Learning a Deep Convolutional Network for Image Super-Resolution

Chao Dong[1], Chen Change Loy[1], Kaiming He[2], and Xiaoou Tang[1]

[1] Department of Information Engineering, The Chinese University of Hong Kong
[2] Microsoft Research

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) [15] 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.

Keywords: Super-resolution, deep convolutional neural networks

1 Introduction

Single image super-resolution (SR) [11] is a classical problem in computer vision. Recent state-of-the-art methods for single image super-resolution are mostly example-based. These methods either exploit internal similarities of the same image [7, 10, 23], or learn mapping functions from external low- and high-resolution exemplar pairs [2, 4, 9, 13, 20, 24–26, 28]. The external example-based methods are often provided with abundant samples, but are challenged by the difficulties of effectively and compactly modeling the data.

The sparse-coding-based method [25, 26] is one of the representative methods for external example-based image super-resolution. This method involves several steps in its pipeline. First, overlapping patches are densely extracted from the image and pre-processed (*e.g.*, subtracting mean). 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 (or averaged) to produce the output. Previous SR methods pay particular attention to learning and optimizing the dictionaries [25, 26] or alternative ways of modeling them [4, 2]. 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 the aforementioned pipeline is equivalent to a deep convolutional neural network [15] (more details in Section 3.2). Motivated by this

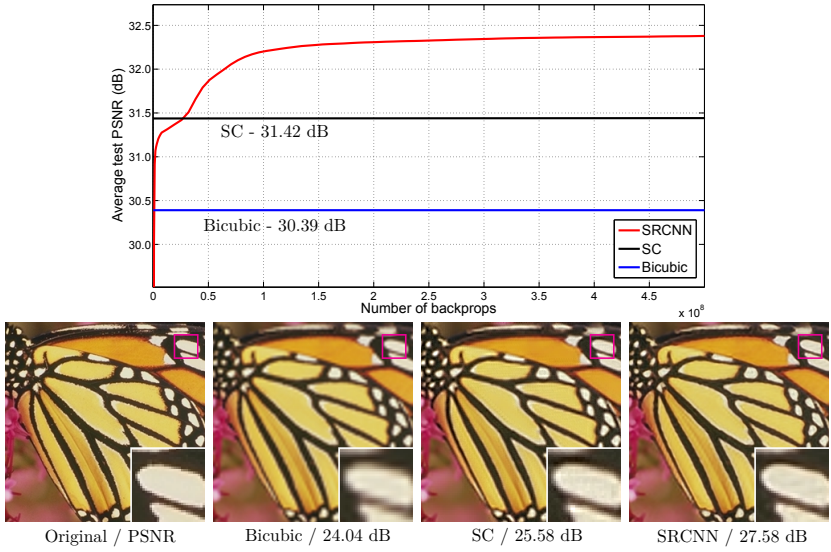

Fig. 1. The proposed Super-Resolution Convolutional Neural Network (SRCNN) surpasses the bicubic baseline with just a few training iterations, and outperforms the sparse-coding-based method (SC) [26] with moderate training. The performance may be further improved with more training iterations. More details are provided in Section 4.1 (the Set5 dataset with an upscaling factor 3). The proposed method provides visually appealing reconstruction from the low-resolution image.

fact, we directly consider a convolutional neural network which is 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 [20, 25, 26] 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] comparing with state-of-the-art example-based methods. Figure 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 series of 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

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

[2] Numerical evaluations in terms of Peak Signal-to-Noise Ratio (PSNR) when the ground truth images are available.

be further improved when (i) larger datasets are available, and/or (ii) a larger model is used. On the contrary, larger datasets/models can present challenges for existing example-based methods.

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

1. We present a 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.

2 Related Work

Image Super-Resolution. A category of state-of-the-art SR approaches [9, 4, 25, 26, 24, 2, 28, 20] learn a mapping between low/high-resolution patches. 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.* [8], 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 [25, 26], the above NN correspondence advances to a more sophisticated sparse coding formulation. This sparse-coding-based method and its several improvements [24, 20] 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.

