Fast and Accurate Single Image Super-Resolution via Information Distillation Network

Zheng Hui, Xiumei Wang[*], Xinbo Gao
School of Electronic Engineering, Xidian University
Xi'an, China
zheng_hui@aliyun.com, wangxm@xidian.edu.cn, xbgao@mail.xidian.edu.cn

Abstract

Recently, deep convolutional neural networks (CNNs) have been demonstrated remarkable progress on single image super-resolution. However, as the depth and width of the networks increase, CNN-based super-resolution methods have been faced with the challenges of computational complexity and memory consumption in practice. In order to solve the above questions, we propose a deep but compact convolutional network to directly reconstruct the high resolution image from the original low resolution image. In general, the proposed model consists of three parts, which are feature extraction block, stacked information distillation blocks and reconstruction block respectively. By combining an enhancement unit with a compression unit into a distillation block, the local long and short-path features can be effectively extracted. Specifically, the proposed enhancement unit mixes together two different types of features and the compression unit distills more useful information for the sequential blocks. In addition, the proposed network has the advantage of fast execution due to the comparatively few numbers of filters per layer and the use of group convolution. Experimental results demonstrate that the proposed method is superior to the state-of-the-art methods, especially in terms of time performance. Code is available at https://github.com/Zheng222/IDN-Caffe.

1. Introduction

Single image super-resolution (SISR) is a classical problem in low-level computer vision, which reconstructs a high-resolution (HR) image from a low-resolution (LR) image. Actually, an infinite number of HR images can get the same LR image by downsampling. Hence, the SR problem is inherently ill-posed and no unique solution exists. In order to mitigate this problem, numerous SISR methods have been proposed in the literature, including interpolation-based methods, reconstruction-based methods and example-based methods. Since the former two kinds of methods usually suffer dramatically drop in restoration performance with larger upscaling factors, the recent SR methods fall into the example-based methods which try to learn prior knowledge from LR and HR pairs.

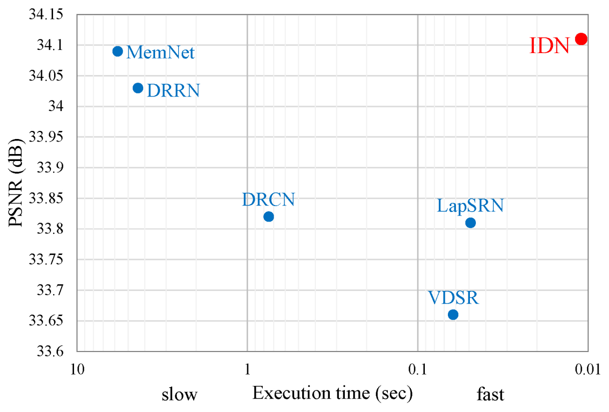


Figure 1. Speed and accuracy trade-off. The average PSNR and the average inference time for upscaling 3× on Set5. The IDN is faster than other methods and achieves the best performance at the same time.

Recently, due to the strength of deep convolutional neural network (CNN), many CNN-based SR methods try to train a deep network to gain better reconstruction performance. Kim *et al*. propose a 20-layer CNN model known as VDSR [12], which adopts residual learning and adaptive gradient clipping to ease training difficulty. To control the model parameters, the authors construct a deeply-recursive convolutional network (DRCN) [13] by adopting recursive layer. To mitigate training difficulty, Mao *et al*. propose a very deep residual encoder-decoder network (RED) [17], which consists of a series of convolutional and subsequent transposed convolution layers to learn end-to-end mappings from the LR images to the ground truths. Tai *et al*. propose

*corresponding author

a deep recursive residual network (DRRN) [22], which employs parameters sharing strategy to alleviate the requirement of enormous parameters of the very deep networks.

Although achieving prominent performance, most of deep networks still have some drawbacks. Firstly, in order to achieve better performance, deepening or widening the network has been a design trend. But the result is that these methods demand large computational cost and memory consumption, which are less applicable in practice, such as mobile and embedded vision applications. Moreover, the traditional convolutional networks usually adopt cascaded network topologies, *e.g*., VDSR [12] and DRCN [13]. In this way, the feature maps of each layer are sent to the sequential layer without distinction. However, Hu *et al*. [9] experimentally demonstrate that adaptively recalibrating channel-wise features responses can improve the representational power of a network.

To address these drawbacks, we propose a novel information distillation network (IDN) with lightweight parameters and computational complexity as illustrated in Figure 2. In the proposed IDN, a feature extraction block (FBlock) first extracts features from the LR image. Then, multiple information distillation blocks (DBlocks) are stacked to progressively distill residual information. Finally, a reconstruction Block (RBlock) aggregates the obtained HR residual representations to generate the residual image. To get a HR image, we implement an element-wise addition operation on the residual image and the upsampled LR image.

