Image Super-Resolution via Deep Recursive Residual Network

Ying Tai* [1], Jian Yang[1], and Xiaoming Liu[2]

[1]Department of Computer Science and Engineering, Nanjing University of Science and Technology
[2]Department of Computer Science and Engineering, Michigan State University

{taiying, csjyang}@njust.edu.cn, liuxm@cse.msu.edu

Abstract

Recently, Convolutional Neural Network (CNN) based models have achieved great success in Single Image Super-Resolution (SISR). Owing to the strength of deep networks, these CNN models learn an effective nonlinear mapping from the low-resolution input image to the high-resolution target image, at the cost of requiring enormous parameters. This paper proposes a very deep CNN model (up to 52 convolutional layers) named Deep Recursive Residual Network (DRRN) that strives for deep yet concise networks. Specifically, residual learning is adopted, both in global and local manners, to mitigate the difficulty of training very deep networks; recursive learning is used to control the model parameters while increasing the depth. Extensive benchmark evaluation shows that DRRN significantly outperforms state of the art in SISR, while utilizing far fewer parameters. Code is available at https://github.com/tyshiwo/DRRN_CVPR17.

1. Introduction

Single Image Super-Resolution (SISR) is a classic computer vision problem, which aims to recover a high-resolution (HR) image from a low-resolution (LR) image. Since SISR restores the high-frequency information, it is widely used in applications such as medical imaging [26], satellite imaging [29], security and surveillance [37], where high-frequency details are greatly desired.

In recent years, due to the powerful learning ability, Deep Learning (DL) models, especially Convolutional Neural Networks (CNN), are widely used to address the ill-posed inverse problem of Super Resolution (SR), and have demonstrated superiority over reconstruction-based methods [4, 35] or other learning paradigms [20, 22, 23, 31]. As the pioneer CNN model for SR, Super-Resolution Convolutional Neural Network (SRCNN) [2] predicts the nonlinear LR-HR mapping via a fully convolutional network, and significantly outperforms classical non-DL methods. However, SRCNN does not consider any self similarity property. To address this issue, the Deep Joint Super Resolution (DJSR) jointly utilizes both the wealth of external examples and the power of self examples unique to the input. Inspired by the learning iterative shrinkage and thresholding algorithm [5], Cascaded Sparse Coding Network (CSCN) [32] is trained end-to-end to fully exploit the natural sparsity of images. Shi et al. [25] observe that the prior models [2, 32] increase LR image's resolution via bicubic interpolation be-

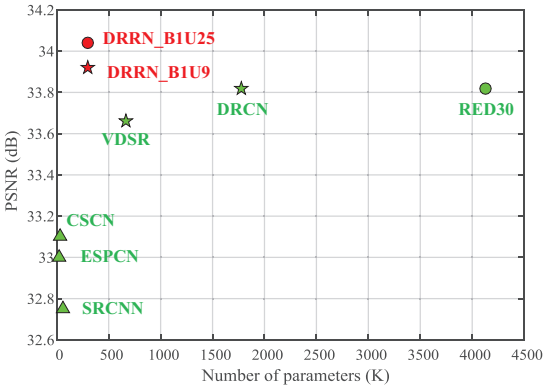


Figure 1. PSNR of recent CNN models for scale factor $\times 3$ on Set5 [1]. Red points are our models. △, ☆, and ○ are models with less than 5 layers, 20 layers, and more than 30 layers, respectively. DRRN_B1U9 means there is 1 recursive block, in which 9 residual units are stacked. With *the same depth but far fewer parameters*, DRRN_B1U9 achieves better performance than the state-of-the-art methods [13, 14]. After increasing the depth without adding any parameters, the 52-layer DRRN_B1U25 further improves the performance and significantly outperforms VDSR [13], DRCN [14] and RED30 [17] by 0.37, 0.21 and 0.21 dB respectively.

*This work was conducted when the first author was a visiting scholar at Michigan State University. It was supported by the National Science Fund of China under Grant Nos. 91420201, 61472187, 61502235, 61233011, 61373063 and 61602244, the 973 Program No.2014CB349303, Program for Changjiang Scholars and Innovative Research Team in University, and partially sponsored by CCF-Tencent Open Research Fund.

fore CNN learning, which increases the computational cost. The Efficient Sub-Pixel Convolutional neural Network (ESPCN) reduces the computational and memory complexity, by increasing the resolution from LR to HR only at the end of the network.

One commonality among the above CNN models is that their networks contain fewer than 5 layers, e.g., SRCNN [2] uses 3 convolutional layers. Their deeper structures with 4 or 5 layers do not achieve better performance, which was attributed to the difficulty of training deeper networks and led to the observation that "the deeper the better" might not be the case in SR. Inspired by the success of very deep networks [8, 27, 28] on ImageNet [21], Kim et al. [13, 14] propose two very deep convolutional networks for SR, both stacking 20 convolutional layers, from the viewpoints of training efficiency and storage, respectively. On the one hand, to accelerate the convergence speed of very deep networks, the VDSR [13] is trained with a very high learning rate (10^{-1}, instead of 10^{-4} in SRCNN) and the authors further use residual learning and adjustable gradient clipping to solve gradient explosion problem. On the other hand, to control the model parameters, the Deeply-Recursive Convolutional Network (DRCN) [14] introduces a very deep recursive layer via a *chain structure* with up to 16 recursions. To mitigate the difficulty of training DRCN, the authors use recursive-supervision and skip-connection, and adopt an ensemble strategy to further improve the performance. Very recently, Mao et al. [17] propose a 30-layer convolutional auto-encoder network named RED30 for image restoration, which uses symmetric skip connections to help training. All of the three models learn the residual image between the input Interpolated LR (ILR) image and the ground truth HR image in the *residual branch*. The residual image is then added to the ILR image from the *identity branch* to estimate the HR image. The three models outperform the previous DL and non-DL methods by a large margin, which demonstrates "the deeper the better" is still true in SR.

