ using neural networks in an ADMM iteration has a complexity of linear order w.r.t. the data dimensionality n.

3.4 Discussion

Algorithm 1 Inner-loop free ADMM with Auxiliary Deep Neural Nets (Inf-ADMM-ADNN)

Training stage:

1: Train net $\mathcal{K}_{\boldsymbol{\phi}}$ for inverting $A^TA + \beta\mathbf{I}$
2: Train net cPSDAE for proximity operator of $\mathcal{R}(\mathbf{x}; \mathbf{y})$

Testing stage:

1: **for** $t = 1, 2, \ldots$ **do**
2: Update $\mathbf{x}$ cf. $\mathbf{x}^{k+1} = \mathcal{F}^{-1}(\mathbf{v})$;
3: Update $\mathbf{z}$ cf. (11);
4: Update $\mathbf{u}$ cf. (5);
5: **end for**

A critical point for learning-based methods is whether the method generalizes to other problems. More specifically, how does a method that is trained on a specific dataset perform when applied to another dataset? To what extent can we reuse the trained network without re-training?

In the proposed method, two deep neural networks are trained to infer $\mathbf{x}$ and $\mathbf{z}$. For the network w.r.t. $\mathbf{z}$, the training only requires the forward model A to generate the training pairs $(\epsilon, A\epsilon)$. The trained network for $\mathbf{z}$ can be applied for any other datasets as long as they share the same A. Thus, this network can be adapted easily to accelerate inference for inverse problems without training data. However, for inverse problems that depends on a different A, a re-trained network is required. It is worth mentioning that the forward model A can be easily learned using training dataset $(\mathbf{x}, \mathbf{y})$, leading to a fully blind estimator associated with the inverse problem. An example of learning $\hat{A}$ can be found in the supplementary materials (see Section 1). For the network w.r.t. $\mathbf{x}$, training requires data pairs $(\mathbf{x}_i, \mathbf{y}_i)$ because of the amortized inference. Note that this is different from training a prior for $\mathbf{x}$ only using training data $\mathbf{x}_i$. Thus, the trained network for $\mathbf{x}$ is confined to the specific tasks constrained by the pairs $(\mathbf{x}, \mathbf{y})$. To extend the generality of the trained network, the amortized setting can be removed, i.e, the measurements $\mathbf{y}$ is removed from the training, leading to a solution to proximity operator $\mathcal{P}_{\mathcal{R}}(\mathbf{v}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2 + \mathcal{R}(\mathbf{x})$. This proximity operation can be regarded as a denoiser which projects the noisy version $\mathbf{v}$ of $\mathbf{x}$ into the subspace imposed by $\mathcal{R}(\mathbf{x})$. The trained network (for the proximity operator) can be used as a plug-and-play prior [25] to regularize other inverse problems

for datasets that share similar statistical characteristics. However, a significant change in the training dataset, e.g., different modalities like MRI and natural images (e.g., ImageNet [11]), would require re-training.

Another interesting point to mention is the scalability of the proposed method to data of different dimensions. The scalability can be adapted using patch-based methods without loss of generality. For example, a neural network is trained for images of size 64×64 but the test image is of size 256×256. To use this pre-trained network, the full image can be decomposed as four 64×64 images and fed to the network. To overcome the possible blocking artifacts, eight overlapping patches can be drawn from the full image and fed to the network. The output of these eight patches are then averaged (unweighted or weighted) over the overlapping parts. A similar strategy using patch stitching can be exploited to feed small patches to the network for higher dimensional datasets.

4 Experiments

In this section, we provide experimental results and analysis on the proposed Inf-ADMM-ADNN and compare the results with a conventional ADMM using inner loops for inverse problems. Experiments on synthetic data have been implemented to show the fast convergence of our method, which comes from the efficient feed-forward propagation through pre-trained neural networks. Real applications using proposed Inf-ADMM-ADNN have been explored, including single image super-resolution, motion deblurring and joint super-resolution and colorization.