The key component of IDN is the information distillation block, which contains an enhancement unit and a compression unit. The enhancement unit mainly comprises two shallow convolutional networks as illustrated in Figure 3. Each of them is a three-layer shallow module. The feature maps of the first module are extracted through a short path (3-layer). Thus, they can be regarded as the local short-path features. Considering that the deep networks have more expressive power, we send a portion of the local short-path features to another module. By this way, the feature maps of the second module naturally become the local long-path features. Different from the approach in [9], we divide feature maps into two parts. One part represents reserved short-path features and another expresses the short-path features that will be enhanced. After getting long and short-path feature maps, we aggregate these two types of features for gaining more abundant and efficient information. In summary, the enhancement unit is mainly to improve the representation power of the network. As for the compression unit, we adopt a simple convolutional layer to compress the redundancy information in features of the enhancement unit.

The main contributions of this work are summarized as follows:

- The proposed IDN extracts feature maps directly from LR images and employs multiple cascaded DBlocks to generate the residual representations in HR space. In each DBlock, the enhancement unit gathers more information as much as possible and the compression unit distills more useful information. As a result, IDN achieves competitive results in spite of using less number of convolutional layer.
- Due to the concise structure of the proposed IDN, it is much faster than several CNN-based SR methods, *e.g*., VDSR [12], DRCN [13], LapSRN [15], DRRN [22] and MemNet [23] as illustrated in Figure 1. Only the proposed method achieves real-time speed and maintains better reconstruction accuracy.

2. Related Work

Single image super-resolution has been extensively studied in these years. In this section, we will focus on recent example-based and neural network based approaches.

2.1. Self-example based methods

The self-example based methods exploit the self-similarity property and extract example pairs merely from the LR image across different scales. This type of methods usually works well in the images containing repetitive patterns and textures but lacks the richness of image structures outside the input image and thus fails to generate satisfactory prediction for images of other classes. Huang *et al*. [10] extend self-similarity based SR to handle the affine and perspective deformation.

2.2. External-example based methods

The external-example based methods learn a mapping between LR and HR patches from external datasets. This type of approaches usually focuses on how to learn a compact dictionary or manifold space to relate LR/HR patches, such as nearest neighbor [7], manifold embedding [2], random forest [20] and sparse representation [25, 26]. While these approaches are effective, the extracted features and mapping functions are not adaptive, which may not be optimal for generating high-quality SR images.

2.3. Convolutional neural networks based methods

Recently, inspired by the achievement of many computer vision tasks tackled with deep learning, neural networks have been achieved dramatic improvement in SR. Dong *et al*. [3, 4] first exploit a three-layer convolutional neural network, named SRCNN, to jointly optimize the feature extraction, non-linear mapping and image reconstruction stages in an end-to-end manner. Afterwards Shi *et al*. [21] propose an efficient sub-pixel convolutional neural network (ESPCN), which extracts feature maps in the LR space and

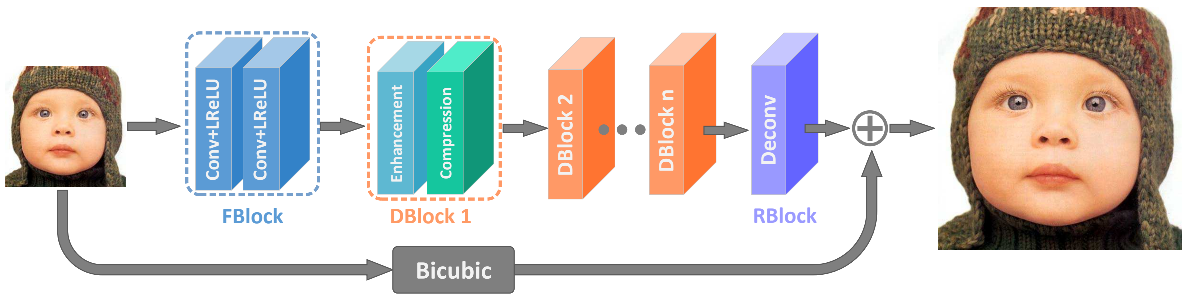


Figure 2. Architecture of the proposed network.

replaces the bicubic upsampling operation with an efficient sub-pixel convolution. Dong *et al.* [5] adopt deconvolution to accelerate SRCNN in combination with smaller filter sizes and more convolution layers. Kim *et al.* [12] propose a very deep CNN model with global residual architecture to achieve superior performance, which utilizes contextual information over large image regions. Another network designed by Kim *et al.* [13], which has recursive convolution with skip connection to avoid introducing additional parameters when the depth is increasing. Mao *et al.* [17] tackle the general image restoration problem with encoder-decoder networks and symmetric skip connections. Lai *et al.* [15] propose the laplacian pyramid super-resolution network (LapSRN) to address the speed and accuracy of SR problem, which takes the original LR images as input and progressively reconstructs the sub-band residuals of HR images. Tai *et al.* [22] propose the deep recursive residual network to effectively build a very deep network structure for SR, which weighs the model parameters against the accuracy. The authors also present a very deep end-to-end persistent memory network (MemNet) [23] for image restoration task, which tackles the long-term dependency problem in the previous CNN architectures. Sajjadi *et al.* [19] propose a novel combination of automated texture synthesis with a perceptual loss focusing on creating realistic textures at a high magnification ratio of 4.

