Image Super-Resolution with Deep Dictionary

Shunta Maeda(✉)

Navier Inc., Tokyo, Japan
shunta@navier.co.jp

Abstract. Since the first success of Dong et al., the deep-learning-based approach has become dominant in the field of single-image super-resolution. This replaces all the handcrafted image processing steps of traditional sparse-coding-based methods with a deep neural network. In contrast to sparse-coding-based methods, which explicitly create high/low-resolution dictionaries, the dictionaries in deep-learning-based methods are implicitly acquired as a nonlinear combination of multiple convolutions. One disadvantage of deep-learning-based methods is that their performance is degraded for images created differently from the training dataset (out-of-domain images). We propose an end-to-end super-resolution network with a deep dictionary (SRDD), where a high-resolution dictionary is explicitly learned without sacrificing the advantages of deep learning. Extensive experiments show that explicit learning of high-resolution dictionary makes the network more robust for out-of-domain test images while maintaining the performance of the in-domain test images. Code is available at https://github.com/shuntama/srdd.

Keywords: Super-resolution · Deep dictionary · Sparse representation

1 Introduction

Single-image super-resolution (SISR) is a classical problem in the field of computer vision that predicts a high-resolution (HR) image from its low-resolution (LR) observation. Because this is an ill-posed problem with multiple possible solutions, obtaining a rich prior based on a large number of data points is beneficial for better prediction. Deep learning is quite effective for such problems. The performance of SISR has been significantly improved by using convolutional neural networks (CNN), starting with the pioneering work of Dong et al. in 2014 [9]. Before the dominance of deep-learning-based methods [1,9,10,19,20,23,26,27,41,42,56,57] in this field, example-based methods [4,12,13,16,21,44,45,51,52] were mainly used for learning priors. Among them, sparse coding, which is a representative example-based method, has shown

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19800-7_27.

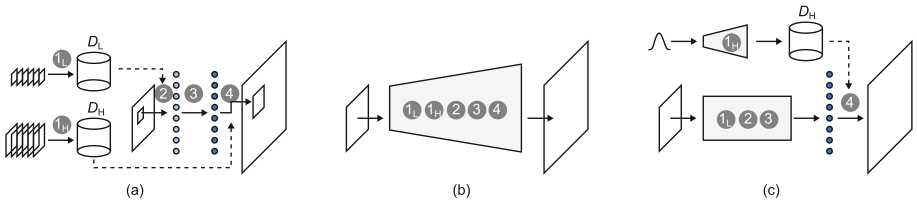

Fig. 1. Schematic illustrations of single image super-resolution with (a) sparse-coding-based approach, (b) conventional deep-learning-based approach, and (c) our approach. The numbers ①-④ indicate each step of the super-resolution process.

state-of-the-art performance [44,45,51]. SISR using sparse coding comprises the following steps, as illustrated in Fig. 1(a): $\textcircled{1_L}$ learn an LR dictionary D_L with patches extracted from LR images, $\textcircled{1_H}$ learn an HR dictionary D_H with patches extracted from HR images, ② represent patches densely cropped from an input image with D_L, ③ map D_L representations to D_H representations, ④ reconstruct HR patches using D_H, then aggregate the overlapped HR patches to produce a final output.

As depicted in Fig. 1(b), Dong et al. [9] replaced all the above handcrafted steps with a multilayered CNN in their proposed method SRCNN to take advantage of the powerful capability of deep learning. Note that, in this method, D_L and D_H are implicitly acquired through network training. Since the SRCNN, various methods have been proposed to improve performance, for example, by deepening the network with residual blocks and skip connections [20,27,41,57], applying attention mechanisms [8,31,34,56], and using a transformer [6,26]. However, most of these studies, including state-of-the-art ones, follow the same formality as SRCNN from a general perspective, where all the processes in the sparse-coding-based methods are replaced by a multilayered network.

One disadvantage of deep-learning-based methods is that their performance is degraded for images created differently from the training dataset [14]. Although there have been several approaches to address this issue, such as training networks for multiple degradations [40,46,49,55,59] and making models agnostic to degradations with iterative optimizations [14,38], it is also important to make the network structure more robust. We hypothesize that D_H implicitly learned inside a multilayered network is fragile to subtle differences in input images from the training time. This hypothesis leads us to the method we propose.

In this study, we propose an end-to-end super-resolution network with a deep dictionary (SRDD), where D_H is explicitly learned through the network training (Fig. 1(c)). The main network predicts the coefficients of D_H and the weighted sum of the elements (or atoms) of D_H produces an HR output. This approach is fundamentally different from the conventional deep-learning-based approach, where the network has upsampling layers inside it. The upsampling process of the proposed method is efficient because the pre-generated D_H can be used as a magnifier for inference. In addition, the main network does not need to maintain the information of the processed image at the pixel level in HR space. Therefore, the network can concentrate only on predicting the coefficients of D_{H}. For in-domain test images, our method shows performance that is not as good as latest ones, but close to the conventional baselines (e.g., CARN). For out-of-domain test images, our method shows superior performance compared to conventional deep-learning-based methods.

2 Related Works

2.1 Sparse-Coding-Based SR

Before the dominance of deep-learning-based methods in the field of SISR, example-based methods showed state-of-the-art performance. The example-based methods exploit internal self-similarity [11,13,16,50] and/or external datasets [4,12,21,44,45,51,52]. The use of external datasets is especially important for obtaining rich prior. In the sparse-coding-based methods [44,45,51,52], which are state-of-the-art example-based methods, high/low-resolution patch pairs are extracted from external datasets to create high/low-resolution dictionaries $D_{\mathrm{H}}/D_{\mathrm{L}}$. The patches cropped from an input image are encoded with D_{L} and then projected onto D_{H} via iterative processing, producing the final output with appropriate patch aggregation.

