

InverseNet: Solving Inverse Problems with Splitting Networks

Kai Fan^{1*}, Qi Wei^{2*}, Wenlin Wang¹, Amit Chakraborty², Katherine Heller¹

¹ Duke University, Durham, North Carolina, USA

{kai.fan, kheller}@stat.duke.edu, wenlin.wang@duke.edu

² Siemens Corporate Technology, Princeton, New Jersey, USA

{qi.wei, amit.chakraborty}@siemens.com

Abstract

We propose a new method that uses deep learning techniques to solve the inverse problems. The inverse problem is cast in the form of learning an end-to-end mapping from observed data to the ground-truth. Inspired by the splitting strategy widely used in regularized iterative algorithm to tackle inverse problems, the mapping is decomposed into two networks, with one handling the inversion of the physical forward model associated with the data term and one handling the denoising of the output from the former network, i.e., the inverted version, associated with the prior/regularization term. The two networks are trained jointly to learn the end-to-end mapping, getting rid of a two-step training. The training is annealing as the intermediate variable between these two networks bridges the gap between the input (degraded version of output) and output and progressively approaches to the ground-truth. The proposed network, referred to as **InverseNet**, is flexible in the sense that most of the existing end-to-end network structure can be leveraged in the first network and most of the existing denoising network structure can be used in the second one. Extensive experiments on both synthetic data and real datasets on the tasks, motion deblurring, super-resolution, and colorization, demonstrate the efficiency and accuracy of the proposed method compared with other image processing algorithms.

1. Introduction

Over the past decades, inverse problems have been widely studied in image and signal processing and computer vision, e.g., denoising [15], deconvolution [2], super-resolution [48] and compressive sensing [18]. An inverse problem is resulted from the forward model which maps unknown signals, i.e., the ground-truth, to acquired/observed information about them, which we call data or measure-

ments. This forward problem, generally relies on a developed physical theory which reveals the link between the ground-truth and the measurements. Solving inverse problems involves learning the inverse mapping from the measurements to the ground-truth. Specifically, it recovers a signal from one or a small number of degraded or noisy measurements, which is usually ill-posed [42]. Mathematically, the goal is to reconstruct a high dimensional ground-truth $\mathbf{x} \in \mathbb{R}^n$ from a low dimensional measurement denoted as $\mathbf{y} \in \mathbb{R}^m$, which is reduced from \mathbf{x} by a forward model A such that $\mathbf{y} = A\mathbf{x}$. This forward model A is constructed to tie the observed data \mathbf{y} to a set of learned model parameters \mathbf{x} . For example, in compressive sensing, \mathbf{y} is a compressive measurement with random sampled regions and A is the measurement matrix, e.g., a random Gaussian matrix; in super-resolution, \mathbf{y} is a low-resolution image and the operation A downsamples high resolution images. The main difficulty of these underdetermined systems comes from the operator A which has a non-trivial null space leading to an infinite number of feasible solutions. Though most of the inverse problems are formulated directly to the setting of an optimization problem associated with the forward model [41], a number of learning-based algorithms have been proposed to solve inverse problems by learning a mapping from the measurement domain of \mathbf{y} to the signal space of \mathbf{x} , with the help of large datasets and neural nets [32] [14]. ¹

More recently, deep learning techniques have arisen as a promising framework and gained great popularity for providing state-of-the-art performance on applications include pattern analysis (unsupervised), classification (supervised), computer vision, image processing, etc [13]. Exploiting deep neural networks to solve inverse problems has been explored recently [14] [40] [1] [24]. In these works, inverse problems are viewed as a pattern mapping problem and most existing learning-based methods propose to learn an

¹These algorithms refer to directly learning optimum mappings from the observed data to their high-resolution correspondents and are different from learning from training datasets some specific priors to be incorporated in the regularized iterative algorithms [15] [47].

*The authors contributed equally to this work.

end-to-end mapping from y to x [37] [40]. By leveraging the powerful approximation ability of deep neural networks, these deep learning based data-driven methods have achieved state-of-the-art performance in many challenging inverse problems like super-resolution [5] [14] [40], image reconstruction [35], automatic colorization [27]. 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. A strong motivation to use neural networks stems from the universal approximation theorem [11], 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. In these recent works [5] [40] [27] [35], an end-to-end mapping from measurements y to ground-truth 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. However, despite their superior performance, these specifically-trained solvers are designed for specific inverse problems and usually cannot be reused to solve other problems without retraining the mapping function - even when the problems are similar. To improve the scope and generality of deep neural network models, more recently, in [7], 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 [4] [31]. This method involved training a deep convolutional auto-encoder network for low-level image modeling, which explicitly imposed regularization that spanned the subspace that the ground-truth images lived in. For the sub-problem that required 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. A similar approach to learn approximate ISTA (Iterative Shrinkage-Thresholding Algorithm) with neural networks was illustrated in [21]. A more flexible splitting strategy of training two reusable neural networks for the two sub-problems within ADMM framework leading to an inner-loop free update rule has been proposed in [17]. The two pre-trained deep neural networks are flexible and reusable in the sense that the trained network for the proximity operator can be used as a plug-and-play prior to regularize other inverse problems sharing similar statistical characteristics and the trained network for the inversion operator can be

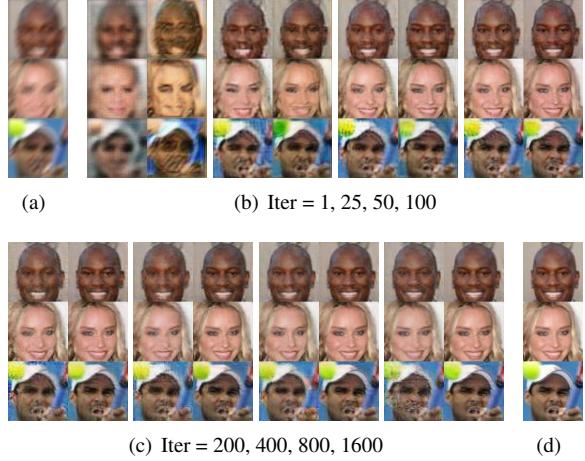


Figure 1. Illustration of Annealing Training. (a) the motion blurred photos. (b-c) the left column is z the results of U-Nets and the right column is \hat{x} the results of DAEs. (d) the ground-truth.

used to solve the same inverse problem for other datasets.

To leverage the advantages of both the end-to-end mapping and the splitting strategy, in this work, we propose to learn an end-to-end mapping consisting of two networks, with one handling the inversion associated with the forward physical model and the other one handling the denoising, respectively. A degraded signal, i.e., an observed data point is fed into the inverse network to output an intermediate update and then fed into the denoising network to refine the result. The intermediate update bridges the information gap between the input, i.e., the degraded signal and the output, i.e., the ground-truth. The training for the proposed InverseNet is annealing in the sense that the input of the denoising network, e.g., the denoising autoencoder, referred to as DAE (as an example explained later) or equivalently, the output of the inversion networks, e.g., the U-Nets (as an example explained later), progressively becomes better (closer to the ground-truth) following by a refined (better) result output by the denoising network, as displayed in Fig. 1. This training leverages both the data term by the U-Nets and a generative prior by the DAEs. More specifically (and heuristically), the inversion network tries to restore the information lost in the forward model, and the denoising network tries to refine the result with learned details, inspired by the two update steps of ADMM.

