UDNet: Up-Down Network for Compact and Efficient Feature Representation in Image Super-Resolution

Chang Chen Xinmei Tian Zhiwei Xiong Feng Wu
University of Science and Technology of China

Abstract

Recently, image super-resolution (SR) using convolutional neural networks (CNNs) have achieved remarkable performance. However, there is a tradeoff between performance and speed of SR, depending on whether feature representation and learning are conducted in high-resolution (HR) or low-resolution (LR) space. Generally, to pursue real-time SR, the number of parameters in CNNs has to be restricted, which results in performance degradation. In this paper, we propose a compact and efficient feature representation for real-time SR, named up-down network (UDNet). Specifically, a novel hourglass-shape structure is introduced by combining transposed convolution and spatial aggregation. This structure enables the network to transfer the feature representations between LR and HR spaces multiple times to learn a better mapping. Comprehensive experiments demonstrate that, compared with existing CNN models, UDNet achieves real-time SR without performance degradation on widely used benchmarks.

1. Introduction

Single image super-resolution (SISR) is a typical inverse problem in low-level image processing, aiming at reconstructing a high-resolution (HR) output from a low-resolution (LR) input. SISR methods can be largely divided into three classes: interpolation based [1, 24], reconstruction based [22, 23], and learning based [5, 6, 7, 13, 14]. Among them, the learning-based methods are widely investigated, which either exploit internal similarities within the same image for self-learning [4, 32] or learn the LR-to-HR mapping from external image patches [18, 27, 29, 30, 31, 33]. As two representatives, Huang et al. [10] utilized the transformed self-similarity to achieve good performance in images with considerable internal patch redundancy, and Dong et al. [5] first trained an end-to-end convolutional neural network (CNN) for external example-based SR with state-of-the-art performance.

Following the seminal work of SRCNN in [5], increasingly more CNN-based methods begin to search for better regression functions in HR space. By introducing the successful deep CNN designed for image recognition (e.g., VGG-Net [21]) into SISR, high reconstruction accuracy has been achieved [13, 14]. In early CNN-based SR models, the LR image is first upsampled to HR space using bicubic interpolation and then the SR operation is performed in HR space. Recently, to accelerate the speed of SR and to avoid the sub-optimal bicubic interpolation, models that reconstruct the HR image directly from the LR image (without bicubic interpolation) become popular. In these models, a single transposed [7] or sub-pixel [20] convolutional layer is adopted as the last layer to transform the LR feature representations into the HR output. Alternatively, Romano et. al [19] decomposed the input image for individual processing and aggregated them into an output image with learned parameters. It notably reduced the computational complexity and can be used on mobile devices.

However, it is still a tradeoff that above models accelerate the speed at the cost of performance degradation. To achieve the real-time speed demanded by practical applications, the number of network parameters has to be reduced to the greatest extent possible. Such restrictions on parameters limit feature representation and learning. Thus, enhancing the efficiency of restricted parameters is important for practical usages. In this paper, we propose a compact and efficient feature representation for real-time SR, named up-down network (UDNet). In our UDNet, a novel hourglass-shape structure is introduced by combining transposed convolution and spatial aggregation. This structure enables the network to transfer the representations between LR and HR spaces multiple times to learn a better mapping. In comparison, existing fast CNN models have at most one transposed or sub-pixel convolution to directly transform LR feature representations to the HR output.

Specifically, we adopt a down-sampling convolutional layer to perform spatial aggregation subsequent to the transposed convolution for upsampling. This structure is named as Up-Down Network (UDNet). By controlling the number of filters in each layer, UDNet realizes a compact and efficient feature representation without additional parameters. Moreover, we investigate a few variants of UDNet for further improved reconstruction accuracy, by expanding it

