Image Super-Resolution Using Deep Convolutional Networks

Chao Dong, Chen Change Loy, *Member, IEEE*, Kaiming He, *Member, IEEE*, and Xiaoou Tang, *Fellow, IEEE*

Abstract—We propose a deep learning method for single image super-resolution (SR). Our method directly learns an end-to-end mapping between the low/high-resolution images. The mapping is represented as a deep convolutional neural network (CNN) that takes the low-resolution image as the input and outputs the high-resolution one. We further show that traditional sparse-coding-based SR methods can also be viewed as a deep convolutional network. But unlike traditional methods that handle each component separately, our method jointly optimizes all layers. Our deep CNN has a lightweight structure, yet demonstrates state-of-the-art restoration quality, and achieves fast speed for practical on-line usage. We explore different network structures and parameter settings to achieve trade-offs between performance and speed. Moreover, we extend our network to cope with three color channels simultaneously, and show better overall reconstruction quality.

Index Terms—Super-resolution, deep convolutional neural networks, sparse coding

1 Introduction

Single image super-resolution (SR), which aims at recovering a high-resolution image from a single low-resolution image, is a classical problem in computer vision. This problem is inherently ill-posed since a multiplicity of solutions exist for any given low-resolution pixel. In other words, it is an underdetermined inverse problem, of which solution is not unique. Such a problem is typically mitigated by constraining the solution space by strong prior information. To learn the prior, recent state-of-the-art methods mostly adopt the example-based [44] strategy. These methods either exploit internal similarities of the same image [5], [13], [16], [19], [45], or learn mapping functions from external low- and high-resolution exemplar pairs [2], [4], [6], [15], [22], [24], [36], [39], [40], [45], [46], [48], [49]. The external example-based methods can be formulated for generic image super-resolution, or can be designed to suit domain specific tasks, i.e., face hallucination [29], [48], according to the training samples provided.

The sparse-coding-based (SC) method [47], [48] is one of the representative external example-based SR methods. This method involves several steps in its solution pipeline. First, overlapping patches are densely cropped from the input image and pre-processed (e.g., subtracting mean and normalization). These patches are then encoded by a low-resolution dictionary. The sparse coefficients are passed into a high-resolution dictionary for reconstructing high-resolution patches. The overlapping reconstructed patches are aggregated (e.g., by weighted averaging) to produce the final output. This pipeline is shared by most external example-based methods, which pay particular attention to learning and optimizing the dictionaries [2], [47], [48] or building efficient mapping functions [24], [39], [40], [45]. However, the rest of the steps in the pipeline have been rarely optimized or considered in an unified optimization framework.

In this paper, we show that the aforementioned pipeline is equivalent to a deep convolutional neural network [26] (more details in Section 3.2). Motivated by this fact, we consider a convolutional neural network that directly learns an end-to-end mapping between low- and high-resolution images. Our method differs fundamentally from existing external example-based approaches, in that ours does not explicitly learn the dictionaries [39], [47], [48] or manifolds [2], [4] for modeling the patch space. These are implicitly achieved via hidden layers. Furthermore, the patch extraction and aggregation are also formulated as convolutional layers, so are involved in the optimization. In our method, the entire SR pipeline is fully obtained through learning, with little pre/post-processing.

We name the proposed model super-resolution convolutional neural network (SRCNN).[1] The proposed SRCNN has several appealing properties. First, its structure is intentionally designed with simplicity in mind, and yet provides superior accuracy[2] compared with state-of-the-art example-based methods. Fig. 1 shows a comparison on an example. Second, with moderate numbers of filters and layers, our method achieves fast speed for practical on-line usage even on a CPU. Our method is faster than a number of

- *C. Dong, C. C. Loy, and X. Tang are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China. E-mail: {dc012, ccloy, xtang}@ie.cuhk.edu.hk.*
- *K. He is with the Visual Computing Group, Microsoft Research Asia, Beijing 100080, China. E-mail: kahe@microsoft.com.*

Manuscript received 30 Dec. 2014; revised 8 Apr. 2015; accepted 18 May 2015. Date of publication 31 May 2015; date of current version 13 Jan. 2016. Recommended for acceptance by M.S. Brown.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2015.2439281

1. The implementation is available at http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html.

2. Numerical evaluations by using different metrics such as the Peak Signal-to-Noise Ratio (PSNR), structure similarity index (SSIM) [41], multi-scale SSIM [42], information fidelity criterion (IFC) [37], when the ground truth images are available.

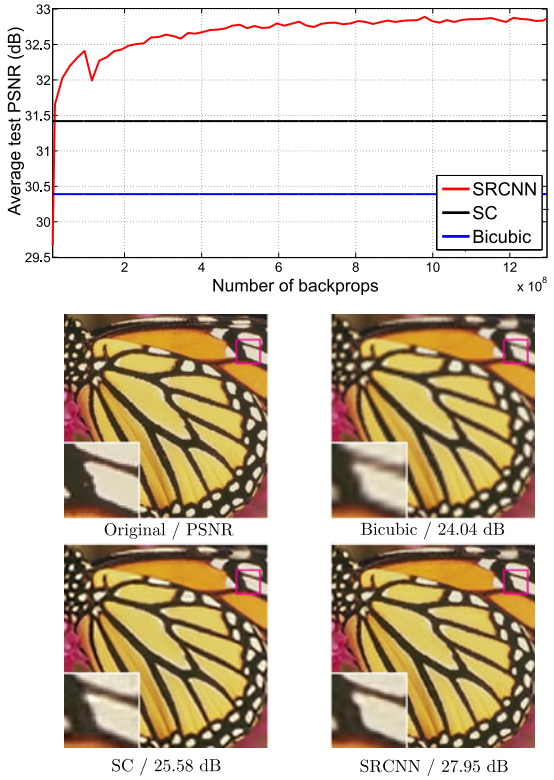


Fig. 1. The proposed super-resolution convolutional neural network surpasses the bicubic baseline with just a few training iterations, and outperforms the sparse-coding-based method [48] with moderate training. The performance may be further improved with more training iterations. More details are provided in Section 4.4.1 (the Set5 dataset with an upscaling factor 3). The proposed method provides visually appealing reconstructed image.

example-based methods, because it is fully feed-forward and does not need to solve any optimization problem on usage. Third, experiments show that the restoration quality of the network can be further improved when (i) larger and more diverse datasets are available, and/or (ii) a larger and deeper model is used. On the contrary, larger datasets/models can present challenges for existing example-based methods. Furthermore, the proposed network can cope with three channels of color images simultaneously to achieve improved super-resolution performance.

Overall, the contributions of this study are mainly in three aspects:

1) We present a fully convolutional neural network for image super-resolution. The network directly learns an end-to-end mapping between low- and high-resolution images, with little pre/post-processing beyond the optimization.
2) We establish a relationship between our deep-learning-based SR method and the traditional sparse-coding-based SR methods. This relationship provides a guidance for the design of the network structure.
3) We demonstrate that deep learning is useful in the classical computer vision problem of super-resolution, and can achieve good quality and speed.

A preliminary version of this work was presented earlier [11]. The present work adds to the initial version in significant ways. First, we improve the SRCNN by introducing larger filter size in the non-linear mapping layer, and explore deeper structures by adding non-linear mapping layers. Second, we extend the SRCNN to process three color channels (either in YCbCr or RGB color space) simultaneously. Experimentally, we demonstrate that performance can be improved in comparison to the single-channel network. Third, considerable new analyses and intuitive explanations are added to the initial results. We also extend the original experiments from *Set5* [2] and *Set14* [49] test images to *BSD200* [31] (200 test images). In addition, we compare with a number of recently published methods and confirm that our model still outperforms existing approaches using different evaluation metrics.

2 Related Work

2.1 Image Super-Resolution

According to the image priors, single-image super resolution algorithms can be categorized into four types—prediction models, edge based methods, image statistical methods and patch based (or example-based) methods. These methods have been thoroughly investigated and evaluated in Yang et al.'s work [44]. Among them, the example-based methods [16], [24], [39], [45] achieve the state-of-the-art performance.

