Robust Single Image Super-Resolution via Deep Networks With Sparse Prior

Ding Liu, *Student Member, IEEE*, Zhaowen Wang, *Member, IEEE*, Bihan Wen, *Student Member, IEEE*, Jianchao Yang, *Member, IEEE*, Wei Han, and Thomas S. Huang, *Fellow, IEEE*

***Abstract*— Single image super-resolution (SR) is an ill-posed problem, which tries to recover a high-resolution image from its low-resolution observation. To regularize the solution of the problem, previous methods have focused on designing good priors for natural images, such as sparse representation, or directly learning the priors from a large data set with models, such as deep neural networks. In this paper, we argue that domain expertise from the conventional sparse coding model can be combined with the key ingredients of deep learning to achieve further improved results. We demonstrate that a sparse coding model particularly designed for SR can be incarnated as a neural network with the merit of end-to-end optimization over training data. The network has a cascaded structure, which boosts the SR performance for both fixed and incremental scaling factors. The proposed training and testing schemes can be extended for robust handling of images with additional degradation, such as noise and blurring. A subjective assessment is conducted and analyzed in order to thoroughly evaluate various SR techniques. Our proposed model is tested on a wide range of images, and it significantly outperforms the existing state-of-the-art methods for various scaling factors both quantitatively and perceptually.**

***Index Terms*— Image super-resolution, deep neural networks, sparse coding.**

I. Introduction

SINGLE image super-resolution is usually cast as an inverse problem of recovering the original high-resolution (HR) image from one low-resolution (LR) observation image. Since the known variables in LR images are greatly outnumbered by the unknowns in typically desired HR images, this problem is highly ill-posed and has limited the use of SR techniques in many practical applications [1], [2].

A large number of single image SR methods have been proposed, exploiting various priors of natual images to regularize the solution of SR. Analytical priors, such as bicubic interpolation, work well for smooth regions; while image models based on statistics of edges [3] and gradients [4] can recover sharper structures. In the patch-based SR methods, HR patch candidates are represented as the sparse linear combination of dictionary atoms trained from external databases [5]–[7], or recovered from similar examples in the LR image itself at different locations and across different scales [8], [9]. A regression model is built between LR and HR patches in [10] and [11]. A comprehensive review of more SR methods can be found in [12].

More recently, inspired by the great success achieved by deep learning [13] in other computer vision tasks, people begin to use neural networks with deep architecture for image SR. Multiple layers of collaborative auto-encoders are stacked together in [14] and [15] for robust matching of self-similar patches. Deep convolutional neural networks (CNN) [16] and deconvolutional networks [17] are designed that directly learn the non-linear mapping from LR space to HR space in a way similar to coupled sparse coding [6]. As these deep networks allow end-to-end training of all the model components between LR input and HR output, significant improvements have been observed over their shadow counterparts.

The networks in [14] and [16] are built with generic architectures, which means all their knowledge about SR is learned from training data. On the other hand, people's domain expertise for the SR problem, such as natural image prior and image degradation model, is largely ignored in deep learning based approaches. It is then worthwhile to investigate whether domain expertise can be used to design better deep model architectures, or whether deep learning can be leveraged to improve the quality of handcrafted models.

In this paper, we extend the conventional sparse coding model [5] using several key ideas from deep learning, and show that domain expertise is complementary to large learning capacity in further improving SR performance. First, based on the learned iterative shrinkage and thresholding algorithm (LISTA) [18], we implement a feed-forward neural network in which each layer strictly correspond to one step in the processing flow of sparse coding based image SR. In this way, the sparse representation prior is effectively encoded in our network structure; at the same time, all the components of sparse coding can be trained jointly through back-propagation. This simple model, which is named sparse coding based network (SCN), achieves notable improvement over the generic CNN model [16] in terms of both recovery accuracy and human perception, and yet has a compact

Manuscript received February 16, 2016; revised April 30, 2016; accepted May 1, 2016. Date of publication May 6, 2016; date of current version May 23, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jie Liang.

D. Liu, W. Han, and T. S. Huang are with the Department of Electrical and Computer Engineering, Beckman Institute, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: dingliu2@illinois.edu; weihan3@illinois.edu; t-huang1@illinois.edu).

Z. Wang is with Adobe Systems Inc., San Jose, CA 95110 USA (e-mail: zhawang@adobe.com).

B. Wen is with the Coordinated Science Laboratory, Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: bwen3@illinois.edu).

J. Yang is with Snapchat Inc., Venice, CA 90291 USA (e-mail: jianchao.yang@snapchat.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2016.2564643

model size. Moreover, with the correct understanding of each layer's physical meaning, we have a more principled way to initialize the parameters of SCN, which helps to improve optimization speed and quality.

A single network is only able to perform image SR by a particular scaling factor. In [16], different networks are trained for different scaling factors. In this paper, we propose a cascade of multiple SCNs to achieve SR for arbitrary factors. This approach, motivated by the self-similarity based SR approach [8], not only increases the scaling flexibility of our model, but also reduces artifacts for large scaling factors. Moreover, inspired by the multi-pass scheme of image denoising [19], we demonstrate that the SR results can be further enhanced by cascading multiple SCNs for SR of a fixed scaling factor. The cascade of SCNs (CSCN) can also benefit from the end-to-end training of deep network with a specially designed multi-scale cost function.

In practical SR scenarios, the real LR measurements usually suffer from various types of corruptions, such as noise and blurring. Sometimes the degradation process is even too complicated or unclear. We propose several schemes using our SCN to robustly handle such practical SR cases. When the degradation mechanism is unknown, we fine-tune the generic SCN with the requirement of only a small amount of real training data and manage to adapt our model to the new scenario. When the forward model for LR generation is clear, we propose an iterative SR scheme incorporating SCN with additional regularization based on priors from the degradation mechanism.

Subjective assessment is important to the SR technology because the commercial products equipped with such technology are usually evaluated subjectively by the end users. In order to thoroughly compare our model with other prevailing SR methods, we conduct a systematic subjective evaluation among these methods, in which the assessment results are statistically analyzed and one score is given for each method.

In short, the contributions of this paper include:

- combining the domain expertise of sparse coding and the merits of deep learning to achieve better SR performance with faster training and smaller model size;
- exploring network cascading for arbitrary scaling factors in order to further enhance SR performance;
- utilizing SCN to robustly handle the practical SR scenarios with non-ideal LR measurements.
- conducting a subjective evaluation on a number of recent state-of-the-art SR methods;

This paper is built upon our previous work in [20] and [21] with several additional contributions. First, we incorporate one more network cascading technique of SCN which further improves the SR performance in [21]. Second, a novel method of coping with the practical SR problems is presented which elaborates both of the training and testing schemes. Third, we introduce in detail the system of subjective assessment and its scoring mechanism. Finally, we provide a more comprehensive experiment section for qualitative and quantitative analysis, which includes extensive experimental results for the practical SR methods.

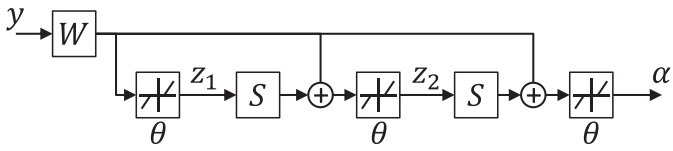


Fig. 1. A LISTA network [18] with 2 time-unfolded recurrent stages, whose output $\boldsymbol{\alpha}$ is an approximation of the sparse code of input signal $\boldsymbol{y}$. The linear weights $\boldsymbol{W}$, $\boldsymbol{S}$ and the shrinkage thresholds $\boldsymbol{\theta}$ are learned from data.

II. Related Work

A. Image SR Using Sparse Coding

The sparse representation based SR method [5] models the transform from each local patch $\boldsymbol{y} \in \mathbb{R}^{m_y}$ in the bicubic-upscaled LR image to the corresponding patch $\boldsymbol{x} \in \mathbb{R}^{m_x}$ in the HR image. The dimension m_y is not necessarily the same as m_x when image features other than raw pixel is used to represent patch $\boldsymbol{y}$. It is assumed that the LR(HR) patch $\boldsymbol{y}(\boldsymbol{x})$ can be represented with respect to an overcomplete dictionary $\boldsymbol{D}_y(\boldsymbol{D}_x)$ using some sparse linear coefficients $\boldsymbol{\alpha}_y(\boldsymbol{\alpha}_x) \in \mathbb{R}^n$, which are known as sparse code. Since the degradation process from $\boldsymbol{x}$ to $\boldsymbol{y}$ is nearly linear, the patch pair can share the same sparse code $\boldsymbol{\alpha}_y = \boldsymbol{\alpha}_x = \boldsymbol{\alpha}$ if the dictionaries $\boldsymbol{D}_y$ and $\boldsymbol{D}_x$ are defined properly. Therefore, for an input LR patch $\boldsymbol{y}$, the HR patch can be recovered as

$$\boldsymbol{x} = \boldsymbol{D}_x\boldsymbol{\alpha}, \quad \text{s.t. } \boldsymbol{\alpha} = \arg\min_{z} \|\boldsymbol{y} - \boldsymbol{D}_y z\|_2^2 + \lambda\|z\|_1, \qquad (1)$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm which is convex and sparsity-inducing, and λ is a regularization coefficient.