Contributions: We propose to learn an end-to-end mapping from the observed data to the ground-truth. Inspired by the splitting strategy widely used to solve inverse problems as discussed above, instead of learning one network to solve them all, we propose to decompose the end-to-end mapping into two parts with one handling the model inversion part and the other one handling the denoising part. There are several benefits to such structure: i) Any existing end-to-end learning algorithm can be regarded as the inversion part

and incorporated with any existing denoising network. ii) The two network can be trained jointly to learn the mapping to its best extent, getting rid of any two-step sub-optimal solution. iii) In the testing phase, only one feed-forward propagation is necessary to solve the inverse problem, getting rid of any further iterations.

2. Inverse Problems

As explained above, the forward model connects the low dimensional measurement $\mathbf{y} \in \mathbb{R}^m$ to high dimensional ground-truth $\mathbf{x} \in \mathbb{R}^n$ by a linear operator A as $\mathbf{y} = A\mathbf{x}$. The fact that $n \geq m$ makes the number of parameters to estimate larger than the number of available data points in hand. Since A is an underdetermined measurement matrix, this imposes an ill-posed problem for finding solution \mathbf{x} on a new observation \mathbf{y} . For instance, the matrix A is a strided Gaussian convolution and not invertible for super-resolution tasks in [37, 40]. To address this issue, computational solutions including approximate inference based on Markov chain Monte Carlo (MCMC) and optimization based on variable splitting under the ADMM framework, were proposed and applied to different kinds of priors, e.g., the empirical Gaussian prior [45, 48], the Total Variation prior [38], etc. The ADMM framework is popular due to its low computational complexity and recent success in solving large scale optimization problems. Mathematically, the optimization problem is formulated as

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

where the introduced auxiliary variable \mathbf{z} is forced to be equal to \mathbf{x} , and $\mathcal{R}(\mathbf{x})$ models the structure promoted by the prior/regularization. To leverage the benefits of ‘big data’, the regularization can be imposed in an empirical Bayesian way, by designing an implicit data dependent prior on \mathbf{x} , i.e., $\mathcal{R}(\mathbf{x}; \mathbf{y})$ for amortized inference [40]. The augmented Lagrangian for Eq. (1) by replacing $\mathcal{R}(\mathbf{x})$ with $\mathcal{R}(\mathbf{x}; \mathbf{y})$ is

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

where \mathbf{u} is the Lagrange multiplier and $\beta > 0$ is the penalty parameter. The conventional augmented Lagrange multiplier method that minimizes \mathcal{L} w.r.t. \mathbf{x} and \mathbf{z} simultaneously, is difficult and does not exploit the fact that the objective function is separable. To tackle this problem, ADMM decomposes the minimization into two subproblems, i.e., minimizations w.r.t. \mathbf{x} and \mathbf{z} , respectively. More specifically, the updates are as follows:

$$\mathbf{z}^{k+1} = \arg \min_{\mathbf{z}} \|\mathbf{y} - A\mathbf{z}\|^2 + \beta \|\mathbf{x}^{k+1} - \mathbf{z} + \frac{\mathbf{u}^k}{2\beta}\|^2 \quad (3)$$

$$\mathbf{x}^{k+1} = \arg \min_{\mathbf{x}} \beta \|\mathbf{x} - \mathbf{z}^k + \mathbf{u}^k / 2\beta\|^2 + \lambda \mathcal{R}(\mathbf{x}; \mathbf{y}) \quad (4)$$

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

For the priors \mathcal{R} of special forms, such as $\|\mathbf{x}\|_1$, a closed-form solution for Eq. (4), i.e., a soft thresholding solution is easily obtained. On the contrary, for some more sophisticated regularizations, e.g., a patch based prior [15, 47], solving Eq. (4) is nontrivial, and may require iterative methods. To solve Eq. (3), a matrix inversion is inevitable, for which gradient descent (GD) method is usually applied to update \mathbf{z} [7]. Thus, solving Eq. (4) and (3) is in general cumbersome as inner loops are required to solve these two sub-minimization problems. More specifically, most of computational complexity comes from the proximity operator due to its intractability and the matrix inversion which is not easy diagonalized.

Motivation to introduce ADMM: The proposed network structure is inspired by the two iterative steps in ADMM updates to solve an inverse problem. More specially, the inverse network imitates the process of solving Eq. (3) and the denoising network imitates an algorithm to solve Eq. (4). In this work, the inverse network exploits the U-Nets structure [33] and the denoising network uses the DAE [43], which will be elaborated in the following section.

3. Approach

The proposed method decomposes the end-to-end mapping from data to ground-truth into two mappings, one corresponding to inversion of the physical model and the other corresponding to the regularized denoising. From a Bayesian perspective, the first mapping handles the likelihood associated with data term and the second mapping handles the prior term. As shown in Fig. 2, the observed data \mathbf{y} is fed into the first network to get an approximation \mathbf{z} of the ground-truth. Note that \mathbf{z} is of the same size as \mathbf{x} and can be regarded as a noisy version of it. The noisy one \mathbf{z} is then fed into the second network to get refined to the clear version $\hat{\mathbf{x}}$. This two-network structure is also echoed from the recent popular refining technique used in [36, 8]. More details about these two networks will be elaborated in the following sections.

3.1. Inversion network mapping \mathbf{y} to \mathbf{z}

The inversion network tries to learn the inversion of the degradation which maps the ground-truth \mathbf{x} to the observed data \mathbf{y} . Note that while it may be straightforward to write down the closed-form solution for sub-problem (3) w.r.t. \mathbf{z} , explicitly computing this solution is nontrivial due to the infeasibility of inverting a big matrix. Similar to the strategy in [17, 40], we design a deep convolutional neural network to learn the inversion. More specifically, we have used a recently developed network structure referred to as U-Nets [33], which was originally developed for medical image segmentation. The U-Net architecture allows low-level information to shortcut by filtering concatenation across different layers, differing from the element-wise addition in

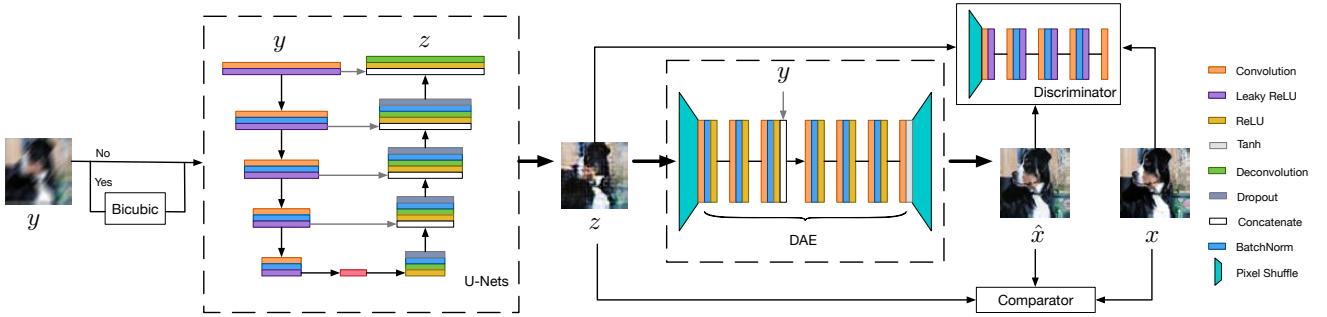


Figure 2. The pipeline architecture of *InverseNet*. Initially, the degraded input y is pre-processed by bicubic resizing depending on the task, e.g., for super-resolution. Thus, the input y of U-Nets has the same size as the output z . Next, the output z is fed into the DAEs with two pixel shuffling blocks. In the end, the intermediate result z and final estimation \hat{x} are both fed into the discriminator and *comparator* with the ground-truth.

