Deep Networks for Image Super-Resolution with Sparse Prior

Zhaowen Wang[†‡] Ding Liu[†] Jianchao Yang[§] Wei Han[†] Thomas Huang[†]
[†]Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL
[‡]Adobe Research, San Jose, CA [§]Snapchat, Venice, CA
[‡]zhawang@adobe.com [†]{dingliu2, weihan3, huang}@ifp.uiuc.edu [§]jcyangenator@gmail.com

Abstract

Deep learning techniques have been successfully applied in many areas of computer vision, including low-level image restoration problems. For image super-resolution, several models based on deep neural networks have been recently proposed and attained superior performance that overshadows all previous handcrafted models. The question then arises whether large-capacity and data-driven models have become the dominant solution to the ill-posed super-resolution problem. In this paper, we argue that domain expertise represented by the conventional sparse coding model is still valuable, and it can be combined with the key ingredients of deep learning to achieve further improved results. We show that a sparse coding model particularly designed for super-resolution can be incarnated as a neural network, and trained in a cascaded structure from end to end. The interpretation of the network based on sparse coding leads to much more efficient and effective training, as well as a reduced model size. Our model is evaluated on a wide range of images, and shows clear advantage over existing state-of-the-art methods in terms of both restoration accuracy and human subjective quality.

1. Introduction

Single image super-resolution (SR) aims at obtaining a high-resolution (HR) image from a low-resolution (LR) input image by inferring all the missing high frequency contents. With the known variables in LR images greatly outnumbered by the unknowns in HR images, SR is a highly ill-posed problem and the current techniques are far from being satisfactory for many real applications [2, 21].

To regularize the solution of SR, people have exploited various priors of natural images. Analytical priors, such as bicubic interpolation, work well for smooth regions; while image models based on statistics of edges [11] and gradients [17, 1] 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 [36, 35], or recovered from similar examples in the LR image itself at different locations and across different scales [13, 12, 32]. A comprehensive review of more SR methods can be found in [33].

More recently, inspired by the great success achieved by deep learning [18, 27, 30] 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 [6] for robust matching of self-similar patches. Deep convolutional neural networks (CNN) [8] and deconvolutional networks [25] are designed that directly learn the non-linear mapping from LR space to HR space in a way similar to coupled sparse coding [35]. 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 [6, 8] 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 worthy 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 [36] 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) [14], we implement a feed-forward neural network whose layers strictly correspond to each 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 [8] in terms

of both recovery accuracy and human perception, and yet has a compact 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 [8], different networks are trained for different scaling factors. In this paper, we also propose a cascade of multiple SCNs to achieve SR for arbitrary factors. This simple approach, motivated by the self-similarity based SR approach [13], not only increases the scaling flexibility of our model, but also reduces artifacts for large scaling factors. 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 short, the contributions of this paper include:

- combine the domain expertise of sparse coding and the merits of deep learning to achieve better SR performance with faster training and smaller model size;
- use network cascading for large and arbitrary scaling factors;
- conduct a subjective evaluation on several recent state-of-the-art methods.

In the following, we will first review related work in Sec. 2. The SCN and CSCN models are introduced in Sec. 3 and Sec. 4, with implementation details in Sec. 5. Extensive experimental results are reported in Sec. 6, and conclusions are drawn in Sec. 7.

2. Related Work

2.1. Image SR Using Sparse Coding

The sparse representation based SR method [36] 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

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

where $\|\cdot\|_1$ denotes the ℓ_1 norm which is convex and sparsity-inducing, and λ is a regularization coefficient. The dictionary pair $(\boldsymbol{D}_y, \boldsymbol{D}_x)$ can be learned alternatively with the inference of training patches' sparse codes in their joint space [36] or through bi-level optimization [35].

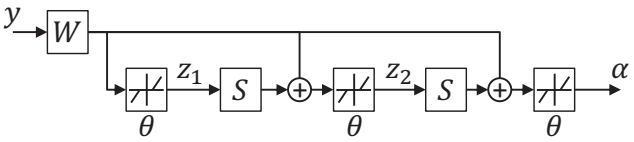


Figure 1. A LISTA network [14] with 2 time-unfolded recurrent stages, whose output $\boldsymbol{\alpha}$ is an approximation of the sparse code of input signal $\boldsymbol{y}$. The linear weights $\boldsymbol{W}$, $\boldsymbol{S}$ and the shrinkage thresholds $\boldsymbol{\theta}$ are learned from data.