In order to learn the dictionary pair $(\boldsymbol{D}_y, \boldsymbol{D}_x)$, the goal is to minimize the recovery error of $\boldsymbol{x}$ and $\boldsymbol{y}$, and thus the loss function L in [6] is defined as

$$L = \frac{1}{2}\left(\gamma\|\boldsymbol{x} - \boldsymbol{D}_x z\|_2^2 + (1-\gamma)\|\boldsymbol{y} - \boldsymbol{D}_y z\|_2^2\right), \qquad (2)$$

where γ $(0 < \gamma \leq 1)$ balances the two reconstruction errors. Then the optimal dictionary pair $\{\boldsymbol{D}_x^*, \boldsymbol{D}_y^*\}$ can found by minimizing the empirical expectation of (2) over all the training LR/HR pairs,

$$\begin{aligned} &\min_{\boldsymbol{D}_x, \boldsymbol{D}_y} \frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{D}_x, \boldsymbol{D}_y, \boldsymbol{x}_i, \boldsymbol{y}_i) \\ &\text{s.t. } z_i = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{y}_i - \boldsymbol{D}_y\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, \quad i = 1, 2, ..., N, \\ &\quad \|D_x(:,k)\|_2 \leq 1, \quad \|D_y(:,k)\|_2 \leq 1, \quad k = 1, 2, ..., K. \end{aligned} \qquad (3)$$

Since the objective function in (2) is highly nonconvex, the dictionary pair $\boldsymbol{D}_y, \boldsymbol{D}_x)$ is usually learned alternatively while keeping the other fixed [6].

B. Network Implementation of Sparse Coding

There is an intimate connection between sparse coding and neural network, which has been well studied in [18] and [22]. A feed-forward neural network as illustrated in Fig. 1 is proposed in [18] to efficiently approximate the sparse code $\boldsymbol{\alpha}$ of input signal $\boldsymbol{y}$ as it would be obtained by solving (1) for a given dictionary $\boldsymbol{D}_y$. The network has a finite number of

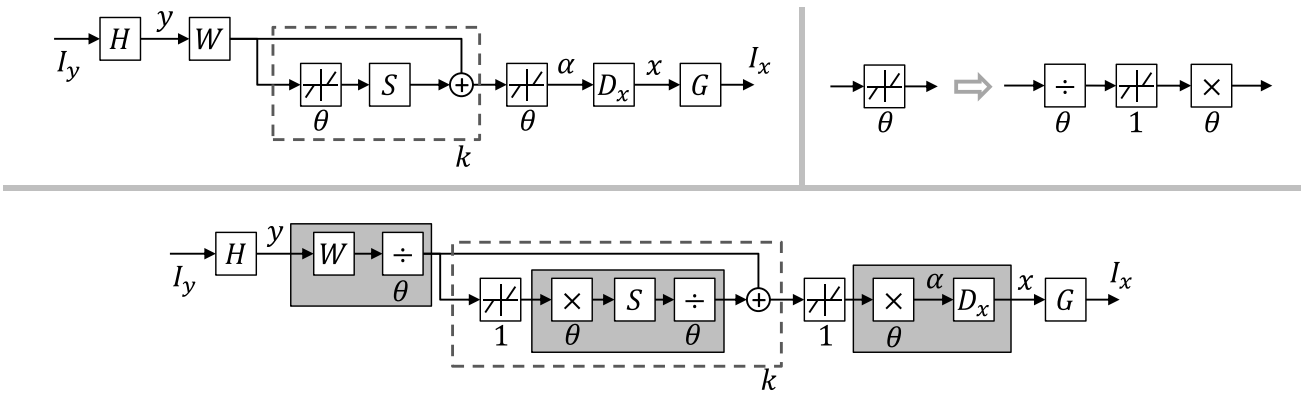


Fig. 2. Top left: the proposed SCN model with a patch extraction layer $\boldsymbol{H}$, a LISTA sub-network for sparse coding (with k recurrent stages denoted by the dashed box), a HR patch recovery layer $\boldsymbol{D}_x$, and a patch combination layer $\boldsymbol{G}$. Top right: a neuron with an adjustable threshold decomposed into two linear scaling layers and a unit-threshold neuron. Bottom: the SCN re-organized with unit-threshold neurons and adjacent linear layers merged together in the gray boxes.

recurrent stages, each of which updates the intermediate sparse code according to

$$z_{k+1} = h_{\boldsymbol{\theta}}(\boldsymbol{W}\boldsymbol{y} + \boldsymbol{S}z_k), \tag{4}$$

where $h_{\boldsymbol{\theta}}$ is an element-wise shrinkage function defined as $[h_{\boldsymbol{\theta}}(\boldsymbol{a})]_i = \text{sign}(a_i)(|a_i| - \theta_i)_+$ with positive thresholds $\boldsymbol{\theta}$.

Different from the iterative shrinkage and thresholding algorithm (ISTA) [23], [24] which finds an analytical relationship between network parameters (weights $\boldsymbol{W}$, $\boldsymbol{S}$ and thresholds $\boldsymbol{\theta}$) and sparse coding parameters ($\boldsymbol{D}_y$ and λ), the authors of [18] learn all the network parameters from training data using a back-propagation algorithm called learned ISTA (LISTA). In this way, a good approximation of the underlying sparse code can be obtained within a fixed number of recurrent stages.

C. Generic Convolutional Neural Network for SR

As an successful example of deep learning for single image SR, Dong *et al.* [16] propose a fully convolutional neural network to directly learn the mapping from the input LR image and the output HR image. It is designed to utilize three convolutional layers to minic the patch extraction and representation, non-linear mapping and reconstruction of the sparse representation based SR methods, respectively. Due to the end-to-end training strategy that jointly optimizes all the parameters and the large learning capacity of neural networks, this method notably outperforms its conventional shadow counterpart.

III. Sparse Coding Based Network for Image SR

A. Network Architecture

Given the fact that sparse coding can be effectively implemented with a LISTA network, it is straightforward to build a multi-layer neural network that mimics the processing flow of the sparse coding based SR method [5]. Same as most patch-based SR methods, our sparse coding based network (SCN) takes the bicubic-upscaled LR image $\boldsymbol{I}_y$ as input, and outputs the full HR image $\boldsymbol{I}_x$. Fig. 2 shows the main network structure, and each of the layers is described in the following.

The input image $\boldsymbol{I}_y$ first goes through a convolutional layer $\boldsymbol{H}$ which extracts feature for each LR patch. There are m_y filters of spatial size $s_y \times s_y$ in this layer, so that our input patch size is $s_y \times s_y$ and its feature representation $\boldsymbol{y}$ has m_y dimensions.

Each LR patch $\boldsymbol{y}$ is then fed into a LISTA network with a finite number of k recurrent stages to obtain its sparse code $\boldsymbol{\alpha} \in \mathbb{R}^n$. Each stage of LISTA consists of two linear layers parameterized by $\boldsymbol{W} \in \mathbb{R}^{n \times m_y}$ and $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, and a nonlinear neuron layer with activation function $h_{\boldsymbol{\theta}}$. The activation thresholds $\boldsymbol{\theta} \in \mathbb{R}^n$ are also to be updated during training, which complicates the learning algorithm. To restrict all the tunable parameters in our linear layers, we do a simple trick to rewrite the activation function as

$$[h_{\boldsymbol{\theta}}(\boldsymbol{a})]_i = \text{sign}(a_i)\theta_i(|a_i|/\theta_i - 1)_+ = \theta_i h_1(a_i/\theta_i). \tag{5}$$

Eq. (5) indicates the original neuron with an adjustable threshold can be decomposed into two linear scaling layers and a unit-threshold neuron, as shown in the top-right of Fig. 2. The weights of the two scaling layers are diagonal matrices defined by $\boldsymbol{\theta}$ and its element-wise reciprocal, respectively.

The sparse code $\boldsymbol{\alpha}$ is then multiplied with HR dictionary $\boldsymbol{D}_x \in \mathbb{R}^{m_x \times n}$ in the next linear layer, reconstructing HR patch $\boldsymbol{x}$ of size $s_x \times s_x = m_x$.

In the final layer $\boldsymbol{G}$, all the recovered patches are put back to the corresponding positions in the HR image $\boldsymbol{I}_x$. This is realized via a convolutional filter of m_x channels with spatial size $s_g \times s_g$. The size s_g is determined as the number of neighboring patches that overlap with the same pixel in each spatial direction. The filter will assign appropriate weights to the overlapped recoveries from different patches and take their weighted average as the final prediction in $\boldsymbol{I}_x$.

As illustrated in the bottom of Fig. 2, after some simple reorganizations of the layer connections, the network described above has some adjacent linear layers which can be merged into a single layer. This helps to reduce the computation load as well as redundant parameters in the network. The layers $\boldsymbol{H}$ and $\boldsymbol{G}$ are not merged because we apply additional nonlinear normalization operations on patches $\boldsymbol{y}$ and $\boldsymbol{x}$, which will be detailed in Sec. VI-A.

Thus, there are totally 5 trainable layers in our network: 2 convolutional layers $\boldsymbol{H}$ and $\boldsymbol{G}$, and 3 linear layers shown as gray boxes in Fig. 2. The k recurrent layers share the same weights and are therefore conceptually regarded as one. Note that all the linear layers are actually implemented as convolutional layers applied on each patch with filter spatial size of 1×1, a structure similar to the network in network [25]. Also note that all these layers have only weights but no biases (zero biases).