4.1 Synthetic data

To evaluate the performance of proposed Inf-ADMM-ADNN, we first test the neural network $\mathcal{K}_{\boldsymbol{\phi}}$, approximating the matrix inversion on synthetic data. More specifically, we assume that the ground-truth $\mathbf{x}$ is drawn from a Laplace distribution Laplace(μ, b), where $\mu = 0$ is the location parameter and b is the scale parameter. The forward model A is a sparse matrix representing convolution with a stride of 4. The architecture of A is available in the supplementary materials (see Section 2). The noise $\mathbf{n}$ is drawn from a standard Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Thus, the observed data is generated as $\mathbf{y} = A\mathbf{x} + \mathbf{n}$. Following Bayes theorem, the maximum a posterior estimate of $\mathbf{x}$ given $\mathbf{y}$, i.e., maximizing $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ can be equivalently formulated as $\arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{y} - A\mathbf{x}\|_2^2 + \frac{1}{b}\|\mathbf{x}\|_1$, where $b = 1$ and $\sigma = 1$ in this setting. Following (3), (4), (5), this problem is reduced to the following three sub-problems:

$$\mathbf{x}^{k+1} = \mathcal{S}_{\frac{1}{2\beta}}(\mathbf{z}^k - \mathbf{u}^k/2\beta) \tag{14}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \|\mathbf{y} - A\mathbf{z}\|_2^2 + \beta\|\mathbf{x}^{k+1} - \mathbf{z} + \mathbf{u}^k/2\beta\|_2^2 \tag{15}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + 2\beta(\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \tag{16}$$

where the soft thresholding operator $\mathcal{S}$ is defined as $\mathcal{S}_\kappa(a) = \begin{cases} 0 & |a| \leq \kappa \\ a - \text{sgn}(a)\kappa & |a| > \kappa \end{cases}$ and sgn(a) extracts the sign of a. The update of $\mathbf{x}^{k+1}$ has a closed-form solution, i.e., soft thresholding of $\mathbf{z}^k - \mathbf{u}^k/2\beta$. The update of $\mathbf{z}^{k+1}$ requires the inversion of a big matrix, which is usually solved using a gradient descent based algorithm. The update of $\mathbf{u}^{k+1}$ is straightforward. Thus, we compare the gradient descent based update, a closed-form solution for matrix inversion[2] and the proposed inner-free update using a pre-trained neural network. The evolution of the objective function w.r.t. the number of iterations and the time has been plotted in the left and middle of Figs. 2. While all three methods perform similarly from iteration to iteration (in the left of Figs. 2), the proposed inner-loop free based and closed-form inversion based methods converge much faster than the gradient based method (in the middle of Figs. 2). Considering the fact that the closed-form solution, i.e., a direct matrix inversion, is usually not available in practice, the learned neural network allows us to approximate the matrix inversion in a very accurate and efficient way.

[2]Note that this matrix inversion can be explicitly computed due to its small size in this toy experiment. In practice, this matrix is not built explicitly.