The internal example-based methods exploit the self-similarity property and generate exemplar patches from the input image. It is first proposed in Glasner's work [16], and several improved variants [13], [43] are proposed to accelerate the implementation. The external example-based methods [2], [4], [6], [15], [36], [39], [46], [47], [48], [49] learn a mapping between low/high-resolution patches from external datasets. These studies vary on how to learn a compact dictionary or manifold space to relate low/high-resolution patches, and on how representation schemes can be conducted in such spaces. In the pioneer work of Freeman et al. [14], the dictionaries are directly presented as low/high-resolution patch pairs, and the nearest neighbour (NN) of the input patch is found in the low-resolution space, with its corresponding high-resolution patch used for reconstruction. Chang et al. [4] introduce a manifold embedding technique as an alternative to the NN strategy. In Yang et al.'s work [47], [48], the above NN correspondence advances to a more sophisticated sparse coding formulation. Other mapping functions such as kernel regression [24], simple function [45], random forest [36] and anchored neighborhood regression [39], [40] are proposed to further improve the mapping accuracy and speed. The sparse-coding-based method and its several improvements [39], [40], [46] are among the state-of-the-art SR methods nowadays. In these methods, the patches are the focus of the optimization; the patch extraction and aggregation steps are considered as pre/post-processing and handled separately.

The majority of SR algorithms [2], [4], [15], [39], [46], [47], [48], [49] focus on gray-scale or single-channel image super-resolution. For color images, the aforementioned methods first transform the problem to a different color space (YCbCr or YUV), and SR is applied only on the luminance channel. There are also works attempting to super-resolve all channels simultaneously. For example, Kim and Kwon [24] and Dai et al. [7] apply their model to each RGB channel and combined them to produce the final results. However, none

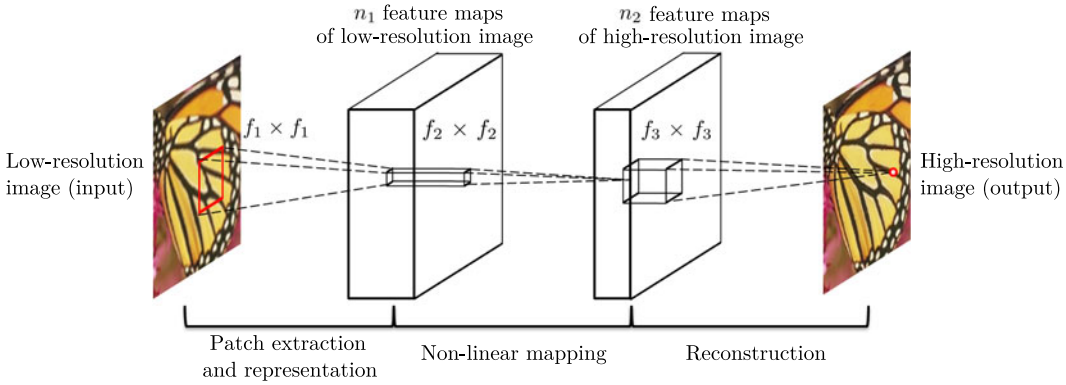


Fig. 2. Given a low-resolution image $\mathbf{Y}$, the first convolutional layer of the SRCNN extracts a set of feature maps. The second layer maps these feature maps nonlinearly to high-resolution patch representations. The last layer combines the predictions within a spatial neighbourhood to produce the final high-resolution image $F(\mathbf{Y})$.

of them has analyzed the SR performance of different channels, and the necessity of recovering all three channels.

2.2 Convolutional Neural Networks (CNN)

Convolutional neural networks date back decades [26] and deep CNNs have recently shown an explosive popularity partially due to its success in image classification [18], [25]. They have also been successfully applied to other computer vision fields, such as object detection [33], [50], face recognition [38], and pedestrian detection [34]. Several factors are of central importance in this progress: (i) the efficient training implementation on modern powerful GPUs [25], (ii) the proposal of the rectified linear unit (ReLU) [32] which makes convergence much faster while still presents good quality [25], and (iii) the easy access to an abundance of data (like ImageNet [9]) for training larger models. Our method also benefits from these progresses.

2.3 Deep Learning for Image Restoration

There have been a few studies of using deep learning techniques for image restoration. The multi-layer perceptron (MLP), whose all layers are fully-connected (in contrast to convolutional), is applied for natural image denoising [3] and post-deblurring denoising [35]. More closely related to our work, the convolutional neural network is applied for natural image denoising [21] and removing noisy patterns (dirt/rain) [12]. These restoration problems are more or less denoising-driven. Cui et al. [5] propose to embed auto-encoder networks in their super-resolution pipeline under the notion internal example-based approach [16]. The deep model is not specifically designed to be an end-to-end solution, since each layer of the cascade requires independent optimization of the self-similarity search process and the auto-encoder. On the contrary, the proposed SRCNN optimizes an end-to-end mapping. Further, the SRCNN is faster at speed. It is not only a quantitatively superior method, but also a practically useful one.

3 Convolutional Neural Networks for Super-Resolution

3.1 Formulation

Consider a single low-resolution image, we first upscale it to the desired size using bicubic interpolation, which is the only pre-processing we perform.[3] Let us denote the interpolated image as $\mathbf{Y}$. Our goal is to recover from $\mathbf{Y}$ an image $F(\mathbf{Y})$ that is as similar as possible to the ground truth high-resolution image $\mathbf{X}$. For the ease of presentation, we still call $\mathbf{Y}$ a "low-resolution" image, although it has the same size as $\mathbf{X}$. We wish to learn a mapping F, which conceptually consists of three operations:

1) *Patch extraction and representation.* this operation extracts (overlapping) patches from the low-resolution image $\mathbf{Y}$ and represents each patch as a high-dimensional vector. These vectors comprise a set of feature maps, of which the number equals to the dimensionality of the vectors.
2) *Non-linear mapping.* this operation nonlinearly maps each high-dimensional vector onto another high-dimensional vector. Each mapped vector is conceptually the representation of a high-resolution patch. These vectors comprise another set of feature maps.
3) *Reconstruction.* this operation aggregates the above high-resolution patch-wise representations to generate the final high-resolution image. This image is expected to be similar to the ground truth $\mathbf{X}$.

We will show that all these operations form a convolutional neural network. An overview of the network is depicted in Fig. 2. Next we detail our definition of each operation.

3.1.1 Patch Extraction and Representation

A popular strategy in image restoration (e.g., [1]) is to densely extract patches and then represent them by a set of pre-trained bases such as PCA, DCT, Haar, etc. This is equivalent to convolving the image by a set of filters, each of which is a basis. In our formulation, we involve the optimization of these bases into the optimization of the network. Formally, our first layer is expressed as an operation F_1:

$$F_1(\mathbf{Y}) = \max(0, W_1 * \mathbf{Y} + B_1), \quad (1)$$

3. Bicubic interpolation is also a convolutional operation, so it can be formulated as a convolutional layer. However, the output size of this layer is larger than the input size, so there is a fractional stride. To take advantage of the popular well-optimized implementations such as *cuda-convnet* [25], we exclude this "layer" from learning.

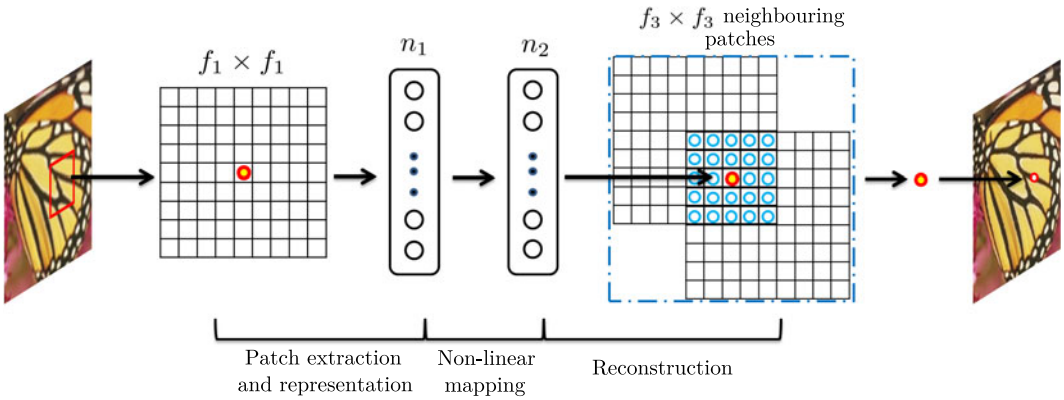


Fig. 3. An illustration of sparse-coding-based methods in the view of a convolutional neural network.

where W_1 and B_1 represent the filters and biases respectively, and '$*$' denotes the convolution operation. Here, W_1 corresponds to n_1 filters of support $c \times f_1 \times f_1$, where c is the number of channels in the input image, f_1 is the spatial size of a filter. Intuitively, W_1 applies n_1 convolutions on the image, and each convolution has a kernel size $c \times f_1 \times f_1$. The output is composed of n_1 feature maps. B_1 is an n_1-dimensional vector, whose each element is associated with a filter. We apply the ReLU ($\max(0, x)$) [32] on the filter responses.[4]

3.1.2 Non-Linear Mapping

The first layer extracts an n_1-dimensional feature for each patch. In the second operation, we map each of these n_1-dimensional vectors into an n_2-dimensional one. This is equivalent to applying n_2 filters which have a trivial spatial support 1×1. This interpretation is only valid for 1×1 filters. But it is easy to generalize to larger filters like 3×3 or 5×5. In that case, the non-linear mapping is not on a patch of the input image; instead, it is on a 3×3 or 5×5 "patch" of the feature map. The operation of the second layer is:

$$F_2(\mathbf{Y}) = \max(0, W_2 * F_1(\mathbf{Y}) + B_2). \tag{2}$$

Here W_2 contains n_2 filters of size $n_1 \times f_2 \times f_2$, and B_2 is n_2-dimensional. Each of the output n_2-dimensional vectors is conceptually a representation of a high-resolution patch that will be used for reconstruction.