2.2. Network Implementation of Sparse Coding

There is an intimate connection between sparse coding and neural network, which has been well studied in [16, 14]. A feed-forward neural network as illustrated in Fig. 1 is proposed in [14] 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 recurrent stages, each of which updates the intermediate sparse code according to

$$\boldsymbol{z}_{k+1} = h_{\boldsymbol{\theta}}(\boldsymbol{W}\boldsymbol{y} + \boldsymbol{S}\boldsymbol{z}_k), \quad (2)$$

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) [7, 26] 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 [14] 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.

3. Sparse Coding based Network for Image SR

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 [36]. 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 $\boldsymbol{S} \in \mathbb{R}^{n \times n}$,

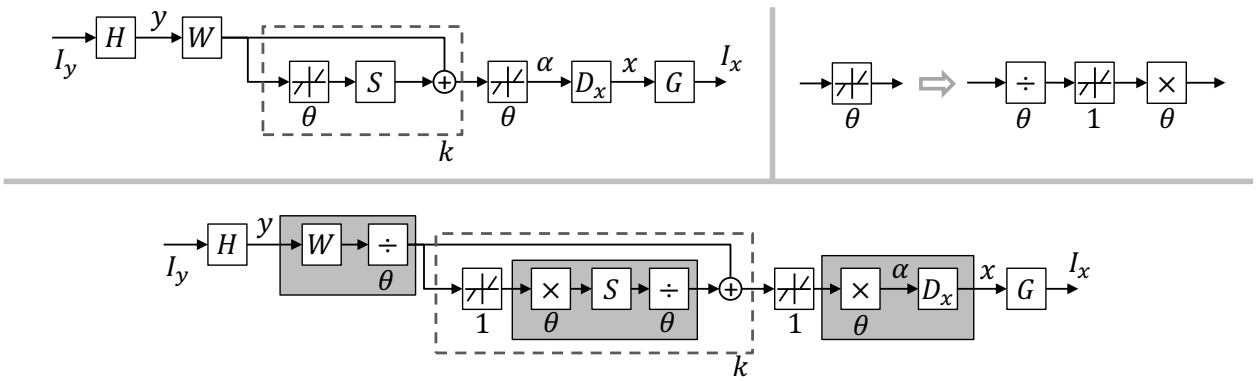


Figure 2. Top left: the proposed SCN model with a patch extraction layer $\boldsymbol{H}$, a LISTA sub-network for sparse coding (with k recurrent stages denoted by the dashed box), a HR patch recovery layer $\boldsymbol{D}_x$, and a patch combination layer $\boldsymbol{G}$. Top right: a neuron with 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.

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

$$[h_{\boldsymbol{\theta}}(\boldsymbol{a})]_i = \text{sign}(a_i)\theta_i(|a_i|/\theta_i - 1)_+ = \theta_i h_1(a_i/\theta_i). \quad (3)$$

Eq. (3) 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. 5.

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 [20]. 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

$$\min_{\boldsymbol{\Theta}} \sum_i \|SCN(\boldsymbol{I}_y^{(i)}; \boldsymbol{\Theta}) - \boldsymbol{I}_x^{(i)}\|_2^2, \quad (4)$$

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.

Advantages over Previous Models

The construction of our SCN follows exactly each step in the sparse coding based SR method [36]. If the network parameters are set according to the dictionaries learned in [36], 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 [36] 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 [8] 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 [8] a generic rectified linear unit (ReLU) [24] 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.

4. 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 [13, 12, 6], 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 (CSCN) trained for small scaling factors can generate even better SR results than a single SCN trained for a large 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). Therefore, each SCN in the cascade serves as a "relaying station" which progressively recovers some useful information lost in bicubic interpolation and compensates for the distortion aggregated from previous stages.