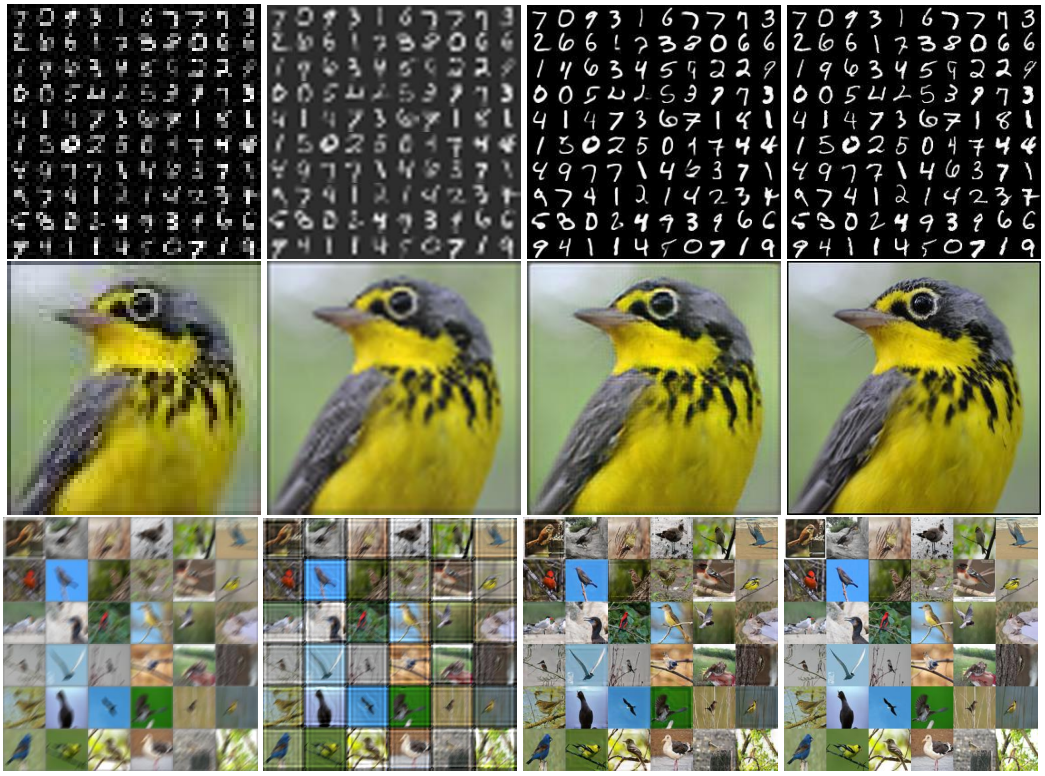


Figure 3: Top two rows : (column 1) LR images, (column 2) bicubic interpolation ($\times 4$), (column 3) results using proposed method ($\times 4$), (column 4) HR image. Bottom row: (column 1) motion blurred images, (column 2) results using Wiener filter with the best performance by tuning regularization parameter, (column 3) results using proposed method, (column 4) ground-truth.

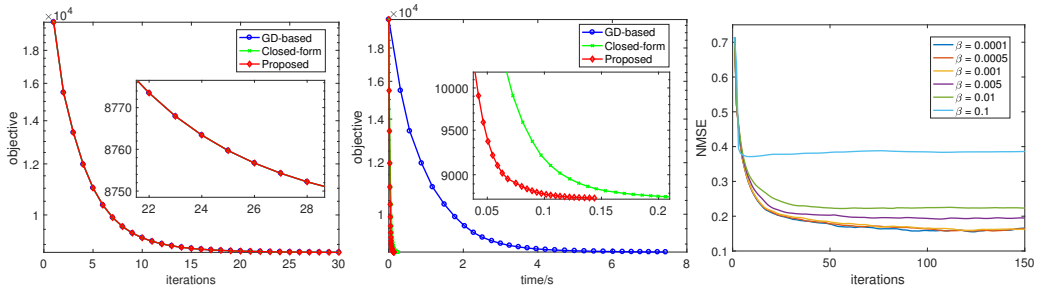


Figure 2: Synthetic data: (left) objective *v.s.* iterations, (middle) objective *v.s.* time. MNIST dataset: (right) NMSE *v.s.* iterations for MNIST image $4\times$ super-resolution.

4.2 Image super-resolution and motion deblurring

In this section, we apply the proposed Inf-ADMM-ADNN to solve the poplar image super-resolution problem. We have tested our algorithm on the MNIST dataset [13] and the 11K images of the Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [26]. In the first two rows of Fig. 3, high resolution images, as shown in the last column, have been blurred (convolved) using a Gaussian kernel of size 3×3 and downsampled every 4 pixels in both vertical and horizontal directions to generate the corresponding low resolution images as shown in the first column. The bicubic interpolation of LR images and results using proposed Inf-ADMM-ADNN on a 20% held-out test set are displayed in column 2 and 3. Visually, the proposed Inf-ADMM-ADNN gives much better results than the bicubic interpolation, recovering more details including colors and edges. A similar task to super-resolution is motion deblurring, in which the convolution kernel is a directional kernel

and there is no downsampling. The motion deblurring results using Inf-ADMM-ADNN are displayed in the bottom of Fig. 3 and are compared with the Wiener filtered deblurring result (the performance of Wiener filter has been tuned to the best by adjusting the regularization parameter). Obviously, the Inf-ADMM-ADNN gives visually much better results than the Wiener filter. Due to space limitations, more simulation results are available in supplementary materials (see Section 3.1 and 3.2).

To explore the convergence speed w.r.t. the ADMM regularization parameter β, we have plotted the normalized mean square error (NMSE) defined as NMSE $= \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 / \|\mathbf{x}\|_2^2$, of super-resolved MNIST images w.r.t. ADMM iterations using different values of β in the right of Fig. 2. It is interesting to note that when β is large, e.g., 0.1 or 0.01, the NMSE of ADMM updates converges to a stable value rapidly in a few iterations (less than 10). Reducing the value of β slows down the decay of NMSE over iterations but reaches a lower stable value. When the value of β is small enough, e.g., $\beta = 0.0001, 0.0005, 0.001$, the NMSE converges to the identical value. This fits well with the claim in Boyd's book [2] that when β is too large it does not put enough emphasis on minimizing the objective function, causing coarser estimation; thus a relatively small β is encouraged in practice. Note that the selection of this regularization parameter is still an open problem.