Convolutional Neural Networks. Convolutional neural networks (CNN) date back decades [15] and have recently shown an explosive popularity partially due to its success in image classification [14]. Several factors are of central importance in this progress: (i) the efficient training implementation on modern powerful GPUs [14], (ii) the proposal of the Rectified Linear Unit (ReLU) [18] which makes convergence much faster while still presents good quality [14], and (iii) the easy access to an abundance of data (like ImageNet [5]) for training larger models. Our method also benefits from these progresses.

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 [19].

More closely related to our work, the convolutional neural network is applied for natural image denoising [12] and removing noisy patterns (dirt/rain) [6]. These restoration problems are more or less denoising-driven. On the contrary, the image super-resolution problem has not witnessed the usage of deep learning techniques to the best of our knowledge.

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]. Denote the interpolated image as $\mathbf{Y}$. Our goal is to recover from $\mathbf{Y}$ an image $F(\mathbf{Y})$ which 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 Figure 2. Next we detail our definition of each operation.

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:

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

[3] Actually, bicubic interpolation is also a convolutional operation, so 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 *convnet* [14], we exclude this "layer" from learning.

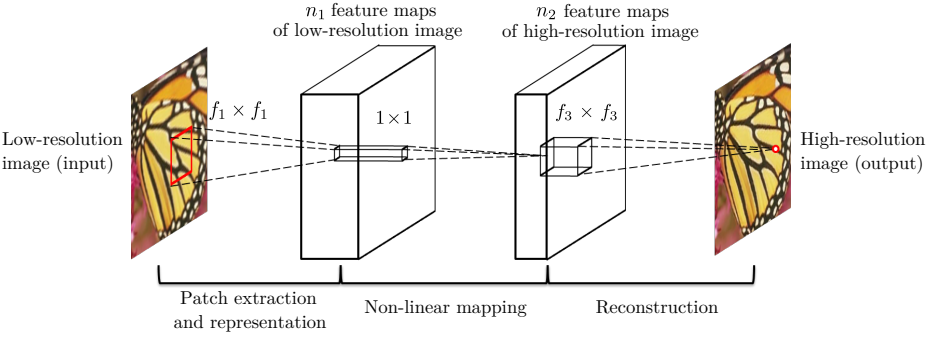

Fig. 2. Given a low-resolution image $\mathbf{Y}$, the first convolutional layer of the SRCNN extracts a set of feature maps. The second layer maps these feature maps nonlinearly to high-resolution patch representations. The last layer combines the predictions within a spatial neighbourhood to produce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

where W_1 and B_1 represent the filters and biases respectively. Here W_1 is of a size $c \times f_1 \times f_1 \times n_1$, where c is the number of channels in the input image, f_1 is the spatial size of a filter, and n_1 is the number of filters. 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 Rectified Linear Unit (ReLU, $\max(0, x)$) [18] on the filter responses[4].

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. The operation of the second layer is:

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

Here W_2 is of a size $n_1 \times 1 \times 1 \times n_2$, 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 (whose spatial supports are 1×1) to increase the non-linearity. But this can significantly increase the complexity of the model, and thus demands more training data and time. In this paper, we choose to use a single convolutional layer in this operation, because it has already provided compelling quality.

[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.