Despite achieving excellent performance, the very deep networks require enormous parameters. Compared to the compact models, large models demand more storage space and are less applicable to mobile systems [6]. To address this issue, we propose a novel Deep Recursive Residual Network (DRRN) to effectively build a very deep network structure, which achieves better performance, but with $2\times$, $6\times$, and $14\times$ fewer parameters than VDSR, DRCN, and RED30, respectively. *In a nutshell, DRRN advances the SR performance with a deeper yet concise network.* Specifically, DRRN has two major algorithmic novelties:

(1) **Both global and local residual learning** are introduced in DRRN. In VDSR and DRCN, the residual image is estimated from the input and output of the networks, termed as *Global Residual Learning* (GRL). Since the SR output is vastly similar to the input, GRL is effective in easing the difficulty of training deep networks. Therefore, we also adopt GRL in our *identity branch*. Further, very deep networks could suffer from the performance degradation problem, as observed in visual recognition [8] and image restoration [17]. The reason may be a significant amount of image details are lost after so many layers. To address this issue, we introduce an enhanced residual unit structure, termed as *multi-path* mode *Local Residual Learning* (LRL), where the identity branch not only carries rich image details to late layers, but also helps gradient flow. GRL and LRL mainly differ in that LRL is performed in every few stacked layers, while GRL is performed between the input and output images, i.e., DRRN has many LRLs and only 1 GRL.

(2) **Recursive learning** of residual units is proposed in DRRN to keep our model compact. In DRCN [14], a deep recursive layer (up to 16 convolutional recursions) is learned and the weights are shared in the 16 convolutional recursions. Our DRRN has two major differences compared to DRCN: (a) Unlike DRCN that shares weights among convolutional layers, DRRN has a *recursive block* consisting of several *residual units*, and the weight set is shared among these residual units. (b) To address the vanishing/exploding gradients problem of very deep models, DRCN supervises every recursion so that the supervision on early recursions help backpropagation. DRRN is relieved from this burden by designing a recursive block with a *multi-path structure*. Our model can be easily trained even with 52 *convolutional layers*. Last but not least, through recursive learning, DRRN can improve accuracy by increasing depth without adding any weight parameters.

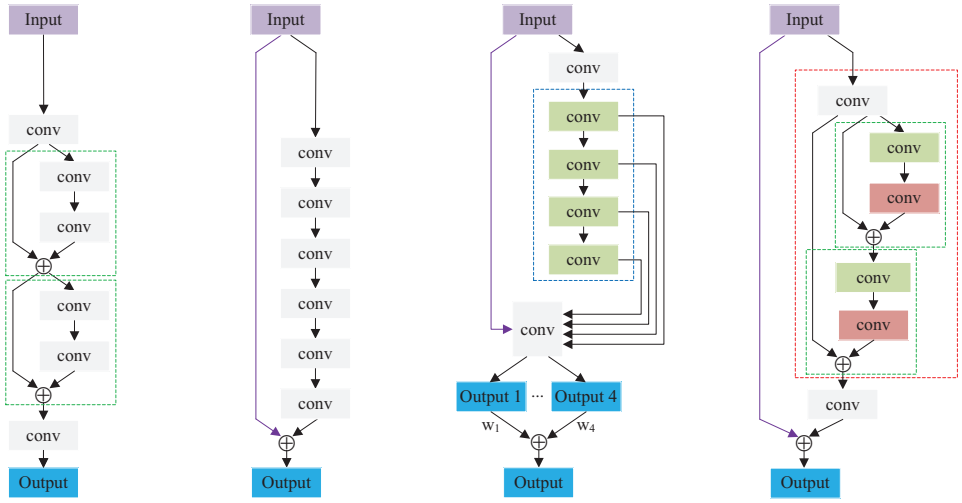
To illustrate the effectiveness of the two strategies used in DRRN, Fig. 1 shows the Peak Signal-to-Noise Ratio (PSNR) performance of several recent CNN models for SR [2, 13, 14, 17, 25, 32] versus the number of parameters, denoted as k. Compared to the prior CNN models, DRRN achieves the best performance with fewer parameters.

2. Related Work

Since Sec. 1 overviews DL-based SISR, this section focuses on three most related work to ours: ResNet [8], VDSR [13] and DRCN [14]. Fig. 2 illustrates these models via simplified network structures with only 6 convolutional layers, where the activation functions, batch normalization (BN) [11] and ReLU [19], are omitted for clarity.

2.1. ResNet

The main idea of ResNet [8] is to use a residual learning framework to ease the training of very deep networks. Instead of hoping every few stacked layers directly fit the desired underlying mapping, the authors explicitly let these layers fit a residual mapping, which is assumed to be easier for optimization. Denoting the input as $\mathbf{x}$ and the underlying mapping as $\mathcal{H}(\mathbf{x})$, the residual mapping is defined as



(a) ResNet (b) VDSR (c) DRCN (d) DRRN (ours)

Figure 2. Simplified structures of (a) ResNet [8]. The green dashed box means a residual unit. (b) VDSR [13]. The purple line refers to a global identity mapping. (c) DRCN [14]. The blue dashed box refers to a recursive layer, among which the convolutional layers (with light green color) share the same weights. (d) DRRN. The red dashed box refers to a recursive block consisting of two residual units. In the recursive block, the corresponding convolutional layers in the residual units (with light green or light red color) share the same weights. In all four cases, the outputs with light blue color are supervised, and $\bigoplus$ is the element-wise addition.