4.3 Joint super-resolution and colorization

While image super-resolution tries to enhance spatial resolution from spatially degraded images, a related application in the spectral domain exists, i.e., enhancing spectral resolution from a spectrally degraded image. One interesting example is the so-called automatic colorization, i.e., hallucinating a plausible color version of a colorless photograph. To the best knowledge of the authors, this is the first time we can enhance both spectral and spatial resolutions from one single band image. In this section, we have tested the ability to perform joint super-resolution and colorization from one single colorless LR image on the celebA-dataset [15]. The LR colorless image, its bicubic interpolation and $\times 2$ HR image are displayed in the top row of Fig. 4. The ADMM updates in the 1st, 4th and 7th iterations (on held-out test set) are displayed in the bottom row, showing that the updated image evolves towards higher quality. More results are in the supplementary materials (see Section 3.3).

Figure 4: (top left) colorless LR image, (top middle) bicubic interpolation, (top right) HR ground-truth, (bottom left to right) updated image in $\mathbf{1}$th, $\mathbf{4}$th and $\mathbf{7}$th ADMM iteration. Note that the colorless LR images and bicubic interpolations are visually similar but different in details noticed by zooming out.

5 Conclusion

In this paper we have proposed an accelerated alternating direction method of multipliers, namely, Inf-ADMM-ADNN to solve inverse problems by using two pre-trained deep neural networks. Each ADMM update consists of feed-forward propagation through these two networks, with a complexity of linear order with respect to the data dimensionality. More specifically, a conditional pixel shuffling denoising auto-encoder has been learned to perform amortized inference for the proximity operator. This auto-encoder leads to an implicit prior learned from training data. A data-independent structured convolutional neural network has been learned from noise to explicitly invert the big matrix associated with the forward model, getting rid of any inner loop in an ADMM update, in contrast to the conventional gradient based method. This network can also be combined with existing proximity operators to accelerate existing ADMM solvers. Experiments and analysis on both synthetic and real dataset demonstrate the efficiency and accuracy of the proposed method. In future work we hope to extend the proposed method to inverse problems related to nonlinear forward models.

Acknowledgments

The authors would like to thank NVIDIA for the GPU donations.

Appendices

A Learning A from training data

The objective to estimate A is formulated as

$$\arg\min_{A} \sum_{i=1}^{N} \|\mathbf{y}_i - A\mathbf{x}_i\|_2^2 + \lambda\phi(A) \tag{17}$$

where $(\mathbf{x}_i, \mathbf{y}_i)_{i=1:N}$ are the training pairs and $\phi(A)$ corresponds to a regularization to A. Empirically, when m is large enough, the regularization plays a less important role. The learned and real kernels for A (of size 4×4) are visually very similar as is shown in Fig. 5.

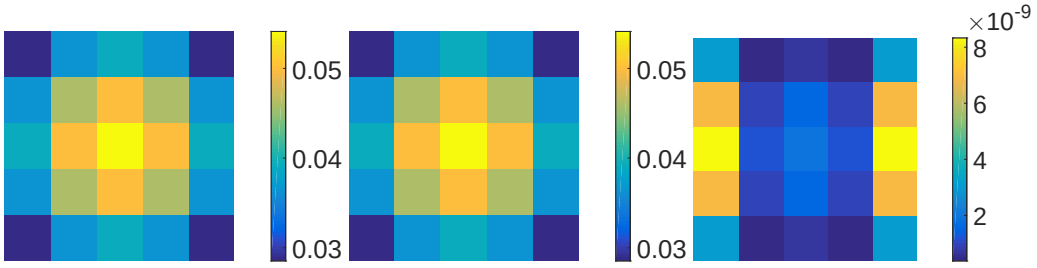


Figure 5: (left) Ground-truth kernel for A , (middle) learned kernel for A, (right) difference of these two.