The CSCN is also a deep network, in which the output of each SCN 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 (4), we can also optimize all of them jointly as shown in Fig. 4. Without loss of generality, we assume each SCN in the cascade has the same scaling factor s. Let $\boldsymbol{I}_0$ denote the input image of original size, and $\hat{\boldsymbol{I}}_j$ $(j>0)$ denote the output image of the j-th SCN upscaled by a total of $\times s^j$ times. Each $\hat{\boldsymbol{I}}_j$ can be compared with its associated ground truth image $\boldsymbol{I}_j$ according to the MSE cost, leading to a multi-scale objective function:

$$\min_{\{\boldsymbol{\Theta}_j\}} \sum_i \sum_j \left\| SCN(\hat{\boldsymbol{I}}_{j-1}^{(i)}\uparrow s; \boldsymbol{\Theta}_j) - \boldsymbol{I}_j^{(i)} \right\|_2^2, \qquad (5)$$

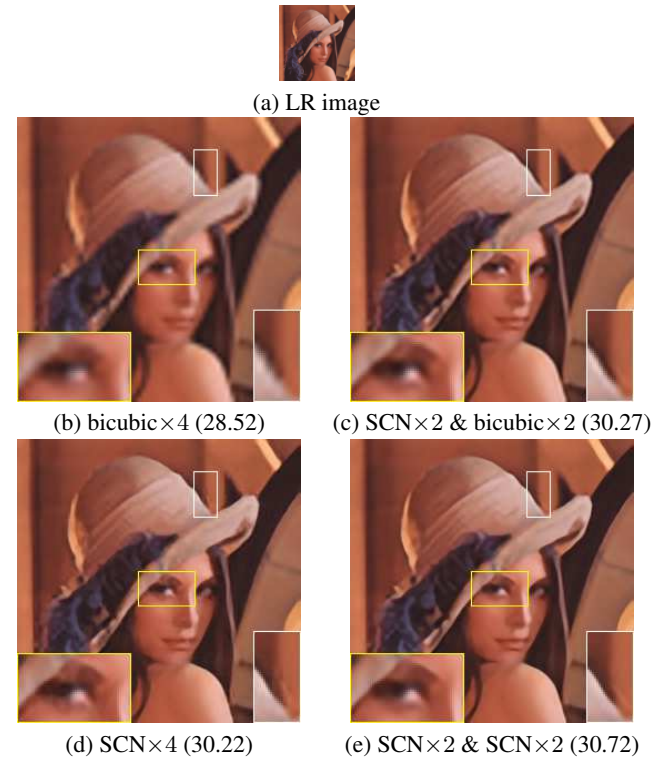


Figure 3. SR results for the "Lena" image upscaled by 4 times. (a) $\rightarrow$ (b) $\rightarrow$ (d) represents the processing flow with a single SCN$\times$4 model. (a) $\rightarrow$ (c) $\rightarrow$ (e) represents the processing flow with two cascaded SCN$\times$2 models. PSNR is given in parentheses.

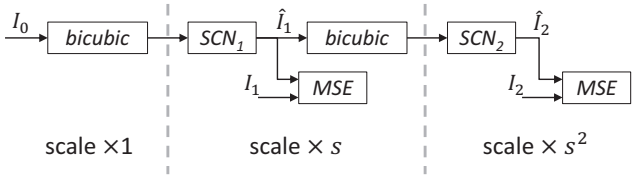


Figure 4. Training cascade of SCNs with multi-scale objectives.

where i denotes the data index, and j denotes the SCN index. $\boldsymbol{I}\uparrow s$ is the bicubic interpolated image of $\boldsymbol{I}$ by a factor of s. This multi-scale objective function makes full use of the supervision information in all scales, sharing a similar idea as heterogeneous networks [19, 5]. All the layer parameters $\{\boldsymbol{\Theta}_j\}$ in (5) could be optimized from end to end by back-propagation. We use a greedy algorithm here to train each SCN 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.

5. Implementation Details

We determine the number of nodes in each layer of our SCN mainly according to the corresponding settings used in sparse coding [35]. 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 [7]:

$$\boldsymbol{w}_1 = C{\cdot}\boldsymbol{D}_y^T,\ \boldsymbol{w}_2 = \boldsymbol{I}{-}\boldsymbol{D}_y^T\boldsymbol{D}_y,\ \boldsymbol{w}_3 = (CL)^{-1}{\cdot}\boldsymbol{D}_x \quad (6)$$

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 [18] 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 [8] 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. To achieve arbitrary upscaling 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. 6.2, 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 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.

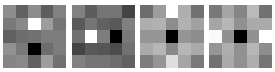


Figure 5. The four learned filters in the first layer $\boldsymbol{H}$.