Method	Key Strategies	Mathematical Formulation
ResNet	*Chain* mode local residual learning	$\mathbf{y} = f_{Rec}(\mathcal{U}_{\mathrm{U}}(\mathcal{U}_{\mathrm{U}-1}(...(\mathcal{U}_1(f_1(\mathbf{x})))...)))$
VDSR	Global residual learning	$\mathbf{y} = f_{Rec}(f_{d-1}(f_{d-2}(...(f_1(\mathbf{x}))...))) + \mathbf{x}$
DRCN	Global residual learning + recursive learning (*single weight layer*) + multi-CNN ensemble	$\mathbf{y} = \Sigma_{t=1}^{\mathrm{T}} w_t \cdot (f_{Rec}(f_2^{(t)}(f_1(\mathbf{x}))) + \mathbf{x})$
DRRN	*Multi-path* mode local residual learning + global residual learning + recursive learning (*multiple weight layers* in the residual unit)	$\mathbf{y} = f_{Rec}(\mathcal{R}_{\mathrm{B}}(\mathcal{R}_{\mathrm{B}-1}(...(\mathcal{R}_1(\mathbf{x}))...))) + \mathbf{x}$

Table 1. Strategies used in ResNet [8], VDSR [13], DRCN [14] and DRRN. U, d, T, and B are the numbers of residual units in ResNet, convolutional layers in VDSR, recursions in DRCN, and recursive blocks in DRRN, respectively. $\mathbf{x}$ and $\mathbf{y}$ are input and output of networks. f denotes function of convolutional layer. $\mathcal{U}$ denotes function of residual unit structure and $\mathcal{R}$ denotes function of our recursive block.

$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$, and a *residual unit* structure is thus:

$$\hat{\mathbf{x}} = \mathcal{U}(\mathbf{x}) = \sigma(\mathcal{F}(\mathbf{x}, W) + h(\mathbf{x})), \tag{1}$$

where $\hat{\mathbf{x}}$ is the output of the residual unit, $h(\mathbf{x})$ is an *identity mapping* [8] : $h(\mathbf{x}) = \mathbf{x}$, W is a set of weights (the biases are omitted to simplify notations), function σ denotes ReLU, $\mathcal{F}(\mathbf{x}, W)$ is the residual mapping to be learned, and $\mathcal{U}$ denotes the function of the residual unit structure. For a basic residual unit that stacks two convolutional layers, $\mathcal{F}(\mathbf{x}, W) = W_2\sigma(W_1\mathbf{x})$. By stacking such structures to construct a very deep 152-layer network, ResNet won the first place in the ILSVRC 2015 classification competition. Since the residual learning in ResNet is adopted in every few stacked layers, this strategy is a form of local residual learning, where residual units are stacked in the *chain* mode.

2.2. VDSR

Differing from ResNet that uses residual learning in every few stacked layers, VDSR [13] introduces GRL, i.e., residual learning between the input ILR image and the output HR image. There are three notes for VDSR: (1) Unlike SRCNN [2] that only uses 3 layers, VDSR stacks 20 weight layers (3×3 for each layer) in the residual branch, which leads to a much larger receptive field (41×41 vs. 13×13). (2) GRL and adjustable gradient clipping enable VDSR to converge very fast ($\sim$4 hours on GPU Titan Z). (3) By adopting scale augmentation, a single network of VDSR is robust to images with different scales. Later, we will show that *VDSR actually is a special case of DRRN, when there's no residual unit in our recursive block.*

2.3. DRCN

DRCN [14] is motivated by the observation that adding more weight layers introduces more parameters, where the model is likely to overfit and also becomes disk hungry. To address these issues, the authors introduce a *recursive layer* into the network, so that the model parameters do not increase while more recursions are performed in the recursive layer. DRCN consists of three parts: embedding net, inference net and reconstruction net, which are illustrated as the first, middle 4, and last convolutional layer(s) in Fig. 2(c), respectively. The embedding net $f_1(\mathbf{x})$ represents a given

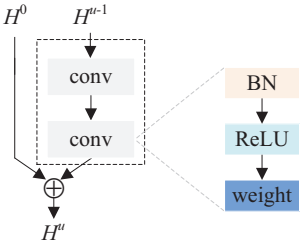


Figure 3. A close look at the u-th residual unit in DRRN. The black dashed box represents the residual function $\mathcal{F}$, which consists of two "conv" layers and each stacked by BN-ReLU-weight layers.

image $\mathbf{x}$ as feature maps H_0. The inference net $f_2(H_0)$ stacks T recursions (T = 16 in [14]) in a recursive layer, with shared weights among these recursions. Finally, the reconstruction net $f_{Rec}(H_{\mathrm{T}})$, where H_{T} is the output of the inference net, generates the intermediate HR image. However, since training such a deep network is difficult, the authors further propose two mitigations, recursive-supervision and skip-connection. Specifically, for the t-th intermediate recursion in the recursive layer, the output after reconstruction net is formulated as

$$\mathbf{y}_t = f_{Rec}(f_2^{(t)}(f_1(\mathbf{x}))) + \mathbf{x}, \tag{2}$$

where $\mathbf{x}$ is skip-connection and basically a GRL. Each intermediate prediction $\mathbf{y}_t$ is learned with supervision. Finally, an ensemble strategy is adopted and the output is the weighted average of all predictions $\mathbf{y} = \Sigma_{t=1}^{\mathrm{T}} w_t \cdot \mathbf{y}_t$, with weights w_t learned during training.

3. Deep Recursive Residual Network

In this section, we present the technical parts of our proposed DRRN. Specifically, we adopt global residual learning in the identity branch and introduce recursive learning into the residual branch by constructing the recursive block structure, in which several residual units are stacked. Noted that in ResNet [8], different residual units use different inputs for the identity branch (green dashed boxes in Fig. 2(a)). However, in our recursive block, a multi-path structure is used and all the residual units share the same input for the identity branch (green dashed boxes in Fig. 2(d)), which further facilitates the learning [16]. We highlight the differences of the network structures between DRRN and the related models in Tab. 1. Now, we will gradually present more details of our model, from the residual unit to the recursive block and finally the whole network structure.