2.2 Deep-Learning-Based SR

Deep CNN. All the handcrafted steps in the traditional sparse-coding-based approach were replaced with an end-to-end CNN in a fully feed-forward manner. Early methods, including SRCNN [9,19,20], adopted pre-upsampling in which LR input images are first upsampled for the SR process. Because the pre-upsampling is computationally expensive, post-upsampling is generally used in recent models [1,26,27,31,56]. In post-upsampling, a transposed convolution or pixelshuffle [37] is usually used to upsample the features for final output. Although there are many proposals to improve network architectures [25,54], the protocol that directly outputs SR images with post-upsampling has been followed in most of those studies. Few studies have focused on the improvement of the upsampling strategy. Tough some recent works [3,5,60] leveraged the pre-trained latent features as a dictionary to improve output fidelity with rich textures, they used standard upsampling strategies in their proposed networks.

Convolutional Sparse Coding. Although methods following SRCNN have been common in recent years, several fundamentally different approaches have been proposed before and after the proposal of SRCNN. Convolutional sparse coding [15,35,39,47] is one of such methods that work on the entire image differently from traditional patch-based sparse coding. The advantage of convolutional sparse coding is that it avoids the boundary effect in patch-based sparse coding. However, it conceptually follows patch-based sparse coding in that the overall SR process is divided into handcrafted steps. Consequently, its performance lags behind that of end-to-end feed-forward CNN.

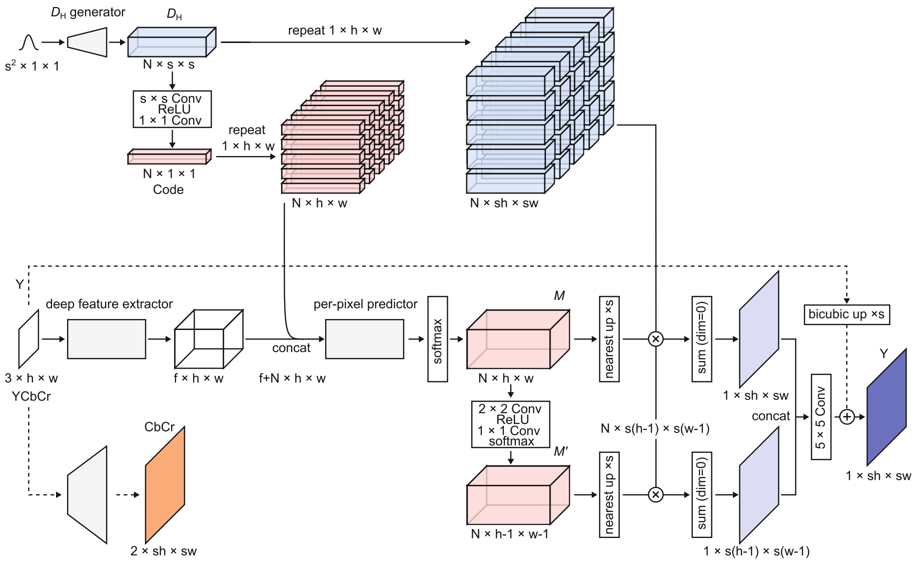

Fig. 2. The overall pipeline of the proposed method. A high-resolution dictionary D_{H} is generated from random noise. An encoded code of D_{H} is then concatenated with an extracted feature to be inputted to a per-pixel predictor. The predictor output is used to reconstruct the final output based on D_{H}.

Robust SR. The performance of deep-learning-based SR is significantly affected by the quality of the input image, especially the difference in conditions from the training dataset [14]. Several approaches have been proposed to make the network more robust against in-domain test images by training with multiple degradations [40,46,49,55,59]. For robustness against out-of-domain test images, some studies aim to make the network agnostic to degradations [14,38]. In these methods, agnostics acquisition is generally limited to specific degradations; therefore, it is important to make the network structure itself more robust.

3 Method

As depicted in Fig. 1(c), the proposed method comprises three components: D_{H} generation, per-pixel prediction, and reconstruction. The D_{H} generator generates an HR dictionary D_{H} from random noise input. The per-pixel predictor predicts the coefficients of D_{H} for each pixel from an LR YCbCr input. In the reconstruction part, the weighted sum of the elements (or atoms) of D_{H} produces an HR Y-channel output as a residual to be added to a bicubically upsampled Y channel. The remaining CbCr channels are upscaled with a shallow SR network. We used ESPCN [37] as the shallow SR network in this work. All of these components can be simultaneously optimized in an end-to-end manner; therefore, the same training procedure can be used as in conventional deep-learning-based SR methods. We use L_1 loss function to optimize the network

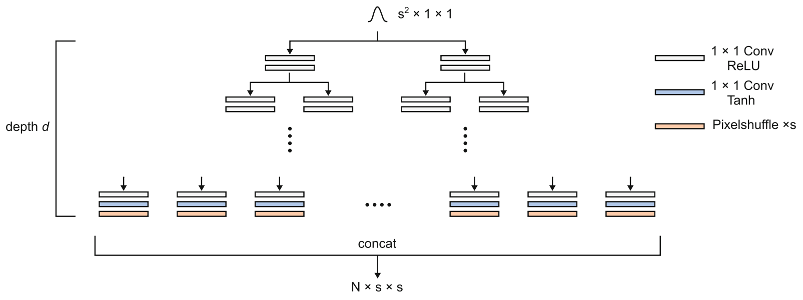

Fig. 3. A generator architecture of a high-resolution dictionary D_H. A tree-like network with depth d generates 2^d atoms of size $1 \times s \times s$ from a random noise input, where s is an upscaling factor.