3. Proposed Method

In this section, we first describe the proposed model architecture and then suggest the enhancement unit and the compression unit, which are the core of the proposed method.

3.1. Network structure

The proposed IDN, as shown in Figure 2, consists of three parts: a feature extraction block (FBlock), multiple stacked information distillation blocks (DBlocks) and a reconstruction block (RBlock). Here, we denote x and y as the input and the output of IDN. With respect to FBlock, two 3×3 convolutional layers are utilized to extract the feature maps from the original LR image. This procedure can be expressed as

$$B_0 = f\left(x\right), \tag{1}$$

where f represents the feature extraction function and B_0 denotes the extracted features and servers as the input to the following stage. The next part is composed of multiple information distillation blocks by using chained mode. Each block contains an enhancement unit and a compression unit with stacked style. This process can be formulated as

$$B_k = F_k\left(B_{k-1}\right), k = 1, \cdots, n, \tag{2}$$

where F_k denotes the k-th DBlock function, B_{k-1} and B_k indicate the input and output of the k-th DBlock respectively. Finally, we take a transposed convolution without activation function as the RBlock. Hence, the IDN can be expressed as

$$y = R\left(F_n\left(B_{n-1}\right)\right) + U\left(x\right), \tag{3}$$

where R, U denote the RBlock and bicubic interpolation operation respectively.

3.1.1 Loss function

We consider two loss functions that measure the difference between the predicted HR image $\hat{I}$ and the corresponding ground-truth I. The first one is mean square error (MSE), which is the most widely used loss function for general image restoration as defined below:

$$l_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left\|I_i - \hat{I}_i\right\|_2^2. \tag{4}$$

However, Lim *et al.* [16] experimentally demonstrate that training with MSE loss is not a good choice. The second loss function is mean absolute error (MAE), which is formulated as follows:

$$l_{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left\|I_i - \hat{I}_i\right\|_1. \tag{5}$$

We empirically found that our model with MSE loss can improve performance of a trained network with MAE loss. Therefore, we first train the network with MAE loss and then fine-tune it by MSE loss.

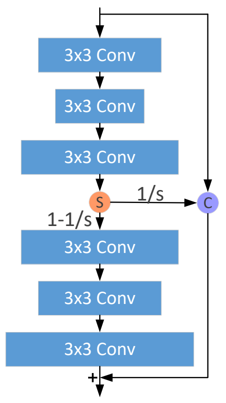


Figure 3. The architecture of enhancement unit in the proposed model. Orange circle represents slice operation and purple circle indicates concatenation operation in channel dimension.

3.2. Enhancement unit

As shown in Figure 3, enhancement unit can be roughly divided into two modules, one is the above three convolutions and another is the below three convolutions. The above module has three 3×3 convolutions, each of them is followed by a leaky rectified linear unit (LReLU) activation function, which is omitted here. Let's denote the feature map dimensions of the i-th layer as $D_i\ (i = 1, \cdots, 6)$. In that way, the relationship of the convolutional layers can be expressed as

$$D_3 - D_1 = D_1 - D_2 = d, \tag{6}$$

where d denotes the difference between the first layer and the second layer or between the first layer and the third layer. Similarly, the dimension of channels in the below module also has this relation and can be described as follows:

$$D_6 - D_4 = D_4 - D_5 = d, \tag{7}$$

where $D_4 = D_3$. The above module is composed of three cascaded convolution layers with LReLUs, and the output of the third convolution layer is sliced into two segments. Supposing the input of this module is B_{k-1}, we have

$$P_1^k = C_a\left(B_{k-1}\right), \tag{8}$$

where B_{k-1} denotes the output of previous block and meanwhile is the input of present block, C_a indicates chained convolutions operation and P_1^k is the output of the above module in the k-th enhancement unit. The feature maps with $\frac{D_3}{s}$ dimensions of P_1^k and the input of the first convolutional layer are concatenated in the channel dimension,

$$R^k = C\left(S\left(P_1^k, 1/s\right), B_{k-1}\right), \tag{9}$$

where C, S represent concatenation operation and slice operation respectively. Specifically, we know the dimension of P_1^k is D_3. Therefore, $S\left(P_1^k, 1/s\right)$ denotes that $\frac{D_3}{s}$ dimensions features are fetched from P_1^k. Moreover, $S\left(P_1^k, 1/s\right)$ concatenates features with B_{k-1} in channel dimension. The purpose is to combine the previous information with some current information. It can be regarded as partially retained local short-path information. We take the rest of local short-path information as the input of the below module, which mainly further extracts long-path feature maps,

$$P_2^k = C_b\left(S\left(P_1^k, 1 - 1/s\right)\right), \tag{10}$$

where P_2^k, C_b are the output and stacked convolution operations of the below module respectively. Finally, as shown in Figure 3, the input information, the reserved local short-path information and the local long-path information are aggregated. Therefore, the enhancement unit can be formulated as

$$\begin{aligned} P^k = P_2^k + R^k = C_b\left(S\left(C_a\left(B_{k-1}\right), 1 - 1/s\right)\right) \\ + C\left(S\left(C_a\left(B_{k-1}\right), 1/s\right), B_{k-1}\right), \end{aligned} \tag{11}$$

where P^k is the output of enhancement unit. At this point, local long-path features P_2^k and the combination of local short-path features and the untreated features R^k are utilized without exception by a compression unit.