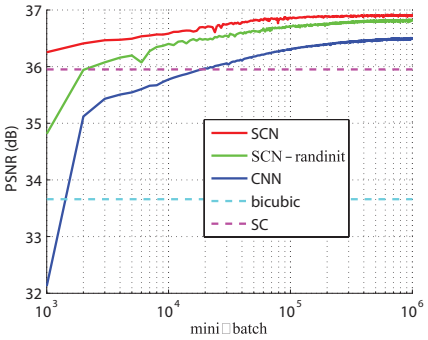


Figure 6. The PSNR change for ×2 SR on Set5 during training using different methods: SCN; SCN with random initialization; CNN. The horizontal dash lines show the benchmarks of bicubic interpolation and sparse coding (SC).

6. 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 [3], Set14 [37] and BSD100 [23] which contain 5, 14 and 100 images respectively. We have also trained on a different larger data set, and observe little performance change (less than 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.

6.1. Algorithm Analysis

We first visualize the four filters learned in the first layer $\boldsymbol{H}$ in Fig. 5. 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. 6. Our SCN improves significantly over sparse coding (SC) [35], as it leverages data more effectively with end-to-end training. The SCN initialized according to (6) can converge faster and better than the same model with random initialization, which indicates that

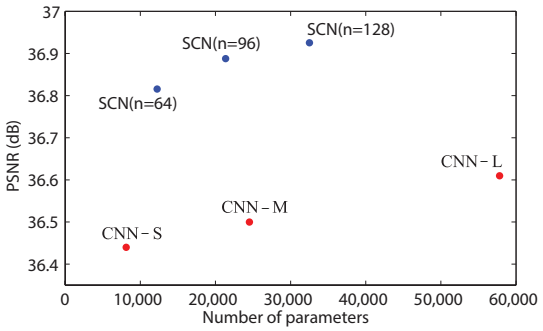


Figure 7. PSNR for $\times 2$ SR on Set5 using SCN and CNN with various network sizes.

Table 1. PSNR of different network cascading schemes on Set5, evaluated for different scaling factors in each column.

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

the understanding of SCN based on sparse coding can help its optimization. We also train a CNN model [8] of the same size as SCN, but find its convergence speed much slower. It is reported in [8] 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. 7. 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. Fig. 7 also shows the performance of CNN with various sizes. Our smallest SCN can achieve higher PSNR than the largest model (CNN-L) in [9] while only using about 20% parameters.

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 1, different network cascade structures (in each row) are compared at different scaling factors (in each column). SCN$\times a$ denotes the simple cascade of SCN with fixed scaling factor a, where an individually trained SCN is applied one or more times for scaling factors other than a. It is observed that SCN$\times 2$ 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$\times 1.5$ does not lead to good results since artifacts quickly get amplified through many repetitive up-scalings. Therefore, we use SCN$\times 2$ as the default building block for CSCN, and drop the notation $\times 2$ when there is no ambiguity. The last row in Table 1 shows that a CSCN trained using the multi-scale objective in (5) 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.

6.2. 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 upscaling factors. Table 2 shows the PSNR and structural similarity (SSIM) [31] for adjusted anchored neighborhood regression (A+) [29], CNN [8], CNN trained with larger model size and more data (CNN-L) [9], the proposed CSCN, and CSCN with our multi-view testing (CSCN-MV). We do not list other methods [35, 28, 37, 17, 15] whose performance is worse than A+ or CNN-L.

It can be seen from Table 2 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.

The visual qualities of the SR results generated by sparse coding (SC) [35], CNN and CSCN are compared in Fig. 8. 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. 9 shows the SR results on the "chip" image compared among more methods including the self-example based method (SE) [12] and the deep network cascade (DNC) [6]. 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.

We also compare CSCN with other sparse coding extensions [22, 10, 38], and consider the blurring effect introduced in downscaling. A PSNR gain of 0.3$\sim$1.6dB is achieved by CSCN in general. Experiment details and source codes are available online[1].