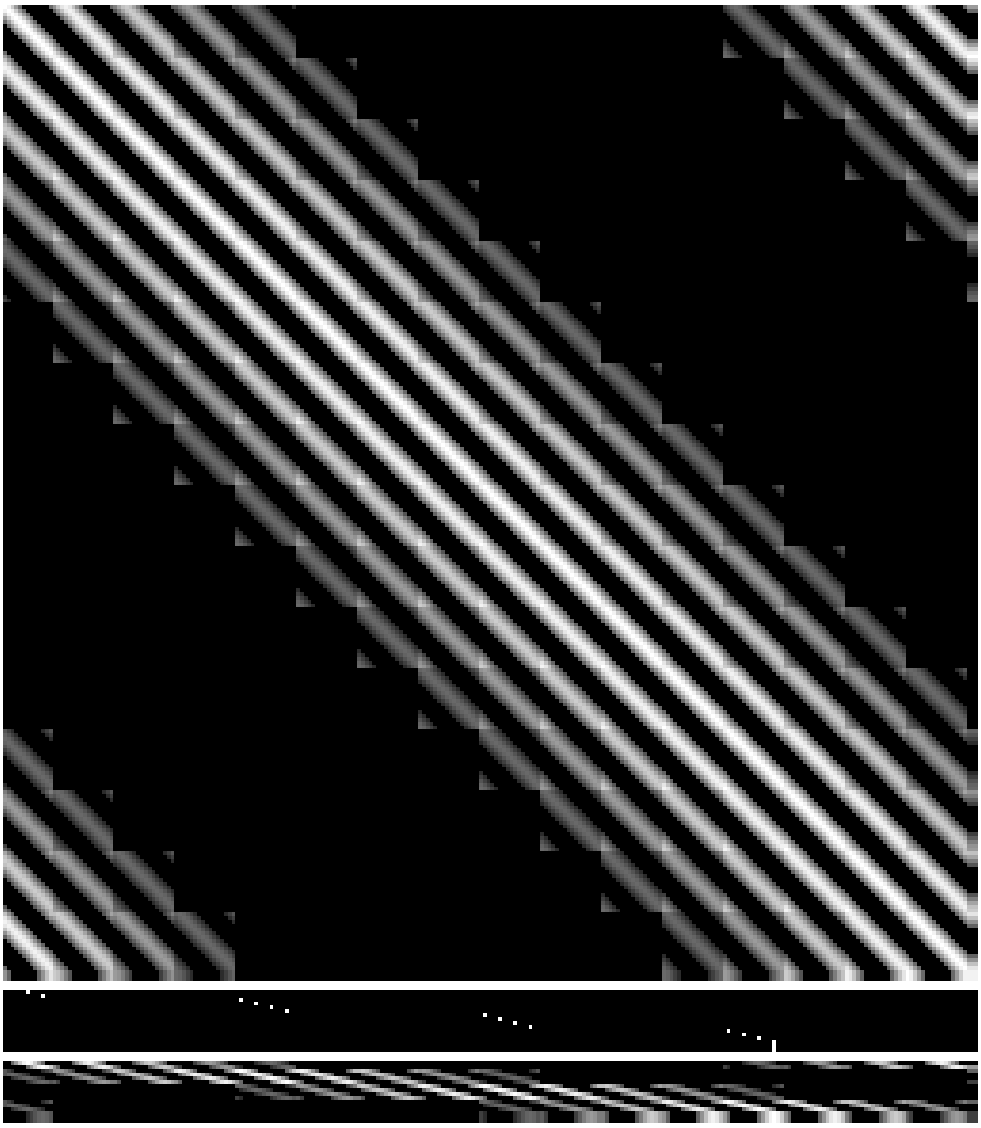


Figure 6: (top) convolution matrix H, (middle) downsample matrix S, (right) strided convolution matrix $A = SH$.

B Structure of matrix A in Section 4.1

The degradation matrix A in strided convolution can be decomposed as the product of H and S, i.e., $A = SH$, where H is a square matrix corresponding to 2-D convolution and S represents the regular 2-D downsampling. In general, the blurring matrix H is a block Toeplitz matrix with Toeplitz blocks. If the convolution is implemented with periodic boundary conditions, i.e., the pixels out of an image is padded with periodic extension of itself, the matrix H is a block circulant matrix with circulant blocks (BCCB). Note that for 1-D case, the matrix B reduces to a circulant matrix. For illustration purposes, an example of matrix B for a 1-D case is given as below.

$$H = \begin{bmatrix}
0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \\
0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.3 \\
0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5
\end{bmatrix}$$

An example of matrix B for 2-D convolution of a 9×9 kernel with a 16×16 image is given in the top of Fig. 6. Clearly, in this huge matrix, a circulant structure is present in the block scale as well as within each block, which clearly demonstrates the self-similar pattern of BCCB matrix.