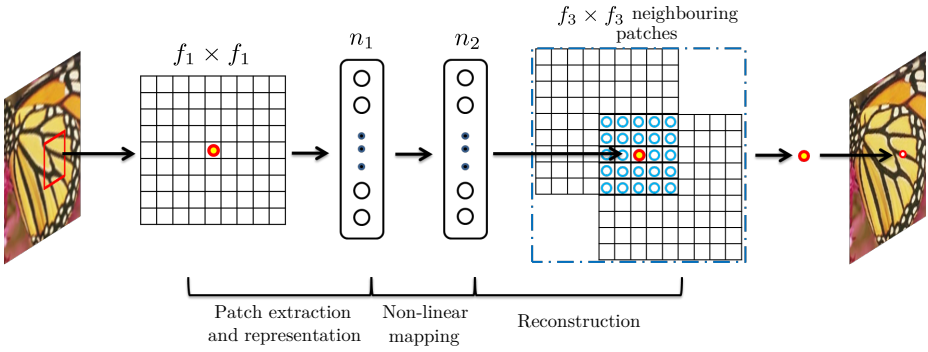

Fig. 3. An illustration of sparse-coding-based methods in the view of a convolutional neural network.

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 pre-defined 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 is of a size $n_2 \times f_3 \times f_3 \times c$, 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 (Figure 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 [25, 26]. 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 [25, 26] can be viewed as a convolutional neural network. Figure 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. This patch is subtracted by its mean, and then is projected 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 Figure 3.

A sparse coding solver will then be applied on the projected n_1 coefficients (*e.g.*, see the Feature-Sign solver [17]). 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 non-linear mapping operator. See the middle part of Figure 3. However, the 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. Our non-linear operator can be considered as a pixel-wise fully-connected layer.

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 Figure 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 setting is $f_1 = 9$, $f_3 = 5$, $n_1 = 64$, and $n_2 = 32$ (we evaluate more settings in the experiment section).

3.3 Loss Function

Learning the end-to-end mapping function F requires the estimation of 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. The loss is minimized using stochastic gradient descent with the standard backpropagation [16].

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. We will study this issue in the future. On the contrary, such a flexibility is in general difficult to achieve for traditional "hand-crafted" methods.

4 Experiments

Datasets. For a fair comparison with traditional example-based methods, we use the same training set, test sets, and protocols as in [20]. Specifically, the training set consists of 91 images. The **Set5** [2] (5 images) is used to evaluate the performance of upscaling factors 2, 3, and 4, and **Set14** [28] (14 images) is used to evaluate the upscaling factor 3. In addition to the 91-image training set, we also investigate a larger training set in Section 5.2.

Comparisons. We compare our **SRCNN** with the state-of-the-art SR methods: the **SC** (sparse coding) method of Yang *et al.* [26], the **K-SVD**-based method [28], **NE+LLE** (neighbour embedding + locally linear embedding) [4], **NE+NNLS** (neighbour embedding + non-negative least squares) [2], and the **ANR** (Anchored Neighbourhood Regression) method [20]. The implementations are all from the publicly available codes provided by the authors. For our implementation, the training is implemented using the *cuda-convnet* package [14].

Implementation Details. As per Section 3.2, we set $f_1 = 9$, $f_3 = 5$, $n_1 = 64$ and $n_2 = 32$ in our main evaluations. We will evaluate alternative settings in the Section 5. For each upscaling factor $\in \{2, 3, 4\}$, we train a specific network for that factor[5].

In the training phase, the ground truth images $\{\mathbf{X}_i\}$ are prepared as 32×32-pixel[6] 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 proper Gaussian kernel, sub-sample it by the upscaling factor, and upscale it by the same factor via bicubic interpolation. The 91 training images provide roughly 24,800 sub-images. The sub-images are extracted from original images with a stride of 14. We attempted smaller strides but did not observe significant performance improvement. From our observation, the training set is sufficient to train the proposed deep network. The training (8×10^8 backpropagations) takes roughly three days, on a GTX 770 GPU.

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

[6] The input size is 33×33 for an upscaling factor 3, so can be divided by 3.

Following [20], we only consider the luminance channel (in YCrCb color space) in our experiments, so $c = 1$ in the first/last layer. The two chrominance channels are bicubic upsampled only for the purpose of displaying, but not for training/testing. Note that our method can be extended to directly training on color images by setting $c = 3$. We use $c = 1$ in this paper mainly for fair comparison with previous methods, as most of them only concern the luminance channels.