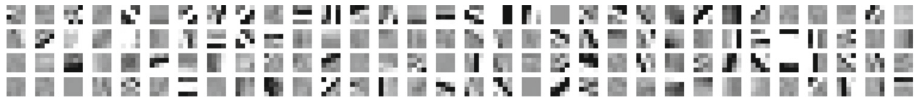

Fig. 4. Learned atoms of ×4 SRDD with $N = 128$. The size of each atom is $1 \times 4 \times 4$. The data range is renormalized to $[0, 1]$ for visualization.

$$L = \frac{1}{M} \sum_{i=1}^{M} ||I_i^{gt} - \Theta(I_i^{lr})||_1, \tag{1}$$

where I_i^{lr} and I_i^{gt} are LR patch and its ground truth. M denotes the number of training image pairs. $\Theta(\cdot)$ represents a function of the SRDD network. Figure 2 illustrates the proposed method in more detail. We describe the design of each component based on Fig. 2 in the following subsections.

3.1 D_H Generation

From random noise $\delta^{s^2 \times 1 \times 1}$ $(\in \mathbb{R}^{s^2 \times 1 \times 1})$ with a standard normal distribution, the D_H generator generates the HR dictionary $D_\mathrm{H}^{N \times s \times s}$, where s is an upscaling factor and N is the number of elements (atoms) in the dictionary. D_H is then encoded by $s \times s$ convolution with groups N, followed by ReLU [33] and 1×1 convolution. Each N element of the resultant code $C_\mathrm{H}^{N \times 1 \times 1}$ represents each $s \times s$ atom as a scalar value. Although the D_H can be trained using a fixed noise input, we found that introducing input randomness improves the stability of the training. A pre-generated fixed dictionary and its code are used in the testing phase. Note that only D_H is generated since low-resolution dictionaries (encoding) can be naturally replaced by convolutional operations without excessive increases in computation.

As illustrated in Fig. 3, the D_H generator has a tree-like structure, where the nodes consist of two 1×1 convolutional layers with ReLU activation. The final layer has a Tanh activation followed by a pixelshuffling layer; therefore, the data range of the output atoms is $[-1, 1]$. To produce N atoms, depth d of the generator is determined as

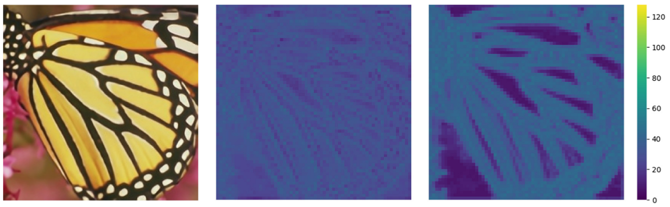

Fig. 5. Visualization of sparsity of a prediction map (center) and its complementary prediction map (right). The number of predicted coefficients larger than 1*e*–2 is counted for each pixel. More atoms are assigned to the high-frequency parts and the low-frequency parts are relatively sparse.

$$d = \log_2 N. \tag{2}$$

Figure 4 shows generated atoms with $s = 4$ and $N = 128$. We observed that the contrast of the output atoms became stronger as training progressed, and they were almost fixed in the latter half of the training.

3.2 Per-pixel Prediction

We utilize UNet++ [61] as a deep feature extractor in Fig. 2. We slightly modify the original UNet++ architecture: the depth is reduced from four to three, and a long skip connection is added. The deep feature extractor outputs a tensor of size $f \times h \times w$ from the input YCbCr image, where h and w are the height and width of the image, respectively. Then the extracted feature is concatenated with the expanded code of D_{H}

$$C_{\mathrm{H}}^{N\times h\times w} = R_{1\times h\times w}(C_{\mathrm{H}}^{N\times 1\times 1}), \tag{3}$$

to be inputted to a per-pixel predictor, where $R_{a\times b\times c}(\cdot)$ denotes the $a\times b\times c$ repeat operations. The per-pixel predictor consists of ten bottleneck residual blocks followed by a softmax function that predicts N coefficients of D_{H} for each input pixel. Both the deep feature extractor and per-pixel predictor contain batch normalization layers [18] before the ReLU activation. The resultant prediction map $M^{N\times h\times w}$ is further convolved with a 2×2 convolution layer to produce a complementary prediction map $M'^{N\times(h-1)\times(w-1)}$. A complementary prediction map is used to compensate for the patch boundaries when reconstructing the final output. The detail of the compensation mechanism is described in the next subsection. Although we tried to replace softmax with ReLU to directly express sparsity, ReLU made the training unstable. We also tried entmax [36], but the performance was similar to that of softmax, so we decided to use softmax for simplicity.

Figure 5 visualizes the sparsity of the prediction map and its complementary prediction map. The number of coefficients larger than $1e-2$ is counted for each pixel to visualize the sparsity. The model with $N = 128$ is used. More atoms are assigned to the high-frequency parts of the image, and the low-frequency parts are relatively sparse. This feature is especially noticeable in the complementary

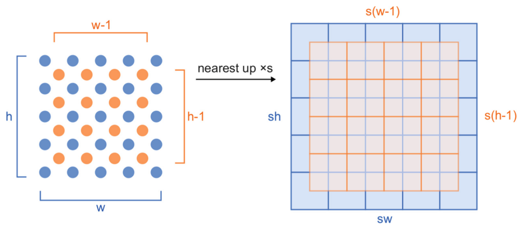

Fig. 6. Schematic illustration of a mechanism to compensate patch boundary with a complementary prediction map, where s is a scaling factor. Left: Prediction map (blue) and its complementary prediction map (orange). Right: Upsampled prediction and complementary prediction maps with centering. (Color figure online)

prediction map. In the high-frequency region, the output image is represented by linear combinations of more than tens of atoms for both maps.