Mean square error (MSE) is employed as the cost function to train the network, and our optimization objective can be expressed as

$$\min_{\boldsymbol{\Theta}} \sum_i \|SCN(\boldsymbol{I}_y^{(i)}; \boldsymbol{\Theta}) - \boldsymbol{I}_x^{(i)}\|_2^2, \tag{6}$$

where $\boldsymbol{I}_y^{(i)}$ and $\boldsymbol{I}_x^{(i)}$ are the i-th pair of LR/HR training data, and $SCN(\boldsymbol{I}_y; \boldsymbol{\Theta})$ denotes the HR image for $\boldsymbol{I}_y$ predicted using the SCN model with parameter set $\boldsymbol{\Theta}$. All the parameters are optimized through the standard back-propagation algorithm. Although it is possible to use other cost terms that are more correlated with human visual perception than MSE, our experimental results show that simply minimizing MSE leads to improvement in subjective quality.

B. Advantages Over Previous Models

The construction of our SCN follows exactly each step in the sparse coding based SR method [5]. If the network parameters are set according to the dictionaries learned in [5], it can reproduce almost the same results. However, after training, SCN learns a more complex regression function and can no longer be converted to an equivalent sparse coding model. The advantage of SCN comes from its ability to jointly optimize all the layer parameters from end to end; while in [5] some variables are manually designed and some are optimized individually by fixing all the others.

Technically, our network is also a CNN and it has similar layers as the CNN model proposed in [16] for patch extraction and reconstruction. The key difference is that we have a LISTA sub-network specifically designed to enforce sparse representation prior; while in [16] a generic rectified linear unit (ReLU) [26] is used for nonlinear mapping. Since SCN is designed based on our domain knowledge in sparse coding, we are able to obtain a better interpretation of the filter responses and have a better way to initialize the filter parameters in training. We will see in the experiments that all these contribute to better SR results, faster training speed and smaller model size than a vanilla CNN.

C. Network Cascade

In this section, we investigate two different network cascade techniques in order to fully exploit our SCN model in SR applications.

1) Network Cascade for SR of a Fixed Scaling Factor: First, we observe that the SR results can be further improved by cascading multiple SCNs trained for the same objective in (6), which is inspired by the multi-pass scheme in [19]. The only difference for training these SCNs is to replace the bicubic interpolated input by its latest HR estimate, while the target output remains the same.

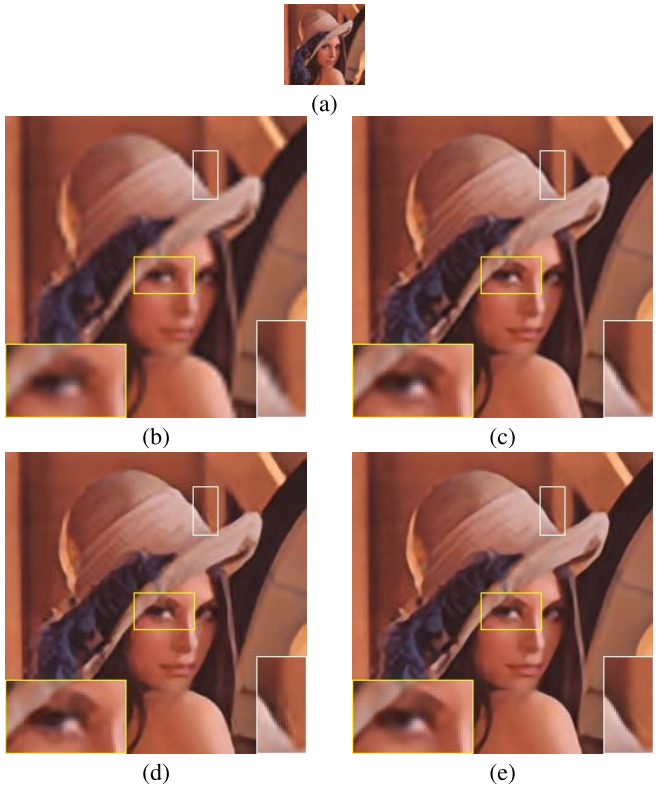


Fig. 3. SR results for the "Lena" image upscaled by 4 times. (a) → (b) → (d) represents the processing flow with a single SCN×4 model. (a) → (c) → (e) represents the processing flow with two cascaded SCN×2 models. PSNR is given in parentheses.

The first SCN plays as a function approximator to model the non-linear mapping from the bicubic upscaled image to the ground-truth image. The following SCN plays as another function approximator, with the starting point changed to a better estimate: the output of its previous SCN.

In other words, the cascade of SCNs as a whole can be considered as a new deeper network having more powerful learning capability, which is able to better approximate the mapping between the LR inputs to the HR counterparts, and these SCNs can be trained jointly to pursue even better SR performance.

2) Network Cascade for Scalable SR: Like most SR models learned from external training examples, the SCN discussed previously can only upscale images by a fixed factor. A separate model needs to be trained for each scaling factor to achieve the best performance, which limits the flexibility and scalability in practical use. One way to overcome this difficulty is to repeatedly enlarge the image by a fixed scale until the resulting HR image reaches a desired size. This practice is commonly adopted in the self-similarity based methods [8], [9], [14], but is not so popular in other cases for the fear of error accumulation during repetitive upscaling.

In our case, however, it is observed that a cascade of SCNs trained for small scaling factors can generate even better SR results than a single SCN trained for a large

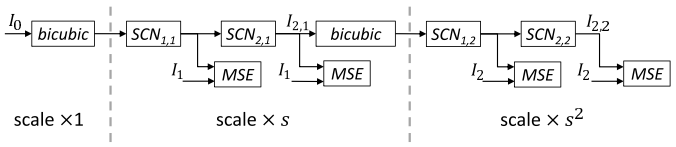


Fig. 4. Training cascade of SCNs with multi-scale objectives.

scaling factor, especially when the target scaling factor is large (greater than 2). This is illustrated by the example in Fig. 3. Here an input image is magnified by $\times 4$ times in two ways: with a single SCN$\times$4 model through the processing flow (a) $\rightarrow$ (b) $\rightarrow$ (d); and with a cascade of two SCN$\times$2 models through (a) $\rightarrow$ (c) $\rightarrow$ (e). It can be seen that the input to the second cascaded SCN$\times$2 in (c) is already sharper and contains less artifacts than the bicubic$\times$4 input to the single SCN$\times$4 in (b), which naturally leads to the better final result in (e) than the one in (d).

To get a better understanding of the above observation, we can draw a loose analogy between the SR process and a communication system. Bicubic interpolation is like a noisy channel through which an image is "transmitted" from LR domain to HR domain. And our SCN model (or any SR algorithm) behaves as a receiver which recovers clean signals from noisy observations. A cascade of SCNs is then like a set of relay stations that enhance signal-to-noise ratio before the signal becomes too weak for further transmission. Therefore, cascading will work only when each SCN can restore enough useful information to compensate for the new artifacts it introduces as well as the magnified artifacts from previous stages.

3) Training Cascade of Networks: Taking into account the two aforementioned cascade techniques, we can consider the cascade of all SCNs as a deeper network (CSCN), in which the final output of the consecutive SCNs of the same ground truth is connected to the input of the next SCN with bicubic interpolation in the between. To construct the cascade, besides stacking several SCNs trained individually with respect to (6), we can also optimize all of them jointly as shown in Fig. 4. Without loss of generality, we assume each stage in Sec. III-C2 has the same scaling factor s. Let $\hat{\boldsymbol{I}}_{j,k}$ $(j > 0, k > 0)$ denote the output image of the j-th SCN in the k-th stage upscaled by a total of $\times s^k$ times. In the same stage, each output of SCNs is compared with the associated ground truth image $\boldsymbol{I}_k$ according to the MSE cost, leading to a multi-scale objective function:

$$\min_{\{\boldsymbol{\Theta}_{j,k}\}} \sum_i \sum_j \sum_k \left\| SCN(\hat{\boldsymbol{I}}^{(i)}_{j-1,k}; \boldsymbol{\Theta}_{j,k}) - \boldsymbol{I}^{(i)}_k \right\|_2^2, \qquad (7)$$

where i denotes the data index, and j, k denotes the SCN index. For simplicity of notation, $\hat{\boldsymbol{I}}_{0,k}$ specially denotes the bicubic interpolated image of the final output in the $(k-1)$-th stage upscaled by a total of $\times s^{k-1}$ times. This multi-scale objective function makes full use of the supervision information in all scales, sharing a similar idea as heterogeneous networks [27]. All the layer parameters $\{\boldsymbol{\Theta}_{j,k}\}$ in (7) could be optimized from end to end by back-propagation. The SCNs share the same training objective can be trained simultaneously, taking advantage of the merit of deep learning. For the SCNs with different training objectives, we use a greedy algorithm here to train them sequentially from the beginning of the cascade so that we do not need to care about the gradient of bicubic layers. Applying back-propagation through a bicubic layer or its trainable surrogate will be considered in future work.

IV. Robust SR for Real Scenarios

Most of recent SR works generate the LR images for both training and testing by downscaling HR images using bicubic interpolation [5], [28]. However, this assumption of the forward model may not always hold in practice. For example, the real LR measurements are usually blurred, or corrupted with noise. Sometimes, the LR generation mechanism may be complicated, or even unknown. We now investigate the practical SR problem, and propose two approaches to handle such non-ideal LR measurements, using the generic SCN. In the case that the underlying mechanism of the real LR generation is unclear or complicated, we propose the data-driven approach by fine-tuning the learned generic SCN with a limited number of real LR measurements as well as their corresponding HR counterparts. On the other hand, if the real training samples are unavailable but the LR generation mechanism is clear, we formulate this inverse problem as the regularized HR image reconstruction problem which can be solved using iterative methods. The proposed methods demonstrate the robustness of our SCN model to different SR scenarios. In the following, we elaborate the details of these two approaches, respectively.