3.3. Compression unit

We achieve compression mechanism by taking advantage of a 1×1 convolution layer. Concretely, the outputs of the enhancement unit are sent to a 1×1 convolution layer, which acts as dimensionality reduction or distilling relevant information for the later network. Thus, the compression unit can be formulated as

$$B_k = f_F^k\left(P^k\right) = \alpha_F^k\left(W_F^k\left(P^k\right)\right), \tag{12}$$

where f_F^k denotes the function of the 1×1 convolution layer (α_F^k denotes the activation function and W_F^k is the weight parameters).

4. Experiments

4.1. Datasets

4.1.1 Training datasets

By following [12, 15, 22, 23], we use 91 images from Yang *et al.* [26] and 200 images from Berkeley Segmentation Dataset (BSD) [18] as the training data. As in [22], to make full use of the training data, we apply data augmentation in three ways: (1) Rotate the images with the degree of 90°, 180° and 270°. (2) Flip images horizontally. (3) Downscale the images with the factor of 0.9, 0.8, 0.7 and 0.6.

4.1.2 Testing datasets

The proposed method is evaluated on four widely used benchmark datasets: Set5 [1], Set14 [27], BSD100 [18], Urban100 [10]. Among these datasets, Set5, Set14 and BSD100 consist of natural scenes and Urban100 contains challenging urban scenes images with details in different frequency bands. The ground truth images are downscaled by bicubic interpolation to generate LR/HR image pairs for both training and testing datasets. We convert each color image into the YCbCr color space and only process the Y-channel, while color components are simply enlarged using bicubic interpolation.

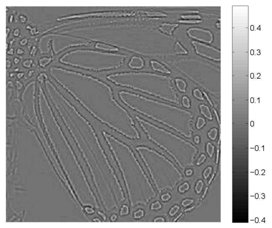
Scale	Training	Fine-tuning
2	$29^2/57^2$	$39^2/77^2$
3	$15^2/43^2$	$26^2/76^2$
4	$11^2/41^2$	$19^2/73^2$

Table 1. The sizes of training and fine-tuning sub-images for different scaling factors.

4.2. Implementation details

For preparing the training samples, we first downsample the original HR images with upscaling factor $m\,(m = 2, 3, 4)$ by using the bicubic interpolation to generate the corresponding LR images and then crop the LR training images into a set of $l_{sub} \times l_{sub}$ size sub-images. The corresponding HR training images are divided into $ml_{sub} \times ml_{sub}$ size sub-images. As the proposed model is trained using the *Caffe* package [11], its transposed convolution filters will generate the output with size $(ml_{sub} - m + 1)^2$ instead of $(ml_{sub})^2$. So we should crop $(m - 1)$-pixel borders on the HR sub-images. Since the minimum size picture “t20” in the 291 dataset is a 78 × 78 size image, the maximum size of the sub-image we can crop on the LR image is 26 × 26 for maintaining data integrity when scaling factor $m = 3$. However, the training process will be unstable due to the larger size training samples equipped with the larger learning rate by using *Caffe* package. Therefore, $15^2/43^2$ training pairs are generated for training stage and $26^2/76^2$ LR/HR sub-images pairs are utilized for fine-tuning phase. The learning rate is initially set to $1e - 4$ and decreases by the factor of 10 during fine-tuning phase. In this way, the sizes of training and fine-tuning samples are shown in Table 1.

Taking into account the trade-off between the execution time and the reconstruction performance, we construct a 31-layer network that denoted as IDN. This model has 4 DBlocks, and the parameters D_3, d and s of enhancement unit in each block are set to 64, 16 and 4 respectively. To reduce the parameters of network, we use the grouped convolution layer [6, 24] in the second and fourth layers in each enhancement unit with 4 groups. In addition, the transposed convolution adopts 17 × 17 filters for all scaling factors and the negative scope of LReLU is set as 0.05. We initialize the weights by using the method proposed in [8] and the biases are set to zero. The proposed network is optimized using Adam [14]. We set the parameters of mini-batch size and weight decay to 64 and $1e - 4$ respectively. In order to get better initialization parameters, we empirically pre-train the proposed model with 10^5 iterations and take these parameters as the initial values of the IDN. Training a IDN roughly takes a day with a TITAN X GPU on the 2× model.