ResNet [22]. Extensive experiments have demonstrated the effectiveness of U-Nets to learn the complex mapping between high-dimensional data, such as images [9] [24].

In general, U-Nets require the input and output to have the same size, for the height and width at least. However, in many inverse problems, for example, super-resolution or compressive sensing, only a low dimensional signal is available as the input. In such cases, we initially apply the bicubic interpolation to obtain the same size input which is suitable for the U-Nets. As a summary, the architecture of our U-Nets is shown in the left dashed box in Fig. 2.

3.2. Denoising network mapping z to \hat{x}

The denoising network plays a role to learn from the dataset a signal prior that can deal with any inverse problems. In optimization algorithms for solving inverse problems, signal priors are usually cast in the form of a denoiser, more specifically, a proximity operator [10]. From the geometrical perspective, the proximity operator projects a noisy data point into the feasible sets spanned by the signal prior. Thus, any existing denoising network can be exploited in the proposed framework. In our work, we propose to a specially designed denoising auto-encoders with pixel shuffling (or sub-pixel) trick [37]. Similar to the inversion net, pixel shuffling convolutional network is designed to deal with inputs and outputs of the same size by periodically reordering the pixels in each channel mapping a high resolution image to the scale as the same as the low dimensional image. The resulting denoising auto-encoders does not have a bottle-neck shape structure as each layer shares the same filter size. This allows us to transform z into a tensor that has the same size as y , and concatenate this tensor with y if desired in the regularization of amortized inference. The detailed architecture of the proposed DAE shown in Fig. 2

3.3. Practical Adversarial Training

From either the perspective of the model structure or the ADMM update rules, z and \hat{x} are both approximates of the ground-truth x . Inspired by the idea in autoencoding beyond pixels using a learned similarity metric [26], standard negative likelihood loss (equivalent to reconstruction loss) incorporating additional generative adversarial loss can practically improve the quality of generated samples. In our work, we also force the outputs of two designed networks to share one discriminator used as a binary classifier, inducing the standard minimax loss [20] in our model, i.e.,

$$\mathcal{L}_D = \mathcal{L}_{GAN:D}(z) + \mathcal{L}_{GAN:D}(\hat{x}) \quad (6)$$

$$\mathcal{L}_G = \mathcal{L}_{GAN:G}(\cdot) + \lambda_l \mathcal{L}_{likelihood}(\cdot), \quad (7)$$

where $\mathcal{L}_{GAN:D}$ is the negative likelihood loss with respect to classification problem, $\mathcal{L}_{GAN:G}$ is the non-saturated generative loss, and “.” in Eq. (7) should be substituted by z or \hat{x} to introduce two different generative loss functions.

Besides using the classifier as the discriminator, many works [34, 3] argued that a pre-trained regression network to match the high level features between the real images and the fake ones can be significantly better than the discriminative loss in practice and theoretically is more robust from the viewpoint of distribution matching. Therefore, a regression network referred to as *comparator* is shared by the outputs of two proposed nets as well. Since the transfer learning is well established in the image domain, such as style learning with feature matching [19] in VGG [39], we prefer to use the pre-trained VGG or AlexNet [25] as the *comparator*. In this case, to leverage both the adversarial training and the *comparator*, we do not modify the discriminator loss in Eq. (6), but add an extra feature matching loss to Eq. (7) as

$$\mathcal{L}_G = \mathcal{L}_{GAN:G}(\cdot) + \lambda_l \mathcal{L}_{likelihood}(\cdot) + \lambda_f \mathcal{L}_{feature}(\cdot). \quad (8)$$

Analogous to the traditional optimization for generative adversarial loss, \mathcal{L}_D and \mathcal{L}_G are iteratively minimized with respect to the parameters of the discriminator, U-Nets, and DAEs.

3.4. Underlying Annealing Training

The proposed adversarial training leads to an underlying annealing training for DAEs where the input (the output of U-Nets) at early training stage is extremely noisy, and then progressively approaches to a clearer version with further training. This incremental tuning nature in adversarial training allows the training to imitate the inner loop of ADMM updates, where the difference \mathbf{u} in Eq. (5) gradually becomes constant and negligible. This is the main reason why we call our model as *InverseNet*. Unlike the traditional DAEs, in which the input data is contaminated by the pre-defined noise, e.g., Gaussian noise with fixed variance, the output of U-Nets or the input of DAEs in the proposed network is contaminated by non-stationary and time-variant noise without an explicit analytic form. If we remove the DAEs part from our model, we found that the performance gradually becomes worse during the training, which can be explained by the instability of adversarial training [3]. The existence of the DAEs make the network be able to purify the complicated noisy output from the inversion network, leading to a much stable and efficient training.

Generality 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? Compared with the reusable matrix inversion neural networks learned with pure noise proposed in [40] [17], this inversion network is less flexible due to the joint training of the inversion and denoising network. However, because the learning is based on problem formulation, though not fully reusable, the well-trained neural networks have better generalization ability compared with the other end-to-end learning networks. For example, the well-trained networks for image motion deblurring on PASCAL VOC dataset [16] can be directly applied to the same task on ImageNet [12] with no necessity of fine-tuning, if the degrade operator A remains the same, since these two datasets are both natural images. However, if a significant domain changes to the dataset, such as MRI, all networks-reusable algorithms will fail and require re-training as well as our approach.