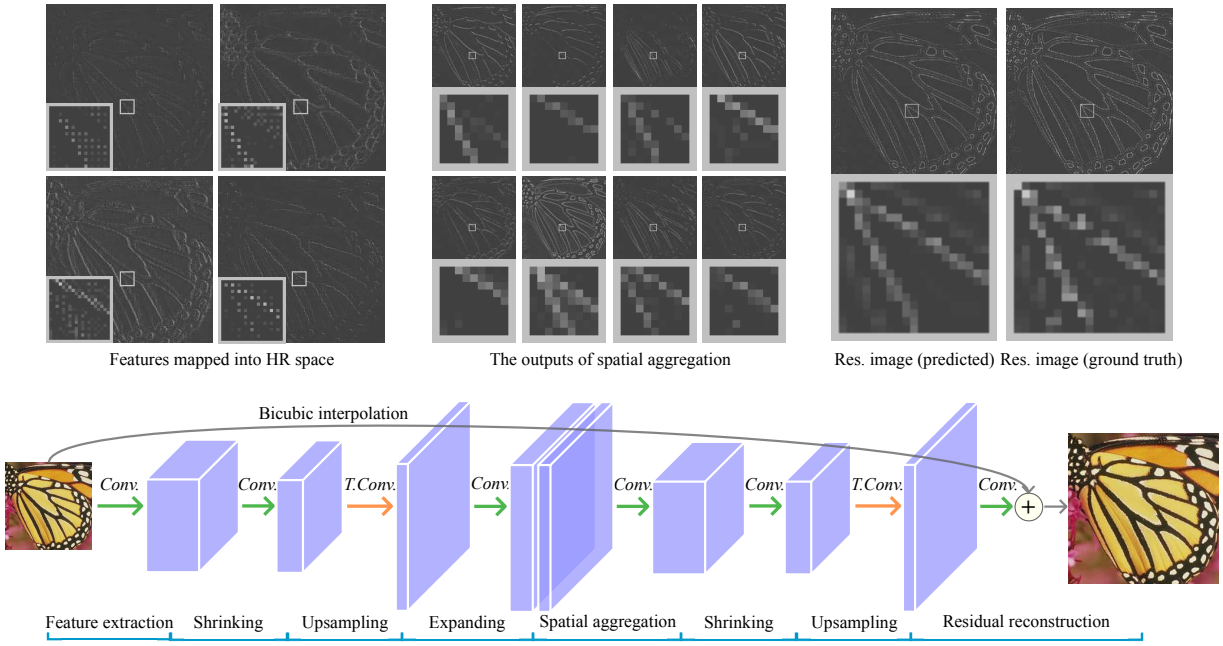


Figure 1. Up-Down Network. Instead of transforming the LR feature representations into the HR output in the last layer, UDNet transfers them between LR and HR spaces multiple times to learn a better mapping. We visualize some typical feature representations from intermediate layers of UDNet to show the processing of network. A representative area is zoomed in at the relative coordinates. And the intensity value of each feature is displayed with a positive offset for better visual experience.

both in width with more filters and in depth with a cascaded structure. Comprehensive experiments demonstrate the superiority of UDNet over previous CNN models on widely used benchmarks, in terms of either speed or performance. We also evaluate the performance of real-time SR of 720p videos, where UDNet outperforms the competitors by a large margin.

2. Related Work

CNN-Based SISR: SRCNN [5] is the first end-to-end convolutional network for SISR that brings the sparse coding algorithm [33] into a trainable network structure. After that, increasingly more network structures are designed for SISR [6, 7, 13, 14]. Although the detailed implementations of these models are different, they all have the same strategy following SRCNN, which contains three main stages: feature extraction, feature integration, and reconstruction. By focusing on designing various topology structures for better feature integration and reconstruction, CNN-based models for SISR have achieved promising results.

Scheme for Acceleration: To accelerate the inference speed for practical applications, the LR image is directly fed into the network, rather than operating on the bicubic interpolated image in HR space. Therefore, a transposed convolution [34] is mostly employed as the last layer to perform upsampling. Dong et al. proposed the FSRCNN model [7] which is a representative. On the other hand, as a notable alternation of transposed convolution, the sub-pixel convolution proposed in ESPCN [20] shifts pixels in a fixed order to perform upsampling. Due to the non-parametric characteristic, a convolution with a kernel size of 1×1 is often adopted to adjust the channels of features. Following the scheme for acceleration, we also adopt an LR image as input. With the proposed UDNet, we alleviate the degradation in a large degree without additional parameters.

Deep residual learning: The concept of residual learning, which is first proposed in ResNet [9] for image classification, has been successfully introduced into image SR [13]. By directly adding the bicubic interpolated image to the output of network, it enables the whole network to learn the residual information. Inspired by that, we introduce an identical connection across the whole network to apply the residual learning.

3. Up-Down Network

UDNet is a fully convolutional network with a lightweight structure (as shown in Fig. 1). Let "*Conv.*" denote the convolutional layer and "*T.Conv.*" denote the transposed convolutional layer. We use three parameters (i.e., kernel size, stride, and the number of filters) to represent each layer in UDNet. Since all of the convolutional layers adopt paddings to keep the boundary from cropping, we omit them for simplifying the expression. Additionally, except for the last layer for residual image reconstruction,