It is possible to add more convolutional layers to increase the non-linearity. But this can increase the complexity of the model ($n_2 \times f_2 \times f_2 \times n_2$ parameters for one layer), and thus demands more training time. We will explore deeper structures by introducing additional non-linear mapping layers in Section 4.3.3.

3.1.3 Reconstruction

In the traditional methods, the predicted overlapping high-resolution patches are often averaged to produce the final full image. The averaging can be considered as a predefined filter on a set of feature maps (where each position is the "flattened" vector form of a high-resolution patch). Motivated by this, we define a convolutional layer to produce the final high-resolution image:

$$F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3. \tag{3}$$

Here W_3 corresponds to c filters of a size $n_2 \times f_3 \times f_3$, and B_3 is a c-dimensional vector.

If the representations of the high-resolution patches are in the image domain (i.e., we can simply reshape each representation to form the patch), we expect that the filters act like an averaging filter; if the representations of the high-resolution patches are in some other domains (e.g., coefficients in terms of some bases), we expect that W_3 behaves like first projecting the coefficients onto the image domain and then averaging. In either way, W_3 is a set of linear filters.

Interestingly, although the above three operations are motivated by different intuitions, they all lead to the same form as a convolutional layer. We put all three operations together and form a convolutional neural network (Fig. 2). In this model, all the filtering weights and biases are to be optimized. Despite the succinctness of the overall structure, our SRCNN model is carefully developed by drawing extensive experience resulted from significant progresses in super-resolution [47], [48]. We detail the relationship in the next section.

3.2 Relationship to Sparse-Coding-Based Methods

We show that the sparse-coding-based SR methods [47], [48] can be viewed as a convolutional neural network. Fig. 3 shows an illustration.

In the sparse-coding-based methods, let us consider that an $f_1 \times f_1$ low-resolution patch is extracted from the input image. Then the sparse coding solver, like Feature-Sign [28], will first project the patch onto a (low-resolution) dictionary. If the dictionary size is n_1, this is equivalent to applying n_1 linear filters ($f_1 \times f_1$) on the input image (the mean subtraction is also a linear operation so can be absorbed). This is illustrated as the left part of Fig. 3.

The sparse coding solver will then iteratively process the n_1 coefficients. The outputs of this solver are n_2 coefficients, and usually $n_2 = n_1$ in the case of sparse coding. These n_2 coefficients are the representation of the high-resolution patch. In this sense, the sparse coding solver behaves as a special case of a non-linear mapping operator, whose spatial support is 1×1. See the middle part of Fig. 3. However, the

4. The ReLU can be equivalently considered as a part of the second operation (Non-linear mapping), and the first operation (Patch extraction and representation) becomes purely linear convolution.

sparse coding solver is not feed-forward, i.e., it is an iterative algorithm. On the contrary, our non-linear operator is fully feed-forward and can be computed efficiently. If we set $f_2 = 1$, then our non-linear operator can be considered as a pixel-wise fully-connected layer. It is worth noting that "the sparse coding solver" in SRCNN refers to the first two layers, but not just the second layer or the activation function (ReLU). Thus the nonlinear operation in SRCNN is also well optimized through the learning process.

The above n_2 coefficients (after sparse coding) are then projected onto another (high-resolution) dictionary to produce a high-resolution patch. The overlapping high-resolution patches are then averaged. As discussed above, this is equivalent to linear convolutions on the n_2 feature maps. If the high-resolution patches used for reconstruction are of size $f_3 \times f_3$, then the linear filters have an equivalent spatial support of size $f_3 \times f_3$. See the right part of Fig. 3.

The above discussion shows that the sparse-coding-based SR method can be viewed as a kind of convolutional neural network (with a different non-linear mapping). But not all operations have been considered in the optimization in the sparse-coding-based SR methods. On the contrary, in our convolutional neural network, the low-resolution dictionary, high-resolution dictionary, non-linear mapping, together with mean subtraction and averaging, are all involved in the filters to be optimized. So our method optimizes an end-to-end mapping that consists of all operations.

The above analogy can also help us to design hyper-parameters. For example, we can set the filter size of the last layer to be smaller than that of the first layer, and thus we rely more on the central part of the high-resolution patch (to the extreme, if $f_3 = 1$, we are using the center pixel with no averaging). We can also set $n_2 < n_1$ because it is expected to be sparser. A typical and basic setting is $f_1 = 9$, $f_2 = 1$, $f_3 = 5$, $n_1 = 64$, and $n_2 = 32$ (we evaluate more settings in the experiment section). On the whole, the estimation of a high resolution pixel utilizes the information of $(9 + 5 - 1)^2 = 169$ pixels. Clearly, the information exploited for reconstruction is comparatively larger than that used in existing external example-based approaches, e.g., using $(5 + 5 - 1)^2 = 81$ pixels[5] [15], [48]. This is one of the reasons why the SRCNN gives superior performance.

3.3 Training

Learning the end-to-end mapping function F requires the estimation of network parameters $\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$. This is achieved through minimizing the loss between the reconstructed images $F(\mathbf{Y}; \Theta)$ and the corresponding ground truth high-resolution images $\mathbf{X}$. Given a set of high-resolution images $\{\mathbf{X}_i\}$ and their corresponding low-resolution images $\{\mathbf{Y}_i\}$, we use mean squared error (MSE) as the loss function:

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i; \Theta) - \mathbf{X}_i||^2, \tag{4}$$

where n is the number of training samples. Using MSE as the loss function favors a high PSNR. The PSNR is a widely-used metric for quantitatively evaluating image restoration quality, and is at least partially related to the perceptual quality. It is worth noticing that the convolutional neural networks do not preclude the usage of other kinds of loss functions, if only the loss functions are derivable. If a better perceptually motivated metric is given during training, it is flexible for the network to adapt to that metric. On the contrary, such a flexibility is in general difficult to achieve for traditional "hand-crafted" methods. Despite that the proposed model is trained favoring a high PSNR, we still observe satisfactory performance when the model is evaluated using alternative evaluation metrics, e.g., SSIM, MSSIM (see Section 4.4.1).

The loss is minimized using stochastic gradient descent with the standard backpropagation [27]. In particular, the weight matrices are updated as

$$\Delta_{i+1} = 0.9 \cdot \Delta_i + \eta \cdot \frac{\partial L}{\partial W_i^\ell}, \quad W_{i+1}^\ell = W_i^\ell + \Delta_{i+1}, \tag{5}$$

where $\ell \in \{1, 2, 3\}$ and i are the indices of layers and iterations, η is the learning rate, and $\frac{\partial L}{\partial W_i^\ell}$ is the derivative. The filter weights of each layer are initialized by drawing randomly from a Gaussian distribution with zero mean and standard deviation 0.001 (and 0 for biases). The learning rate is 10^{-4} for the first two layers, and 10^{-5} for the last layer. We empirically find that a smaller learning rate in the last layer is important for the network to converge (similar to the denoising case [21]).

In the training phase, the ground truth images $\{\mathbf{X}_i\}$ are prepared as $f_{sub} \times f_{sub} \times c$-pixel sub-images randomly cropped from the training images. By "sub-images" we mean these samples are treated as small "images" rather than "patches", in the sense that "patches" are overlapping and require some averaging as post-processing but "sub-images" need not. To synthesize the low-resolution samples $\{\mathbf{Y}_i\}$, we blur a sub-image by a Gaussian kernel, sub-sample it by the upscaling factor, and upscale it by the same factor via bicubic interpolation.

To avoid border effects during training, all the convolutional layers have no padding, and the network produces a smaller output ($(f_{sub} - f_1 - f_2 - f_3 + 3)^2 \times c$). The MSE loss function is evaluated only by the difference between the central pixels of $\mathbf{X}_i$ and the network output. Although we use a fixed image size in training, the convolutional neural network can be applied on images of arbitrary sizes during testing.

We implement our model using the *cuda-convnet* package [25]. We have also tried the *Caffe* package [23] and observed similar performance.