A. Data-Driven SR by Fine-Tuning

Deep learning models can be efficiently transferred from one task to another by re-using the intermediate representation in the original neural network [29]. This method has proven successful on a number of high-level vision tasks, even if there is a limited amount of training data in the new task [30].

The success of super-resolution algorithms usually highly depends on the accuracy of the model of the imaging process. When the underlying mechanism of the generation of LR images is not clear, we can take advantage of the aforementioned merit of deep learning models by learning our model in a data-driven manner, to adapt it for a particular task. Specifically, we start training from the generic SCN model while using very limited amount of training data from a new SR scenario, and manage to adapt it to the new SR scenario and obtain promising results. In this way, it is demonstrated that the SCN has the strong capability of learning complex mappings between the non-ideal LR measurements to their HR counterparts as well as the high flexibility of adapting to various SR tasks.

B. Iterative SR With Regularization

The second approach considers the case that the mechanism of generating the real LR images is relatively simple and clear, indicating the training data is always available if we synthesize

LR images with the known degradation process. We propose an iterative SR scheme which incorporates the generic SCN model with additional regularization based on task-related priors (e.g. the known kernel for deblurring, or the data sparsity for denoising). In this section, we specifically discuss handling blurred and noisy LR measurements in details as examples, though the iterative SR methods can be generalized to other practical imaging models.

1) Blurry Image Upscaling: The real LR images can be generated with various types of blurring. Directly applying the generic SCN model is obviously not optimal. Instead, with the known blurring kernel, we propose to estimate the regularized version of the HR image $\hat{\boldsymbol{I}}_x$ based on the directly upscaled image $\tilde{\boldsymbol{I}}_x$ by the learned SCN as follows:

$$\hat{\boldsymbol{I}}_x = \arg\min_{\boldsymbol{I}} \|\boldsymbol{I} - \tilde{\boldsymbol{I}}_x\|_2, \quad \text{s.t. } D{\cdot}B \cdot \boldsymbol{I} = \boldsymbol{I}_y^0 \tag{8}$$

where $\boldsymbol{I}_y^0$ is the original blurred LR input, and the operators B and D are blurring and sub-sampling respectively. Similar to the previous work [5], we use back-projection to iteratively estimate the regularized HR input on which our model can perform better. Specifically, given the regularized estimate $\hat{\boldsymbol{I}}_x^{i-1}$ at iteration $i-1$, we estimate a less blurred LR image $\boldsymbol{I}_y^{i-1}$ by downsampling $\hat{\boldsymbol{I}}_x^i$ using bicubic interpolation. The upscaled $\tilde{\boldsymbol{I}}_x^i$ by learned SCN serves the regularizer for the i-th iteration as following:

$$\hat{\boldsymbol{I}}_x^i = \arg\min_{\boldsymbol{I}} \|\boldsymbol{I} - \tilde{\boldsymbol{I}}_x^i\|_2^2 + \|D{\cdot}B \cdot \boldsymbol{I} - \boldsymbol{I}_y^0\|_2^2 \tag{9}$$

Here we use penalty method to form an unconstrained problem. The upscaled HR image $\tilde{\boldsymbol{I}}_x^i$ can be computed as $SCN(\boldsymbol{I}_y^{i-1}, \boldsymbol{\Theta})$. The same process is repeated until convergence. We have applied the proposed iterative scheme to LR images generated from Gaussian blurring and sub-sampling as an example. The empirical performance is illustrated in Sec. VI.

2) Noisy Image Upscaling: Noise is a ubiquitous cause of corruption in image acquisition. State-of-the-art image denoising methods usually adopt priors such as patch similarity [31], patch sparsity [19], [32], or both [33], as regularizer in image restoration. In this section, we propose a regularized noisy image upscaling scheme, for specifically handling noisy LR images, in order to obtain improved SR quality. Though any denoising algorithm can be used in our proposed scheme, here we apply spatial similarity combined with transform domain image patch group-sparsity as our regularizer [33], to form the regularized iterative SR problem as an example.

Similar to the method in Sec. IV-B1, we iteratively estimate the less noisy HR image from the denoised LR image. Given the denoised LR estimate $\hat{\boldsymbol{I}}_y^{i-1}$ at iteration $i-1$, we directly upscale it, using the learned generic SCN, to obtain the HR image $\hat{\boldsymbol{I}}_x^{i-1}$. It is then downsampled using bicubic interpolation, to generate the LR image $\tilde{\boldsymbol{I}}_y^i$, which is used in the fidelity term in the i-th iteration of LR image denoising. The same process is repeated until convergence. The iterative LR image denoising problem is formulated as follows:

$$\left\{\hat{\boldsymbol{I}}_y^i, \{\hat{\alpha}_i\}\right\} = \arg\min_{\boldsymbol{I}, \{\boldsymbol{\alpha}_i\}} \|\boldsymbol{I} - \tilde{\boldsymbol{I}}_y^i\|_2^2 + \sum_{j=1}^{N} \left\{\|W_{3D} G_j \boldsymbol{I} - \alpha_j\|_2^2 + \tau\|\alpha_j\|_0\right\} \tag{10}$$

where the operator G_j generates the 3D vectorized tensor, which groups the j-th overlapping patch from the LR image I, together with the spatially similar patches within its neighborhood by block matching [33]. The codes $\{\alpha_j\}$ of the patch groups in the domain of 3D sparsifying transform W_{3D} are sparse, which is enforced by the l_0 norm penalty [34]. The weight τ controls the sparsity level, which normally depends on the remaining noise level in $\tilde{I}_y^i$ [34], [35].

In (10), we use the patch group sparsity as our denoising regularizer. The 3D sparsifying transform W_{3D} can be commonly used analytical transforms, such as discrete cosine transform (DCT) or Wavelets. The state-of-the-art BM3D denoising algorithm [33] is based on such an approach, but further improved by more sophisticated engineering stages. In order to achieve the best practical SR quality, we demonstrate the empirical performance comparison using BM3D as the regularizer in Sec. VI. Additionally, our proposed iterative method is a general practical SR framework, which is not dedicated to SCN. One can conveniently extend it to other SR methods, which generates $\tilde{I}_y^i$ in i^{th} iteration. The performance comparison of these methods is illustrated in Sec. VI.

V. Subjective Evaluation Protocol

Subjective perception is an important metric to evaluate SR techniques for commercial use, other than the quantitative evaluation. In order to more thoroughly compare various SR methods and quantify the subjective perception, we utilize an online platform for subjective evaluation of SR results from several methods [36], including bicubic, SC [6], SE [9], self-example regression (SER) [37], CNN [16] and CSCN. Each participant is invited to conduct several pair-wise comparisons of SR results from different methods. The SR methods of displayed SR images in each pair are randomly selected. Ground truth HR images are also included when they are available as references. For each pair, the participant needs to select the better one in terms of perceptual quality. A snapshot of our evaluation web page[1] is shown in Fig. 5.

Specifically, there are SR results over 6 images with different scaling factors: "kid"×4, "chip"×4, "statue"×4, "lion"×3, "temple"×3 and "train"×3. The images are shown in Fig. 6. All the visual comparison results are then summarized into a 7×7 winning matrix for 7 methods (including ground truth). A Bradley–Terry [38] model is calculated based to these results and the subjective score is estimated for each method according to this model. In the Bradley-Terry model, the probability that an object X is favored over Y is assumed to be

$$p(X \succ Y) = \frac{e^{s_X}}{e^{s_X} + e^{s_Y}} = \frac{1}{1 + e^{s_Y - s_X}}, \tag{11}$$

[1] www.ifp.illinois.edu/~wang308/survey

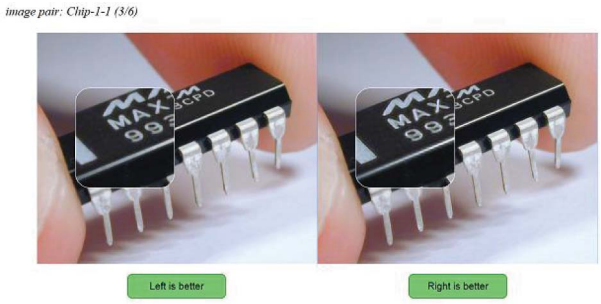


Fig. 5. The user interface of a web-based image quality evaluation, where two images are displayed side by side and local details can be magnified by moving mouse over the corresponding region.

Fig. 6. The 6 images used in subjective evaluation.

where s_X and s_Y are the subjective scores for X and Y. The scores $\boldsymbol{s}$ for all the objects can be jointly estimated by maximizing the log likelihood of the pairwise comparison observations:

$$\max_{\boldsymbol{s}} \sum_{i,j} w_{ij} \log\left(\frac{1}{1+e^{s_j - s_i}}\right), \tag{12}$$

where w_{ij} is the (i, j)-th element in the winning matrix $\boldsymbol{W}$, meaning the number of times when method i is favored over method j. We use the Newton-Raphson method to solve Eq. (12) and set the score for ground truth method as 1 to avoid the scale ambiguity.

The experiment results are detailed in Sec. VI.

VI. Experiments

We evaluate and compare the performance of our models using the same data and protocols as in [28], which are commonly adopted in SR literature. All our models are learned from a training set with 91 images, and tested on Set5 [39], Set14 [40] and BSD100 [41] which contain 5, 14 and 100 images respectively. We have also trained on other different larger data sets, and observe marginal performance change (around 0.1dB). The original images are downsized by bicubic interpolation to generate LR-HR image pairs for both training and evaluation. The training data are augmented with translation, rotation and scaling.