(a) residual image

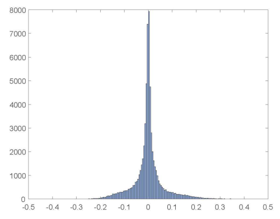
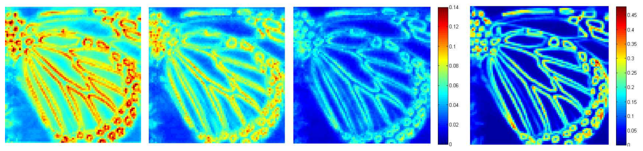

(b) data distribution histogram

Figure 4. The residual image and its data distribution of the “butterfly” image from Set5 dataset.


(a) the average feature maps of enhancement units

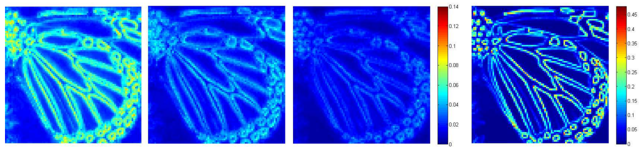

(b) the average feature maps of compression units

Figure 5. Visualization of the average feature maps.

4.3. Network analysis

The proposed model with a global residual structure mainly learns a residual image. As show in Figure 4(a), the ground truth residual image mainly contains details and texture information and its normalized pixel value ranges from -0.4 to 0.5. From Figure 4(b), we find that there are positive and negative values in the residual image, and the number of positive pixels is intuitively similar to that of negative ones. Obviously, the number of zero value and its neighbors is the most, which suggests that smooth region in residual image is almost eliminated. Therefore, the task of our network is

Dataset	Scale	Bicubic	VDSR [12]	DRCN [13]	LapSRN [15]	DRRN [22]	MemNet [23]	IDN (Ours)
Set5	×2	33.66/0.9299	37.53/0.9587	37.63/0.9588	37.52/0.9591	37.74/0.9591	37.78/0.9597	37.83/0.9600
	×3	30.39/0.8682	33.66/0.9213	33.82/0.9226	33.81/0.9220	34.03/0.9244	34.09/0.9248	34.11/0.9253
	×4	28.42/0.8104	31.35/0.8838	31.53/0.8854	31.54/0.8852	31.68/0.8888	31.74/0.8893	31.82/0.8903
Set14	×2	30.24/0.8688	33.03/0.9124	33.04/0.9118	32.99/0.9124	33.23/0.9136	33.28/0.9142	33.30/0.9148
	×3	27.55/0.7742	29.77/0.8314	29.76/0.8311	29.79/0.8325	29.96/0.8349	30.00/0.8350	29.99/0.8354
	×4	26.00/0.7027	28.01/0.7674	28.02/0.7670	28.09/0.7700	28.21/0.7721	28.26/0.7723	28.25/0.7730
BSD100	×2	29.56/0.8431	31.90/0.8960	31.85/0.8942	31.80/0.8952	32.05/0.8973	32.08/0.8978	32.08/0.8985
	×3	27.21/0.7385	28.82/0.7976	28.80/0.7963	28.82/0.7980	28.95/0.8004	28.96/0.8001	28.95/0.8013
	×4	25.96/0.6675	27.29/0.7251	27.23/0.7233	27.32/0.7275	27.38/0.7284	27.40/0.7281	27.41/0.7297
Urban100	×2	26.88/0.8403	30.76/0.9140	30.75/0.9133	30.41/0.9103	31.23/0.9188	31.31/0.9195	31.27/0.9196
	×3	24.46/0.7349	27.14/0.8279	27.15/0.8276	27.07/0.8275	27.53/0.8378	27.56/0.8376	27.42/0.8359
	×4	23.14/0.6577	25.18/0.7524	25.14/0.7510	25.21/0.7562	25.44/0.7638	25.50/0.7630	25.41/0.7632

Table 2. Average PSNR/SSIMs for scale 2×, 3× and 4×. Red color indicates the best and blue color indicates the second best performance.

Dataset	Scale	Bicubic	VDSR [12]	DRCN [13]	LapSRN [15]	DRRN [22]	MemNet [23]	IDN (Ours)
Set5	×2	6.083	8.580	8.783	9.010	8.670	8.850	9.252
	×3	3.580	5.203	5.336	5.194	5.394	5.503	5.620
	×4	2.329	3.542	3.543	3.559	3.700	3.787	3.826
Set14	×2	6.105	8.159	8.370	8.501	8.280	8.469	8.839
	×3	3.473	4.691	4.782	4.662	4.870	4.958	5.062
	×4	2.237	3.106	3.098	3.145	3.249	3.309	3.354
BSD100	×2	5.619	7.494	7.577	7.715	7.513	7.665	7.931
	×3	3.138	4.151	4.184	4.057	4.235	4.300	4.398
	×4	1.978	2.679	2.633	2.677	2.746	2.778	2.837
Urban100	×2	6.245	8.629	8.959	8.907	8.889	9.122	9.594
	×3	3.620	5.159	5.314	5.156	5.440	5.560	5.676
	×4	2.361	3.462	3.465	3.530	3.669	3.786	3.789