4 Experiments

We first investigate the impact of using different datasets on the model performance. Next, we examine the filters learned by our approach. We then explore different architecture designs of the network, and study the relations between super-resolution performance and factors like depth, number of filters, and filter sizes. Subsequently, we compare our method with recent state-of-the-arts both quantitatively and qualitatively. Following [40], super-resolution is only applied on the luminance channel (Y channel in YCbCr color space) in Sections 4.1-4.4, so $c = 1$ in the

5. The patches are overlapped with 4 pixels at each direction.

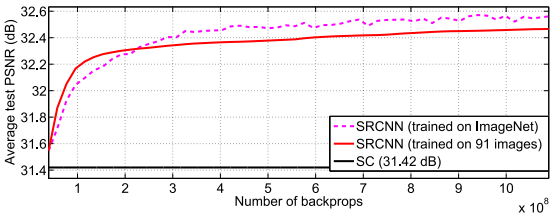


Fig. 4. Training with the much larger ImageNet dataset improves the performance over the use of 91 images.

first/last layer, and performance (e.g., PSNR and SSIM) is evaluated on the Y channel. At last, we extend the network to cope with color images and evaluate the performance on different channels.

4.1 Training Data

As shown in the literature, deep learning generally benefits from big data training. For comparison, we use a relatively small training set [39], [48] that consists of 91 images, and a large training set that consists of 395,909 images from the ILSVRC 2013 ImageNet detection training partition. The size of training sub-images is $f_{sub} = 33$. Thus the 91-image dataset can be decomposed into 24,800 sub-images, which are extracted from original images with a stride of 14. Whereas the ImageNet provides over 5 million sub-images even using a stride of 33. We use the basic network settings, i.e., $f_1 = 9$, $f_2 = 1$, $f_3 = 5$, $n_1 = 64$, and $n_2 = 32$. We use the *Set5* [2] as the validation set. We observe a similar trend even if we use the larger *Set14* set [49]. The upscaling factor is 3. We use the sparse-coding-based method [48] as our baseline, which achieves an average PSNR value of 31.42 dB.

The test convergence curves of using different training sets are shown in Fig. 4. The training time on ImageNet is about the same as on the 91-image dataset since the number of backpropagations is the same. As can be observed, with the same number of backpropagations (i.e., 8×10^8), the SRCNN+ImageNet achieves **32.52 dB**, higher than 32.39 dB yielded by that trained on 91 images. The results positively indicate that SRCNN performance may be further boosted using a larger training set, but the effect of big data is not as impressive as that shown in high-level vision problems [25]. This is mainly because that the 91 images have already captured sufficient variability of natural images. On the other hand, our SRCNN is a relatively small network (8,032 parameters), which could not overfit the the 91 images (24,800 samples). Nevertheless, we adopt the ImageNet, which contains more diverse data, as the default training set in the following experiments.

4.2 Learned Filters for Super-Resolution

Fig. 5 shows examples of learned first-layer filters trained on the ImageNet by an upscaling factor 3. Please refer to our published implementation for upscaling factors 2 and 4. Interestingly, each learned filter has its specific functionality. For instance, the filters g and h are like Laplacian/Gaussian filters, the filters a - e are like edge detectors at different directions, and the filter f is like a texture extractor. Example feature maps of different layers are shown in Fig. 6. Obviously, feature maps of the first layer contain different structures (e.g., edges at different directions), while that of the second layer are mainly different on intensities.

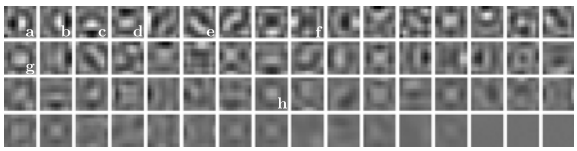


Fig. 5. The figure shows the first-layer filters trained on ImageNet with an upscaling factor 3. The filters are organized based on their respective variances.

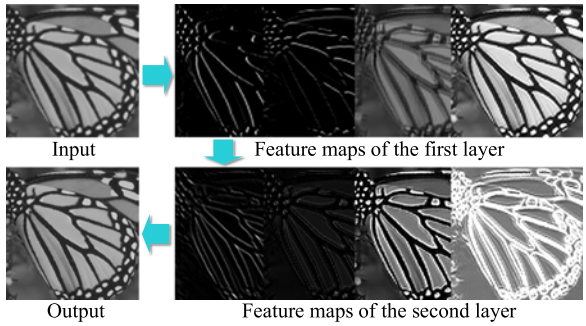


Fig. 6. Example feature maps of different layers.

4.3 Model and Performance Trade-Offs

Based on the basic network settings (i.e., $f_1 = 9$, $f_2 = 1$, $f_3 = 5$, $n_1 = 64$, and $n_2 = 32$), we will progressively modify some of these parameters to investigate the best trade-off between performance and speed, and study the relations between performance and parameters.

4.3.1 Filter Number

In general, the performance would improve if we increase the network width,[6] i.e., adding more filters, at the cost of running time. Specifically, based on our network default settings of $n_1 = 64$ and $n_2 = 32$, we conduct two experiments: (i) one is with a larger network with $n_1 = 128$ and $n_2 = 64$, and (ii) the other is with a smaller network with $n_1 = 32$ and $n_2 = 16$. Similar to Section 4.1, we also train the two models on ImageNet and test on Set5 with an upscaling factor 3. The results observed at 8×10^8 backpropagations are shown in Table 1. It is clear that superior performance could be achieved by increasing the width. However, if a fast restoration speed is desired, a small network width is preferred, which could still achieve better performance than the sparse-coding-based method (31.42 dB).

4.3.2 Filter Size

In this section, we examine the network sensitivity to different filter sizes. In previous experiments, we set filter size $f_1 = 9$, $f_2 = 1$ and $f_3 = 5$, and the network could be denoted as 9-1-5. First, to be consistent with sparse-coding-based methods, we fix the filter size of the second layer to be $f_2 = 1$, and enlarge the filter size of other layers to $f_1 = 11$ and $f_3 = 7$ (11-1-7). All the other settings remain the same with Section 4.1. The results with an upscaling factor 3 on Set5 are 32.57 dB, which is slightly higher than the 32.52 dB reported in Section 4.1. This indicates that a reasonably

6. We use 'width' to term the number of filters in a layer, following [17]. The term 'width' may have other meanings in the literature.

TABLE 1
The Results of Using Different Filter Numbers in SRCNN

$n_1 = 128$		$n_1 = 64$		$n_1 = 32$	
$n_2 = 64$		$n_2 = 32$		$n_2 = 16$	
PSNR	Time (sec)	PSNR	Time (sec)	PSNR	Time (sec)
32.60	0.60	32.52	0.18	32.26	0.05

Training is performed on ImageNet whilst the evaluation is conducted on the Set5 dataset.

larger filter size could grasp richer structural information, which in turn lead to better results.

Then we further examine networks with a larger filter size of the second layer. Specifically, we fix the filter size $f_1 = 9$, $f_3 = 5$, and enlarge the filter size of the second layer to be (i) $f_2 = 3$ (9-3-5) and (ii) $f_2 = 5$ (9-5-5). Convergence curves in Fig. 7 show that using a larger filter size could significantly improve the performance. Specifically, the average PSNR values achieved by 9-3-5 and 9-5-5 on Set5 with 8×10^8 backpropagations are 32.66 and 32.75 dB, respectively. The results suggest that utilizing neighborhood information in the mapping stage is beneficial.

However, the deployment speed will also decrease with a larger filter size. For example, the number of parameters of 9-1-5, 9-3-5, and 9-5-5 is 8,032, 24,416, and 57,184 respectively. The complexity of 9-5-5 is almost twice of 9-3-5, but the performance improvement is marginal. Therefore, the choice of the network scale should always be a trade-off between performance and speed.

4.3.3 Number of Layers

Recent study by He and Sun [17] suggests that CNN could benefit from increasing the depth of network moderately. Here, we try deeper structures by adding another non-linear mapping layer, which has $n_{22} = 16$ filters with size $f_{22} = 1$. We conduct three controlled experiments, i.e., 9-1-1-5, 9-3-1-5, 9-5-1-5, which add an additional layer on 9-1-5, 9-3-5, and 9-5-5, respectively. The initialization scheme and learning rate of the additional layer are the same as the second layer. From Figs. 13a, 13b and 8c, we can observe that the four-layer networks converge slower than the three-layer network. Nevertheless, given enough training time, the deeper networks will finally catch up and converge to the three-layer ones.