To avoid border effects during training, all the convolutional layers have no padding, and the network produces a smaller output (20×20). The MSE loss function is evaluated only by the difference between the central 20×20 crop of $\mathbf{X}_i$ and the network output. In processing test images, the convolutional neural network can be applied on images of arbitrary sizes. All the convolutional layers are given sufficient zero-padding during testing, so that the output image is of the same size as the input. To address the border effects, in each convolutional layer, the output (before ReLU) at each pixel is normalized by the number of valid input pixels, which can be computed beforehand.

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 [12]).

4.1 Quantitative Evaluation

As shown in Tables 1 and 2, the proposed SRCNN yields the highest average PSNR in all experiments. Note that our SRCNN results are based on the checkpoint of 8×10^8 backpropagations. Specifically, as shown in the Table 1 (Set5), the average gains achieved by SRCNN are 0.51 dB, 0.47 dB, and 0.40 dB, higher than the next best approach, ANR [20], on all the three upscaling factors. We note that Set5 may not be a conclusive test set due to the limited number of test samples, but the results are indicative that the proposed model can handle different upscaling factors well. On the larger Set14 dataset, our SRCNN consistently outperforms other methods by a large margin (≥ 0.3 dB on average). A similar trend is observed when we used SSIM [22, 21] as the performance metric, the results of which could be found in the supplementary file. It is worth pointing out that SRCNN surpasses the bicubic baseline at the very beginning of the learning stage (see Figure 1), and with moderate training, SRCNN outperforms existing state-of-the-art methods (see Figure 6). Yet, the performance is far from converge. We conjecture that better results can be obtained given longer training time (see Figure 6). In Section 5.2, we will show that our method also benefits from more training data.

Figures 7, 8 and 9 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. In spite of the best average PSNR values, the proposed SRCNN does not achieve the highest PSNR on images "baby" and "head" from Set5. Nevertheless, our results are still visually appealing (see Figure 10).

Table 1. The results of PSNR (dB) and test time (sec) on the Set5 dataset.

Set5 [2]		Bicubic		SC [26]		K-SVD [28]		NE+NNLS [2]		NE+LLE [4]		ANR [20]		SRCNN	
images	Scale	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
baby	2	37.07	-	-	-	38.25	7.0	38.00	68.6	38.33	13.6	**38.44**	2.1	38.30	0.38
bird	2	36.81	-	-	-	39.93	2.2	39.41	22.5	40.00	4.2	40.04	0.62	**40.64**	0.14
butterfly	2	27.43	-	-	-	30.65	1.8	30.03	16.6	30.38	3.3	30.48	0.50	**32.20**	0.10
head	2	34.86	-	-	-	35.59	2.1	35.48	19.2	35.63	3.8	**35.66**	0.57	35.64	0.13
woman	2	32.14	-	-	-	34.49	2.1	34.24	19.3	34.52	3.8	34.55	0.57	**34.94**	0.13
average	2	33.66	-	-	-	35.78	3.03	35.43	29.23	35.77	5.74	35.83	0.87	**36.34**	0.18
baby	3	33.91	-	34.29	76.0	35.08	3.3	34.77	28.3	35.06	6.0	**35.13**	1.3	35.01	0.38
bird	3	32.58	-	34.11	30.4	34.57	1.0	34.26	8.9	34.56	1.9	34.60	0.39	**34.91**	0.14
butterfly	3	24.04	-	25.58	26.8	25.94	0.81	25.61	7.0	25.75	1.4	25.90	0.31	**27.58**	0.10
head	3	32.88	-	33.17	21.3	33.56	1.0	33.45	8.2	33.60	1.7	**33.63**	0.35	33.55	0.13
woman	3	28.56	-	29.94	25.1	30.37	1.0	29.89	8.7	30.22	1.9	30.33	0.37	**30.92**	0.13
average	3	30.39	-	31.42	35.92	31.90	1.42	31.60	12.21	31.84	2.58	31.92	0.54	**32.39**	0.18
baby	4	31.78	-	-	-	33.06	2.4	32.81	16.2	32.99	3.6	**33.03**	0.85	32.98	0.38
bird	4	30.18	-	-	-	31.71	0.68	31.51	4.7	31.72	1.1	31.82	0.27	**31.98**	0.14
butterfly	4	22.10	-	-	-	23.57	0.50	23.30	3.8	23.38	0.90	23.52	0.24	**25.07**	0.10
head	4	31.59	-	-	-	32.21	0.68	32.10	4.5	32.24	1.1	**32.27**	0.27	32.19	0.13
woman	4	26.46	-	-	-	27.89	0.66	27.61	4.3	27.72	1.1	27.80	0.28	**28.21**	0.13
average	4	28.42	-	-	-	29.69	0.98	29.47	6.71	29.61	1.56	29.69	0.38	**30.09**	0.18