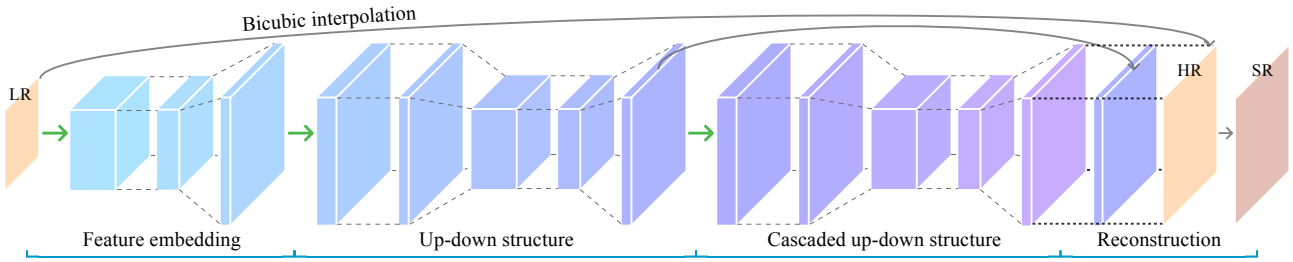


Figure 2. UDNet-C_1: The cascaded variant of UDNet with double up-down structures. We concatenate the features of the last layers from each up-down structure to construct the residual image.

each layer adopts ReLU [17] as the activation function. We divide the components of UDNet into six classes: feature extraction, shrinking, upsampling, expanding, spatial aggregation, and residual reconstruction.

To better clarify the structure of UDNet, we use FSRCNN [7] for reference. Both of them has a lightweight structure and the number of filters is restricted to control the computational complexity for real-time applications. The main different between them is the implementation of mapping. In FSRCNN, the same convolutional layers are stacked for mapping in LR space only. While our proposed UDNet transfers the feature representations between LR and HR spaces. We describe the details of implementation as follows.

Feature extraction and shrinking: UDNet adopts two convolutional layers to extract features, which is a widely used approach in existing CNN-based SR models [5, 6, 7, 13, 14, 20]. These two layers can be represented as *Conv*(3, 1, n_1) and *Conv*(3, 1, n_2). To avoid the sub-optimal bicubic interpolation and reduce the computational complexity, UDNet extracts features in LR space directly. The shrinking strategy aims to reduce the dimension of the extracted features and accelerate the speed of subsequent inference.

Upsampling and expanding: Most existing fast models (e.g., FSRCNN [7]) operate the mapping in LR space only and perform upsampling in the last layer with a single transposed convolutional layer. To better model the inverse one-to-many mapping, we introduce another transposed convolutional layer before the final reconstruction to transform the representations from LR space to HR space. The setting of stride is related to the scale factor. For instance, we adopt *Conv*(4, 2, n_3) for 2x SR and *Conv*(5, 3, n_3) for 3x SR. The expanding strategy is implemented by two convolutional layers which can be represented as *Conv*(3, 1, n_2)-*Conv*(3, 1, n_3). Inspired by [7], it is designed to refine the mapped HR feature representations in the contextual area.

Spatial aggregation and residual reconstruction: A single convolutional layer is adopted to learn spatial aggregation (i.e., down-sampling). The value of stride is related to the scale of factor. The residual reconstruction is composed of two parts: bicubic-interpolated image and predicted residual image. It not only facilitates the convergence but also prevents the low-frequency context from copying through the entire network.

Loss function and visualization: Let x denote the input LR image and y denotes the ground truth. The predicted image is represented as $\hat{x}$, which is the approximation of y (i.e., $\hat{x} \rightarrow y$). Then, the inference can be represented as:

$$\hat{x} = \mathcal{R}_\Theta(x) + \mathcal{B}(x). \tag{1}$$

where $\mathcal{R}_\Theta(\cdot)$ stands for the residual output from the network with parameters Θ and $\mathcal{B}(\cdot)$ stands for the bicubic interpolated image. Following the majority of previous works, the loss function chosen for optimizing the parameters is the mean square error (MSE):

$$\mathcal{L}(\Theta) = \frac{1}{2N}\sum_{i=1}^{N} ||\hat{x}_i - y_i||_2^2 \tag{2}$$

where Θ stands for the parameters in UDNet and N denotes the size of mini-batch for the algorithm of stochastic gradient descent.

To visually understand the feature representations in UDNet, we illustrate the typical feature mappings selected from intermediate layers of UDNet and the predicted image in comparison with the residual image (as shown in Fig. 1).

4. Variants of UDNet

In this section, we explore the variants of UDNet in three aspects: increased number of filters, deeper network structures with cascading and the different scale factors.

UDNet with added filters: The number of filters is an essential parameter of CNN-based models. Let three variables n_1, n_2 and n_3 denote the number of filters as described in Sec. 3. In UDNet, we set $n_1 = 24$, $n_2 = 12$ and $n_3 = 6$. With additional filters, we design five settings for these variables to explore the expansibility of filters: $n_1 = 32$, $n_1 = 48$, $n_1 = 64$, $n_1 = 80$ and $n_1 = 96$. Corresponding to the setting of n_1, n_2 is set as $n_1/2$ and n_3 is set as $n_1/4$. We name this series of variants as UDNet-n_1. The experimental results in Sec. 5.5 show that UDNet can achieve higher reconstruction accuracy by releasing the restriction on parameters.