4. Experiments and Results

In this section, we provide experimental results and analysis on the proposed *InverseNet* for solving inverse problems. The datasets used in our experiments include Caltech-UCSD Birds-200-2011 (CUB) [44], Large-scale CelebFaces Attributes (CelebA) [29], PASCAL VOC and Im-

PSNR	CUB	CelebA	ImageNet
Wiener filter (baseline)	22.42	20.92	20.44
Robust motion deblur [46]	25.03	25.06	23.37
Neural motion deblur [6]	25.73	25.76	24.74
Pix2Pix [23] w. comparator	23.67	23.59	22.05
Ours	28.39	34.02	28.87
SSIM	CUB	CelebA	ImageNet
Wiener filter (baseline)	0.6572	0.7020	0.6357
Robust motion deblur [46]	0.7459	0.8052	0.7283
Neural motion deblur [6]	0.8853	0.9649	0.9074
Pix2Pix [23] w. comparator	0.7554	0.8553	0.7335
Ours	0.9421	0.9738	0.9446

Table 1. PSNR and SSIM Comparison on motion deblurring

geNet [25], and the learning tasks considered include motion deblurring, $\times 4$ super-resolution, and joint $\times 2$ super-resolution and colorization. Note that the degraded operator A is unknown during training, and is only used to generate the measurement and ground-truth paired data. Our code will be available on the repository <https://github.com/>

4.1. Motion Deblurring

The target of motion deblurring is to recover a sharp non-blurred image from a single motion-blurred image, which has been a fundamental problem in computational imaging [46]. The motion deblurring is challenging as both the blurring kernel and the latent non-blurred images are unknown in most cases, leading to an ill-posed inverse problem. Note that the size of an image after motion deblurring remains the same. Thus, any resizing of the input images, e.g., bicubic interpolation or pixel shuffling for the initialization of our approach is unnecessary and can be simply removed from the pipeline. We trained our model on all datasets with size 64×64 images without any other pre-processing.

Qualitative Results The proposed *InverseNet* is compared with the baseline optimization methods by Wiener filtering, the robust non-blind motion deblurring [46], and another deep learning approach method neural motion deblurring [6] visually in Fig. 3. Note that in [46], the algorithm for a single image motion deblurring requires to feed the blurring kernel, while other methods including ours are all blind recoveries. As shown in Fig. 3, the traditional single image processing algorithms including Wiener filtering and robust motion deblurring suffered heavily from ring artifacts while the neural motion deblurring and the proposed *InverseNet* gave much better and high-quality recoveries. The ring artifact can be explained by the fact that the chosen convolutional kernel to blur images has larger variance and is more challenging.

Quantitative Results The PSNR and SSIM were calcu-

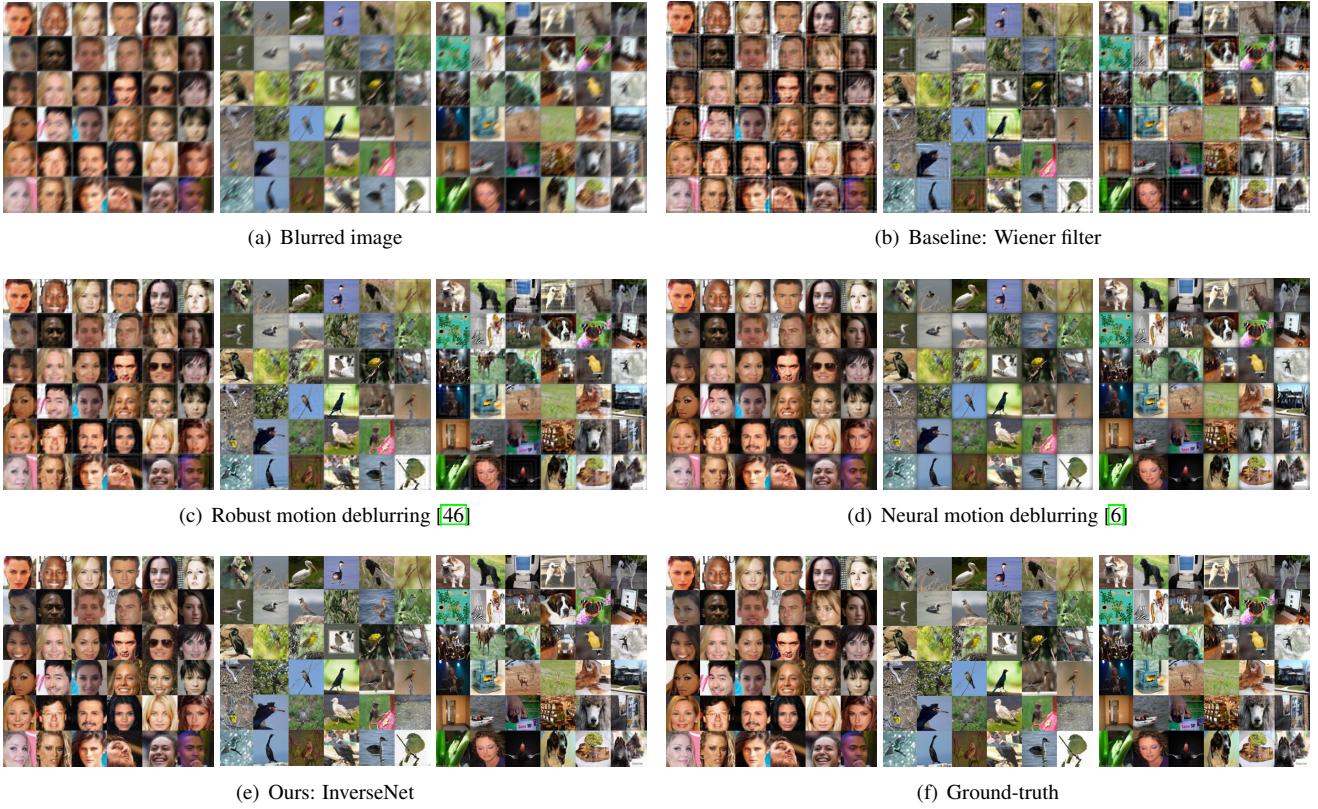


Figure 3. Example deblurred results from different methods and three different datasets (zooming in can see details).

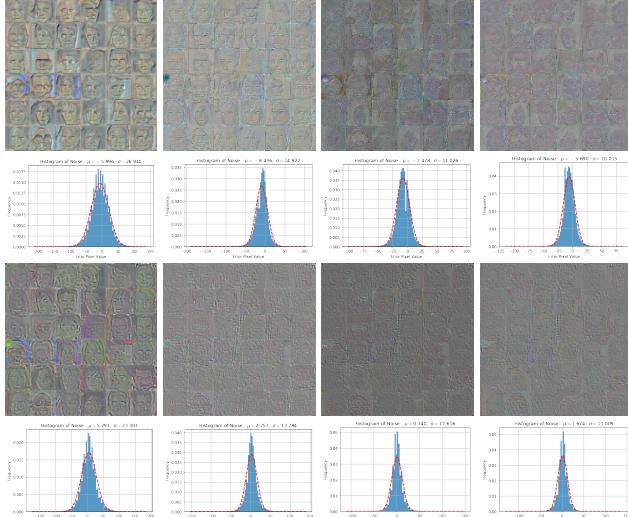


Figure 4. Annealing training at iteration 1, 25, 50, 100 on celebA dataset. 1st and 3rd Rows: the error images between \mathbf{z} and model estimation $\hat{\mathbf{x}}$, and the ones between \mathbf{z} and ground-truth \mathbf{x} . 2nd and 4th Rows: the annealing empirical distribution of pixel level noise w.r.t $\bar{\mathbf{x}} - \mathbf{z}$ and $\mathbf{x} - \mathbf{z}$.

lated on the same testing datasets across all compared algorithms and summarized in Table 1. Clearly, the proposed



Figure 5. **Transfer Learning:** Testing on the same images from Fig. 3(a) with well-trained model on VOC dataset.