3.1. Residual Unit

In ResNet [8], the basic residual unit is formulated as Eq. 1 and the activation functions (BN [11] and ReLU [19]) are performed *after* the weight layers. In contrast to such a "post-activation" structure, He et al. [9] propose a "pre-activation" structure, which performs the activation *before* the weight layers. They claim that the pre-activation version

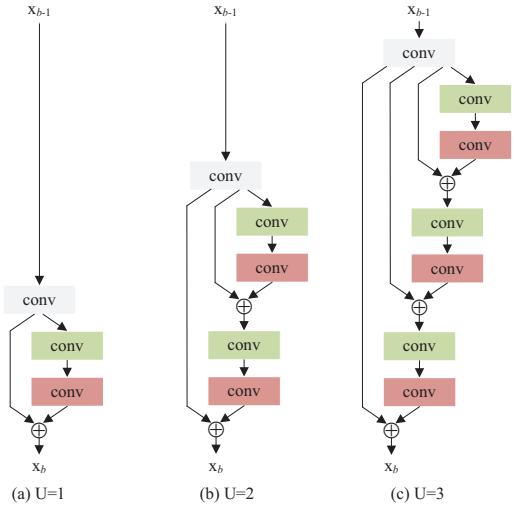


Figure 4. Structures of our recursive blocks. U means number of residual units in the recursive block.

is much easier to train and generates better performance than the post-activation version. Specifically, the residual unit with pre-activation structure is formulated as

$$H^u = \mathcal{F}(H^{u-1}, W^u) + H^{u-1}, \tag{3}$$

where $u = 1, 2, ..., \mathrm{U}$, U is the number of residual units in a recursive block, H^{u-1} and H^u are the input and output of the u-th residual unit, and $\mathcal{F}$ denotes the residual function.

Instead of directly using the above residual unit, we modify Eq. 3 so that the inputs to the identity branch and the residual branch are different. As described in the beginning of Sec. 3, the inputs to all of the identity branches of the residual units in one recursive block are kept the same, i.e., H^0 in Fig. 3. As a result, there are multiple paths between the input and output of our recursive block, as shown in Fig. 4. The residual paths help to learn highly complex features and the identity paths help gradient backpropagation during training. Compared to the chain mode, this multi-path mode facilitates the learning and is less prone to overfitting [16]. Therefore, we formulate our residual unit as

$$H^u = \mathcal{G}(H^{u-1}) = \mathcal{F}(H^{u-1}, W) + H^0, \tag{4}$$

where $\mathcal{G}$ denotes the function of our residual unit, H^0 is the result of the first convolutional layer in the recursive block. Since the residual unit is recursively learned, the weight set W is shared among the residual units within a recursive block, but different across different recursive blocks.

3.2. Recursive Block

We now introduce the details of our recursive block. First, we illustrate the structure of our recursive blocks in Fig. 4. Motivated by [16], we introduce a convolutional layer at the beginning of the recursive block, and then several residual units mentioned in Sec. 3.1 are stacked. We

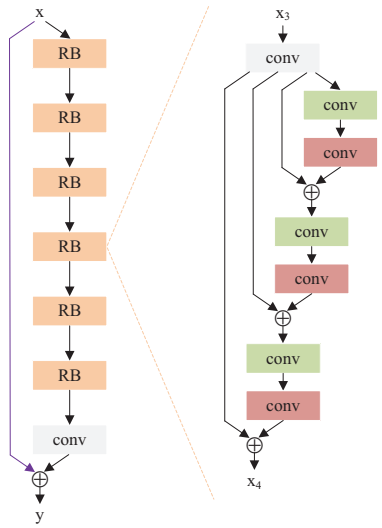


Figure 5. An example network structure of DRRN with B = 6 and U = 3. Here, "RB" layer refers to a recursive block.

denote B as the number of recursive blocks, $\mathbf{x}_{b-1}$ and $\mathbf{x}_b$ ($b = 1, 2, ..., \mathrm{B}$) as the input and output of the b-th recursive block, and $H_b^0 = f_b(\mathbf{x}_{b-1})$ as the result after passing $\mathbf{x}_{b-1}$ through the first convolutional layer, whose function is f_b. According to Eq. 4, the result of u-th residual unit is

$$H_b^u = \mathcal{G}(H^{u-1}) = \mathcal{F}(H_b^{u-1}, W_b) + H_b^0. \quad (5)$$

Thus, the output of the b-th recursive block $\mathbf{x}_b$ is

$$\mathbf{x}_b = H_b^{\mathrm{U}} = \mathcal{G}^{(\mathrm{U})}(f_b(\mathbf{x}_{b-1})) = \mathcal{G}(\mathcal{G}(...(\mathcal{G}(f_b(\mathbf{x}_{b-1})))...)), \quad (6)$$

where U-fold operations of $\mathcal{G}_b$ are performed.

3.3. Network Structure

Finally, we simply stack several recursive blocks, followed by a convolutional layer reconstructing the residual between the LR and HR images. The residual image is then added to the global identity mapping from the input LR image. The entire network structure of DRRN is illustrated in Fig. 5. Actually, VDSR [13] can be viewed as a special case of DRRN, i.e., when U = 0, DRRN becomes VDSR.

DRRN has two key parameters: the recursive block number B and the residual unit number U in each recursive block. Given different B and U, we can learn DRRN with different depths – the number of convolutional layers. Specifically, the depth of DRRN d is calculated as:

$$d = (1 + 2 \times \mathrm{U}) \times \mathrm{B} + 1. \quad (7)$$

Denoting $\mathbf{x}$ and $\mathbf{y}$ to be the input and output of DRRN, $\mathcal{R}$ to be the function of the b-th recursive block, we have

$$\mathbf{x}_b = \mathcal{R}_b(\mathbf{x}_{b-1}) = \mathcal{G}^{(\mathrm{U})}(f_b(\mathbf{x}_{b-1})). \quad (8)$$

When $b = 1$, we define $\mathbf{x}_0 = \mathbf{x}$. Then, DRRN can be formulated as

$$\mathbf{y} = \mathcal{D}(\mathbf{x}) = f_{Rec}(\mathcal{R}_{\mathrm{B}}(\mathcal{R}_{\mathrm{B}-1}(...(\mathcal{R}_1(\mathbf{x}))...))) + \mathbf{x}, \quad (9)$$

where f_{Rec} is a function for the last convolutional layer in DRRN to reconstruct the residual. Tab. 1 lists the mathematical formulations of ResNet, VDSR, DRCN and DRRN.