3.3 Reconstruction

The prediction map $M^{N\times h\times w}$ is upscaled to $N\times sh\times sw$ by nearest-neighbor interpolation, and the element-wise multiplication of that upscaled prediction map $U_s(M^{N\times h\times w})$ with the expanded dictionary $R_{1\times h\times w}(D_{\mathrm{H}}^{N\times s\times s})$ produces $N\times sh\times sw$ tensor T consists of weighted atoms. The $U_s(\cdot)$ denotes $\times s$ nearest-neighbor upsampling. Finally, tensor T is summed over the first dimension, producing output x as

$$x^{1\times sh\times sw} = \sum_{k=0}^{N-1} T^{N\times sh\times sw}[k,:,:], \tag{4}$$

$$T^{N\times sh\times sw} = U_s(M^{N\times h\times w}) \otimes R_{1\times h\times w}(D_{\mathrm{H}}^{N\times s\times s}). \tag{5}$$

The same sequence of operations is applied to the complementary prediction map to obtain the output x' as follows:

$$x'^{1\times s(h-1)\times s(w-1)} = \sum_{k=0}^{N-1} T'^{N\times s(h-1)\times s(w-1)}[k,:,:], \tag{6}$$

$$T'^{N\times s(h-1)\times s(w-1)} = U_s(M'^{N\times(h-1)\times(w-1)}) \otimes R_{1\times(h-1)\times(w-1)}(D_{\mathrm{H}}^{N\times s\times s}). \tag{7}$$

Note that the same dictionary, D_{H}, is used to obtain x and x'. By centering x and x', as illustrated in Fig. 6, the imperfections at the patch boundaries can complement each other. The final output residual is obtained by concatenating the overlapping parts of the centered x and x' and applying a 5×5 convolution. For non-overlapping parts, x is simply used as the final output.

4 Experiments

4.1 Implementation Details

We adopt a model with 128 atoms (SRDD-128) and a small model with 64 atoms (SRDD-64). The number of filters of the models is adjusted according to the number of atoms. Our network is trained by inputting 48×48 LR patches with a mini-batch size of 32. Following previous studies [1,27,56], random flipping and rotation augmentation is applied to each training sample. We use Adam optimizer [22] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate of the network except for the D_{H} generator is initialized as $2e{-}4$ and halved at [200k, 300k, 350k, 375k]. The total training iterations is 400k. The learning rate of the D_{H} generator is initialized as $5e{-}3$ and halved at [50k, 100k, 200k, 300k, 350k]. Parameters of the D_{H} generator are frozen at 360k iteration. In addition, to stabilize training of the D_{H} generator, we randomly shuffle the order of output atoms for the first 1k iterations. We use PyTorch to implement our model with an NVIDIA P6000 GPU. Training takes about two/three days for SRDD-64/128, respectively. More training details are provided in the supplementary material.

4.2 Dataset and Evaluation

Training Dataset. Following previous studies, we use 800 HR-LR image pairs of the DIV2K [43] training dataset to train our models. LR images are created from HR images by Matlab bicubic downsampling. For validation, we use initial ten images from the DIV2K validation dataset.

Test Dataset. For testing, we evaluate the models on five standard benchmarks: Set5 [2], Set14 [53], BSD100 [29], Urban100 [16], and Manga109 [30]. In addition to standard test images downsampled with Matlab bicubic function same as in training, we use test images that downsampled by OpenCV bicubic, bilinear, and area functions to evaluate the robustness of the models. In addition, we evaluate the models on real-world ten historical photographs.

Evaluation. We use common image quality metrics peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) [48] calculated on the Y channel (luminance channel) of YCbCr color space. For evaluation of real-world test images, no-reference image quality metric NIQE [32] is used since there are no ground-truth images. Following previous studies, we ignore s pixels from the border to calculate all the metrics, where s is an SR scale factor.

4.3 Ablation Study

We conduct ablation experiments to examine the impact of individual elements in SRDD. We report the results of SRDD-64 throughout this section. The results of the ablation experiments on Set14 downsampled with Matlab bicubic function are summarized in Table 1.

Batch Normalization. We show the validation curves of SRDD-64 with and without batch normalization layers in Fig. 7. The performance of the proposed model is substantially improved by using batch normalization. This result is in contrast to conventional deep-learning-based SR methods, where batch normalization generally degrades performance [27]. Unlike conventional methods where the network directly outputs the SR image, the prediction network in SRDD

Table 1. Results of ablation experiments on Set14 downsampled with Matlab bicubic function.

	PSNR	SSIM
SRDD-64	28.54	0.7809
SRDD-64 − batch norm	28.49	0.7792
SRDD-64 − bottleneck blocks	28.48	0.7790
SRDD-64 − compensation	28.51	0.7801

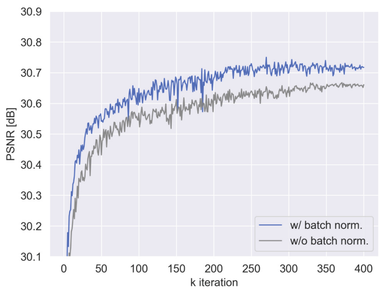

Fig. 7. Validation curves during the training of SRDD-64 with and without batch normalization layers.

predicts the coefficients of D_{H} for each pixel, which is rather similar to the semantic segmentation task. In this sense, it is natural that batch normalization, which is effective for semantic segmentation [7,24,58], is also effective for the proposed model.