A. Implementation Details

We determine the number of nodes in each layer of our SCN mainly according to the corresponding settings used in sparse coding [6]. Unless otherwise stated, we use input LR patch size s_y=9, LR feature dimension m_y=100, dictionary size n=128, output HR patch size s_x=5, and patch aggregation filter size s_g=5. All the convolution layers have a stride of 1. Each LR patch $\boldsymbol{y}$ is normalized by its mean and variance, and the same mean and variance are used to restore the final HR patch $\boldsymbol{x}$. We crop 56×56 regions from each image to obtain fixed-sized input samples to the network, which produces outputs of size 44×44.

To reduce the number of parameters, we implement the LR patch extraction layer $\boldsymbol{H}$ as the combination of two layers: the first layer has 4 trainable filters each of which is shifted to 25 fixed positions by the second layer. Similarly, the patch combination layer $\boldsymbol{G}$ is also split into a fixed layer which aligns pixels in overlapping patches and a trainable layer whose weights are used to combine overlapping pixels. In this way, the number of parameters in these two layers are reduced by more than an order, and there is no observable loss in performance.

We employ a standard stochastic gradient descent algorithm to train our networks with mini-batch size of 64. Based on the understanding of each layer's role in sparse coding, we use Harr-like gradient filters to initialize layer $\boldsymbol{H}$, and use uniform weights to initialize layer $\boldsymbol{G}$. All the remaining three linear layers are related to the dictionary pair $(\boldsymbol{D}_x, \boldsymbol{D}_y)$ in sparse coding. To initialize them, we first randomly set $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ with Gaussian noise, and then find the corresponding layer weights as in ISTA [23]:

$$\boldsymbol{w}_1 = C \cdot \boldsymbol{D}_y^T, \quad \boldsymbol{w}_2 = \boldsymbol{I} - \boldsymbol{D}_y^T \boldsymbol{D}_y, \quad \boldsymbol{w}_3 = (CL)^{-1} \cdot \boldsymbol{D}_x \tag{13}$$

where $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ denote the weights of the three subsequent layers after layer $\boldsymbol{H}$. L is the upper bound on the largest eigenvalue of $\boldsymbol{D}_y^T \boldsymbol{D}_y$, and C is the threshold value before normalization. We empirically set L=C=5.

The proposed models are all trained using the CUDA ConvNet package [13] on a workstation with 12 Intel Xeon 2.67GHz CPUs and 1 GTX680 GPU. Training a SCN usually takes less than one day. Note that this package is customized for classification networks, and its efficiency can be further optimized for our SCN model.

In testing, to make the entire image covered by output samples, we crop input samples with overlap and extend the boundary of original image by reflection. Note we shave the image border in the same way as [16] for objective evaluations to ensure fair comparison. Only the luminance channel is processed with our method, and bicubic interpolation is applied to the chrominance channels, as their high frequency components are less noticeable to human eyes. To achieve arbitrary scaling factors using CSCN, we upscale an image by ×2 times repeatedly until it is at least as large as the desired size. Then a bicubic interpolation is used to downscale it to the target resolution if necessary.

When reporting our best results in Sec. VI-C, we also use the multi-view testing strategy commonly employed in image classification. For patch-based image SR, multi-view testing is implicitly used when predictions from multiple overlapping

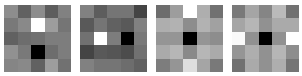


Fig. 7. The four learned filters in the first layer $\boldsymbol{H}$.

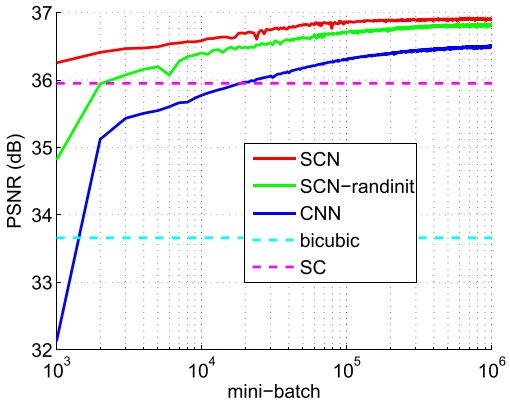


Fig. 8. The PSNR change for $\times 2$ SR on Set5 during training using different methods: SCN; SCN with random initialization; CNN. The horizontal dash lines show the benchmarks of bicubic interpolation and sparse coding (SC).

patches are averaged. Here, besides sampling overlapping patches, we also add more views by flipping and transposing the patch. Such strategy is found to improve SR performance for general algorithms at the sheer cost of computation.

B. Algorithm Analysis

We first visualize the four filters learned in the first layer $\boldsymbol{H}$ in Fig. 7. The filter patterns do not change much from the initial first and second order gradient operators. Some additional small coefficients are introduced in a highly structured form that capture richer high frequency details.

The performance of several networks during training is measured on Set5 in Fig. 8. Our SCN improves significantly over sparse coding (SC) [6], as it leverages data more effectively with end-to-end training. The SCN initialized according to (13) can converge faster and better than the same model with random initialization, which indicates that the understanding of SCN based on sparse coding can help its optimization. We also train a CNN model [16] of the same size as SCN, but find its convergence speed much slower. It is reported in [16] that training a CNN takes 8×10^8 back-propagations (equivalent to 12.5×10^6 mini-batches here). To achieve the same performance as CNN, our SCN requires less than 1% back-propagations.

The network size of SCN is mainly determined by the dictionary size n. Besides the default value n=128, we have tried other sizes and plot their performance versus the number of network parameters in Fig. 9. The PSNR of SCN does not drop too much as n decreases from 128 to 64, but the model size and computation time can be reduced significantly, as shown in Table I. Fig. 9 also shows the performance of CNN with various sizes. Our smallest SCN can achieve higher PSNR than the largest model (CNN-L) in [42] while only using about 20% parameters.

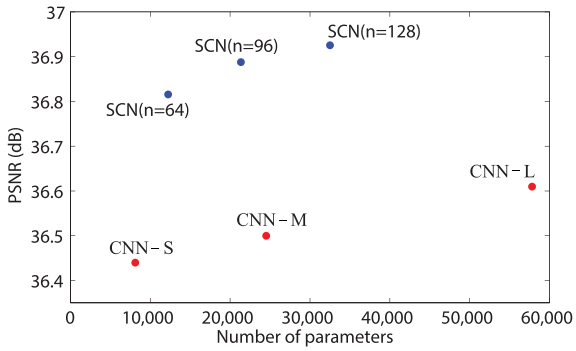


Fig. 9. PSNR for $\times 2$ SR on Set5 using SCN and CNN with various network sizes.

TABLE I

TIME CONSUMPTION FOR SCN TO UPSCALE THE "BABY" IMAGE FROM 256×256 TO 512×512 USING DIFFERENT DICTIONARY SIZE n

n	64	96	128	256	512
time (s)	0.159	0.192	0.230	0.445	1.214

TABLE II

PSNR OF DIFFERENT NETWORK CASCADING SCHEMES ON Set5, EVALUATED FOR DIFFERENT SCALING FACTORS IN EACH COLUMN

scaling factor	×1.5	×2	×3	×4
SCN×1.5	40.14	36.41	30.33	29.02
SCN×2	**40.15**	**36.93**	32.99	30.70
SCN×3	39.88	36.76	32.87	30.63
SCN×4	39.69	36.54	32.76	30.55
CSCN	**40.15**	**36.93**	**33.10**	**30.86**

TABLE III

EFFECT OF VARIOUS TRAINING SETS ON THE PSNR OF ×2 UPSCALING WITH SINGLE VIEW SCN

Training Set	Test Set			
	Set5	Set14	BSD100	ILSVRC (100)
Set91	36.93	32.56	31.40	32.13
BSD200	**36.97**	**32.69**	**31.55**	32.27
ILSVRC (7.5k)	36.84	32.67	31.51	**32.31**

Different numbers of recurrent stages k have been tested for SCN, and we find increasing k from 1 to 3 only improves performance by less than 0.1dB. As a tradeoff between speed and accuracy, we use k=1 throughout the paper.

In Table II, different network structures with cascade for scalable SR in Sec. III-C2 (in each row) are compared at different scaling factors (in each column). SCN×a denotes the model trained with fixed scaling factor a without any cascade technique. For a fixed a, we use SCN×a as a basic module and apply it one or more times to super-resolve images for different upscaling factors, which is shown in each row of Table II. It is observed that SCN×2 can perform as well as the scale-specific model for small scaling factor (1.5), and much better for large scaling factors (3 and 4). Note that the cascade of SCN×1.5 does not lead to good results since artifacts quickly

TABLE IV

PSNR (SSIM) COMPARISON ON THREE TEST DATA SETS AMONG DIFFERENT METHODS. RED INDICATES THE BEST AND BLUE INDICATES THE SECOND BEST PERFORMANCE. THE PERFORMANCE GAIN OF OUR BEST MODEL OVER ALL THE OTHERS' BEST IS SHOWN IN THE LAST ROW