InverseNet outperformed the other methods with a significant improvement, which is in consistence with the qualitative results in Fig. 3.

Transfer Training We also checked to what extent the well-trained model could be adapted to other datasets. Specifically, we trained the *InverseNet* on PASCAL VOC and tested it on CUB, celebA and ImageNet, with the results shown in Fig. 5. The quantitative results is shown in Table 2. As we argued before, the similar image domain between ImageNet and VOC leads the PSNR does not decrease much, i.e., from 28.87 to 28.52, compared with using the *InverseNet* trained on ImageNet itself. This demonstrated the generality of the learned *InverseNet*.

Annealing Training We also validated the importance of annealing training on the datasets (more intensive results



Figure 6. **Ring Effect Remover** on ImageNet for $64 \times 64 \rightarrow 256 \times 256$. 1st Row: LR images; 2nd Row: The bicubic interpolation results (having ring artifacts); 3rd Row: Results by *InverseNet*; 4th Row: HR ground-truth.

Metric	CUB	CelebA	ImageNet
PSNR	25.65	27.90	28.52
SSIM	0.9105	0.9373	0.9303

Table 2. Performance with well-trained model on VOC

are available in the supplementary materials). If we only trained the model with U-Nets with a discriminator (this will recover the model *pix2pix* [23]) and a *comparator*, the training was usually very unstable and we have to do cherry-pick from results in every iteration. However, with the help of refining network DAEs, the performance was boosted up significantly, reflected by both the quantitative results summarized in Table 1 and the qualitative results displayed in Fig. 1. Additionally, we also visualized the residual images between the output of U-Nets and the final output of DAEs or the ground-truth in Fig. 4. The pixel level noise approximately followed a Gaussian distribution at each iteration, but with varying mean and variance, leading to a non-stationary and time variant noise. It was observed that the mean and variance gradually became smaller and stable

at the final training stage. This potentially enables the DAEs to depress noise with variances of different scales added to the input, which is different from traditional DAEs, and allows more robust training.

4.2. Super-Resolution

Super-resolution is another popular inverse problem in image processing, and it aims at the restoration of high frequencies to enrich details in an image based on a set of prior examples with low resolution (LR) and corresponding high resolution (HR) images. The degraded operator A can source from quantization error, limitation of the sensor from the capturing camera, the presence of blurriness and the use of downsampling operators to reduce the image resolution for storage purposes. It is well known that super-resolution is ill-posed, since for each LR image the space of corresponding HR images can be very large.

In this experiment, we first synthesized a challenge $\times 4$ downsampling operator A , which was a channel-wise strided convolution and may result in the *ring artifact* on low resolution images (see detailed form in supplementary materials) and trained the super-resolution model on two

datasets, CUB and ImageNet. Our task is trying to super-resolve images from 64×64 to 256×256 . In addition, we also compared with SRGAN [28] in supplementary materials.

Ring Artifact Remover The difficulty of the super-resolution is highly affected by the shape and scale of the convolution kernel in the degradation model. Convolving an image with a wider and flatter convolution kernel leads to more aliasing of high frequency information in spectral domain thus leads to a more challenging problem. Usually, ring artifact can be resulted from this heavy aliasing on the spectral domain. On the contrary, a narrower and sharper convolutional kernel results to less information loss thus is less challenging. In this task, we used a flat kernel of size 7×7 which can cause heavy ring artifact. Fig. 6 shows the super-resolution results on held-out testing ImageNet dataset. In this case, the bicubic interpolation failed to remove the ring effects produced during downsampling. However, our end-to-end trained *InverseNet* can well tackle this problem by greatly depressing the ring effect.

4.3. Jointly Super-Resolution and Colorization

A more challenging task is to enhance both spectral and spatial resolutions from one single band image. More specifically, the enhancement is a combination of super-resolution and colorization (hallucinating a plausible color version of a colorless image), which can be considered as the compound of two degraded operators. Thus, our single channel colorless low resolution image was obtained by convolving a kernel A of size $9 \times 9 \times 3 \times 1$ and downsampling with a stride 2 in both horizontal and vertical directions. Note that this is different from and more challenging than the conventional multi-band image fusion, e.g., fusing a multispectral image and a high spatial panchromatic image (also referred to as pansharpening [30]), in the sense that only one low-spatial low-spectral resolution image is available. Additionally, our inverse task is also different from the traditional colorization in the CIE Lab color space. In this section, we have tested the ability to perform joint $\times 2$ super-resolution and colorization from one single colorless LR image on the dataset celebA and CUB. The celebA $32 \times 32 \rightarrow 64 \times 64$ mainly includes faces images with less color variance, so it is an easier dataset compared with CUB wild bird images $64 \times 64 \rightarrow 128 \times 128$. The results are displayed in Fig. 7. Visually, the joint super-resolved and colorized images are very similar with their ground-truth with variances for some details. It is especially interesting to observe that the restored high-resolution images looked more natural than the ground-truth of some ‘outlier’ images. This results from the fact that the model was trained from massive images in which most look normal. For example, the person in the third column had purple hairs which rarely

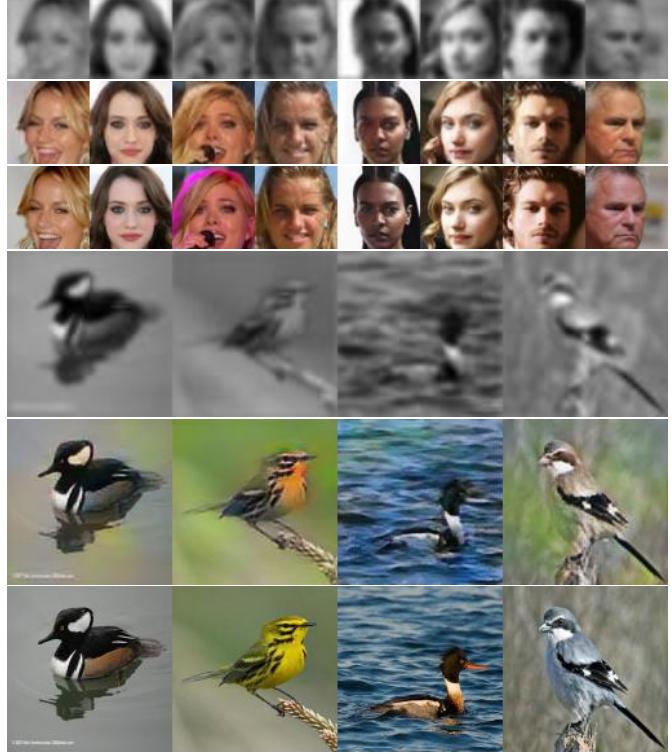


Figure 7. Joint Super-resolution and colorization for CelebA and CUB datasets. (top): colorless LR images; (middle): recovery by *InverseNet*; bottom: full color HR images.

appeared in the dataset. Feeding its degraded version, i.e., a blurred colorless image to the trained *InverseNet* gave a face with most probable, popular and thus ‘natural’ brown hairs as shown in the second row. More results are available in the supplementary materials.