The downsampling matrix S corresponds to downsampling the original signal and its transpose S^T interpolates the decimated signal with zeros. Similarly, a 1-D example of downsampling matrix is shown in (18) for an illustrative purpose. An example of matrix S for downsampling a 16×16 image to the size of 4×4, i.e., $S \in \mathbb{R}^{16 \times 256}$, is displayed in the middle of Fig. 6. The resulting degradation matrix A, which is the product of S and H is shown in the bottom of Fig. 6.

$$S = \left[\begin{array}{cccc|cccc|cccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right] \tag{18}$$

C More experimental results

C.1 Motion deblurring

The motion blurring kernel is shown in Fig. 7. More results of motion deblurring on held-out testing data for CUB dataset are displayed in Fig. 8, 9.

C.2 Super-resolution

More results of super-resolution on held-out testing data for CUB dataset are displayed in Fig. 10, 11.

C.3 Joint super-resolution and colorization

More results of joint super-resolution and colorization on held-out testing data for CelebA dataset are displayed in Fig. 12.

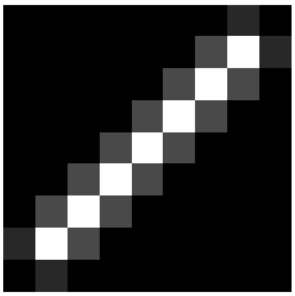
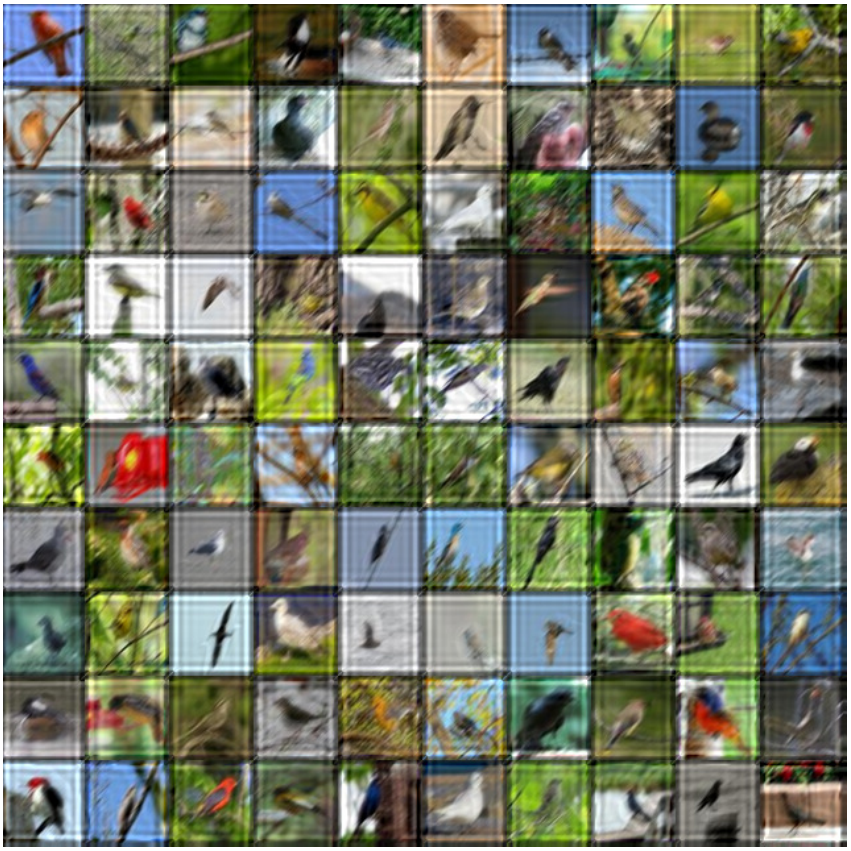


Figure 7: 9×9 motion blurring kernel.

(a) ground-truth image for motion deblurring task

(b) motion blurred images

Figure 8: motion blurring



(a) results using Wiener filter with the best performance by tuning regularization parameter.