Bottleneck Blocks. We eliminate bottleneck blocks and D_{H} code injection from the per-pixel predictor. The prediction network becomes close to the plane UNet++ structure with this modification. The performance drops as shown in Table 1, but still demonstrates a certain level of performance.

Compensation Mechanism. As shown in Table 1, removing the compensation mechanism from SRDD-64 degrades the performance. However, the effect is marginal indicates that our model can produce adequate quality outputs without boundary compensation. This result is in contrast to the sparse-coding-based methods, which generally require aggregation with overlapping patch sampling to reduce imperfection at the patch boundary. Because the computational complexity of our compensation mechanism is very small compared to that of the entire model, we adopt it even if the effect is not so large.

4.4 Results on In-Domain Test Images

We conduct experiments on five benchmark datasets, where the LR input images are created by Matlab bicubic downsampling same as in the DIV2K training dataset. Because SRDD is quite shallow and fast compared to current state-of-the-art models, we compare SRDD to relatively shallow and fast models with

Table 2. Quantitative comparison for ×4 SR on benchmark datasets. Best and second best results are highlighted in red and blue, respectively.

Method	Set5	Set14	BSD100	Urban100	Manga109
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
Bicubic	28.42/0.8104	26.00/0.7027	25.96/0.6675	23.14/0.6577	24.89/0.7866
A+ [45]	30.28/0.8603	27.32/0.7491	26.82/0.7087	24.32/0.7183	- / -
SRCNN [9]	30.48/0.8628	27.50/0.7513	26.90/0.7101	24.52/0.7221	27.58/0.8555
FSRCNN [10]	30.72/0.8660	27.61/0.7550	26.98/0.7150	24.62/0.7280	27.90/0.8610
VDSR [19]	31.35/0.8830	28.02/0.7680	27.29/0.0726	25.18/0.7540	28.83/0.8870
DRCN [20]	31.53/0.8854	28.02/0.7670	27.23/0.7233	25.14/0.7510	- / -
LapSRN [23]	31.54/0.8850	28.19/0.7720	27.32/0.7270	25.21/0.7560	29.09/0.8900
DRRN [41]	31.68/0.8888	28.21/0.7720	27.38/0.7284	25.44/0.7638	- / -
MemNet [42]	31.74/0.8893	28.26/0.7723	27.40/0.7281	25.50/0.7630	29.42/0.8942
CARN [1]	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837	30.47/0.9084
IMDN [17]	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838	30.45/ 0.9075
LatticeNet [28]	32.30/0.8962	28.68/0.7830	27.62/0.7367	26.25/0.7873	- / -
SRDD-64	32.05/0.8936	28.54/0.7809	27.54/0.7353	25.89/0.7812	30.16/0.9043
SRDD-128	32.25/0.8958	28.65/0.7838	27.61/0.7378	26.10/0.7877	30.44/0.9084
RCAN [56]	(32.63/0.9002)	(28.87/0.7889)	(27.77/0.7436)	(26.82/0.8087)	(31.22/0.9173)
NLSA [31]	(32.59/0.9000)	(28.87/0.7891)	(27.78/0.7444)	(26.96/0.8109)	(31.27/0.9184)
SwinIR [26]	(32.72/0.9021)	(28.94/0.7914)	(27.83/0.7459)	(27.07/0.8164)	(31.67/0.9226)

Table 3. Execution time of representative models on an Nvidia P4000 GPU for ×4 SR with input size 256 × 256.

	Running time [s]
SRCNN [9]	0.0669
FSRCNN [10]	0.0036
VDSR [19]	0.2636
LapSRN [23]	0.1853
CARN [1]	0.0723
IMDN [17]	0.0351
SRDD-64	0.0842
SRDD-128	0.2196
RCAN [56]	1.5653
NLSA [31]	1.7139
SwinIR [26]	2.1106

roughly 50 layers or less. Note that recent deep SR models usually have hundreds of convolutional layers [56]. We select ten models for comparison: SRCNN [9], FSRCNN [10], VDSR [19], DRCN [20], LapSRN [23], DRRN [41], MemNet [42], CARN [1], IMDN [17], and LatticeNet [28]. We also compare our model with a representative sparse coding-based method A+ [45]. Results for the representative very deep models RCAN [56], NLSA [31], and SwinIR [26] are also shown.

The quantitative results for ×4 SR on benchmark datasets are shown in Table 2. SRDD-64 and SRDD-128 show comparable performances to CARN/IMDN and LatticeNet, respectively. As shown in Table 3, the inference speed of SRDD-64 is also comparable to that of CARN, but slower than IMDN.