The effectiveness of deeper structures for super resolution is found not as apparent as that shown in image classification [17]. Furthermore, we find that deeper networks do not always result in better performance. Specifically, if we add an additional layer with $n_{22} = 32$ filters on 9-1-5 network, then the performance degrades and fails to surpass the three-layer network (see Fig. 9a). If we go deeper by adding

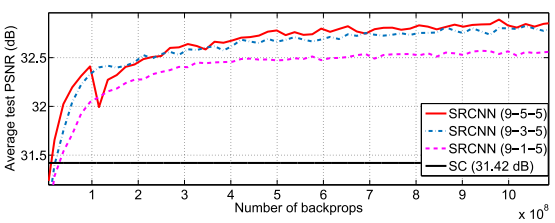
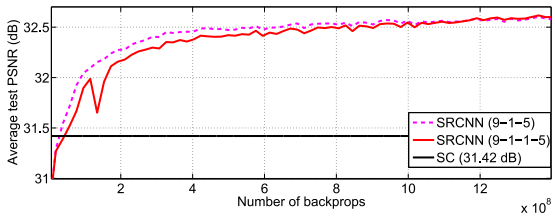
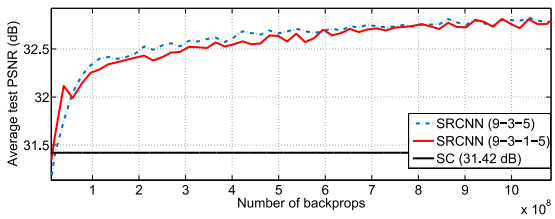


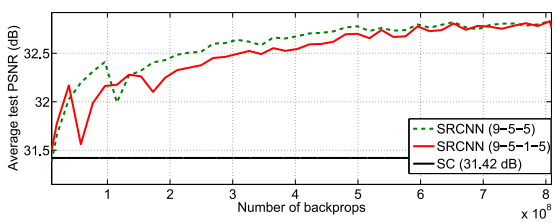
Fig. 7. A larger filter size leads to better results.



(a) 9-1-5 vs. 9-1-1-5



(b) 9-3-5 vs. 9-3-1-5

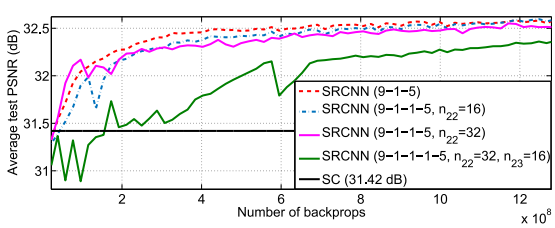


(c) 9-5-5 vs. 9-5-1-5

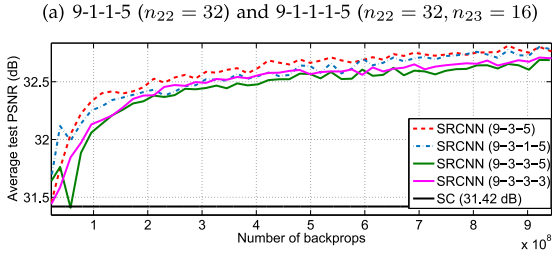
Fig. 8. Comparisons between three-layer and four-layer networks.

two non-linear mapping layers with $n_{22} = 32$ and $n_{23} = 16$ filters on 9-1-5, then we have to set a smaller learning rate to ensure convergence, but we still do not observe superior performance after a week of training (see Fig. 9a). We also tried to enlarge the filter size of the additional layer to $f_{22} = 3$, and explore two deep structures—9-3-3-5 and 9-3-3-3. However, from the convergence curves shown in Fig. 9b, these two networks do not show better results than the 9-3-1-5 network.

All these experiments indicate that it is not "the deeper the better" in this deep model for super-resolution. It may be caused by the difficulty of training. Our CNN network contains no pooling layer or full-connected layer, thus it is sensitive to the initialization parameters and learning rate. When we go deeper (e.g., four or five layers), we find it hard to set appropriate learning rates that guarantee convergence. Even



(a) 9-1-1-5 ($n_{22} = 32$) and 9-1-1-1-5 ($n_{22} = 32, n_{23} = 16$)



(b) 9-3-3-5 and 9-3-3-3

Fig. 9. Deeper structure does not always lead to better results.

TABLE 2
The Average Results of PSNR (dB), SSIM, IFC, NQM, WPSNR (dB) and MSSIM on the Set5 Dataset

Eval. Mat	Scale	Bicubic	SC [48]	NE+LLE [4]	KK [24]	ANR [39]	A+ [39]	SRCNN
PSNR	2	33.66	-	35.77	36.20	35.83	36.54	**36.66**
	3	30.39	31.42	31.84	32.28	31.92	32.59	**32.75**
	4	28.42	-	29.61	30.03	29.69	30.28	**30.49**
SSIM	2	0.9299	-	0.9490	0.9511	0.9499	**0.9544**	0.9542
	3	0.8682	0.8821	0.8956	0.9033	0.8968	0.9088	**0.9090**
	4	0.8104	-	0.8402	0.8541	0.8419	0.8603	**0.8628**
IFC	2	6.10	-	7.84	6.87	8.09	**8.48**	8.05
	3	3.52	3.16	4.40	4.14	4.52	**4.84**	4.58
	4	2.35	-	2.94	2.81	3.02	**3.26**	3.01
NQM	2	36.73	-	42.90	39.49	43.28	**44.58**	41.13
	3	27.54	27.29	32.77	32.10	33.10	**34.48**	33.21
	4	21.42	-	25.56	24.99	25.72	**26.97**	25.96
WPSNR	2	50.06	-	58.45	57.15	58.61	**60.06**	59.49
	3	41.65	43.64	45.81	46.22	46.02	**47.17**	47.10
	4	37.21	-	39.85	40.40	40.01	41.03	**41.13**
MSSSIM	2	0.9915	-	0.9953	0.9953	0.9954	**0.9960**	0.9959
	3	0.9754	0.9797	0.9841	0.9853	0.9844	**0.9867**	0.9866
	4	0.9516	-	0.9666	0.9695	0.9672	0.9720	**0.9725**

it converges, the network may fall into a bad local minimum, and the learned filters are of less diversity even given enough training time. This phenomenon is also observed in [16], where improper increase of depth leads to accuracy saturation or degradation for image classification. Why "deeper is not better" is still an open question, which requires investigations to better understand gradients and training dynamics in deep architectures. Therefore, we still adopt three-layer networks in the following experiments.

4.4 Comparisons to State-of-the-Arts

In this section, we show the quantitative and qualitative results of our method in comparison to state-of-the-art methods. We adopt the model with good performance-speed trade-off: a three-layer network with $f_1 = 9$, $f_2 = 5$, $f_3 = 5$, $n_1 = 64$, and $n_2 = 32$ trained on the ImageNet. For each upscaling factor $\in \{2, 3, 4\}$, we train a specific network for that factor.[7]

Comparisons. We compare our *SRCNN* with the state-of-the-art SR methods:

- *SC* - sparse coding-based method of Yang et al. [48]
- *NE+LLE* - neighbour embedding + locally linear embedding method [4]
- *ANR* - Anchored Neighbourhood Regression method [39]
- *A+* - Adjusted Anchored Neighbourhood Regression method [40], and
- *KK* - the method described in [24], which achieves the best performance among external example-based methods, according to the comprehensive evaluation conducted in Yang et al.'s work [44]

The implementations are all from the publicly available codes provided by the authors, and all images are downsampled using the same bicubic kernel.

7. In the area of denoising [3], for each noise level a specific network is trained.

Test set. The *Set5* [2] (5 images), *Set14* [49] (14 images) and *BSD200* [31] (200 images)[8] are used to evaluate the performance of upscaling factors 2, 3, and 4.

Evaluation metrics. Apart from the widely used PSNR and SSIM [41] indices, we also adopt another four evaluation matrices, namely IFC [37], noise quality measure (NQM) [8], weighted peak signal-to-noise ratio (WPSNR) and multi-scale structure similarity index (MSSSIM) [42], which obtain high correlation with the human perceptual scores as reported in [44].

4.4.1 Quantitative and Qualitative Evaluation

As shown in Tables 2, 3 and 4, the proposed SRCNN yields the highest scores in most evaluation matrices in all experiments.[9] Note that our SRCNN results are based on the checkpoint of 8×10^8 backpropagations. Specifically, for the upscaling factor 3, the average gains on PSNR achieved by SRCNN are 0.15, 0.17, and 0.13 dB, higher than the next best approach, A+ [40], on the three datasets. When we take a look at other evaluation metrics, we observe that SC, to our surprise, gets even lower scores than the bicubic interpolation on IFC and NQM. It is clear that the results of SC are more visually pleasing than that of bicubic interpolation. This indicates that these two metrics may not truthfully reveal the image quality. Thus, regardless of these two metrics, SRCNN achieves the best performance among all methods and scaling factors.