Data Set	Set5			Set14			BSD100		
Upscaling	×2	×3	×4	×2	×3	×4	×2	×3	×4
A+ [43]	36.55 (0.9544)	32.59 (0.9088)	30.29 (0.8603)	32.28 (0.9056)	29.13 (0.8188)	27.33 (0.7491)	30.78 (0.8773)	28.18 (0.7808)	26.77 (0.7085)
CNN [16]	36.34 (0.9521)	32.39 (0.9033)	30.09 (0.8530)	32.18 (0.9039)	29.00 (0.8145)	27.20 (0.7413)	31.11 (0.8835)	28.20 (0.7794)	26.70 (0.7018)
CNN-L [42]	36.66 (0.9542)	32.75 (0.9090)	30.49 (0.8628)	32.45 (0.9067)	29.30 (0.8215)	27.50 (0.7513)	31.36 (0.8879)	28.41 (0.7863)	26.90 (0.7103)
CSCN	37.00 (0.9557)	33.18 (0.9153)	30.94 (0.8755)	32.65 (0.9081)	29.41 (0.8234)	27.71 (0.7592)	31.46 (0.8891)	28.52 (0.7883)	27.06 (0.7167)
CSCN-MV	37.21 (0.9571)	33.34 (0.9173)	31.14 (0.8789)	32.80 (0.9101)	29.57 (0.8263)	27.81 (0.7619)	31.60 (0.8915)	28.60 (0.7905)	27.14 (0.7191)
Our Improvement	0.55 (0.0029)	0.59 (0.0083)	0.65 (0.0161)	0.35 (0.0034)	0.27 (0.0048)	0.31 (0.0106)	0.24 (0.0036)	0.19 (0.0042)	0.24 (0.0088)

get amplified through many repetitive upscalings. Therefore, we use SCN×2 as the default building block for CSCN, and drop the notation ×2 when there is no ambiguity. The last row in Table II shows that a CSCN trained using the multi-scale objective in (7) can further improve the SR results for scaling factors 3 and 4, as the second SCN in the cascade is trained to be robust to the artifacts generated by the first one.

As shown in [42], the amount of training data plays an important role in the field of deep learning. In order to evaluate the effect of various amount of data on training CSCN, we change the training set from a relatively small set of 91 images (Set91) [28] to two other sets: the 199 out of 200 training images[2] in BSD500 dataset (BSD200) [41], and a subset of 7,500 images from the ILSVRC2013 dataset [44]. A model of exactly the same architecture without any cascade is trained on each data set, and another 100 images from the ILSVRC2013 dataset are included as an additional test set. From Table III, we can observe that the CSCN trained on BSD200 consistently outperforms its counterpart trained on Set91 by around 0.1dB on all test data. However, the performance of the model trained on ILSVRC2013 is slightly different from the one trained on BSD200, which shows the saturation of the performance as the amount of training data increases. The inferior quality of images in ILSVRC2013 may be a hurdle to further improve the performance. Therefore, our method is robust to training data and can benefit marginally from a larger set of training images.

C. Comparison With State of the Arts

We compare the proposed CSCN with other recent SR methods on all the images in Set5, Set14 and BSD100 for different scaling factors. Table IV shows the PSNR and structural similarity (SSIM) [45] for adjusted anchored neighborhood regression (A+) [43], CNN [16], CNN trained with larger model size and much more data (CNN-L) [42], the proposed CSCN, and CSCN with our multi-view testing (CSCN-MV). We do not list other methods [6], [10], [28], [40], [46] whose performance is worse than A+ or CNN-L.

It can be seen from Table IV that CSCN performs consistently better than all previous methods in both PSNR and SSIM, and with multi-view testing the results can be further improved. CNN-L improves over CNN by increasing model parameters and training data. However, it is still not as good as CSCN which is trained with a much smaller size and on a much smaller data set. Clearly, the better model structure of CSCN makes it less dependent on model capacity and training data in improving performance. Our models are generally more advantageous for large scaling factors due to the cascade structure. A larger performance gain is observed on Set5 than the other two test sets because Set5 has more similar statistics as the training set.

The visual qualities of the SR results generated by sparse coding (SC) [6], CNN and CSCN are compared in Fig. 10. Our approach produces image patterns with shaper boundaries and richer textures, and is free of the ringing artifacts observable in the other two methods.

Fig. 11 shows the SR results on the "chip" image compared among more methods including the self-example based method (SE) [9] and the deep network cascade (DNC) [14]. SE and DNC can generate very sharp edges on this image, but also introduce artifacts and blurs on corners and fine structures due to the lack of self-similar patches. On the contrary, the CSCN method recovers all the structures of the characters without any distortion.

D. Robustness to Real SR Scenarios

We evaluate the performance of the proposed practical SR methods in Sec. IV, by providing the empirical results of several experiments for the two aforementioned approaches.

1) Data-Driven SR by Fine-Tuning: The proposed method in Sec. IV-A is data-driven, and thus the generic SCN can be easily adapted for a particular task, with a small amount of training samples. We demonstrate the performance of this method in the application of enlarging low-DPI scanned document images with heavy noise. We first obtain several pairs of

[2]Since one out of 200 training images coincides with one image in Set5, we exclude it from our training set.

Fig. 10. SR results given by SC [6] (first row), CNN [16] (second row) and our CSCN (third row). Images from left to right: the "monarch" image upscaled by $\times 3$; the "zebra" image upscaled by $\times 3$; the "comic" image upscaled by $\times 3$.

LR and HR images by scanning a document under two settings of 150DPI and 300DPI. Then we fine-tune our generic CSCN model using only one pair of scanned images for a few iterations. Fig. 13 illustrates the visualization of the upscaled image from the 150DPI scanned image. As shown by the SR results in Fig. 13, the CSCN before adaptation is very sensitive to LR measurement corruption, so the enlarged texts in (b) are much more corrupted than they are in the nearest neighbor upscaled image (a). However, the adapted CSCN model removes almost all the artifacts and can restore clear texts in (c), which is promising for practical applications such as quality enhancement of online scanned books and restoration of legacy documents.

2) Regularized Iterative SR: We now show experimental results of practical SR for blurred and noisy LR images, using the proposed regularized iterative methods in Sec. IV-B. We first compare the SR performance on blurry images using the proposed method in Sec. IV-B1 with several other recent methods [47]–[49], using the same test images and settings. All these methods are designed for blurry LR input, while our model is trained on sharp LR input. As shown in Table V, our model achieves much better results than the competitors. Note the speed of our model is also much faster than the conventional sparse coding based methods.

To test the performance of upscaling noisy LR images, we simulate additive Gaussian noise for the LR input images at 4 different noise levels ($\sigma = 5$, 10, 15, 20) as the noisy input images. We compare the practical SR results in Set5 obtained from the following algorithms: directly using SCN, our proposed iterative SCN method using BM3D as denoising regularizer (iterative BM3D-SCN), and fine-tuning SCN with additional noisy training pairs. Note that knowing the underlying corruption model of real LR image (e.g., noise distribution or blurring kernel), one can always synthesizes real training pairs for fine-tuning the generic SCN. In other words, once the iterative SR method is feasible, one can always apply our proposed data-driven method for SR alternatively. However, the other way around is not true. Therefore, the knowledge of the corruption model of real measurements can be considered as a stronger assumption, compared to providing real training image pairs. Correspondingly, the SR performances of these two methods are evaluated when both can be applied. We also provide the results of methods directly using another generic SR model: CNN-L [42], and the similar iterative SR method involving CNN-L (iterative BM3D-CNN-L).

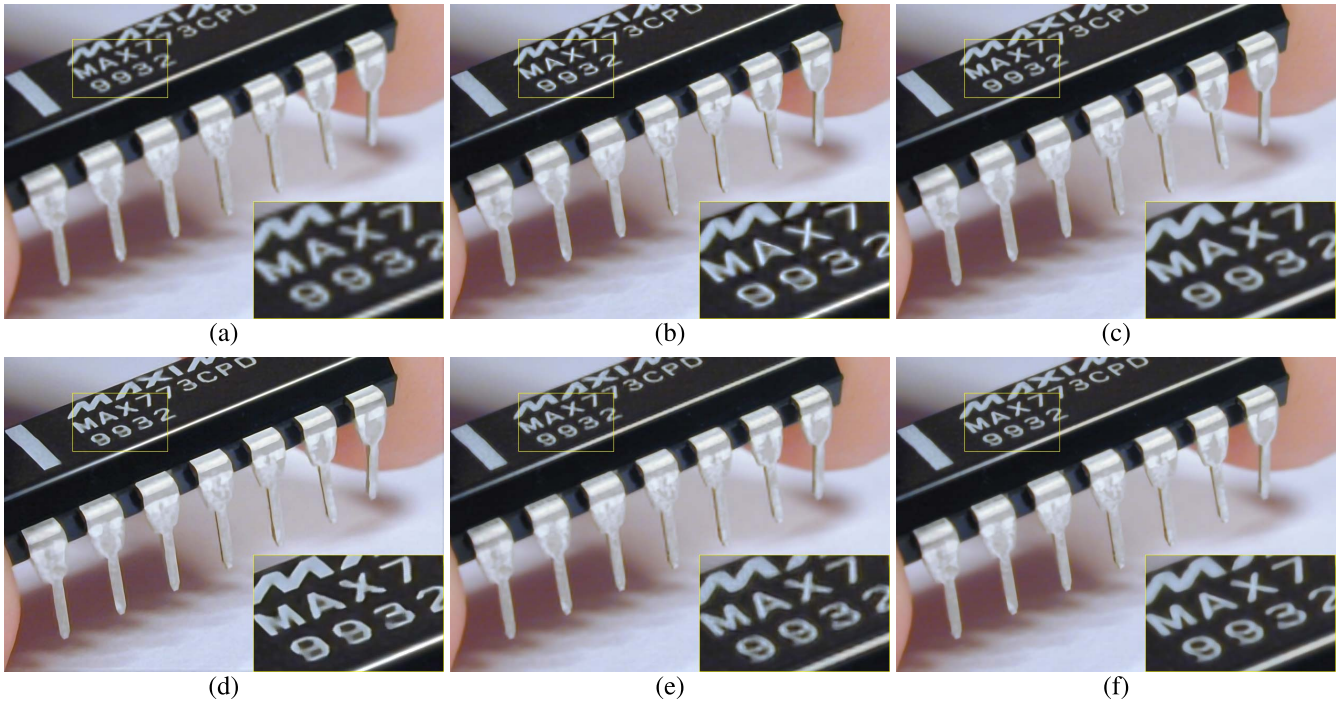


Fig. 11. The "chip" image upscaled by $\times 4$ times using different methods. (a) Bicubic. (b) SE [9]. (c) SC [6]. (d) DNC [14]. (e) CNN [16]. (f) CSCN.