Table 2. The results of PSNR (dB) and test time (sec) on the Set14 dataset.

Set14 [28]		Bicubic		SC [26]		K-SVD [28]		NE+NNLS [2]		NE+LLE [4]		ANR [20]		SRCNN	
images	scale	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
baboon	3	23.21	-	23.47	126.3	23.52	3.6	23.49	29.0	23.55	5.6	23.56	1.1	**23.60**	0.40
barbara	3	26.25	-	26.39	127.9	**26.76**	5.5	26.67	47.6	26.74	9.8	26.69	1.7	26.66	0.70
bridge	3	24.40	-	24.82	152.7	25.02	3.3	24.86	30.4	24.98	5.9	25.01	1.1	**25.07**	0.44
coastguard	3	26.55	-	27.02	35.6	27.15	1.3	27.00	11.6	27.07	2.6	27.08	0.45	**27.20**	0.17
comic	3	23.12	-	23.90	54.5	23.96	1.2	23.83	11.0	23.98	2.0	24.04	0.42	**24.39**	0.15
face	3	32.82	-	33.11	20.4	33.53	1.1	33.45	8.3	33.56	1.7	**33.62**	0.34	33.58	0.13
flowers	3	27.23	-	28.25	76.4	28.43	2.3	28.21	20.2	28.38	4.0	28.49	0.81	**28.97**	0.30
foreman	3	31.18	-	32.04	25.9	33.19	1.3	32.87	10.8	33.21	2.2	33.23	0.44	**33.35**	0.17
lenna	3	31.68	-	32.64	68.4	33.00	3.3	32.82	29.3	33.01	6.0	33.08	1.1	**33.39**	0.44
man	3	27.01	-	27.76	111.2	27.90	3.4	27.72	29.5	27.87	6.1	27.92	1.1	**28.18**	0.44
monarch	3	29.43	-	30.71	112.1	31.10	4.9	30.76	43.3	30.95	8.8	31.09	1.6	**32.39**	0.66
pepper	3	32.39	-	33.32	66.3	34.07	3.3	33.56	28.9	33.80	6.6	33.82	1.1	**34.35**	0.44
ppt3	3	23.71	-	24.98	96.1	25.23	4.0	24.81	36.0	24.94	7.8	25.03	1.4	**26.02**	0.58
zebra	3	26.63	-	27.95	114.4	28.49	2.9	28.12	26.3	28.31	5.5	28.43	1.0	**28.87**	0.38
average	3	27.54	-	28.31	84.88	28.67	2.95	28.44	25.87	28.60	5.35	28.65	0.97	**29.00**	0.39

4.2 Running Time

Figure 4 shows the running time comparisons of several state-of-the-art methods, along with their restoration performance. All baseline methods are obtained from the corresponding authors' MATLAB implementation, whereas ours are in C++. We profile the running time of all the algorithms using the same machine (Intel CPU 3.10 GHz and 16 GB memory)[7]. Our method takes 0.39 sec per image on average in Set14 (Table 2), whereas other methods are several times or even orders of magnitude slower. 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. We also 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.