UDNet with cascaded structures: Cascading is widely used in several methods [4,29] to further enhance the performance. Our proposed UDNet is also suitable for cascading. We design the network based on the UDNet-64 and name this series of variants as UDNet-C$_k$, where k represents the number of cascaded structures. We illustrate the structure of UDNet-C$_1$ in Fig. 2 for a better understanding. In contrast to the method proposed in [29], the cascaded structure of UDNet only works at a fixed scale factor.

Meanwhile, with the cascaded structure, the depth of network is increased. It thus results in difficulty in convergence. To make the network converge to a stable equilibrium point, batch normalization (BN) [11] is appended after each layer following [9]. Experimental results demonstrate that the cascaded variants of UDNet achieve state-of-the-art reconstruction accuracy on widely used benchmarks.

Applying recursion for large scale factors: Facility of different scale factors is often required in SISR. UDNet supports larger scale factors by setting the value of stride in the transposed convolutional layer. For instance, when the scale factor is 3, we set the stride as 3 correspondingly. However, the dimensionality gap between LR and HR spaces will increase along with the scale factors. Thus, inspired by [29], we apply recursion to UDNet for alleviating this issue. For instance, to scale up the input LR image at a factor of 4, we first pass the input image into UDNet and obtain the HR image with a scale factor of 2 as output. Then, we feed the output image as the input into the same UDNet to obtain the target result. As for the scale factor of 3, we obtain it by down-sampling from the 4x image.

5. Experiments and Results

5.1. Datasets

Training datasets: There are four main datasets for training: 91 images from Yang et al. [33], 100 images from "General-100" [7], 200 images from "BSD200" [16], and millions of images from ImageNet. We assume that more training data stands a better probability of achieving state-of-the-art performance. However, various methods have proposed their best results which are trained on different datasets. For eliminating the influence of different training datasets, we train our models on the "General-100" dataset to evaluate the real-time performance following [7] and train on the 291 images (i.e., containing 91 images from [33] and 200 images from [16]) for comparison with state-of-the-art methods following [13]. Moreover, detailed results trained on each dataset can be found in the supplementary document.

As for the implementation, we convert all of the image patches into HDF5 format to adapt for the I/O interface of Caffe [12]. We partition them into the sub-image patches with a size of 96×96. And we set the number of each mini-batch as 32 for stochastic gradient decent.

Models	PNet	UDNet
Input	I_{LR}	
Stage 1	C(3,24,1) C(3,12,24)	
	C(5,6,12) $s=1$	TC(5,6,12) $s=3$
Stage 2	C(3,12,6) C(3,6,12)	
	C(3,24,6) $s=1$	C(3,24,6) $s=3$
	C(3,12,24)	
Stage 3	TC(5,6,12) $s=3$ C(3,1,6)	
PNSR / SSIM (Set5 & Set14)	33.10 / 0.9143 29.43 / 0.8249	33.27 / 0.9165 29.53 / 0.8266

Table 1. Ablation experiments of UDNet. We evaluate the mean PSNR (dB) and SSIM on "Set5" [2] and "Set14" [35] datasets at the scale factor of 3. Here, "C" and "TC" denote convolution and transposed convolution, respectively. And we use (kernel size, output channels, input channels) to denote each layer. By default, the stride is set as 1 and we adopt the symbol "s" to emphasize the different settings of strides.

Benchmarks: Four commonly used datasets, "Set5" [2], "Set14" [35], "BSD100" [16] and "Urban100" [10], are employed as benchmarks for SISR. To evaluate real-time SR in videos, we adopt six 720p videos from Xiph [1] as the testing dataset.

5.2. Settings for Training

We adopt "Adam" [15] as our solver for optimization with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the coefficient of weight decay (l_2 norm) is set at 0.0001. The global basic learning rate is set at 0.001, and we multiply by 0.1 for biases in each layer in particular. The decay policy of the learning rate follows a polynomial function:

$$l_b \times (1 - i/N)^p, \tag{3}$$

where l_b stands for the basic learning rate, i denotes the current time of iterations, N denotes the total number of iterations, and we set $p = 1$. We stop the training procedure when no notable improvement can be observed after 1.2×10^6 iterations. For initialization, we use the method proposed in [8] to initialize the weights and set zeros for the initialization of biases.