5. Conclusion

In this paper we proposed the *InverseNet* to solve inverse problems by end-to-end mapping. To take the advantage of the efficiency (in testing phase) of end-to-end learning and the flexibility brought by splitting strategy simultaneously, the mapping was decomposed into two neural networks, i.e., the inversion network and the denoising one. The former one was designed to learn the inversion of the physical forward model associated with the data term and the latter one was to learn a proximity operator or a projection onto the ground-truth signal space associated with the prior term. The two pre-trained deep neural networks were trained jointly using prepared data pairs $(\mathbf{x}_i, \mathbf{y}_i)$, getting rid of any two-step separate training. Experiments and analysis on various datasets demonstrated the efficiency and accuracy of the proposed method. In future work we hope to further extend the proposed method to tackle the ‘learn to learn’ problem.

References

- [1] J. Adler and O. Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *arXiv preprint arXiv:1704.04058*, 2017.
- [2] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Trans. Image Process.*, 19(9):2345–2356, 2010.
- [3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.
- [4] 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*, 3(1):1–122, 2011.
- [5] J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. *arXiv preprint arXiv:1511.05666*, 2015.
- [6] A. Chakrabarti. A neural approach to blind motion deblurring. In *European Conference on Computer Vision*, pages 221–235. Springer, 2016.
- [7] J. Chang, C.-L. Li, B. Poczos, B. Kumar, and A. C. Sankaranarayanan. One network to solve them all—solving linear inverse problems using deep projection models. *arXiv preprint arXiv:1703.09912*, 2017.
- [8] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. *arXiv preprint arXiv:1707.09405*, 2017.
- [9] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.
- [10] P. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer Optimization and Its Applications, pages 185–212. Springer New York, 2011.
- [11] B. C. Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.
- [12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [13] L. Deng, D. Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.
- [14] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, Feb 2016.
- [15] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Process.*, 15(12):3736–3745, 2006.
- [16] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [17] K. Fan*, Q. Wei*, L. Carin, and K. Heller. An inner-loop free solution to inverse problems using deep neural networks. In *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec. 2017.
- [18] M. A. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE J. Sel. Topics Signal Process.*, 1(4):586–597, 2007.
- [19] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.
- [20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- [21] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 399–406, 2010.
- [22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Int. Conf. Comp. Vision and Pattern Recognition (CVPR)*, June 2016.
- [23] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.
- [24] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Trans. Image Process.*, 26(9):4509–4522, Sept 2017.
- [25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [26] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 1558–1566. JMLR. org, 2016.
- [27] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *Proc. European Conf. Comp. Vision (ECCV)*, pages 577–593. Springer, 2016.
- [28] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.
- [29] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proc. IEEE Int. Conf. Comp. Vision (ICCV)*, pages 3730–3738, 2015.
- [30] L. Loncan, L. B. Almeida, J. M. Bioucas-Dias, X. Briottet, J. Chanussot, N. Dobigeon, S. Fabre, W. Liao, G. Licciardi, M. Simoes, J.-Y. Tourneret, M. Veganzones, G. Vivone, Q. Wei, and N. Yokoya. Hyperspectral pansharpening: a review. *IEEE Geosci. Remote Sens. Mag.*, 3(3):27–46, Sept. 2015.

- [31] S. Lu, M. Hong, and Z. Wang. A nonconvex splitting method for symmetric nonnegative matrix factorization: Convergence analysis and optimality. *IEEE Transactions on Signal Processing*, 65(12):3120–3135, June 2017.
- [32] Y. Lu, M. Inamura, and M. del Carmen Valdés. Super-resolution of the undersampled and subpixel shifted image sequence by a neural network. *International Journal of Imaging Systems and Technology*, 14(1):8–15, 2004.
- [33] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.
- [34] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.
- [35] J. Schlemper, J. Caballero, J. V. Hajnal, A. Price, and D. Rueckert. A deep cascade of convolutional neural networks for MR image reconstruction. *arXiv preprint arXiv:1703.00555*, 2017.
- [36] S. Shankar, D. Robertson, Y. Ioannou, A. Criminisi, and R. Cipolla. Refining architectures of deep convolutional neural networks. In *Proc. IEEE Int. Conf. Comp. Vision and Pattern Recognition (CVPR)*, pages 2212–2220, Las Vegas, NV, USA, 2016.
- [37] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. 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.
- [38] M. Simoes, J. Bioucas-Dias, L. 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.
- [39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [40] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised MAP inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.
- [41] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.
- [42] A. Tikhonov and V. Arsenin. *Solutions of ill-posed problems*. Scripta series in mathematics. Winston, 1977.
- [43] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 1096–1103. ACM, 2008.
- [44] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.
- [45] Q. Wei, N. Dobigeon, and J.-Y. Tourneret. Bayesian fusion of multi-band images. *IEEE J. Sel. Topics Signal Process.*, 9(6):1117–1127, Sept. 2015.
- [46] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *European Conference on Computer Vision*, pages 157–170. Springer, 2010.
- [47] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Trans. Image Process.*, 19(11):2861–2873, 2010.
- [48] N. Zhao, Q. Wei, A. Basarab, N. Dobigeon, D. Kouamé, and J. Y. Tourneret. Fast single image super-resolution using a new analytical solution for $\ell_2 - \ell_2$ problems. *IEEE Trans. Image Process.*, 25(8):3683–3697, Aug. 2016.

Appendices

A. Training details

A.1. Adversarial training algorithm

We define a soft GAN loss with the following form

$$\mathcal{L}_{GAN}^{soft}(\mathbf{x}, l) = -l \cdot \log D(\mathbf{x}) - (1 - l) \cdot \log(1 - D(\mathbf{x})) \quad (9)$$

where $l \in [0, 1]$ is the soft label for data \mathbf{x} . The explicit forms of the Discriminator (D) loss and the generator (G) loss are

$$\mathcal{L}_D = \mathcal{L}_{GAN}^{soft}(\mathbf{x}, 0.99) + \mathcal{L}_{GAN}^{soft}(\mathbf{z}, 0.01) + \mathcal{L}_{GAN}^{soft}(\hat{\mathbf{x}}, 0.01) \quad (10)$$

$$\mathcal{L}_G(\mathbf{z}) = \mathcal{L}_{GAN}^{soft}(\mathbf{z}, 0.99) + \lambda_r \cdot \|\mathbf{z} - \mathbf{x}\|_2^2 + \lambda_f \cdot \|C(\mathbf{z}) - C(\mathbf{x})\|_2^2 \quad (11)$$

$$\mathcal{L}_G(\hat{\mathbf{x}}) = \mathcal{L}_{GAN}^{soft}(\hat{\mathbf{x}}, 0.99) + \lambda_r \cdot \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 + \lambda_f \cdot \|C(\hat{\mathbf{x}}) - C(\mathbf{x})\|_2^2 \quad (12)$$

where \mathbf{z} is the output of U-Nets, $\hat{\mathbf{x}}$ is the output of DAE and $C(\cdot)$ is the comparator output. In our experiment, we used FC6 layer of the AlexNet. The tunable hyper-parameters $\lambda_r = \lambda_f = 0.5$ were used in our experiment. Experimentally, this soft loss can alleviate the vulnerability of neural networks to adversarial examples.