Table 3. Average IFCs for scale 2×, 3× and 4×. Red color indicates the best and blue color indicates the second best performance.

to gradually subtract the smooth area of the original input image. In order to verify our intuition, we need inspect the outputs of enhancement and compression units. For better visualizing the intermediary of the proposed model, we consider an operation T that can transform a $3D$ tensor A to a flattened $2D$ tensor defined over the spatial dimensions, which can be formulated as follows:

$$T : R^{c\times h\times w} \rightarrow R^{h\times w}. \tag{13}$$

Specifically, in this work, we will consider the mean of the feature maps in channel dimension, which can be described by

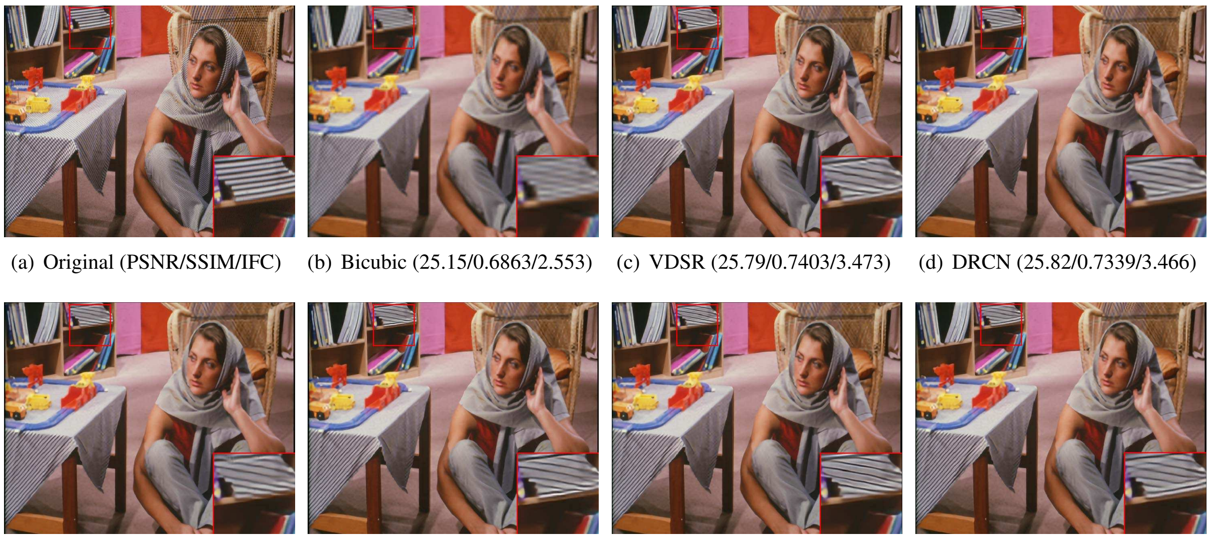
$$T_{mean}\left(A\right) = \frac{1}{c}\sum\nolimits_{i=1}^{c} A_i, \tag{14}$$

where $A_i = A\left(i, :, :\right)$ (using Matlab notation). The average feature map can roughly represent the situations of the whole feature maps. To explore the functions of enhancement unit and compression unit, we visualize the outputs of each enhancement unit and compression unit by utilizing above-mentioned method. As illustrated in Figure 5(a), from the first subpicture to the third subpicture, average feature maps gradually reduce the pixel values, especially in smooth areas. According to Figure 5(a), we can easily see that the first subfigure holds larger pixel values but has rough outline of the butterfly. The second and the third subfigures show that the later enhancement units continue decreasing the pixel values to obtain the features with a relatively clear contour profile. In addition, the last subfigure obviously surpasses the former figures, which brings the better inputs for the sequential compression unit that directly connects to RBlock. In summary, the function of the enhancement unit mainly enhances the outline areas of input LR image. As for the effect of compression unit, comparing Figure 5(a) with Figure 5(b), we find that the pixel values of features are mapped into a smaller range through the compression unit. From the second subfigure in Figure 5(b) and the third subfigure in Figure 5(a), we can see some regions of the average feature map of compression unit are enhanced by the following enhancement unit. This indicates that the process of the first three stacked blocks is to reduce the pixel value as a whole, while the last block greatly enhances the contrast between the contour and the smooth areas.

The RBlock, a transposed convolution layer, assembles the output of the final DBlock to generate the residual image. The bias term of this transposed convolution can auto-

Dataset	Scale	VDSR [12]	DRCN [13]	LapSRN [15]	DRRN [22]	MemNet [23]	IDN (Ours)
Set5	×2	0.054	0.735	0.032	4.343	5.715	0.016
	×3	0.062	0.748	0.049	4.380	5.761	0.011
	×4	0.054	0.735	0.040	4.450	5.728	0.009
Set14	×2	0.113	1.579	0.035	8.540	12.031	0.025
	×3	0.122	1.569	0.061	8.298	11.543	0.014
	×4	0.112	1.526	0.040	8.540	11.956	0.010
BSD100	×2	0.071	0.983	0.018	4.430	5.875	0.015
	×3	0.071	0.996	0.037	4.430	5.897	0.009
	×4	0.071	0.984	0.023	4.373	5.887	0.007
Urban100	×2	0.451	5.010	0.082	26.699	35.871	0.062
	×3	0.514	5.054	0.122	26.693	35.803	0.034
	×4	0.448	5.048	0.100	26.702	37.404	0.022

Table 4. Comparison the running time (sec) on the 4 benchmark datasets with scale factors 2×, 3× and 4×. Red color indicates the fastest algorithm and blue color indicates the second fastest method. Our IDN achieves the best time performance.