6.3. Subjective Evaluation

We conducted a subjective evaluation of SR results for several methods including bicubic, SC [35], SE [12],

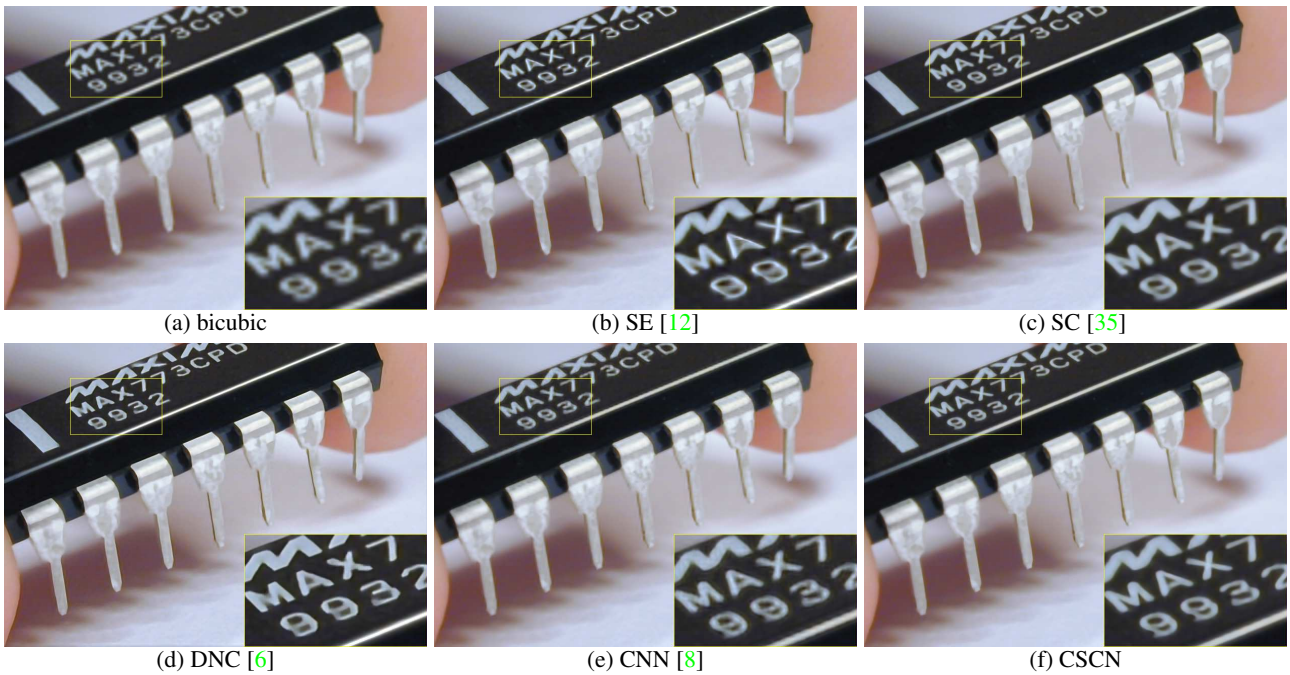
[1] www.ifp.illinois.edu/~dingliu2/iccv15

Table 2. 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+ [29]	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 [8]	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 [9]	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	36.93 (0.9552)	33.10 (0.9144)	30.86 (0.8732)	32.56 (0.9074)	29.41 (0.8238)	27.64 (0.7578)	31.40 (0.8884)	28.50 (0.7885)	27.03 (0.7161)
CSCN-MV	37.14 (0.9567)	33.26 (0.9167)	31.04 (0.8775)	32.71 (0.9095)	29.55 (0.8271)	27.76 (0.7620)	31.54 (0.8908)	28.58 (0.7910)	27.11 (0.7191)
Our Improvement	0.48 (0.0023)	0.51 (0.0077)	0.55 (0.0147)	0.26 (0.0028)	0.25 (0.0056)	0.26 (0.0107)	0.18 (0.0029)	0.17 (0.0047)	0.21 (0.0088)

Figure 8. SR results given by SC [35] (first row), CNN [8] (second row) and our CSCN (third row). Images from left to right: the "monarch" image upscaled by ×3; the "zebra" image upscaled by ×3; the "comic" image upscaled by ×3.

self-example regression (SER) [34], CNN [8] and CSCN. Ground truth HR images are also included when they are available as references. Each of the participants in the evaluation is shown a set of HR image pairs, which are



(a) bicubic (b) SE [12] (c) SC [35]

(d) DNC [6] (e) CNN [8] (f) CSCN

Figure 9. The "chip" image upscaled by $\times 4$ times using different methods.