5.3. Ablation Experiments of UDNet

In this section, we discuss a plain network for comparison with UDNet. We replace the intermediate transposed convolution with a standard convolution of same hyper-parameters to derive the plain network (named as PNet). As listed in Table 1, we decompose the overall structure into three stages: feature extraction, feature integration, and reconstruction. The PNet operates the feature integration

[1] Xiph.org Video. https://media.xiph.org/video/derf/

Models	SRCNN [5]	SRCNN-Ex [6]	CSCN [29]	ESPCN [20]	FSRCNN [7]	UDNet (Ours)
Input	I_{HR}	I_{HR}	I_{HR}	I_{LR}	I_{LR}	I_{LR}
Stage 1	C(9,64,1)	C(9,64,1)	C(5,4,1) C(5,100,4)	C(5,64,1)	C(5,56,1) C(1,12,56)	C(3,24,1) C(3,12,24) TC(5,6,12)
Stage 2	C(1,32,64)	C(5,32,64)	$3 \times$ *linear layers*	C(3,32,64) C(3,9,32)	$4 \times C$(3,12,12) C(1,56,12)	C(3,12,6) C(3,6,12) C(3,24,6) C(3,12,24)
Stage 3	C(5,1,32)	C(5,1,32)	C(1,25,1) C(5,1,25)	*sub-pixel*	TC(9,1,56)	TC(5,6,12) C(3,1,6)
Parameters	8,032	57,184	$\approx$ 32,000	22,624	12,464	11,646
Time (s)	0.2648	0.8163	1.1302	0.1016	0.1004	**0.0924**
PSNR/SSIM	32.39/0.9033	32.75/0.9090	33.10/0.9144	33.01/0.9121	33.16/0.9140	**33.27/0.9165**

Table 2. Comparisons of mean PSNR (dB) and SSIM between UDNet and existing CNN-based methods. We decompose the overall structure into three stages: feature extraction, feature integration, and reconstruction. The linear layers in CSCN [29] are related to the over-complete dictionary. We train UDNet on the "General-100" dataset following [7] and evaluate the results on the dataset "Set5" [2] at the scale factor of 3 on the uniform test platform. The mean running times are calculated using a single CPU following [20].

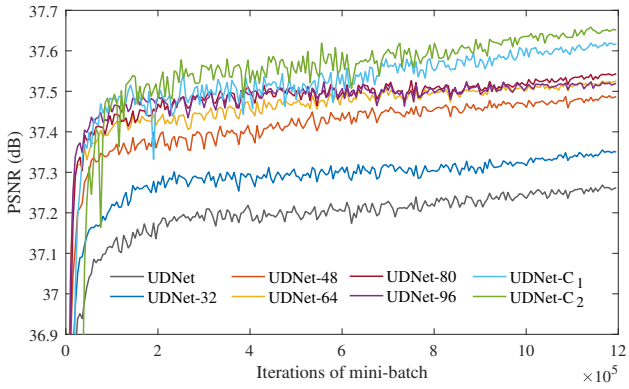


Figure 3. Convergence curves of UDNet and its variants. We display the change of mean PSNR on "Set5" at a scale factor of 2.

in LR space only, which is similar to FSRCNN [7]. While the UDNet transfers the feature representations between LR and HR spaces. We train both of them on the "General-100" dataset [7]. Experimental results demonstrate the advantage of UDNet over the plain network in Table 1.

5.4. UDNet with Restricted Parameters

Uniform test platform: We establish a uniform platform based on the libraries of MatConvNet[2] [28] for evaluating CNN-based models with publicly available implementations. We convert the trained filters of various models into the same format to prevent error in evaluating the running time on different platforms. To measure the running time of each model exactly, we repeat the testing procedure of each model 20 times and report the average running time. For evaluation, we crop the boundary of each image in the benchmarks as the same pixel as the scale factor following the majority of previous works. We calculate the PSNR and SSIM using only the luminance components, following the evaluation platform first established in [25].

[2]MatConvNet. http://www.vlfeat.org/matconvnet/

Models	UDNet	UDNet-32	UDNet-48	UDNet-64
PSNR	27.88	27.93	28.02	28.05
Models	UDNet-80	UDNet-96	UDNet-C_1	UDNet-C_2
PSNR	28.08	28.07	28.10	28.14

Table 3. Comparisons between UDNet and its variants. We evaluate the mean PSNR results on "Set14" at a scale factor of 4.

Enhancing performance without adding parameters: Let "C" denote the convolution and "TC" denote the transposed convolution. We use three values inside the brackets to represent the primary variables of convolutional layers (kernel size (S), output channels (C_{out}), input channels (C_{in})). The number of parameters (P) is the product of these three values: $P = S^2 \times C_{out} \times C_{in}$. Although the parameter number is quite essential for measuring the capacity of a model, it cannot exactly represent the computational complexity in practice. Thus, we further evaluate the average running time on the uniform test platform using a single kernel of a general CPU running at 4GHz (as listed in Table 2). We adopt "Set5" [2] as the testing dataset and train our model on the "General-100" dataset following [7].

We list the experimental results of the comparison with several CNN-based models in Tabel 2. By receiving the LR image I_{LR} directly as input without pre-interpolation, UDNet has a clear advantage in performance compared to previous models. It has neither additional parameters nor an increase in computational complexity.