Given a training set $\{\mathbf{x}^{(i)}, \tilde{\mathbf{x}}^{(i)}\}_{i=1}^N$, where N is the number of training patches and $\tilde{\mathbf{x}}^{(i)}$ is the ground truth HR patch of the LR patch $\mathbf{x}^{(i)}$, the loss function of DRRN is

$$\mathcal{L}(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \|\tilde{\mathbf{x}}^{(i)} - \mathcal{D}(\mathbf{x}^{(i)})\|^2, \quad (10)$$

where Θ denotes the parameter set. The objective function is optimized via the mini-batch stochastic gradient descent (SGD) with backpropagation [15]. We implement DRRN via Caffe [12].

4. Experiments

4.1. Datasets

By following [13, 23], we use a training dataset of 291 images, where 91 images are from Yang et al. [35] and other 200 images are from Berkeley Segmentation Dataset [18]. For testing, we utilize four widely used benchmark datasets, Set5 [1], Set14 [36], BSD100 [18] and Urban100 [10], which have 5, 14, 100 and 100 images respectively.

4.2. Implementation Details

Data augmentation is performed on the 291-image training dataset. Inspired by [30], the flipped and rotated versions of the training images are considered. Specifically, we rotate the original images by 90°, 180°, 270° and flip them horizontally. After that, for each original image, we have 7 additional augmented versions. Besides, inspired by VDSR [13], we also use scale augmentation to train our model, and images with different scales ($\times 2$, $\times 3$ and $\times 4$) are all included in the training set. Therefore, for all different scales, we only need to train a *single* model.

Training images are split into 31×31 patches, with the stride of 21, by considering both the training time and storage complexities. We set the mini-batch size of SGD to 128, momentum parameter to 0.9, and weight decay to 10^{-4}. Every weight layer has 128 filters of the size 3×3.

For weight initialization, we use the same method as He et al. [7], which is shown to be suitable for networks utilizing ReLU. The initial learning rate is set to 0.1 and then decreased to half every 10 epochs. Since a large learning rate is used in our work, we adopt the adjustable gradient clipping [13] to boost the convergence rate while suppressing exploding gradients. Specifically, the gradients are clipped to $[-\frac{\theta}{\gamma}, \frac{\theta}{\gamma}]$, where γ is the current learning rate and $\theta = 0.01$ is the gradient clipping parameter. Training a DRRN of $d = 20$ roughly takes 4 days with 2 Titan X GPUs.

4.3. Study of B and U

In this subsection, we explore various combinations of B and U to construct different DRRN structures with different

Dataset	Scale	Bicubic	SRCNN [2]	SelfEx [10]	RFL [23]	VDSR [13]	DRCN [14]	DRRN_B1U9	DRRN_B1U25
Set5	×2	33.66/0.9299	36.66/0.9542	36.49/0.9537	36.54/0.9537	37.53/0.9587	37.63/0.9588	37.66/0.9589	37.74/0.9591
	×3	30.39/0.8682	32.75/0.9090	32.58/0.9093	32.43/0.9057	33.66/0.9213	33.82/0.9226	33.93/0.9234	34.03/0.9244
	×4	28.42/0.8104	30.48/0.8628	30.31/0.8619	30.14/0.8548	31.35/0.8838	31.53/0.8854	31.58/0.8864	31.68/0.8888
Set14	×2	30.24/0.8688	32.45/0.9067	32.22/0.9034	32.26/0.9040	33.03/0.9124	33.04/0.9118	33.19/0.9133	33.23/0.9136
	×3	27.55/0.7742	29.30/0.8215	29.16/0.8196	29.05/0.8164	29.77/0.8314	29.76/0.8311	29.94/0.8339	29.96/0.8349
	×4	26.00/0.7027	27.50/0.7513	27.40/0.7518	27.24/0.7451	28.01/0.7674	28.02/0.7670	28.18/0.7701	28.21/0.7720
BSD100	×2	29.56/0.8431	31.36/0.8879	31.18/0.8855	31.16/0.8840	31.90/0.8960	31.85/0.8942	32.01/0.8969	32.05/0.8973
	×3	27.21/0.7385	28.41/0.7863	28.29/0.7840	28.22/0.7806	28.82/0.7976	28.80/0.7963	28.91/0.7992	28.95/0.8004
	×4	25.96/0.6675	26.90/0.7101	26.84/0.7106	26.75/0.7054	27.29/0.7251	27.23/0.7233	27.35/0.7262	27.38/0.7284
Urban100	×2	26.88/0.8403	29.50/0.8946	29.54/0.8967	29.11/0.8904	30.76/0.9140	30.75/0.9133	31.02/0.9164	31.23/0.9188
	×3	24.46/0.7349	26.24/0.7989	26.44/0.8088	25.86/0.7900	27.14/0.8279	27.15/0.8276	27.38/0.8331	27.53/0.8378
	×4	23.14/0.6577	24.52/0.7221	24.79/0.7374	24.19/0.7096	25.18/0.7524	25.14/0.7510	25.35/0.7576	25.44/0.7638

Table 2. Benchmark results. Average PSNR/SSIMs for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, BSD100 and Urban100. Red color indicates the best performance of our methods and blue color indicates the best performance of previous methods.

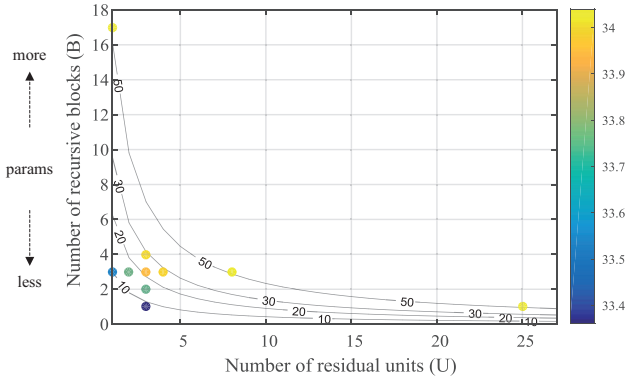


Figure 6. PSNR of various DRRNs at B and U combinations. The color of the point indicates the PSNR that corresponds to the bar on the right and 4 depth contours ($d = 50, 30, 20, 10$) are also plotted. The tests are conducted for scale factor ×3 on Set5.

depths, and see how the two parameters affect the performance. In Fig. 6, we build a grid of B and U, and sample several points in the grid with the depth ranging from 8 to 52 layers. The parameter number keeps the same when more residual units are used in one recursive block, and linearly increases when more recursive blocks are stacked.