It is worth pointing out that SRCNN surpasses the bicubic baseline at the very beginning of the learning stage (see Fig. 1), and with moderate training, SRCNN outperforms existing state-of-the-art methods (see Fig. 4). Yet, the performance is far from converge. We conjecture that better results can be obtained given longer training time (see Fig. 10).

8. We use the same 200 images as in [44].

9. The PSNR value of each image can be found in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2015.2439281.

TABLE 3
The Average Results of PSNR (dB), SSIM, IFC, NQM, WPSNR (dB) and MSSIM on the Set14 Dataset

Eval. Mat	Scale	Bicubic	SC [48]	NE+LLE [4]	KK [24]	ANR [39]	A+ [39]	SRCNN
PSNR	2	30.23	-	31.76	32.11	31.80	32.28	**32.45**
	3	27.54	28.31	28.60	28.94	28.65	29.13	**29.30**
	4	26.00	-	26.81	27.14	26.85	27.32	**27.50**
SSIM	2	0.8687	-	0.8993	0.9026	0.9004	0.9056	**0.9067**
	3	0.7736	0.7954	0.8076	0.8132	0.8093	0.8188	**0.8215**
	4	0.7019	-	0.7331	0.7419	0.7352	0.7491	**0.7513**
IFC	2	6.09	-	7.59	6.83	7.81	**8.11**	7.76
	3	3.41	2.98	4.14	3.83	4.23	**4.45**	4.26
	4	2.23	-	2.71	2.57	2.78	**2.94**	2.74
NQM	2	40.98	-	41.34	38.86	41.79	**42.61**	38.95
	3	33.15	29.06	37.12	35.23	37.22	**38.24**	35.25
	4	26.15	-	31.17	29.18	31.27	**32.31**	30.46
WPSNR	2	47.64	-	54.47	53.85	54.57	**55.62**	55.39
	3	39.72	41.66	43.22	43.56	43.36	44.25	**44.32**
	4	35.71	-	37.75	38.26	37.85	38.72	**38.87**
MSSSIM	2	0.9813	-	0.9886	0.9890	0.9888	0.9896	**0.9897**
	3	0.9512	0.9595	0.9643	0.9653	0.9647	0.9669	**0.9675**
	4	0.9134	-	0.9317	0.9338	0.9326	0.9371	**0.9376**

TABLE 4
The Average Results of PSNR (dB), SSIM, IFC, NQM, WPSNR (dB) and MSSIM on the BSD200 Dataset

Eval. Mat	Scale	Bicubic	SC [48]	NE+LLE [4]	KK [24]	ANR [39]	A+ [39]	SRCNN
PSNR	2	28.38	-	29.67	30.02	29.72	30.14	**30.29**
	3	25.94	26.54	26.67	26.89	26.72	27.05	**27.18**
	4	24.65	-	25.21	25.38	25.25	25.51	**25.60**
SSIM	2	0.8524	-	0.8886	0.8935	0.8900	0.8966	**0.8977**
	3	0.7469	0.7729	0.7823	0.7881	0.7843	0.7945	**0.7971**
	4	0.6727	-	0.7037	0.7093	0.7060	0.7171	**0.7184**
IFC	2	5.30	-	7.10	6.33	7.28	**7.51**	7.21
	3	3.05	2.77	3.82	3.52	3.91	**4.07**	3.91
	4	1.95	-	2.45	2.24	2.51	**2.62**	2.45
NQM	2	36.84	-	41.52	38.54	41.72	**42.37**	39.66
	3	28.45	28.22	34.65	33.45	34.81	**35.58**	34.72
	4	21.72	-	25.15	24.87	25.27	**26.01**	25.65
WPSNR	2	46.15	-	52.56	52.21	52.69	53.56	**53.58**
	3	38.60	40.48	41.39	41.62	41.53	42.19	**42.29**
	4	34.86	-	36.52	36.80	36.64	37.18	**37.24**
MSSSIM	2	0.9780	-	0.9869	0.9876	0.9872	0.9883	**0.9883**
	3	0.9426	0.9533	0.9575	0.9588	0.9581	0.9609	**0.9614**
	4	0.9005	-	0.9203	0.9215	0.9214	0.9256	**0.9261**

Figs. 14, 15 and 16 show the super-resolution results of different approaches by an upscaling factor 3. As can be observed, the SRCNN produces much sharper edges than other approaches without any obvious artifacts across the image.

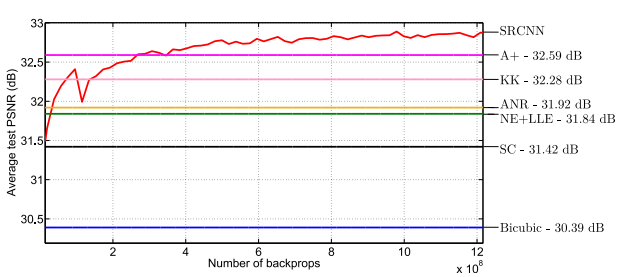


Fig. 10. The test convergence curve of SRCNN and results of other methods on the Set5 dataset.

In addition, we report to another recent deep learning method for image super-resolution (DNC) of Cui et al. [5]. As they employ a different blur kernel (a Gaussian filter with a standard deviation of 0.55), we train a specific network (9-5-5) using the same blur kernel as DNC for fair quantitative comparison. The upscaling factor is 3 and the training set is the 91-image dataset. From the convergence curve shown in Fig. 11, we observe that our SRCNN surpasses DNC with just 2.7×10^7 backprops, and a larger margin can be obtained given longer training time. This also demonstrates that the end-to-end learning is superior to DNC, even if that model is already "deep".

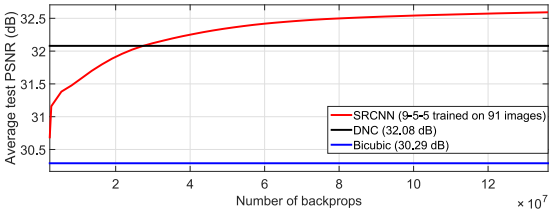


Fig. 11. The test convergence curve of SRCNN and the result of DNC on the Set5 dataset.

4.4.2 Running Time

Fig. 12 shows the running time comparisons of several state-of-the-art methods, along with their restoration performance on *Set14*. All baseline methods are obtained from the corresponding authors' MATLAB+MEX implementation, whereas ours are in pure C++. We profile the running time of all the algorithms using the same machine (Intel CPU 3.10 GHz and 16 GB memory). Note that the processing time of our approach is highly linear to the test image resolution, since all images go through the same number of convolutions. Our method is always a trade-off between performance and speed. To show this, we train three networks for comparison, which are 9-1-5, 9-3-5, and 9-5-5. It is clear that the 9-1-5 network is the fastest, while it still achieves better performance than the next state-of-the-art A+. Other methods are several times or even orders of magnitude slower in comparison to 9-1-5 network. Note the speed gap is not mainly caused by the different MATLAB/C++ implementations; rather, the other methods need to solve complex optimization problems on usage (e.g., sparse coding or embedding), whereas our method is completely feed-forward. The 9-5-5 network achieves the best performance but at the cost of the running time. The test-time speed of our CNN can be further accelerated in many ways, e.g., approximating or simplifying the trained networks [10], [30], with possible slight degradation in performance.

4.5 Experiments on Color Channels

In previous experiments, we follow the conventional approach to super-resolve color images. Specifically, we first transform the color images into the YCbCr space. The SR algorithms are only applied on the Y channel, while the Cb, Cr channels are upscaled by bicubic interpolation.

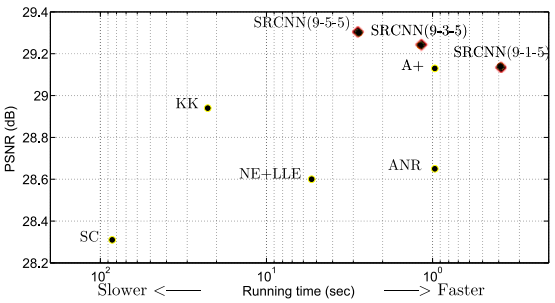


Fig. 12. The proposed SRCNN achieves the state-of-the-art super-resolution quality, whilst maintains high and competitive speed in comparison to existing external example-based methods. The chart is based on Set14 results summarized in Table 3. The implementation of all three SRCNN networks are available on our project page.