(b) deblurred results using Inf-ADMM-ADNN

Figure 9: motion deblurring

(a) LR images

(b) bicubic interpolations

Figure 10: super-resolution

(a) super-resolved ($\times 4$) images using Inf-ADMM-ADNN

(b) HR groundtruth

Figure 11: super-resolution

(a) colorless images

(b) joint super-resolution ($\times 2$) and colorization using Inf-ADMM-ADNN

(c) HR groundtruth

Figure 12: joint super-resolution and colorization

D Networks Setting

D.1 Network for updating $\mathbf{x}$

For MNIST dataset, we did not use the pixel shuffling strategy, since each data point is a 28×28 grayscale image, which is relatively small. Alternatively, we used a standard denoising auto-encoder with architecture specifications in Table 1.

Table 1: Network Hyper-Parameters of DAE for MNIST

Input Dim	Layer	Output Dim
$28 \times 28 \times 1$	Conv$(5, 5, 1, 32)$-Stride$(2, 2)$-'SAME'-Relu	$14 \times 14 \times 32$
$14 \times 14 \times 32$	Conv$(5, 5, 32, 64)$-Stride$(2, 2)$-'SAME'-Relu	$7 \times 7 \times 64$
$7 \times 7 \times 64$	Conv$(5, 5, 64, 128)$-Stride$(2, 2)$-'VALID'-Relu	$2 \times 2 \times 128$
$2 \times 2 \times 128$	Conv$(2, 2, 128, 64)$-Stride$(1, 1)$-'VALID'-None	$1 \times 1 \times 64$
$1 \times 1 \times 64$	Conv_trans$(3, 3, 64, 128)$-Stride$(1, 1)$-'VALID'-Relu	$3 \times 3 \times 128$
$3 \times 3 \times 128$	Conv_trans$(5, 5, 128, 64)$-Stride$(1, 1)$-'VALID'-Relu	$7 \times 7 \times 64$
$7 \times 7 \times 64$	Conv_trans$(5, 5, 64, 32)$-Stride$(2, 2)$-'SAME'-Relu	$14 \times 14 \times 32$
$14 \times 14 \times 32$	Conv_trans$(5, 5, 32, 1)$-Stride$(2, 2)$-'SAME'-Sigmoid	$28 \times 28 \times 1$

For CUB-200-2011 dataset, we applied a periodical pixel shuffling layer to the input image of size $256 \times 256 \times 3$ with the output of size $64 \times 64 \times 48$. Note that we did not use any stride here since we keep the image scale in each layer identical. The architecture of the cPSDAE is given in Table 2. For CelebA dataset, we applied the periodical pixel shuffling layer to the input image of size $128 \times 128 \times 3$ with the output of size $32 \times 32 \times 48$, and the rest of setting is the same as CUB-200-2011 dataset, as shown in Table 3. In terms of the discriminator, we fed the pixel shuffled images. The architecture of the disriminator is the same as the one in DCGAN.

Table 2: Network hyper-parameters of cPSDAE for CUB-200-2011

Input Dim	Layer	Output Dim
$256 \times 256 \times 3$	periodical pixel shuffling	$64 \times 64 \times 48$
$64 \times 64 \times 48$	Conv$(4, 4, 48, 128)$-'SAME'-Batch_Norm-Relu	$64 \times 64 \times 128$
$64 \times 64 \times 128$	Conv$(4, 4, 128, 64)$-'SAME'-Batch_Norm-Relu	$64 \times 64 \times 64$
$64 \times 64 \times 64$	Conv$(4, 4, 64, 32)$-'SAME'-Batch_Norm-Relu	$64 \times 64 \times 32$
$64 \times 64 \times \{32, 3\}$	Concatenate in Channel	$64 \times 64 \times 35$
$64 \times 64 \times 35$	Conv$(4, 4, 35, 64)$-'SAME'-Batch_Norm-Relu	$64 \times 64 \times 64$
$64 \times 64 \times 64$	Conv$(4, 4, 64, 128)$-'SAME'-Batch_Norm-Relu	$64 \times 64 \times 128$
$64 \times 64 \times 128$	Conv$(4, 4, 128, 48)$-'SAME'-Batch_Norm-Relu	$64 \times 64 \times 48$
$64 \times 64 \times 48$	periodical pixel shuffling	$256 \times 256 \times 3$