5.5. Comparison between UDNet and its Variants

Investigation of variants with added parameters: As shown in Fig. 3, UDNet and its variants converge fast and achieve high accuracy after a few iterations. By releasing the restriction on parameters, even better performance is achieved. According to the results shown in Table 3, a moderate number of parameters can help to achieve better performance. However, too many parameters (such as UDNet-96) is not an efficient choice since it may cause redundancy

(a) Original (PSNR/SSIM) (b) Bicubic (23.30/0.750) (c) ANR(24.16/0.808) (d) A+ (24.32/0.819) (e) SRCNN (24.39/0.821)

(f) CSCN (24.28/0.816) (g) FSRCNN (24.62/0.830) (h) VDSR (24.87/0.843) (i) UDNet (24.71/0.836) (j) UDNet-C_2 (24.89/0.843)

Figure 4. Comparisons of image "58060" from "BSD100" [16] dataset at a scale factor of 2.

Dataset	Scale	Bicubic PSNR/SSIM	A+ [27] PSNR/SSIM	RAISR [19] PSNR/SSIM	CSCN [29] PSNR/SSIM	ESPCN [20] PSNR/SSIM	FSRCNN [7] PSNR/SSIM	VDSR [13] PSNR/SSIM	UDNet PSNR/SSIM	UDNet-C_2 PSNR/SSIM
Set5	×2	33.66/0.9299	36.54/0.9544	36.15/0.951	36.93/0.9552	-/-	37.00/0.9558	37.53/0.9587	37.29/0.9576	**37.67/0.9591**
	×3	30.39/0.8682	32.58/0.9088	32.21/0.901	33.10/0.9144	33.01/0.9121	33.16/0.9140	33.66/0.9213	33.32/0.9176	**33.74/0.9222**
	×4	28.42/0.8104	30.28/0.8603	29.84/0.848	30.86/0.8732	30.78/0.8681	30.71/0.8657	31.35/0.8838	31.04/0.8772	**31.42/0.8850**
Set14	×2	30.23/0.8687	32.28/0.9056	32.13/0.902	32.56/0.9074	-/-	32.63/0.9088	33.03/0.9124	32.88/0.9110	**33.08/0.9128**
	×3	27.54/0.7736	29.13/0.8188	28.86/0.812	29.41/0.8238	29.44/0.8255	29.43/0.8242	29.77/0.8314	29.63/0.8291	**29.83/0.8324**
	×4	26.00/0.7019	27.32/0.7491	27.00/0.738	27.64/0.7587	27.70/0.7591	27.59/0.7535	28.01/0.7674	27.88/0.7639	**28.14/0.7705**
B100	×2	29.56/0.8431	31.21/0.8863	-/-	31.40/0.8884	-/-	31.50/0.8906	31.90/**0.8960**	31.72/0.8935	**31.90**/0.8959
	×3	27.21/0.7384	28.29/0.7835	-/-	28.50/0.7885	28.51/0.7895	28.52/0.7893	**28.82**/0.7976	28.66/0.7949	28.81/**0.7986**
	×4	25.96/0.6674	26.82/0.7087	-/-	27.03/0.7161	27.03/0.7166	26.96/0.7128	27.29/0.7251	27.15/0.7226	**27.30/0.7277**
U100	×2	26.88/0.8403	29.20/0.8938	-/-	29.76/0.9009	-/-	29.83/0.9016	30.76/0.9140	30.31/0.9078	**30.88/0.9155**
	×3	24.46/0.7349	26.03/0.7973	-/-	26.45/0.8093	-/-	26.42/0.8064	**27.14**/0.8279	26.72/0.8161	27.09/**0.8290**
	×4	23.14/0.6577	24.32/0.7183	-/-	24.75/0.7372	-/-	24.58/0.7269	**25.18**/0.7524	24.91/0.7409	25.15/**0.7562**