First, to clearly show how a single parameter affects DRRN, we fix one parameter to 3 and change the other from 1 to 4. Fig. 6 shows that increasing B or U results in deeper models and achieves better performance, which indicates *deeper is still better*. Despite different structures, these models are comparable as long as their depths are similar, e.g., B2U3 ($d = 15$, $k = 784$K) and B3U2 ($d = 16$, $k = 1,182$K) achieve 33.76 and 33.77 dB, respectively.

The structures mentioned above all use the recursive learning strategy. Next, we test three very different structures to demonstrate the effectiveness of such a strategy. Specifically, we fix one parameter to 1 and change the other to construct networks with $d = 52$. This results in two extreme structures: B1U25 ($k = 297$K) and B17U1 ($k = 7,375$K). For B1U25, only one recursive block is used, in which 25 residual units are recursively learned. For B17U1, 17 recursive blocks are stacked, with no recursive learning. We also construct a normal structure B3U8 ($d = 52$, $k = 1,182$K). Fig. 6 shows that despite different structures, the three networks achieve comparable performance (B17U1 34.03 dB, B3U8 34.04 dB and B1U25 34.03 dB) and outperform the pervious shallow networks. Thanks to the recursive learning strategy, B1U25 can achieve state-of-the-art results using far fewer parameters.

4.4. Comparison with State-of-the-Art Models

We now provide quantitative and qualitative comparisons. Considering both the performance and number of parameters, we choose DRRN_B1U25 ($d = 52$, $k = 297$K) as our best model. For fair comparison, we also construct a DRRN_B1U9 ($d = 20$, $k = 297$K) structure, which has the same depth as VDSR and DRCN, but fewer parameters. Both the DL [2, 13, 14] and non-DL [10, 20, 23] methods in recent years are used for benchmark. Experimental setting is kept the same as those previous methods. Specifically, we first apply bicubic interpolation to the color components of an image and all models are applied to its luminance component only. Therefore, the input and output images are of the same size. For fair comparison, similar to [2, 13, 14, 23], we crop pixels near image boundary before evaluation, although this is unnecessary for DRRN.

Tab. 2 summarizes quantitative results on the four testing sets, by citing the results of prior methods from [13, 14]. The two DRRN models outperforms all existing methods in all datasets and scale factors, in both PSNR and Structural SIMilarity (SSIM) [1]. Especially on the recent difficult Urban100 dataset [10], DRRN significantly advances the state of the art, with the improvement margin of 0.47, 0.38, and 0.26 dB on scale factor ×2, ×3 and ×4 respectively.

Further, we also use another metric: Information Fidelity Criterion (IFC) [24] for comparison, which claims to have the highest correlation with perceptual scores for SR evaluation [34]. The results are presented in Tab. 3. Note that

[1] With two convolutional layers in the residual branch, DRRN achieves state-of-the-art performance. More complex designs have the potential to improve performance but are not the focus of this work.

Dataset	Scale	Bicubic	SRCNN [2]	SelfEx [10]	RFL [23]	PSyCo [20]	VDSR [13]	DRRN_B1U9	DRRN_B1U25
Set5	×2	6.083	8.036	7.811	8.556	8.642	8.569	8.583	8.671
	×3	3.580	4.658	4.748	4.926	5.083	5.221	5.241	5.397
	×4	2.329	2.991	3.166	3.191	3.379	3.547	3.581	3.703
Set14	×2	6.105	7.784	7.591	8.175	8.280	8.178	8.181	8.320
	×3	3.473	4.338	4.371	4.531	4.660	4.730	4.732	4.878
	×4	2.237	2.751	2.893	2.919	3.055	3.133	3.147	3.252
Urban100	×2	6.245	7.989	7.937	8.450	8.589	8.645	8.653	8.917
	×3	3.620	4.584	4.843	4.801	5.031	5.194	5.259	5.456
	×4	2.361	2.963	3.314	3.110	3.351	3.496	3.536	3.676

Table 3. Benchmark results. Average IFCs for the scale factor ×2, ×3 and ×4 on datasets Set5, Set14 and Urban100. Red color indicates the best performance of our methods and blue color indicates the best performance of previous methods.