[7] The running time may be slightly different from that reported in [20] due to different machines.

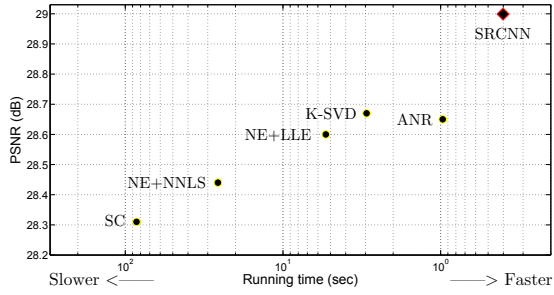

Fig. 4. The proposed SRCNN achieves the state-of-the-art super-resolution quality, whilst maintains high and competitive speed in comparison to existing external example-based methods. The chart is based on Set14 results summarized in Table 2.

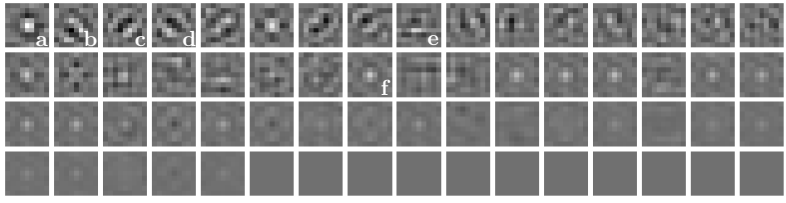

Fig. 5. The figure shows first-layer filters trained on 91 images with an upscaling factor 2. The filters are organized based on their respective variances.

5 Further Analyses

5.1 Learned Filters for Super-resolution

Figure 5 shows examples of learned first-layer filters trained on 91 images (24,800 sub-images) by an upscaling factor 2. Please refer to our published implementation for the patterns of upscaling factors 3 and 4. Interestingly, each learned filter has its specific functionality. For instance, the filters a and f are like Laplacian/Gaussian filters, the filters b, c, and d are like edge detectors at different directions, and the filter e is like a texture extractor. We observe some "dead" filters, whose weights are all nearly zeros, similar to those observed in [27]. Nevertheless, patterns may emerge in some of these dead filters given long enough training time. We will investigate this phenomenon in future work.

5.2 Learning Super-resolution from ImageNet

As shown in the literature, deep learning generally benefits from big data training. In the above experiments, we use a standard training set that consists of 91 images to ensure fair comparison with existing methods. In this section, we show that our deep model could achieve better performance given a large training set. We use a total of 395,909 images from the ILSVRC 2013 ImageNet detection training partition for SRCNN learning. These images are decomposed into over

5 million sub-images using a stride of 33. We use the same network settings as the above experiments, *i.e.* $f_1 = 9$, $f_3 = 5$, $n_1 = 64$ and $n_2 = 32$. The training time on ImageNet is about the same as on 91 images since the number of back-propagations is the same. The experiments are tested on Set5 with an upscaling factor 3. The test convergence curve trained on ImageNet and results of other methods are shown in Figure 6. 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 the original SRCNN trained on 91 images (or 24,800 sub-images). The results positively indicate that SRCNN performance may be further boosted using a larger and more diverse image training set.

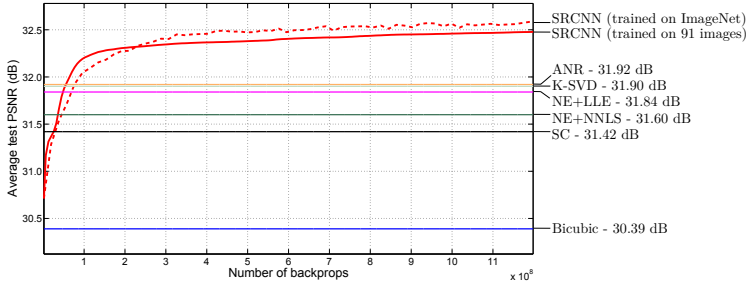

Fig. 6. The test convergence curve trained on ImageNet and results of other methods on the Set5 dataset.