Instead of pre-training the generator as in SRGAN, we jointly trained the whole model, i.e., the generator and discriminator simultaneously. The detailed optimization updates for adversarial training are summarized in Algorithm 1. We chose $K = 1$ in all experiments, and Adam [?] optimizer was applied in the gradient descent for all parameter updates. For Adam, the learning rate is 10^{-4} , and $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. In addition, the gradient is clipped by norm 5 for each net. The batch size of each iteration is 36 for all experiments. For batch normalization, momentum is 0.9 and $\epsilon = 10^{-5}$.

Algorithm 1 Adversarial training for InverseNet

```

for  $t = 1, 2, \dots$  do
    Randomly get batch data pairs  $(\mathbf{x}, \mathbf{y})$ ;
    for  $k = 1, 2, \dots, K$  do
        Update parameter of Discriminator with gradient  $\nabla \mathcal{L}_D$ ;
    end for
    Update parameter of U-Nets with gradient  $\nabla \mathcal{L}_G(\mathbf{z})$ ;
    Update parameter of DAEs with gradient  $\nabla \mathcal{L}_G(\hat{\mathbf{x}})$ ;
end for

```

A.2. Model structure

In Table 3, we describe the model structure for U-Nets with input of size 256×256 . The structure of DAEs is summarized in Table 4). Beside of the pixel shuffling layer, all the other layers can share the same structure as the sizes, i.e., the height and width of an input image and an output image are the same.

B. More results for motion deblurring

We tested our motion deblurred model on another natural image as shown in Fig. 8. The model was trained on 128×128 patches from ImageNet, and tested on one Wonder Woman poster image with size 256×256 and 512×512 . We also compared it with the blind neural motion deblurring algorithm [6] on 256×256 ². As shown in Fig. 8, the deblurred image using InverseNet is much more clear and closer to the original image than the neural deblurred one. In the 512×512 size deblurring, visually, the restored image by InverseNet is almost exactly the same with the original one as displayed in Fig. 9.

C. More results for ring artifact remover super-resolution

We include the results of SRGAN [28] for comparison as shown in Fig. 10. For SRGAN, except downgraded kernel, we follow the training instruction in the paper by first pretraining the generator for 10^6 iterations and then fine-tuning the model

²We used the codes offered by the authors in <https://github.com/ayanc/nndeblur>.

Table 3. Network hyper-parameters of U-Nets

Layer Name	Dimension	Layer Operations
data	$256 \times 256 \times 3$	Conv(4, 4, 3, 16)-‘SAME’
e1	$128 \times 128 \times 16$	Leaky Relu-Conv(4, 4, 16, 32)-‘SAME’-Batch_Norm
e2	$64 \times 64 \times 32$	Leaky Relu-Conv(4, 4, 32, 64)-‘SAME’-Batch_Norm
e3	$32 \times 32 \times 64$	Leaky Relu-Conv(4, 4, 64, 128)-‘SAME’-Batch_Norm
e4	$16 \times 16 \times 128$	Leaky Relu-Conv(4, 4, 128, 128)-‘SAME’-Batch_Norm
e5	$8 \times 8 \times 128$	Leaky Relu-Conv(4, 4, 128, 128)-‘SAME’-Batch_Norm
e6	$4 \times 4 \times 128$	Leaky Relu-Conv(4, 4, 128, 128)-‘SAME’-Batch_Norm
e7	$2 \times 2 \times 128$	Leaky Relu-Conv(4, 4, 128, 128)-‘SAME’-Batch_Norm
e8	$1 \times 1 \times 128$	Relu-Conv_Trans(4, 4, 128, 128)-‘SAME’-Batch_Norm-Cancat(e7)
d1	$2 \times 2 \times 256$	Relu-Conv_Trans(4, 4, 256, 128)-‘SAME’-Batch_Norm-Cancat(e6)
d2	$4 \times 4 \times 256$	Relu-Conv_Trans(4, 4, 256, 128)-‘SAME’-Batch_Norm-Cancat(e5)
d3	$8 \times 8 \times 256$	Relu-Conv_Trans(4, 4, 256, 128)-‘SAME’-Batch_Norm-Cancat(e4)
d4	$16 \times 16 \times 256$	Relu-Conv_Trans(4, 4, 256, 64)-‘SAME’-Batch_Norm-Cancat(e3)
d5	$32 \times 32 \times 128$	Relu-Conv_Trans(4, 4, 128, 32)-‘SAME’-Batch_Norm-Cancat(e2)
d6	$64 \times 64 \times 64$	Relu-Conv_Trans(4, 4, 64, 16)-‘SAME’-Batch_Norm-Cancat(e1)
d7	$128 \times 128 \times 32$	Relu-Conv_Trans(4, 4, 32, 3)-‘SAME’-Tanh
d8	$256 \times 256 \times 3$	

Table 4. Network hyper-parameters of DAEs

Input Dimension	Layer	Output Dimension
$256 \times 256 \times 3$	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	$256 \times 256 \times 3$

Figure 8. Motion deblurring for a 256×256 image with model trained on 128×128 images.

with discriminator included for another 10^6 iterations. Since the author did not release the codes, this implementation was based on our own and we tried our best to fine tune these parameters to the best performance. As shown in Fig. 10, the super-resolved result by SRGAN did not remove the ring effect but could sharpen some details of images. On the contrary, the results of InverseNet successfully removed the ring artifact while keeping the details.



Figure 9. Motion deblurring for 512×512 image with model trained on 128×128 images. From left to right: blurred image, deblurred image by InverseNet, and original image.

D. Visualization of degraded kernel A

D.1. Motion deblurring

The degradation matrix A in motion deblurring is a square matrix corresponding to a 2-D convolution. 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). The first row of the matrix A is the motion blurring convolution kernel and the other row vectors are rotated one element to the right relative to the preceding row vector. The 9×9 motion blurring convolution kernel is displayed as below.

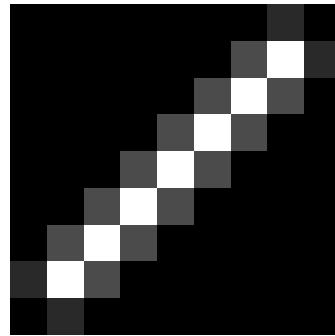


Figure 11. 9×9 motion blurring kernel.