TABLE 5
Average PSNR (dB) of Different Channels and Training Strategies on the Set5 Dataset

Training Strategies	PSNR of different channel(s)			
	Y	Cb	Cr	RGB color image
Bicubic	30.39	45.44	45.42	34.57
Y only	**32.39**	45.44	45.42	36.37
YCbCr	29.25	43.30	43.49	33.47
Y pre-train	32.19	**46.49**	**46.45**	36.32
CbCr pre-train	32.14	46.38	45.84	36.25
RGB	32.33	46.18	46.20	**36.44**
KK	32.37	44.35	44.22	36.32

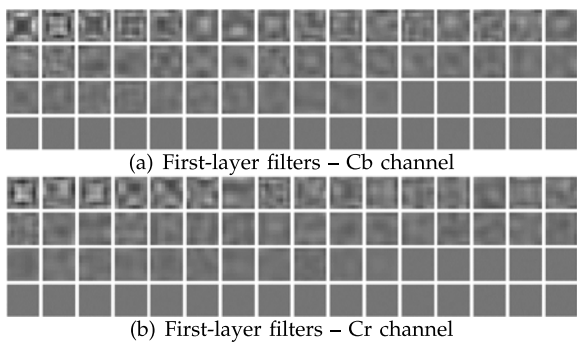
It is interesting to find out if super-resolution performance can be improved if we jointly consider all three channels in the process.

Our method is flexible to accept more channels without altering the learning mechanism and network design. In particular, it can readily deal with three channels simultaneously by setting the input channels to $c = 3$. In the following experiments, we explore different training strategies for color image super-resolution, and subsequently evaluate their performance on different channels.

Implementation details. Training is performed on the 91-image dataset, and testing is conducted on the *Set5* [2]. The network settings are: $c = 3$, $f_1 = 9$, $f_2 = 1$, $f_3 = 5$, $n_1 = 64$, and $n_2 = 32$. As we have proved the effectiveness of SRCNN on different scales, here we only evaluate the performance of upscaling factor 3.

Comparisons. We compare our method with the state-of-art color SR method—KK [24]. We also try different learning strategies for comparison:

- **Y only:** This is our baseline method, which is a single-channel ($c = 1$) network trained only on the luminance channel. The Cb, Cr channels are upscaled using bicubic interpolation.
- **YCbCr:** Training is performed on the three channels of the YCbCr space.
- **Y pre-train:** First, to guarantee the performance on the Y channel, we only use the MSE of the Y channel as the loss to pre-train the network. Then we employ the MSE of all channels to fine-tune the parameters.
- **CbCr pre-train:** We use the MSE of the Cb, Cr channels as the loss to pre-train the network, then fine-tune the parameters on all channels.



(a) First-layer filters – Cb channel

(b) First-layer filters – Cr channel

Fig. 13. Chrominance channels of the first-layer filters using the "Y pre-train" strategy.

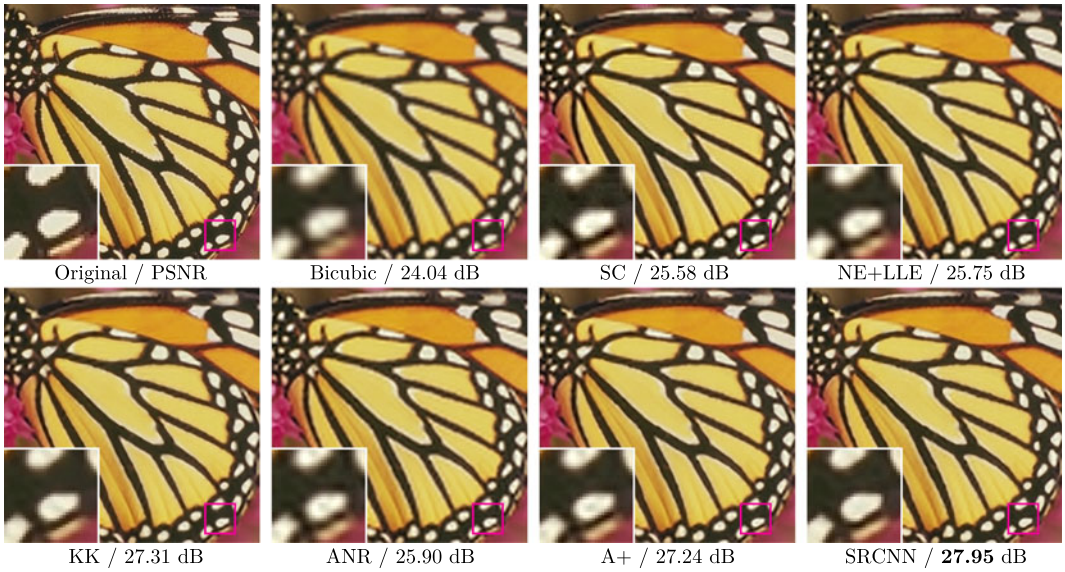


Fig. 14. The “butterfly” image from Set5 with an upscaling factor 3.

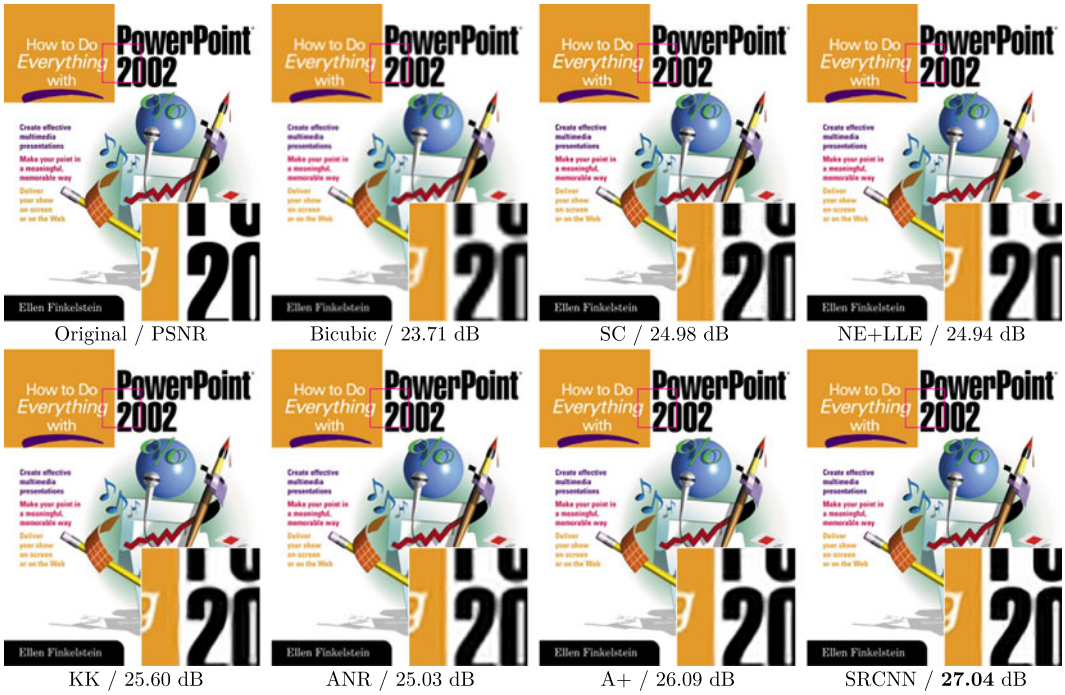


Fig. 15. The “ppt3” image from Set14 with an upscaling factor 3.

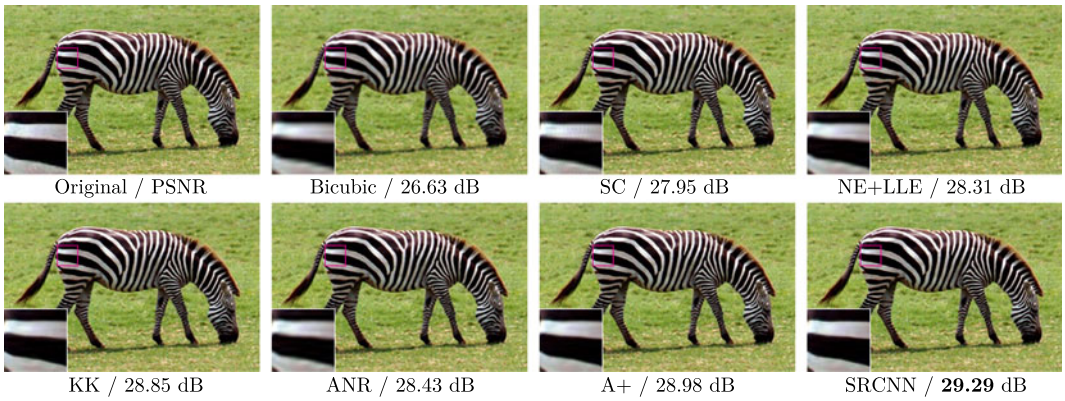


Fig. 16. The “zebra” image from Set14 with an upscaling factor 3.