(a) Original (PSNR/SSIM/IFC) (b) Bicubic (25.15/0.6863/2.553) (c) VDSR (25.79/0.7403/3.473) (d) DRCN (25.82/0.7339/3.466)

(e) LapSRN (25.77/0.7430/3.546) (f) DRRN (25.74/0.7403/3.589) (g) MemNet (25.61/0.7399/3.639) (h) IDN (**25.84/0.7442/3.749**)

Figure 6. The "barbara" image from the Set14 dataset with an upscaling factor 4.

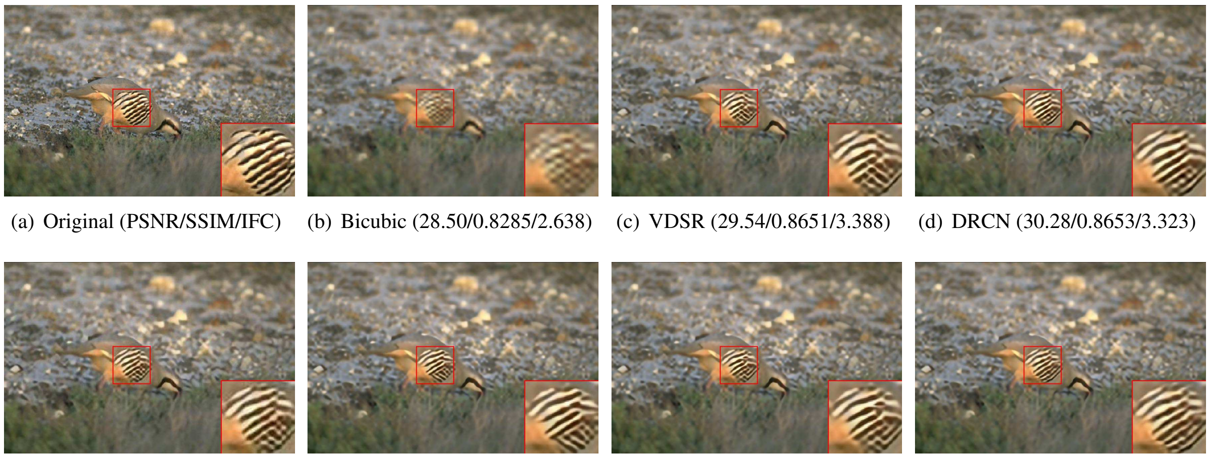
matically adjust the central value of the residual image data distribution to approach the ground-truth.

4.4. Comparisons with state-of-the-arts

We compare the proposed method with other SR methods, including bicubic, SRCNN [3, 4], VDSR [12], DRCN [13], LapSRN [15], DRRN [22] and MemNet [23]. Table 2 shows the average peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) values on four benchmark datasets. The proposed method performs favorably against state-of-the-art results on most datasets. In addition, we measure all methods with information fidelity criterion (IFC) metric, which assesses the image quality based on natural scene statistics and correlates well with human perception of image super-resolution. Table 3 shows the proposed method achieves the best performance and outperforms MemNet [23] by a considerable margin.

Figure 6, 7 and 8 show visual comparisions. The "barbara" image has serious artifacts in the read box due to the loss of high frequency information, which can be seen from the result of bicubic interpolation. Only the proposed method recovers roughly the outline of several stacked books as shown in Figure 6. From Figure 7, we can obviously see that the proposed method gains clearer contour without serious artifacts while other methods have different degrees of the fake information. In Figure 8, the building structure on image "img085" of Urban100 dataset is relatively clear in the proposed method.