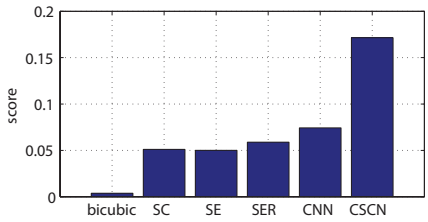


Figure 10. Subjective SR quality scores for different methods including bicubic, SC [35], SE [12], SER [34], CNN [8] and the proposed CSCN. The score for ground truth result is 1.

upscaled from the same LR images using two randomly selected methods. For each pair, the subject needs to decide which one is better in terms of perceptual quality.

We have a total of 270 participants giving 720 pairwise comparisons over 6 images with different scaling factors. Not every participant completed all the comparisons but their partial responses are still useful. All the evaluation results can be summarized into a 7×7 winning matrix for 7 methods (including ground truth), based on which we fit a Bradley-Terry [4] model to estimate the subjective score for each method so that they can be ranked.

Fig. 10 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.

7. 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 propose a cascaded network for better flexibility in scaling factors as well as more robustness to artifacts.

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 will also be explored.

References

[1] H. A. Aly and E. Dubois. Image up-sampling using total-variation regularization with a new observation model. *IEEE TIP*, 14(10):1647–1659, 2005. 1

[2] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE TPAMI*, 24(9):1167–1183, 2002. 1

[3] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012. 5

[4] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, pages 324–345, 1952. 8

[5] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *ACM SIGKDD*. ACM, 2015. 4

[6] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen. Deep network cascade for image super-resolution. In *ECCV*, pages 49–64, 2014. 1, 4, 6, 8

[7] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004. 2, 5

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199, 2014. 1, 2, 3, 4, 5, 6, 7, 8

[9] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 2015. 6, 7

[10] W. Dong, L. Zhang, and G. Shi. Centralized sparse representation for image restoration. In *ICCV*, pages 1259–1266, 2011. 6

[11] R. Fattal. Image upsampling via imposed edge statistics. In *ACM Transactions on Graphics*, volume 26:3, page 95, 2007. 1

[12] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics*, 30(2):12, 2011. 1, 4, 6, 8

[13] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009. 1, 2, 4

[14] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, 2010. 1, 2

[15] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 6

[16] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. *arXiv preprint arXiv:1010.3467*, 2010. 2

[17] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE TPAMI*, 32(6):1127–1133, 2010. 1, 6

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1, 5

[19] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*, 2014. 4

[20] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 3

[21] Z. Lin and H.-Y. Shum. Fundamental limits of reconstruction-based superresolution algorithms under local translation. *IEEE TPAMI*, 26(1):83–97, 2004. 1

[22] X. Lu, H. Yuan, P. Yan, Y. Yuan, and X. Li. Geometry constrained sparse coding for single image super-resolution. In *CVPR*, pages 1648–1655, 2012. 6

[23] 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 *ICCV*, volume 2, pages 416–423, July 2001. 5

[24] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, pages 807–814, 2010. 4

[25] C. Osendorfer, H. Soyer, and P. van der Smagt. Image super-resolution with fast approximate convolutional sparse coding. In *NIPS*, pages 250–257. Springer, 2014. 1

[26] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 20(10):2526–2563, 2008. 2

[27] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 1

[28] R. Timofte, V. De, and L. V. Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV*, pages 1920–1927, 2013. 5, 6

[29] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014. 6, 7

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008. 1

[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 6

[32] Z. Wang, Y. Yang, Z. Wang, S. Chang, J. Yang, and T. S. Huang. Learning super-resolution jointly from external and internal examples. *IEEE TIP*, 24(11):4359–4371, 2015. 1

[33] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: a benchmark. In *ECCV*, pages 372–386, 2014. 1

[34] J. Yang, Z. Lin, and S. Cohen. Fast image super-resolution based on in-place example regression. In *CVPR*, pages 1059–1066. IEEE, 2013. 7, 8

[35] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *IEEE TIP*, 21(8):3467–3478, 2012. 1, 2, 4, 5, 6, 7, 8

[36] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE TIP*, 19(11):1–8, 2010. 1, 2, 3

[37] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. 2012. 5, 6

[38] K. Zhang, X. Gao, D. Tao, and X. Li. Single image super-resolution with non-local means and steering kernel regression. *IEEE TIP*, 21(11):4544–4556, 2012. 6