Table 3: Network hyper-parameters of cPSDAE for CelebA

Input Dim	Layer	Output Dim
$64 \times 64 \times 3$	periodical pixel shuffling	$32 \times 32 \times 12$
$32 \times 32 \times 12$	Conv$(4, 4, 12, 128)$-'SAME'-Batch_Norm-Relu	$32 \times 32 \times 128$
$32 \times 32 \times 128$	Conv$(4, 4, 128, 64)$-'SAME'-Batch_Norm-Relu	$32 \times 32 \times 64$
$32 \times 32 \times 64$	Conv$(4, 4, 64, 32)$-'SAME'-Batch_Norm-Relu	$32 \times 32 \times 32$
$32 \times 32 \times \{32, 3\}$	Concatenate in Channel	$32 \times 32 \times 35$
$32 \times 32 \times 35$	Conv$(4, 4, 35, 64)$-'SAME'-Batch_Norm-Relu	$32 \times 32 \times 64$
$32 \times 32 \times 64$	Conv$(4, 4, 64, 128)$-'SAME'-Batch_Norm-Relu	$32 \times 32 \times 128$
$32 \times 32 \times 128$	Conv$(4, 4, 128, 12)$-'SAME'-Batch_Norm-Relu	$32 \times 32 \times 12$
$32 \times 32 \times 12$	periodical pixel shuffling	$64 \times 64 \times 3$

D.2 Network for updating $\mathbf{z}$

As described in Section 3.2, the neural network to update $\mathbf{z}$ was designed to have symmetric architecture. The details of this architecture is given in Table 4. Note that $W \times H$ represents the size of the width and height of measurement $\mathbf{y}$.

Table 4: Symmetric network hyper-parameters for updating $\mathbf{z}$

Input Dim	Layer	Output Dim
$H \times W \times 3$	Conv_trans(4,4,3,32, W_0)-'SAME'-Relu	$H \times W \times 32$
$H \times W \times 32$	Conv_trans(4,4,32,64, W_1)-'SAME'-Relu	$H \times W \times 64$
$H \times W \times 64$	Conv(4,4,3,32, W_1)-'SAME'-Relu	$H \times W \times 32$
$H \times W \times 32$	Conv(4,4,32,64, W_0)-'SAME'	$H \times W \times 3$

References

[1] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *arXiv preprint arXiv:1704.04058*, 2017.

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

[3] Joan Bruna, Pablo Sprechmann, and Yann LeCun. Super-resolution with deep convolutional sufficient statistics. *arXiv preprint arXiv:1511.05666*, 2015.

[4] JH Chang, Chun-Liang Li, Barnabas Poczos, BVK Kumar, and Aswin C Sankaranarayanan. One network to solve them all—solving linear inverse problems using deep projection models. *arXiv preprint arXiv:1703.09912*, 2017.

[5] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[6] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[7] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[8] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Process.*, 15(12):3736–3745, 2006.

[9] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proc. IEEE Int. Conf. Comp. Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[12] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Proc. European Conf. Comp. Vision (ECCV)*, pages 577–593. Springer, 2016.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[14] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proc. IEEE Int. Conf. Comp. Vision (ICCV)*, pages 4247–4255, 2015.

[15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. IEEE Int. Conf. Comp. Vision (ICCV)*, pages 3730–3738, 2015.

[16] Helmut Maurer and Jochem Zowe. First and second-order necessary and sufficient optimality conditions for infinite-dimensional programming problems. *Math. Progam.*, 16(1):98–110, 1979.

[17] Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

[18] Jo Schlemper, Jose Caballero, Joseph V Hajnal, Anthony Price, and Daniel Rueckert. A deep cascade of convolutional neural networks for MR image reconstruction. *arXiv preprint arXiv:1703.00555*, 2017.

[19] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proc. IEEE Int. Conf. Comp. Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[20] M. Simoes, J. Bioucas-Dias, L.B. Almeida, and J. Chanussot. A convex formulation for hyperspectral image superresolution via subspace-based regularization. *IEEE Trans. Geosci. Remote Sens.*, 53(6):3373–3388, Jun. 2015.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

[23] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.

[24] A.N. Tikhonov and V.I.A. Arsenin. *Solutions of ill-posed problems*. Scripta series in mathematics. Winston, 1977.

[25] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *Proc. IEEE Global Conf. Signal and Information Processing (GlobalSIP)*, pages 945–948. IEEE, 2013.

[26] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[27] Q. Wei, N. Dobigeon, and Jean-Yves Tourneret. Bayesian fusion of multi-band images. *IEEE J. Sel. Topics Signal Process.*, 9(6):1117–1127, Sept. 2015.