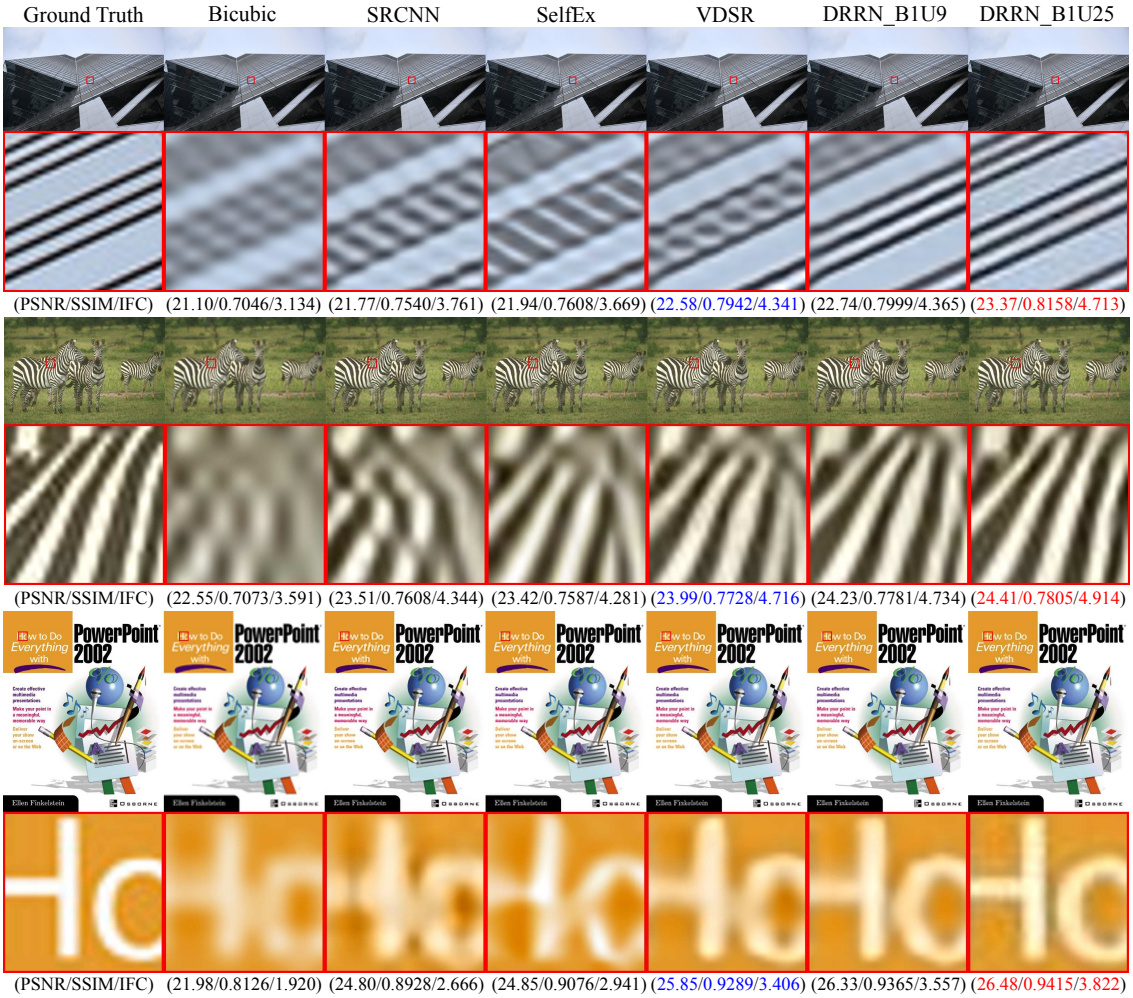


Figure 7. Qualitative comparison. (1) The first row shows image "img059" (Urban100 with scale factor ×3). DRRN recovers sharp lines, while others all give blurry results. (2) The second row shows image "253027" (BSD100 with scale factor ×3). DRRN accurately recovers the pattern. (3) The last row shows image "ppt3" (Set14 with scale factor ×4). Texts in DRRN are sharp, while others are blurry.

the results of [2, 10, 20, 23] are cited from [20] [2], while the results of VDSR come from our re-implementation. Similar to DRRN, the VDSR re-implementation also uses BN and ReLU as the activation functions, unlike the original VDSR [13] that does not use BN. These results are faithful since our VDSR re-implementation achieves similar benchmark performance as [13] reported in Tab. 2. Since only Set5, Set14 and Urban100 are used in [20], we omit BSD100 in this test. It is clear that DRRN still outperforms all existing methods in all datasets and scale factors. Regarding the speed, our 20-layer B1U9 network takes 0.25 second to process a 288 × 288 image on a Titan X GPU.

[2] Since PSyCo [20] does not present complete PSNR/SSIM performance on the four benchmarks, we do not include it in Tab. 2.

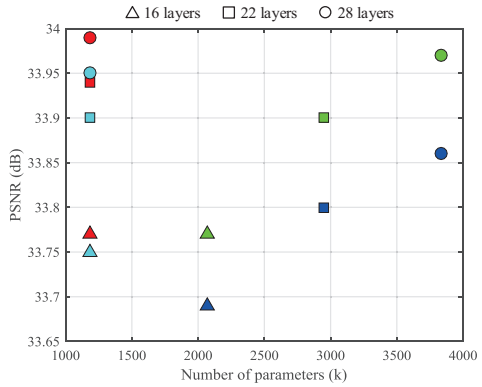


Figure 8. PSNR for scale factor $\times 3$ on Set5 using VDSR (blue), DRRN_NS (green), DRRN_C (cyan) and DRRN (red).

Methods	VDSR	DRRN_NS_C	DRRN_NS	DRRN_C	DRRN
Loc. Res. L	×	√	√	√	√
Recu. L	×	×	×	√	√
Multi-path	×	×	√	×	√
PSNR	33.86	33.92	33.97	33.95	33.99

Table 4. Average PSNR when different DRRN components are turned on or off, for scale factor $\times 3$ on dataset Set5.

Qualitative comparisons among SRCNN [2], SelfEx [10], VDSR [13] and DRRN are illustrated in Fig. 7. For SRCNN and SelfEx, we use their public codes. For VDSR, we use our re-implementation. As we can see, our method produces relatively sharper edges with respect to patterns, while other methods may give blurry results.

4.5. Discussions

Since global residual learning has been well discussed in [13], in this section, we mainly focus on local residual learning (LRL), recursive learning and multi-path structure.

Local Residual Learning To demonstrate the effectiveness of LRL, DRRN is compared with VDSR [13], which has no LRL. For fair comparison, the depth and number of parameters are kept the same for both methods. Specifically, we evaluate three depths: 16(B3U2), 22(B3U3), and 28(B3U4) convolutional layers. Each convolutional layer has 128 filters with the size 3×3. To keep the parameter number the same, in this test we do not share the weight set of the residual units in one recursive block, and denote this DRRN structure as DRRN_NS. Fig. 8 shows the PSNR of both methods in different depths. We see that the LRL strategy consistently improves VDSR at all depths.