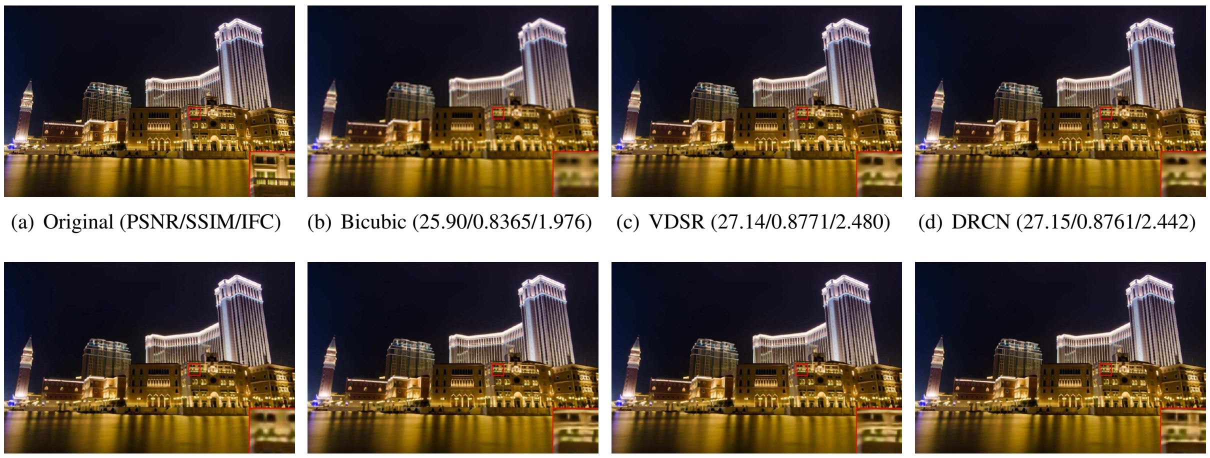
From Table 2, the performance of the proposed IDN is lower than that of MemNet in Urban100 dataset and 3×, 4× scale factors, while our IDN can achieve slightly better performance in other benchmark datasets. The main reason is that MemNet takes an interpolated LR image as its input so that more information is fed into the network and the



(a) Original (PSNR/SSIM/IFC) (b) Bicubic (28.50/0.8285/2.638) (c) VDSR (29.54/0.8651/3.388) (d) DRCN (30.28/0.8653/3.323)

(e) LapSRN (30.10/0.8710/3.432) (f) DRRN (29.74/0.8671/3.509) (g) MemNet (30.14/0.8697/3.563) (h) IDN (**30.40/0.8715/3.703**)

Figure 7. The "8023" image from the BSD100 dataset with an upscaling factor 4.



(a) Original (PSNR/SSIM/IFC) (b) Bicubic (25.90/0.8365/1.976) (c) VDSR (27.14/0.8771/2.480) (d) DRCN (27.15/0.8761/2.442)

(e) LapSRN (27.11/0.8809/2.593) (f) DRRN (26.65/0.8739/2.399) (g) MemNet (26.83/0.8750/2.444) (h) IDN (**27.26/0.8824/2.705**)

Figure 8. The "img085" image from the Urban100 dataset with an upscaling factor 4.

process of the SR only needs to correct the interpolated image. The algorithms that take the original LR image as input demand predicting more pixels from scratch, especially in larger images and larger magnification factors.

As for inference time, we use the public codes of the compared algorithms to evaluate the runtime on the machine with 4.2GHz Intel i7-7700K CPU (32G RAM) and Nvidia TITAN X (Pascal) GPU (12G memory). Since we note that official implementations of MemNet and DRRN have the condition of out of the GPU memory when testing the images on BSD100 and Urban100 datasets, we divide 100 images into several parts and evaluate on these parts and then collect them for these two datasets. Table 4 shows the average execution time on four benchmark datasets. It is noteworthy that the proposed IDN is approximately 500 times faster than MemNet [23] with 2× magnification on the Urban100 dataset.

5. Conclusions

In this paper, we propose a novel network that employs distillation blocks to gradually extract abundant and efficient features for the reconstruction of HR images. The proposed approach achieves competitive results on four benchmark datasets in terms of PSNR, SSIM and IFC. Meanwhile the inference time substantially exceeds the state-of-the-art methods such as DRRN [22] and MemNet [23]. This compact network will be more widely applicable in practice. In the future, this approach of image super-resolution will be explored to facilitate other image restoration problems such as denosing and compression artifacts reduction.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 61472304, 61432014 and U1605252.

References

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

[2] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *CVPR*, 2004. 2

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

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016. 2, 7

[5] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, pages 391–407, 2016. 3

[6] C. Francois. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1251–1258, 2017. 5

[7] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002. 2

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015. 5

[9] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *arXiv:1709.01507*, 2017. 2

[10] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015. 2, 5

[11] 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 *ACMMM*, pages 675–678, 2014. 5

[12] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016. 1, 2, 3, 4, 6, 7

[13] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, pages 1637–1645, 2016. 1, 2, 3, 6, 7

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 5

[15] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, pages 624–632, 2017. 2, 3, 4, 6, 7

[16] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshop*, pages 136–144, 2017. 3

[17] X.-J. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, 2016. 1, 3

[18] 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 *CVPR*, pages 416–423, 2001. 4, 5

[19] M. S. M. Sajjadi, B. Scholkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, pages 4491–4500, 2017. 3

[20] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *CVPR*, pages 3791–3799, 2015. 2

[21] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016. 2

[22] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, pages 3147–3155, 2017. 2, 3, 4, 6, 7, 8

[23] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, pages 3147–3155, 2017. 2, 3, 4, 6, 7, 8

[24] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 5

[25] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *CVPR*, 2008. 2

[26] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010. 2, 4

[27] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730, 2010. 5