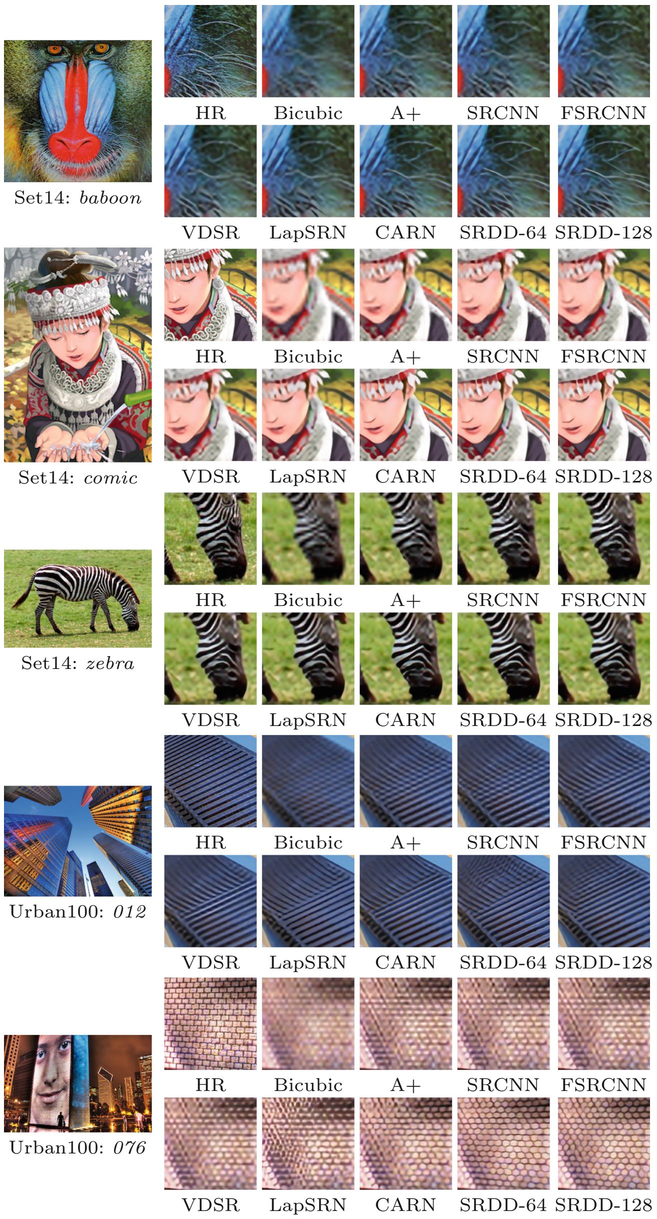

Fig. 8. Visual comparison for ×4 SR on Set14 and Urban100 dataset. Zoom in for a better view.

These results indicate that the overall performance of our method on in-domain test images is close to that of conventional baselines (not as good as state-of-the-art models). The running time of representative deep models are also shown

Table 4. Quantitative results of ×4 SR on Set14 downsampled with three different OpenCV resize functions. Note that the models are trained with Matlab bicubic downsampling.

	Bicubic	Bilinear	Area
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
CARN [1]	21.17/0.6310	22.76/0.6805	26.74/0.7604
IMDN [17]	20.99/0.6239	22.54/0.6741	26.60/0.7589
IKC [14]	20.10/0.6031	21.71/0.6558	26.40/0.7554
SRDD-64	21.52/0.6384	23.13/0.6871	27.05/0.7630

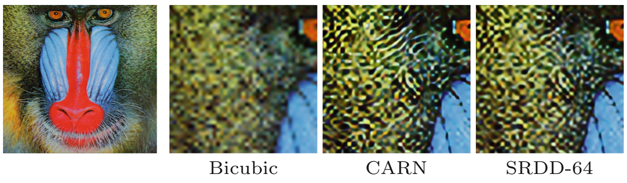

Fig. 9. Visual comparison for ×4 SR on Set14 *baboon* downsampled with OpenCV bicubic function.

for comparison. They are about 20 times slower than CARN and SRDD-64. The visual results are provided in Fig. 8.

4.5 Results on Out-of-Domain Test Images

Synthetic Test Images. We conduct experiments on Set14, where the LR input images are created differently from training time. We use bicubic, bilinear, and area downsampling with OpenCV resize functions. The difference between Matlab and OpenCV resize functions mainly comes from the anti-aliasing option. The anti-aliasing is default enabled/unenabled in Matlab/OpenCV, respectively. We mainly evaluate CARN and SRDD-64 because these models have comparable performance on in-domain test images. The state-of-the-art lightweight model IMDN [17] and the representative blind SR model IKC [14] are also evaluated for comparison. The results are shown in Table 4. SRDD-64 outperformed these models by a large margin for the three different resize functions. This result implies that our method is more robust for the out-of-domain images than conventional deep-learning-based methods. The visual comparison on a test image downsampled with OpenCV bicubic function is shown in Fig. 9. CARN overly emphasizes high-frequency components of the image, while SRDD-64 outputs a more natural result.

Real-World Test Images. We conduct experiments on widely used ten historical images to see the robustness of the models on unknown degradations. Because there is no ground-truth image, we adopt a no-reference image quality metric NIQE for evaluation. Table 5 shows average NIQE for representative methods.

Table 5. Results of no-reference image quality metric NIQE on real-world historical images. Note that the models are trained with Matlab bicubic downsampling.

	NIQE (lower is better)
Bicubic	7.342
A+ [45]	6.503
SRCNN [9]	6.267
FSRCNN [10]	6.130
VDSR [19]	6.038
LapSRN [23]	6.234
CARN [1]	5.921
EDSR [27]	5.852
SRDD-64	5.877
SRDD-128	5.896

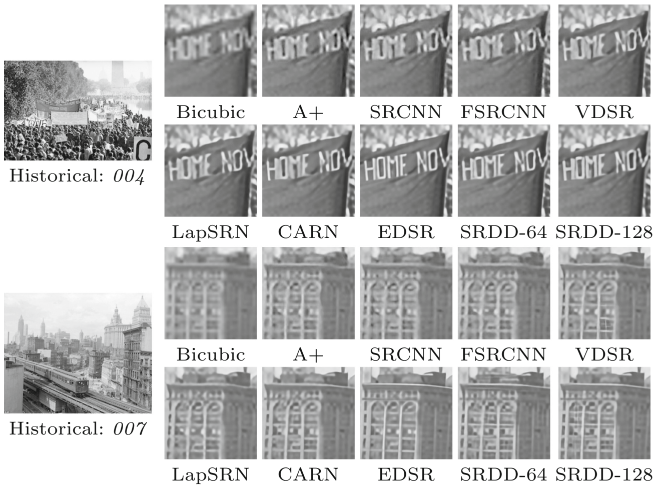

Fig. 10. Visual comparison for $\times 4$ SR on real-world historical images. Zoom in for a better view.