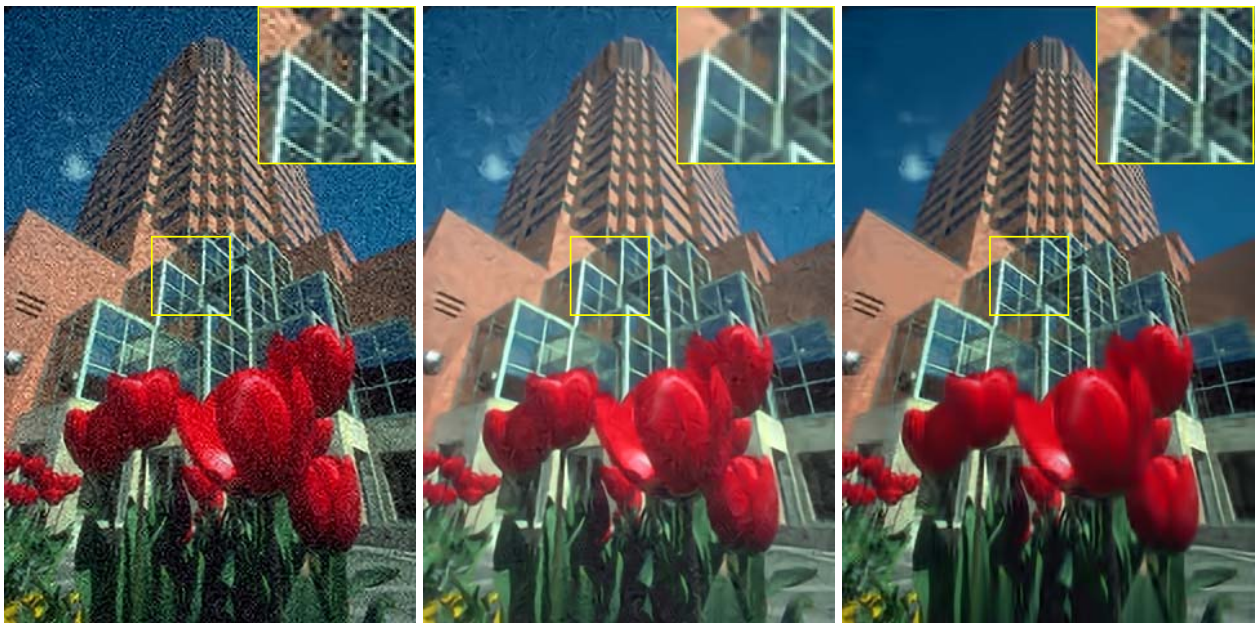


Fig. 12. The "building" image corrupted by additive Gaussian noise of $\sigma = 10$ and then upscaled by $\times 2$ times using different methods. (a) Direct SCN PSNR=24.00dB. (b) Fine-tuning SCN PSNR=27.54dB. (c) Iterative BM3D-SCN PSNR=27.86dB.

The practical SR results are listed in Table VI. We observed the improved PSNR using our proposed regularized iterative SR method over all noise levels. The proposed iterative BM3D-SCN achieves much higher PSNR than the method of directly using SCN. The performance gap (in terms of SR PSNR) between iterative BM3D-SCN and direct SCN becomes larger, as the noise level increases. Similar observation can be found in the result comparison of iterative BM3D-CNN-L and direct CNN-L. Compared to the method of fine-tuning SCN, the iterative BM3D-SCN method demonstrates better empirical performance, with 0.3 dB improvement on average. The iterative BM3D-CNN-L method provides comparable results, compared to the iterative BM3D-SCN method, which demonstrates that our proposed regularized iterative SCN scheme can be easily extended for other SR methods, and is able to effectively handle noisy LR measurements.

An example of upscaling noisy LR images using the aforementioned methods is demonstrated in Fig. 12. Both find-tuning SCN and iterative BM3D-SCN are able to significantly

TABLE V
PSNR OF ×3 UPSCALING ON LR IMAGES WITH DIFFERENT BLURRING KERNELS

Kernel	Gaussian $\sigma = 1.0$			Gaussian $\sigma = 1.6$		
Method	CSR [47]	NLM [48]	SCN	CSR [47]	GSC [49]	SCN
Butterfly	27.87	26.93	**28.70**	28.19	25.48	**29.03**
Parrots	30.17	29.93	**30.75**	30.68	29.20	**30.83**
Parthenon	26.89	–	**27.06**	27.23	26.44	**27.40**
Bike	24.41	24.38	**24.81**	24.72	23.78	**25.11**
Flower	29.14	28.86	**29.50**	29.54	28.30	**29.78**
Girl	**33.59**	33.44	33.57	**33.68**	33.13	33.65
Hat	31.09	30.81	**31.32**	31.33	30.29	**31.62**
Leaves	26.99	26.47	**27.45**	27.60	24.78	**27.87**
Plants	33.92	33.27	**34.35**	34.00	32.33	**34.53**
Raccoon	**29.09**	–	28.99	**29.29**	28.81	29.16
Average	29.32	29.26	**29.65**	29.63	28.25	**29.90**

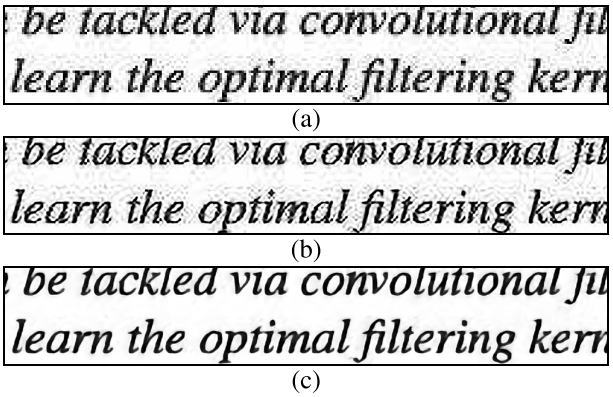


Fig. 13. Low-DPI scanned document upscaled by ×4 times using different methods. (a) Nearest neighbor. (b) CSCN. (c) Adapted CSCN.

TABLE VI
PSNR VALUES FOR ×2 UPSCALING NOISY LR IMAGES IN Set5 BY DIRECTLY USING SCN (DIRECT SCN), DIRECTLY USING CNN-L (DIRECT CNN-L), SCN AFTER FINE-TUNING ON NEW NOISY TRAINING DATA (FINE-TUNING SCN), THE ITERATIVE METHOD OF BM3D & SCN (ITERATIVE BM3D-SCN), AND THE ITERATIVE METHOD OF BM3D & CNN-L (ITERATIVE BM3D-CNN-L)

σ	5	10	15	20
Direct SCN	30.23	25.11	21.81	19.45
Direct CNN-L	30.47	25.32	21.91	19.46
Fine-tuning SCN	33.03	31.00	29.46	28.44
Iterative BM3D-SCN	**33.51**	**31.22**	**29.65**	**28.61**
Iterative BM3D-CNN-L	33.42	31.16	29.62	28.59

suppress the additive noise, while many artifacts induced by noise are observed in the SR result of direct SCN. It is notable that the fine-tuning SCN method performs better recovering the texture and the iterative BM3D-SCN method is preferable in smooth regions.

E. Subjective Evaluation

We have a total of 270 participants giving 720 pairwise comparisons over 6 images with different scaling factors, which are shown in Fig. 6. Not every participant completed all the comparisons but their partial responses are still useful.

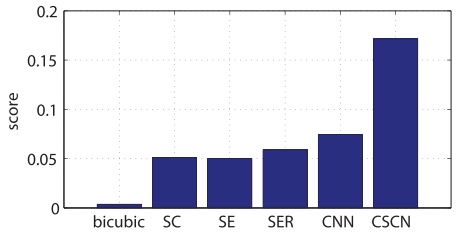


Fig. 14. Subjective SR quality scores for different methods including bicubic, SC [6], SE [9], SER [37], CNN [16] and the proposed CSCN. The score for ground truth result is 1.

Fig. 14 shows the estimated scores for the 6 SR methods in our evaluation, with the score for ground truth method normalized to 1. As expected, all the SR methods have much lower scores than ground truth, showing the great challenge in SR problem. The bicubic interpolation is significantly worse than other SR methods. The proposed CSCN method outperforms other previous state-of-the-art methods by a large margin, demonstrating its superior visual quality. It should be noted that the visual difference between some image pairs is very subtle. Nevertheless, the human subjects are able to perceive such difference when seeing the two images side by side, and therefore make consistent ratings. The CNN model becomes less competitive in the subjective evaluation than it is in PSNR comparison. This indicates that the visually appealing image appearance produced by CSCN should be attributed to the regularization from sparse representation, which can not be easily learned by merely minimizing reconstruction error as in CNN.