5.3 Filter Number

In comparison to other CNN structures [14], we use a relatively small network scale to achieve the state-of-the-art performance in super-resolution. In general, the performance would still improve if we enlarge the network scale, *e.g.* adding more layers and filters, at the cost of running time. Here, we evaluate the performance of using different numbers of filters. Specifically, based on our network default setting of $n_1 = 64$ and $n_2 = 32$, we conduct two additional 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 5.2, we also train the two models on ImageNet and test on Set5 with an upscaling factor 3. The results are shown in Table 3. It is clear that superior performance could be achieved using more filters. However, if a fast restoration speed is desired, a small network scale is preferred, which could still achieve better performance than the state-of-the-art.

Table 3. The results of using different filter numbers in SRCNN. Training is performed on ImageNet whilst the evaluation is conducted on the Set5 dataset.

Set5 [2]	$n_1 = 128$, $n_2 = 64$		$n_1 = 64$, $n_2 = 32$		$n_1 = 32$, $n_2 = 16$	
images	PSNR	Time	PSNR	Time	PSNR	Time
	32.60	0.60	32.52	0.18	32.26	0.05

5.4 Filter Size

In this section, we examine the network sensitivity to different filter sizes. In previous experiments, we set filter size of the first layer as $f_1 = 9$ and that of the last layer as $f_3 = 5$. Here, we enlarge the filter size to $f_1 = 11$ and $f_3 = 7$. All the other settings remain the same with Section 5.2. 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 5.2. This suggests that a reasonably larger filter size could grasp richer structural information, which in turn lead to better results. However, the deployment speed will also decrease with a larger filter size. Therefore, the choice of the network scale should always be a trade-off between performance and speed.

6 Conclusion

We have presented a novel deep learning approach for single image super-resolution (SR). We show that conventional sparse-coding-based image super-resolution 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 hidden layers/filters in the network, 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.

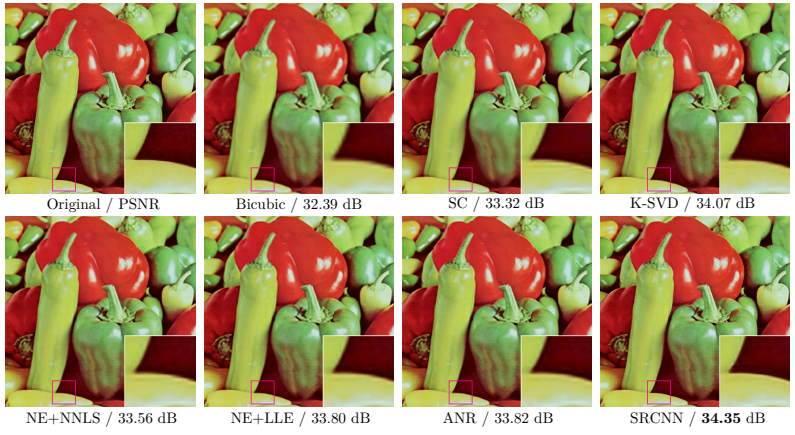

Fig. 7. "Pepper" image from Set14 with an upscaling factor 3.

Fig. 8. "Butterfly" image from Set5 with an upscaling factor 3.