Recursive Learning To contrast our recursive learning strategy, the three DRRN_NS versions are compared with the three weight-shared versions (Fig. 8). Storage is an important factor to consider when building a deep model. The recursive learning strategy can reduce the storage demand and keep a concise model while increasing its depth. Interestingly the weight-shared DRRN versions achieve comparable or even better performance than DRRN_NS versions

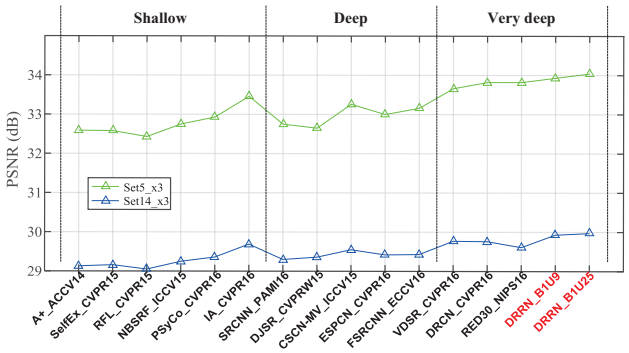


Figure 9. Comparing deep and shallow models proposed in recent three years that report PSNR for scale factor $\times 3$ on Set5 and Set14.

while using only *a small fraction of parameters*, which indicates when limited training set (e.g., 291 images) is used, the recursive learning is indeed effective under the same structure, and less prone to overfitting [16].

Multi-Path Structure To demonstrate the effectiveness of multi-path structure, we compare DRRN with the chain structure, denoted as DRRN_C. As shown in Fig. 8, with the same depth and parameter number, the multi-path structures achieve higher PSNR than the corresponding chain structures in all three cases. Further, Tab. 4 presents the comprehensive study on performance gains using network B3U4 as the example. It shows how *different* technical parts improve the performance compared to the baseline VDSR.

Deep vs. Shallow Finally, we give a comparison of the deep and shallow SISR models, which are published in recent three years (2014 to 2016) that report PSNR for scale factor $\times 3$ on datasets Set5 and Set14. Shallow (non-DL) models include A+ [31], SelfEx [10], RFL [23], NBSRF [22], PSyCo [20] and IA [30]. The deep models ($d \leq 8$) include SRCNN [2], DJSR [33], CSCN [32], ESPCN [25] and FSRCNN [3]. Very deep models ($d \geq 20$) include VDSR [13], DRCN [14], RED [17] and DRRN with $d = 20$ and 52. Fig. 9 shows that 1) very deep models significantly outperform the shallow models; 2) DRRN_B1U9 ($d = 20$, $k = 297$K) already outperforms the state of the arts with the same depth but fewer parameters; 3) a deeper DRRN_B1U25 ($d = 52$, $k = 297$K) further improves the performance without adding any parameters.

5. Conclusions

In this paper, we propose Deep Recursive Residual Network (DRRN) for single image super-resolution. In DRRN, an enhanced residual unit structure is recursively learned in a recursive block, and we stack several recursive blocks to learn the residual image between the HR and LR images. The residual image is then added to the input LR image from a global identity branch to estimate the HR image. Extensive benchmark experiments and analysis show that DRRN is a deep, concise, and superior model for SISR.

References

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

[2] C. Dong, 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. 1, 2, 3, 6, 7, 8

[3] C. Dong, C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016. 8

[4] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009. 1

[5] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, 2010. 1

[6] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015. 5

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 4

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *arXiv:1603.05027v2*, 2016. 4

[10] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 5, 6, 7, 8

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2, 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. *arXiv:1408.5093*, 2014. 5

[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, 5, 6, 7, 8

[14] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016. 1, 2, 3, 4, 6, 8

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998. 5

[16] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *CVPR*, 2015. 4, 8

[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, 2, 8

[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 *ICCV*, 2001. 5

[19] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 2, 4

[20] E. Perez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn. PSyCo: Manifold span reduction for super resolution. In *CVPR*, 2016. 1, 6, 7, 8

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and et al. ImageNet large scale visual recognition challenge. *arXiv:1409.0575*, 2014. 2

[22] J. Salvador and E. Perez-Pellitero. Naive bayes super-resolution forest. In *ICCV*, 2015. 1, 8

[23] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *CVPR*, 2015. 1, 5, 6, 7, 8

[24] H. Sheikh, A. Bovik, and G. de Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Transactions on image processing*, 14(12):2117–2128, 2005. 6

[25] 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, 8

[26] W. Shi, J. Caballero, C. Ledig, X. Zhuang, W. Bai, K. Bhatia, A. Marvao, T. Dawes, D. ORegan, and D. Rueckert. Cardiac image super-resolution with global correspondence using multi-atlas patchmatch. In *MICCAI*, 2013. 1

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, and S. Reed. Going deeper with convolutions. In *CVPR*, 2015. 2

[29] M. W. Thornton, P. M. Atkinson, and D. a. Holland. Sub-pixel mapping of rural land cover objects from fine spatial resolution satellite sensor imagery using super-resolution pixel-swapping. *International Journal of Remote Sensing*, 27(3):473–491, 2006. 1

[30] R. Timofte, R. Rothe, and L. V. Gool. Seven ways to improve example-based single image super resolution. In *CVPR*, 2016. 5, 8

[31] R. Timofte, V. D. Smet, and L. V. Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014. 1, 8

[32] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *ICCV*, 2015. 1, 2, 8

[33] Z. Wang, Y. Yang, Z. Wang, S. Chang, W. Han, J. Yang, and T. Huang. Self-tuned deep super resolution. In *CVPR workshop*, 2015. 8

[34] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In *ECCV*, 2014. 6

[35] J. Yang, J. Wright, T. Huang, and Y. Ma. Image superresolution via sparse representation. *IEEE Transactions on image processing*, 19(11):2861–2873, 2010. 1, 5

[36] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. *Curves and Surfaces*, pages 711–730, 2012. 5

[37] W. Zou and P. C. Yuen. Very low resolution face recognition problem. *IEEE Transactions on image processing*, 21(1):327–340, 2012. 1