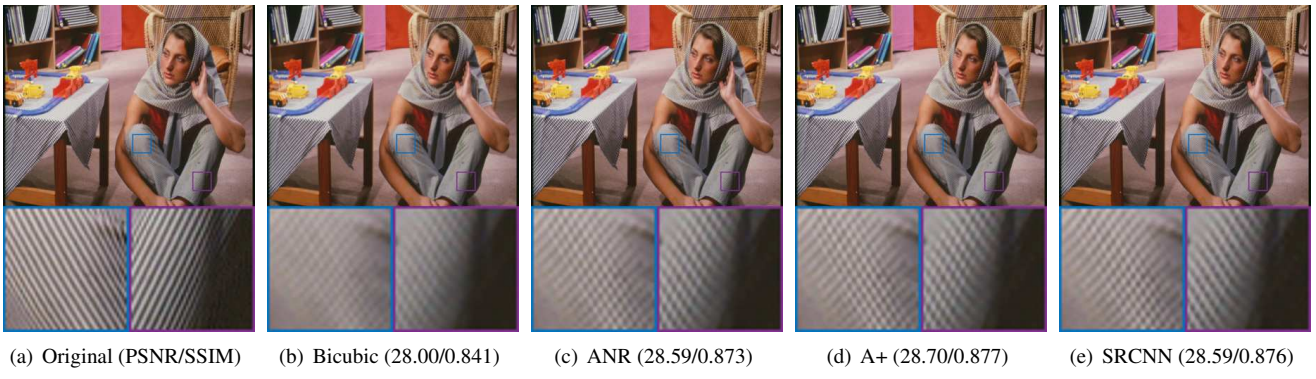
Table 4. Comparisons of mean PSNR/SSIM on four widely used benchmarks at scale factors of 2, 3 and 4. The best results reported in the corresponding papers are presented. For ESPCN [20], we measure the results using the given images from its supplementary material.

in the feature representation.

Cascading – the deeper, the better: The hyper-parameters of each layer for cascading are the same as UDNet-64, where $n_1 = 64$, $n_2 = 32$ and $n_3 = 16$. As shown in Fig. 3, it is more difficult for the UDNet-C_k to find a stable way to converge at the beginning of training due to the increase of depth. Along with the gradual reduction of learning rate during training, they have a notable tendency to achieve better performance than the variants with added parameters. This result suggests that by optimizing the policy of learning rate, the reconstruction accuracy of the cascaded variant can be further improved. Moreover, according to the results listed in Table 3, we believe that the deeper the cascaded structure is, the larger improvement can be achieved.

5.6. Comparison with State-of-the-Art Methods

Single image super-resolution: As shown in Table 4, we adopt six representative models for comparison. Among these models, A+ [27] is a representative of traditional SR methods. RAISR [19], ESPCN [20] and FSRCNN [7] are typical models with compact structures for real-time performance. CSCN [29] and VDSR [13] are models with high reconstruction accuracy. The experimental results show that the cascaded variants of UDNet achieve state-of-the-art performance at various scales on four widely used benchmarks. In Fig. 6, we further evaluate the efficiency of various models considering both performance and running time evaluated using a single GPU 980Ti. Among them, additional models (i.e., DRCN [14], IA [26], ANR [25] and NE+LLE [3]) are adopted for comparison.



(a) Original (PSNR/SSIM) (b) Bicubic (28.00/0.841) (c) ANR (28.59/0.873) (d) A+ (28.70/0.877) (e) SRCNN (28.59/0.876)

(f) CSCN (28.31/0.870) (g) FSRCNN (28.31/0.875) (h) VDSR (28.41/0.878) (i) UDNet (28.50/0.877) (j) UDNet-C_2 (28.60/0.878)

Figure 5. Comparisons of the image "barbara" from the "Set14" [35] dataset at a scale factor of 2.

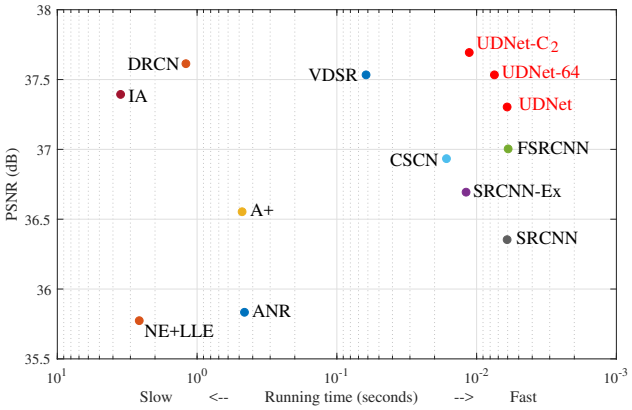


Figure 6. Illustration of the trade-off between accuracy and speed for various models. We evaluate the average running time and mean PSNR on "Set5" at a scale factor of 2. From this figure, we can have the following conclusions. Compared with fast (real-time) SR models, our methods can largely boost the reconstruction accuracy. When compared with SR models which have high reconstruction accuracy, our methods can greatly accelerate the speed without performance degradation.

Also, several examples of visual comparison are presented. As shown in Fig. 4, with the same training dataset, UDNet-C_2 has a distinct advantage over VDSR [13] in the areas with streaks. In Fig. 5, previous methods do not perform well in the pinstripe areas, and UDNet-C_2 is the only model that reconstructs them with correct directions. More comparisons can be found in the supplementary document.