VII. CONCLUSIONS

We propose a new model for image SR by combining the strengths of sparse coding and deep network, and make considerable improvement over existing deep and shallow SR models both quantitatively and qualitatively. Besides producing good SR results, the domain knowledge in the form of sparse coding can also benefit training speed and model compactness. Furthermore, we investigate the cascade of network for both fixed and incremental scaling factors so as to enhance

SR performance. In addition, the robustness to real SR scenarios is discussed for handling non-ideal LR measurements. More generally, our observation is in line with other recent extensions made to CNN with better domain knowledge for different tasks.

In future work, we will apply the SCN model to other problems where sparse coding can be useful. The interaction between deep networks for low-level and high-level vision tasks, such as [50], will also be explored.

REFERENCES

[1] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1167–1183, Sep. 2002.

[2] Z. Lin and H.-Y. Shum, "Fundamental limits of reconstruction-based superresolution algorithms under local translation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 83–97, Jan. 2004.

[3] R. Fattal, "Image upsampling via imposed edge statistics," *ACM Trans. Graph.*, vol. 26, no. 3, 2007, Art. no. 95.

[4] H. A. Aly and E. Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Trans. Image Process.*, vol. 14, no. 10, pp. 1647–1659, Oct. 2005.

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

[6] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE Trans. Image Process.*, vol. 21, no. 8, pp. 3467–3478, Aug. 2012.

[7] X. Gao, K. Zhang, D. Tao, and X. Li, "Image super-resolution with sparse neighbor embedding," *IEEE Trans. Image Process.*, vol. 21, no. 7, pp. 3194–3205, Jul. 2012.

[8] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. ICCV*, Sep./Oct. 2009, pp. 349–356.

[9] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Trans. Graph.*, vol. 30, no. 2, 2011, Art. no. 12.

[10] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1127–1133, Jun. 2010.

[11] C. Deng, J. Xu, K. Zhang, D. Tao, X. Gao, and X. Li, "Similarity constraints-based structured output regression machine: An approach to image super-resolution," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

[12] C.-Y. Yang, C. Ma, and M.-H. Yang, "Single-image super-resolution: A benchmark," in *Proc. ECCV*, 2014, pp. 372–386.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[14] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *Proc. ECCV*, 2014, pp. 49–64.

[15] Z. Wang *et al.*, "Self-tuned deep super resolution," in *Proc. IEEE Conf. CVPR Workshops*, Jun. 2015, pp. 1–8.

[16] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. ECCV*, 2014, pp. 184–199.

[17] C. Osendorfer, H. Soyer, and P. van der Smagt, "Image super-resolution with fast approximate convolutional sparse coding," in *Neural Information Processing*. New York, NY, USA: Springer, 2014, pp. 250–257.

[18] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. ICML*, 2010, pp. 399–406.

[19] B. Wen, S. Ravishankar, and Y. Bresler, "Structured overcomplete sparsifying transform learning with convergence guarantees and applications," *Int. J. Comput. Vis.*, vol. 114, no. 2, pp. 137–167, 2015.

[20] Z. Wang *et al.*, *Sparse Coding and Its Applications in Computer Vision*. Singapore: World Scientific, 2015.

[21] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. CVPR*, Dec. 2015, pp. 370–378.

[22] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. (2010). "Fast inference in sparse coding algorithms with applications to object recognition." [Online]. Available: http://arxiv.org/abs/1010.3467

[23] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004.

[24] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Comput.*, vol. 20, no. 10, pp. 2526–2563, 2008.

[25] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: http://arxiv.org/abs/1312.4400

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.

[27] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proc. ACM SIGKDD*, 2015, pp. 119–128.

[28] R. Timofte, V. De, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. ICCV*, Dec. 2013, pp. 1920–1927.

[29] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proc. IEEE ICASSP*, May 2013, pp. 8595–8598.

[30] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. CVPR*, Jun. 2014, pp. 1717–1724.

[31] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. CVPR*, Jun. 2005, pp. 60–65.

[32] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[33] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[34] B. Wen, S. Ravishankar, and Y. Bresler, "Video denoising by online 3D sparsifying transform learning," in *Proc. IEEE ICIP*, Sep. 2015, pp. 118–122.

[35] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning—Part I: Algorithms," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 625–636, Jun. 2015.

[36] Z. Wang, Y. Yang, Z. Wang, S. Chang, J. Yang, and T. S. Huang, "Learning super-resolution jointly from external and internal examples," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4359–4371, Nov. 2015.

[37] J. Yang, Z. Lin, and S. Cohen, "Fast image super-resolution based on in-place example regression," in *Proc. CVPR*, Jun. 2013, pp. 1059–1066.

[38] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. The method of paired comparisons," *Biometrika*, vol. 39, nos. 3–4, pp. 324–345, 1952.

[39] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. BMVC*, 2012, pp. 135.1–135.10.

[40] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces*. Heidelberg, Germany: Springer, 2012, pp. 711–730.

[41] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. ICCV*, vol. 2. Jul. 2001, pp. 416–423.

[42] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.

[43] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. ACCV*, 2014, pp. 111–126.

[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.

[45] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[46] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. CVPR*, Jun. 2015, pp. 5197–5206.

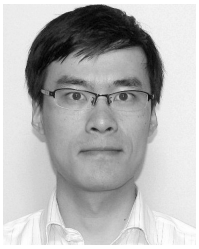
[47] W. Dong, L. Zhang, and G. Shi, "Centralized sparse representation for image restoration," in *Proc. ICCV*, Nov. 2011, pp. 1259–1266.

[48] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4544–4556, Nov. 2012.

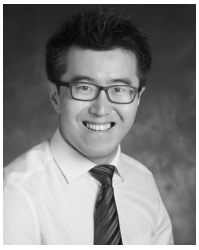
[49] X. Lu, H. Yuan, P. Yan, Y. Yuan, and X. Li, "Geometry constrained sparse coding for single image super-resolution," in *Proc. CVPR*, Jun. 2012, pp. 1648–1655.

[50] Z. Wang, S. Chang, Y. Yang, D. Liu, and T. S. Huang, "Studying very low resolution recognition using deep networks," in *Proc. CVPR*, 2016, pp. 1–9.

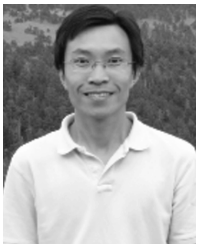
Ding Liu (S'15) received the B.S. degree from the Chinese University of Hong Kong, Hong Kong, in 2012, and the M.S. degree from the University of Illinois at Urbana–Champaign, USA, in 2014, where he is currently pursuing the Ph.D. degree under the supervision of Prof. T. S. Huang. His research experience encompasses using deep learning to solve low-level vision problems, including image superresolution, image restoration, and image denoising. He has research interests in the broad areas of computer vision, image processing, and deep learning.



Zhaowen Wang (M'14) received the B.E. and M.S. degrees from Shanghai Jiao Tong University, China, in 2006 and 2009, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, in 2014. He is currently a Research Scientist with the Imagination Laboratory, Adobe Systems Inc. His research has been focused on understanding and enhancing images via machine learning algorithms, with a special interest in sparse coding and deep learning.



Bihan Wen received the B.Eng. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2012, and the M.S. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2015, where he is currently pursuing the Ph.D. degree. His current research interests include signal and image processing, machine learning, sparse representation, and big data applications.



Jianchao Yang (M'14) received the B.E. degree from the University of Science and Technology of China, in 2006, and the Ph.D. degree from the Electrical and Computer Engineering Department, University of Illinois at Urbana–Champaign, in 2011, under the supervision of Prof. T. S. Huang. He is currently a Research Scientist with Snapchat Inc., Venice, CA. In particular, he has extensive experience in the following research areas, such as image categorization, object recognition and detection, and image retrieval; image and video superresolution, denoising, and deblurring; face recognition and soft biometrics; sparse coding and sparse representation; and unsupervised learning, supervised learning, and deep learning. His research interests are in the broad areas of computer vision, machine learning, and image processing.

Wei Han received the B.Eng. and M.S. degrees from the Department of Computer Science, Shanghai Jiao Tong University, China, in 2009 and 2012, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign. His research interests include computer vision with a focus on image object detection and recognition, and video event detection.

Thomas S. Huang (F'01) received the B.S. degree from National Taiwan University, Taipei, Taiwan, and the M.S. and D.Sc. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, all in electrical engineering. He was a Faculty Member with the Department of Electrical Engineering, MIT, from 1963 to 1973, and a Faculty Member with the School of Electrical Engineering and the Director of its Laboratory for Information and Signal Processing with Purdue University from 1973 to 1980. In 1980, he joined the University of Illinois at Urbana–Champaign, where he is a William L. Everitt Distinguished Professor of Electrical and Computer Engineering, and a Research Professor with the Coordinated Science Laboratory, and at the Beckman Institute for Advanced Science, he is Technology and Co-Chair of the Institutes major research theme Human Computer Intelligent Interaction. His professional interests lie in the broad areas of information technology, especially the transmission and processing of multidimensional signals. He has published 21 books, and over 600 papers in network theory, digital filtering, image processing, and computer vision. He is a member of the National Academy of Engineering and the Academia Sinica, China, a Foreign Member of the Chinese Academies of Engineering and Sciences, and a fellow of the International Association of Pattern Recognition and the Optical Society of America. Among his many honors and awards, such as the Honda Lifetime Achievement Award, the IEEE Jack Kilby Signal Processing Medal, and the King-Sun Fu Prize of the International Association for Pattern Recognition.