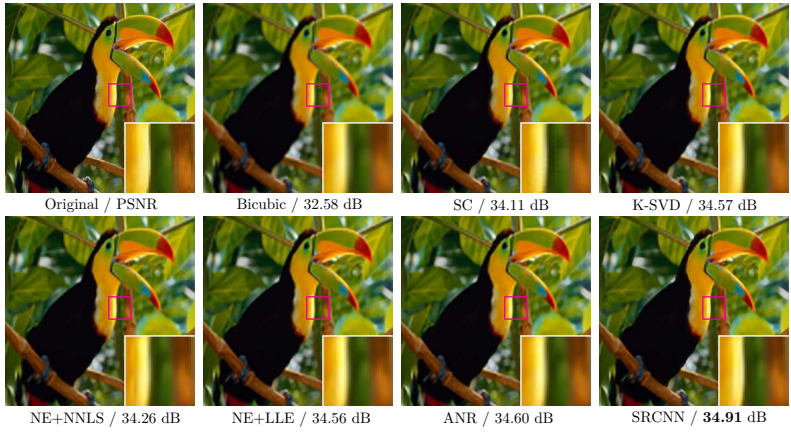

Fig. 9. "Bird" image from Set5 with an upscaling factor 3.

Fig. 10. "Baby" image from Set5 with an upscaling factor 3.

References

1. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. TSP 54(11), 4311–4322 (2006)
2. Bevilacqua, M., Roumy, A., Guillemot, C., Morel, M.L.A.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: BMVC (2012)
3. Burger, H.C., Schuler, C.J., Harmeling, S.: Image denoising: Can plain neural networks compete with BM3D? In: CVPR. pp. 2392–2399 (2012)
4. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: CVPR (2004)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
6. Eigen, D., Krishnan, D., Fergus, R.: Restoring an image taken through a window covered with dirt or rain. In: ICCV. pp. 633–640 (2013)
7. Freedman, G., Fattal, R.: Image and video upscaling from local self-examples. TOG 30(2), 12 (2011)
8. Freeman, W.T., Jones, T.R., Pasztor, E.C.: Example-based super-resolution. Computer Graphics and Applications 22(2), 56–65 (2002)
9. Freeman, W.T., Pasztor, E.C., Carmichael, O.T.: Learning low-level vision. IJCV 40(1), 25–47 (2000)
10. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: ICCV. pp. 349–356 (2009)
11. Irani, M., Peleg, S.: Improving resolution by image registration. CVGIP 53(3), 231–239 (1991)
12. Jain, V., Seung, S.: Natural image denoising with convolutional networks. In: NIPS. pp. 769–776 (2008)
13. Jia, K., Wang, X., Tang, X.: Image transformation based on learning dictionaries across image spaces. TPAMI 35(2), 367–380 (2013)
14. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: NIPS. pp. 1097–1105 (2012)
15. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation pp. 541–551 (1989)
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
17. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS. pp. 801–808 (2006)
18. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML. pp. 807–814 (2010)
19. Schuler, C.J., Burger, H.C., Harmeling, S., Scholkopf, B.: A machine learning approach for non-blind image deconvolution. In: CVPR. pp. 1067–1074 (2013)
20. Timofte, R., De Smet, V., Van Gool, L.: Anchored neighborhood regression for fast example-based super-resolution. In: ICCV. pp. 1920–1927 (2013)
21. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. TIP 13(4), 600–612 (2004)
22. Yang, C.Y., Yang, M.H.: Fast direct super-resolution by simple functions. In: ICCV. pp. 561–568 (2013)
23. Yang, J., Lin, Z., Cohen, S.: Fast image super-resolution based on in-place example regression. In: CVPR. pp. 1059–1066 (2013)

24. Yang, J., Wang, Z., Lin, Z., Cohen, S., Huang, T.: Coupled dictionary training for image super-resolution. TIP 21(8), 3467–3478 (2012)
25. Yang, J., Wright, J., Huang, T., Ma, Y.: Image super-resolution as sparse representation of raw image patches. In: CVPR. pp. 1–8 (2008)
26. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. TIP 19(11), 2861–2873 (2010)
27. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional neural networks. Tech. rep. (2013)
28. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Curves and Surfaces, pp. 711–730. Springer (2012)