Dataset	Scale	Bicubic	SRCNN	FSRCNN	UDNet	UDNet-32
4People	×2	35.66	38.41	39.33	39.54	**39.86**
K.&S.	×2	35.46	38.32	40.11	40.18	**40.44**
Mobcal	×2	30.79	32.55	32.95	33.13	**33.14**
P.Run	×2	25.86	27.08	27.41	27.43	**27.46**
Shields	×2	31.54	33.18	33.72	33.78	**33.85**
STO	×2	32.21	33.72	34.20	34.33	**34.38**
Average	×2	31.92	33.88	34.62	34.73	**34.86**
4People	×4	30.17	31.25	31.79	32.06	**32.25**
K.&S.	×4	29.86	31.26	31.81	32.15	**32.32**
Mobcal	×4	26.48	27.19	27.74	27.78	**27.90**
P.Run	×4	22.26	22.71	22.71	22.84	**22.89**
Shields	×4	26.94	27.78	28.25	28.16	**28.28**
STO	×4	27.63	28.23	28.32	28.55	**28.63**
Average	×4	27.22	28.07	28.44	28.59	**28.71**

Table 5. Comparisons of the mean PSNR for evaluating real-time SR. We evaluate the results on six 720p videos from Xiph database. All models are trained on the "General-100" dataset [7]. UDNet and UDNet-32 have clear advantage over the competitors.

Toward real-time SR for 720p videos: To evaluate real-time ($>$ 24 FPS) SR in videos, SRCNN [5], FSRCNN [7] and the bicubic interpolation are employed for comparison, as listed in Table 5. We adopt six 720p videos from Xiph database for evaluation. UDNet and UDNet-32 achieve 38.1 and 27.8 FPS on a single GPU 980Ti, respectively. Under the premise of real-time speed for 720p videos, UDNet outperforms the competitors by a large margin.

6. Conclusion

In this paper, we propose a novel Up-Down Network for compact and efficient feature representation in real-time SR. By transferring the feature representations between LR and HR spaces multiple times, UDNet learns a better mapping for this inverse problem. Comprehensive experiments demonstrate the superiority of UDNet and its variants over previous CNN models on widely used benchmarks, in terms of either speed or performance. These results make it a solid step for exploring more efficient feature representation and learning in SISR.

References

[1] H. A. Aly and E. Dubois. Image up-sampling using total-variation regularization with a new observation model. In *TIP*, 2014. 1

[2] M. Bevilacqua, A. Roumy, C. Guillemot, and M. A. Morel. Low complexity single-image super-resolution based on non-negative neighbor embedding. In *BMVC*, 2012. 4, 5

[3] H. Chang, D. Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *CVPR*, 2004. 6

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

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

[6] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. In *TPAMI*, 2015. 1, 2, 3, 5

[7] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 7

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *CoRR*, 2015. 4

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4

[10] J. B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 1, 4

[11] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[12] 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 *arXiv:1408.5093*, 2014. 4

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

[14] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016. 1, 2, 3, 6

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *arXiv:1412.6980*, 2014. 4

[16] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 4, 6

[17] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3

[18] G. Riegler, S. Schulter, M. Ruther, and H. Bischof. Conditioned regression models for non-blind single image super-resolution. In *ICCV*, 2015. 1

[19] Y. Romano, J. Isidoro, and P. Milanfar. Rapid and accurate image super resolution. In *TCI*, 2016. 1, 6

[20] W. Shi, J. Caballero, F. Huszar, 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*, 2016. 1, 2, 3, 5, 6

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv:1409.1556*, 2014. 1

[22] J. Sun, Z. Xu, and H. Y. Shum. Image super-resolution using gradient profile prior. In *CVPR*, 2008. 1

[23] J. Sun, N. N. Zheng, H. Tao, and H. Y. Shum. Image hallucination with primal sketch priors. In *CVPR*, 2003. 1

[24] Y. W. Tai, S. Liu, M. S. Brown, and S. Lin. Super resolution using edge prior and single image detail synthesis. In *CVPR*, 2010. 1

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

[26] R. Timofte, R. Rothe, and L. V. Gool. Seven ways to improve example-based single image super resolution. In *CVPR*, 2016. 6

[27] R. Timofte, V. D. Smet, and L. V. Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014. 1, 6

[28] A. Vedaldi and K. Lenc. Matconvnet - convolutional neural networks for matlab. In *CoRR*, 2014. 5

[29] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *ICCV*, 2015. 1, 4, 5, 6

[30] Z. Xiong, X. Sun, and F. Wu. Image hallucination with feature enhancement. In *CVPR*, 2009. 1

[31] Z. Xiong, D. Xu, X. Sun, and F. Wu. Example-based super-resolution with soft information and decision. In *TMM*, 2013. 1

[32] J. Yang, Z. Lin, and S. Cohen. Fast image super-resolution based on in-place example regression. In *CVPR*, 2013. 1

[33] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. In *TIP*, 2010. 1, 2, 4

[34] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011. 2

[35] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representation. In *Curves and Surfaces*, 2012. 4, 7