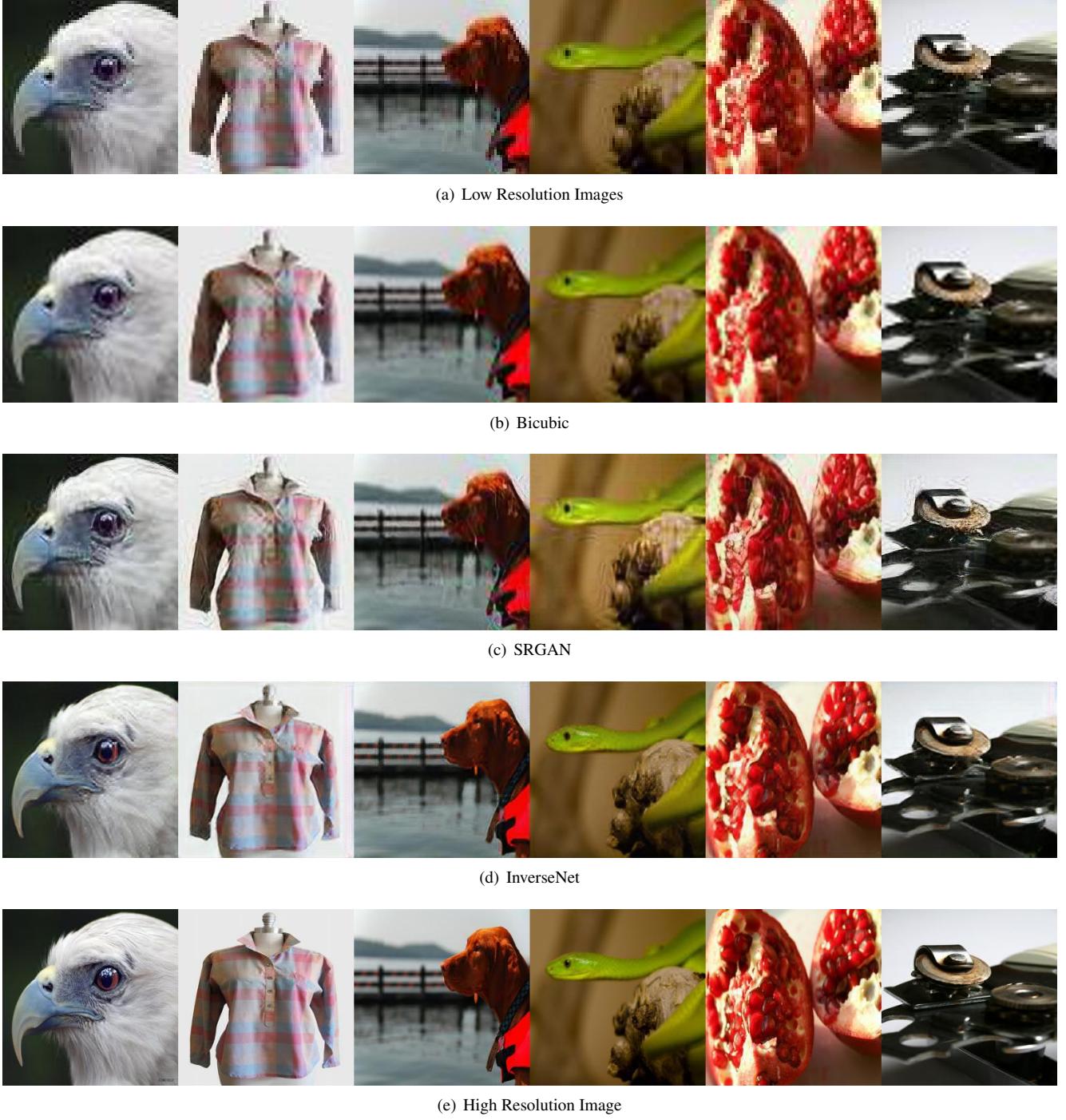


Figure 10. **Ring Effect Remover** on ImageNet for $64 \times 64 \rightarrow 256 \times 256$.

D.2. Super-resolution

The degradation matrix A in super-resolution can be decomposed as the multiplication of a convolution matrix H (as in motion blurring) and a down-sampling matrix S , i.e., $A = SH$. The down-sampling matrix S represents the regular 2-D decimation by keeping one pixel every d pixels in both horizontal and vertical directions, where d is the sampling ratio. The matrix A is equivalent to the strided convolution widely used in CNN architecture. The 3×3 convolution kernel used in our experiment is displayed as below. The down-sampling ratio d is fixed to 4 in the experiments, corresponding to shrinking the

size of an image from 256×256 to 64×64 .

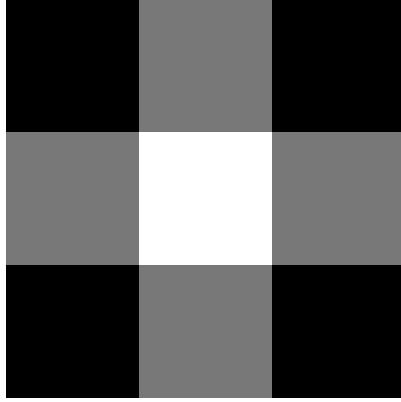


Figure 12. 3×3 convolution kernel in super-resolution.

D.3. Joint super-resolution and colorization

The degradation matrix A in joint super-resolution and colorization can be decomposed as the product of a convolution matrix H , a down-sampling matrix S and a spectral degradation matrix L . The matrices H and S are similarly defined as in Section D.2. The main difference here is that the kernel of H is channel wise in the sense that the convolution kernel for each channel is different, as shown in Fig. 13. The role of spectral degradation L is making the average of the R, G, B channels to get a one-channel grayscale image.

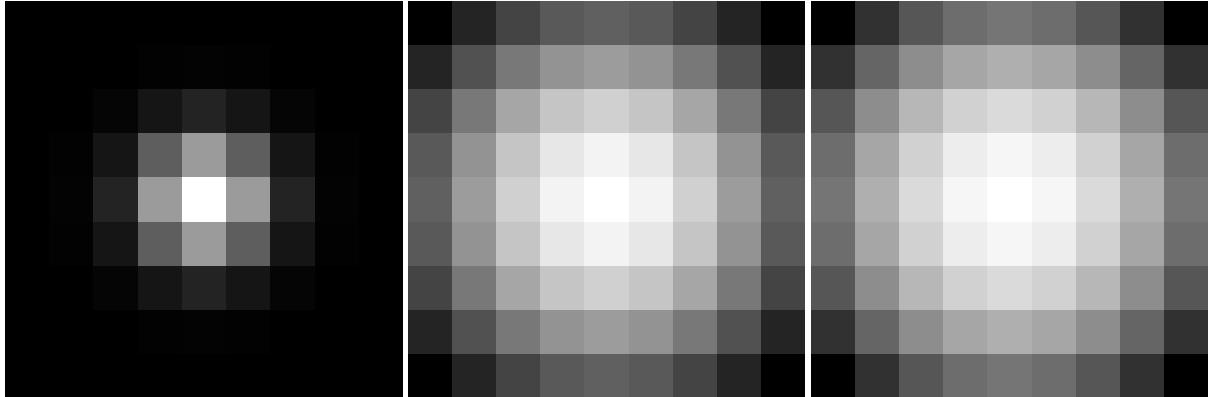


Figure 13. Three 9×9 convolution kernels for red (left), green (middle) and blue (right) channels.