As seen in the previous subsection, our SRDD-64 shows comparable performance to CARN if compared with in-domain test images. However, on the realistic datasets with the NIQE metric, SRDD-64 clearly outperforms CARN and is close to EDSR. Interestingly, unlike the results on the in-domain test images, the performance of SRDD-64 is better than that of SRDD-128 for realistic degradations. This is probably because representing an HR image with a small number of atoms makes the atoms more versatile. The visual results are provided in Fig. 10.

4.6 Experiments of $\times 8$ SR

To see if our method would work at different scaling factors, we also experiment with the $\times 8$ SR case. We use DIV2K dataset for training and validation. The test

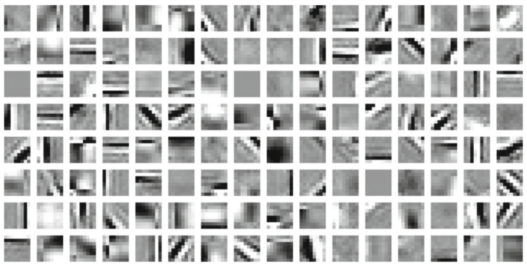

Fig. 11. Learned atoms of ×8 SRDD with $N = 128$. The size of each atom is $1 \times 8 \times 8$. The data range is renormalized to $[0, 1]$ for visualization.

Table 6. Quantitative comparison for ×8 SR on benchmark datasets. Best and second best results are highlighted in red and blue, respectively.

Method	Set5	Set14	BSD100	Urban100	Manga109
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
Bicubic	24.40/0.6580	23.10/0.5660	23.67/0.5480	20.74/0.5160	21.47/0.6500
SRCNN [9]	25.33/0.6900	23.76/0.5910	24.13/0.5660	21.29/0.5440	22.46/0.6950
FSRCNN [10]	20.13/0.5520	19.75/0.4820	24.21/0.5680	21.32/0.5380	22.39/0.6730
VDSR [19]	25.93/0.7240	24.26/0.6140	24.49/0.5830	21.70/0.5710	23.16/0.7250
LapSRN [23]	26.15/0.7380	24.35/0.6200	24.54/0.5860	21.81/0.5810	23.39/0.7350
MemNet [42]	26.16/0.7414	24.38/0.6199	24.58/0.5842	21.89/0.5825	23.56/0.7387
EDSR [27]	26.96/0.7762	24.91/0.6420	24.81/0.5985	22.51/0.6221	24.69/0.7841
SRDD-64	26.66/0.7652	24.75/0.6345	24.71/0.5926	22.20/0.6034	24.14/0.7621
SRDD-128	26.76/0.7677	24.79/0.6369	24.75/0.5947	22.25/0.6073	24.25/0.7672

images are prepared with the same downsampling function (i.e. Matlab bicubic function) as the training dataset. Figure 11 shows generated atoms of SRDD with $s = 8$ and $N = 128$. The structure of atoms with $s = 8$ is finer than that with $s = 4$, while the coarse structures of both cases are similar. The quantitative results on five benchmark datasets are shown in Table 6. SRDD performs better than the representative shallow models though its performance does not reach representative deep model EDSR.

5 Conclusions

We propose an end-to-end super-resolution network with a deep dictionary (SRDD). An explicitly learned high-resolution dictionary (D_{H}) is used to upscale the input image as in the sparse-coding-based methods, while the entire network, including the D_{H} generator, is simultaneously optimized to take full advantage of deep learning. For in-domain test images (images created by the same procedure as the training dataset), the proposed SRDD shows performance that is not as good as latest ones, but close to the conventional baselines (e.g., CARN). For out-of-domain test images, SRDD outperforms conventional deep-learning-based methods, demonstrating the robustness of our model.

The proposed method is not limited to super-resolution tasks but is potentially applicable to other tasks that require high-resolution output, such as high-resolution image generation. Hence, the proposed method is expected to have a broad impact on various tasks. Future works will be focused on the application of the proposed method to other vision tasks. In addition, we believe that our method still has much room for improvement compared to the conventional deep-learning-based approach.

Acknowledgements. I thank Uday Bondi for helpful comments on the manuscript.

References

1. Ahn, N., Kang, B., Sohn, K.-A.: Fast, accurate, and lightweight super-resolution with cascading residual network. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11214, pp. 256–272. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01249-6_16
2. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: BMVC (2012)
3. Chan, K.C., Wang, X., Xu, X., Gu, J., Loy, C.C.: Glean: generative latent bank for large-factor image super-resolution. In: CVPR (2021)
4. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: CVPR (2004)
5. Chen, C., et al.: Real-world blind super-resolution via feature matching with implicit high-resolution priors. arXiv preprint arXiv:2202.13142 (2022)
6. Chen, H., et al.: Pre-trained image processing transformer. In: CVPR (2021)
7. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
8. Dai, T., Cai, J., Zhang, Y., Xia, S.T., Zhang, L.: Second-order attention network for single image super-resolution. In: CVPR (2019)
9. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8692, pp. 184–199. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10593-2_13
10. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 391–407. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_25
11. Freedman, G., Fattal, R.: Image and video upscaling from local self-examples. TOG **30**(2), 1–11 (2011)
12. Freeman, W.T., Jones, T.R., Pasztor, E.C.: Example-based super-resolution. CG&A **22**(2), 56–65 (2002)
13. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: ICCV (2009)
14. Gu, J., Lu, H., Zuo, W., Dong, C.: Blind super-resolution with iterative kernel correction. In: CVPR (2019)
15. Gu, S., Zuo, W., Xie, Q., Meng, D., Feng, X., Zhang, L.: Convolutional sparse coding for image super-resolution. In: ICCV (2015)

16. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: CVPR (2015)
17. Hui, Z., Gao, X., Yang, Y., Wang, X.: Lightweight image super-resolution with information multi-distillation network. In: ACM Multimedia (2019)
18. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
19. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. In: CVPR (2016)
20. Kim, J., Lee, J.K., Lee, K.M.: Deeply-recursive convolutional network for image super-resolution. In: CVPR (2016)
21. Kim, K.I., Kwon, Y.: Single-image super-resolution using sparse regression and natural image prior. TPAMI **32**(6), 1127–1133 (2010)
22. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2014)
23. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep Laplacian pyramid networks for fast and accurate super-resolution. In: CVPR (2017)
24. Li, H., Xiong, P., An, J., Wang, L.: Pyramid attention network for semantic segmentation. In: BMVC (2018)
25. Li, Y., et al.: NTIRE 2022 challenge on efficient super-resolution: methods and results. In: CVPRW (2022)
26. Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., Timofte, R.: SwinIR: image restoration using Swin transformer. In: ICCV (2021)
27. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: CVPRW (2017)
28. Luo, X., Xie, Y., Zhang, Y., Qu, Y., Li, C., Fu, Y.: LatticeNet: towards lightweight image super-resolution with Lattice Block. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12367, pp. 272–289. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58542-6_17
29. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV (2001)
30. Matsui, Y., et al.: Sketch-based manga retrieval using manga109 dataset. Multimed. Tools App. **76**(20), 21811–21838 (2017)
31. Mei, Y., Fan, Y., Zhou, Y.: Image super-resolution with non-local sparse attention. In: CVPR (2021)
32. Mittal, A., Soundararajan, R., Bovik, A.C.: Making a "completely blind" image quality analyzer. Signal Process. Lett. **20**(3), 209–212 (2012)
33. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML (2010)
34. Niu, B., et al.: Single image super-resolution via a holistic attention network. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12357, pp. 191–207. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58610-2_12
35. Osendorfer, C., Soyer, H., Smagt, P.: Image super-resolution with fast approximate convolutional sparse coding. In: ICONIP (2014)
36. Peters, B., Niculae, V., Martins, A.F.: Sparse sequence-to-sequence models. In: ACL (2019)
37. Shi, W., et al.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: CVPR (2016)
38. Shocher, A., Cohen, N., Irani, M.: "zero-shot" super-resolution using deep internal learning. In: CVPR (2018)

39. Simon, D., Elad, M.: Rethinking the CSC model for natural images. In: NeurIPS (2019)
40. Soh, J.W., Cho, S., Cho, N.I.: Meta-transfer learning for zero-shot super-resolution. In: CVPR (2020)
41. Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: CVPR (2017)
42. Tai, Y., Yang, J., Liu, X., Xu, C.: MemNet: a persistent memory network for image restoration. In: ICCV (2017)
43. Timofte, R., Agustsson, E., Van Gool, L., Yang, M.H., Zhang, L.: NTIRE 2017 challenge on single image super-resolution: methods and results. In: CVPRW (2017)
44. Timofte, R., De Smet, V., Van Gool, L.: Anchored neighborhood regression for fast example-based super-resolution. In: ICCV (2013)
45. Timofte, R., De Smet, V., Van Gool, L.: A+: adjusted anchored neighborhood regression for fast super-resolution. In: ACCV (2014)
46. Wang, L., et al.: Unsupervised degradation representation learning for blind super-resolution. In: CVPR (2021)
47. Wang, Z., Liu, D., Yang, J., Han, W., Huang, T.: Deep networks for image super-resolution with sparse prior. In: ICCV (2015)
48. 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)
49. Xu, Y.S., Tseng, S.Y.R., Tseng, Y., Kuo, H.K., Tsai, Y.M.: Unified dynamic convolutional network for super-resolution with variational degradations. In: CVPR (2020)
50. Yang, J., Lin, Z., Cohen, S.: Fast image super-resolution based on in-place example regression. In: CVPR (2013)
51. Yang, J., Wang, Z., Lin, Z., Cohen, S., Huang, T.: Coupled dictionary training for image super-resolution. TIP **21**(8), 3467–3478 (2012)
52. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. TIP **19**(11), 2861–2873 (2010)
53. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Curves and Surfaces (2010)
54. Zhang, K., et al.: Aim 2020 challenge on efficient super-resolution: methods and results. In: ECCVW (2020)
55. Zhang, K., Zuo, W., Zhang, L.: Learning a single convolutional super-resolution network for multiple degradations. In: CVPR (2018)
56. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 294–310. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_18
57. Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y.: Residual dense network for image super-resolution. In: CVPR (2018)
58. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
59. Zhou, R., Susstrunk, S.: Kernel modeling super-resolution on real low-resolution images. In: ICCV (2019)
60. Zhou, S., Chan, K.C., Li, C., Loy, C.C.: Towards robust blind face restoration with codebook lookup transformer. arXiv preprint arXiv:2206.11253 (2022)
61. Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: UNet++: a nested u-net architecture for medical image segmentation. In: Stoyanov, D., et al. (eds.) DLMIA/ML-CDS -2018. LNCS, vol. 11045, pp. 3–11. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00889-5_1