- **RGB:** Training is performed on the three channels of the RGB space.

The results are shown in Table 5, where we have the following observations. (i) If we directly train on the YCbCr channels, the results are even worse than that of bicubic interpolation. The training falls into a bad local minimum, due to the inherently different characteristics of the Y and Cb, Cr channels. (ii) If we pre-train on the Y or Cb, Cr channels, the performance finally improves, but is still not better than "Y only" on the color image (see the last column of Table 5, where PSNR is computed in RGB color space). This suggests that the Cb, Cr channels could decrease the performance of the Y channel when training is performed in a unified network. (iii) We observe that the Cb, Cr channels have higher PSNR values for "Y pre-train" than for "CbCr pre-train". The reason lies on the differences between the Cb, Cr channels and the Y channel. Visually, the Cb, Cr channels are more blurry than the Y channel, thus are less affected by the downsampling process. When we pre-train on the Cb, Cr channels, there are only a few filters being activated. Then the training will soon fall into a bad local minimum during fine-tuning. On the other hand, if we pre-train on the Y channel, more filters will be activated, and the performance on Cb, Cr channels will be pushed much higher. Fig. 13 shows the Cb, Cr channels of the first-layer filters with "Y pre-train", of which the patterns largely differ from that shown in Fig. 5. (iv) Training on the RGB channels achieves the best result on the color image. Different from the YCbCr channels, the RGB channels exhibit high cross-correlation among each other. The proposed SRCNN is capable of leveraging such natural correspondences between the channels for reconstruction. Therefore, the model achieves comparable result on the Y channel as "Y only", and better results on Cb, Cr channels than bicubic interpolation. (v) In KK [24], super-resolution is applied on each RGB channel separately. When we transform its results to YCbCr space, the PSNR value of Y channel is similar as "Y only", but that of Cb, Cr channels are poorer than bicubic interpolation. The result suggests that the algorithm is biased to the Y channel. On the whole, our method trained on RGB channels achieves better performance than KK and the single-channel network ("Y only"). It is also worth noting that the improvement compared with the single-channel network is not that significant (i.e., 0.07 dB). This indicates that the Cb, Cr channels barely help in improving the performance.

5 Conclusion

We have presented a novel deep learning approach for single image SR. We show that conventional sparse-coding-based SR methods can be reformulated into a deep convolutional neural network. The proposed approach, SRCNN, learns an end-to-end mapping between low- and high-resolution images, with little extra pre/post-processing beyond the optimization. With a lightweight structure, the SRCNN has achieved superior performance than the state-of-the-art methods. We conjecture that additional performance can be further gained by exploring more filters and different training strategies. Besides, the proposed structure, with its advantages of simplicity and robustness, could be applied to other low-level vision problems, such as image deblurring or simultaneous SR+denoising. One could also investigate a network to cope with different upscaling factors.

References

[1] 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.

[2] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. A. Morel, "Low-complexity single-image super-resolution based on non-negative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 1–10.

[3] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 2392–2399.

[4] H. Chang, D. Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," presented at the *IEEE Conf. Comput. Vis. Pattern Recog.*, Washington, DC, USA, 2004.

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

[6] D. Dai, R. Timofte, and L. Van Gool, "Jointly optimized regressors for image super-resolution," *Eurographics*, vol. 7, p. 8, 2015.

[7] S. Dai, M. Han, W. Xu, Y. Wu, Y. Gong, and A. K. Katsaggelos, "Softcuts: A soft edge smoothness prior for color image super-resolution," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 969–981, May 2009.

[8] N. Damera-Venkata, T. D. Kite, W. S. Geisler, B. L. Evans, and A. C. Bovik, "Image quality assessment based on a degradation model," *IEEE Trans. Image Process.*, vol. 9, no. 11, pp. 636–650, Apr. 2000.

[9] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 248–255.

[10] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

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

[12] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 633–640.

[13] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Trans. Graph.*, vol. 30, no. 11, p. 12, 2011.

[14] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *Comput. Graph. Appl.*, vol. 22, no. 11, pp. 56–65, 2002.

[15] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *Int. J. Comput. Vis.*, vol. 40, no. 11, pp. 25–47, 2000.

[16] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 349–356.

[17] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3791–3799.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 346–361.

[19] J. B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 5197–5206.

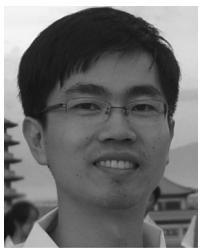
[20] M. Irani and S. Peleg, "Improving resolution by image registration," *Graph. Models Image Process.*, vol. 53, no. 11, pp. 231–239, 1991.

[21] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 769–776.

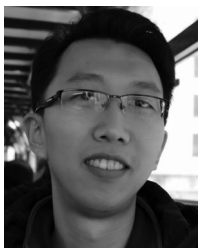
[22] K. Jia, X. Wang, and X. Tang, "Image transformation based on learning dictionaries across image spaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 367–380, Feb. 2013.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[24] 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.
[25] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.
[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[28] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 801–808.
[29] C. Liu, H. Y. Shum, and W. T. Freeman, "Face hallucination: Theory and practice," *Int. J. Comput. Vis.*, vol. 75, no. 11, pp. 115–134, 2007.
[30] F. Mamalet and C. Garcia, "Simplifying convNets for fast learning," in *Proc. Int. Conf. Artif. Neural Netw.*, 2012, pp. 58–65.
[31] 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. IEEE Int. Conf. Comput. Vis.*, 2001, vol. 2, pp. 416–423.
[32] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[33] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, and X. Tang, "Deepid-net: Deformable deep convolutional neural networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2015, pp. 2403–2412.
[34] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2056–2063.
[35] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Scholkopf, "A machine learning approach for non-blind image deconvolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1067–1074.
[36] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3791–3799.
[37] H. R. Sheikh, A. C. Bovik, and G. De Veciana, "An information fidelity criterion for image quality assessment using natural scene statistics," *IEEE Trans. Image Process.*, vol. 14, no. 11, pp. 2117–2128, Dec. 2005.
[38] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1988–1996.
[39] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1920–1927.
[40] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. IEEE Asian Conf. Comput. Vis.*, 2014, pp. 111–126.
[41] 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. 11, pp. 600–612, Apr. 2004.
[42] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. IEEE Conf. Rec. 37th Asilomar Conf. Signals, Syst. Comput.*, 2003, vol. 2, pp. 1398–1402.
[43] C. Y. Yang, J. B. Huang, and M. H. Yang, "Exploiting self-similarities for single frame super-resolution," in *Proc. IEEE Asian Conf. Comput. Vis.*, 2010, pp. 497–510.
[44] C. Y. Yang, C. Ma, and M. H. Yang, "Single-image super-resolution: A benchmark," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 372–386.
[45] J. Yang, Z. Lin, and S. Cohen, "Fast image super-resolution based on in-place example regression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1059–1066.
[46] 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. 11, pp. 3467–3478, Aug. 2012.
[47] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2008, pp. 1–8.
[48] 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.
[49] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. 7th Int. Conf. Curves Surfaces*, 2012, pp. 711–730.
[50] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.



Chao Dong received the BS degree in information engineering from the Beijing Institute of Technology, China, in 2011. He is currently working toward the PhD degree in the Department of Information Engineering, Chinese University of Hong Kong. His research interests include image super-resolution and denoising.



Chen Change Loy received the PhD degree in computer science from the Queen Mary University of London in 2010. He is currently a research assistant professor in the Department of Information Engineering, Chinese University of Hong Kong. Previously, he was a postdoctoral researcher at Vision Semantics Ltd. His research interests include computer vision and pattern recognition, with focus on face analysis, deep learning, and visual surveillance. He is a member of the IEEE.

Kaiming He received the BS degree from Tsinghua University in 2007, and the PhD degree from the Chinese University of Hong Kong in 2011. He is a researcher at Microsoft Research Asia (MSRA) since 2011. His research interests include computer vision and computer graphics. He has won the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He is a member of the IEEE.

Xiaoou Tang (S'93-M'96-SM'02-F'09) received the BS degree from the University of Science and Technology of China, Hefei, in 1990, the MS degree from the University of Rochester, New York, in 1991, and the PhD degree from the Massachusetts Institute of Technology, Cambridge, in 1996. He is a professor in the Department of Information Engineering and an associate dean (Research) of the Faculty of Engineering, Chinese University of Hong Kong. He was the group manager of the Visual Computing Group, Microsoft Research Asia, from 2005 to 2008. His research interests include computer vision, pattern recognition, and video processing. He received the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR. 2009. He was a program chair of the IEEE International Conference on Computer Vision (ICCV) 2009 and he is an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and the *International Journal of Computer Vision*. He is a fellow of the IEEE.