Accurate Image Super-Resolution Using Very Deep Convolutional Networks

Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee
Department of ECE, ASRI, Seoul National University, Korea
{j.kim, deruci, kyoungmu}@snu.ac.kr

Abstract

We present a highly accurate single-image super-resolution (SR) method. Our method uses a very deep convolutional network inspired by VGG-net used for ImageNet classification [19]. We find increasing our network depth shows a significant improvement in accuracy. Our final model uses 20 weight layers. By cascading small filters many times in a deep network structure, contextual information over large image regions is exploited in an efficient way. With very deep networks, however, convergence speed becomes a critical issue during training. We propose a simple yet effective training procedure. We learn residuals only and use extremely high learning rates (10^4 times higher than SRCNN [6]) enabled by adjustable gradient clipping. Our proposed method performs better than existing methods in accuracy and visual improvements in our results are easily noticeable.

1. Introduction

We address the problem of generating a high-resolution (HR) image given a low-resolution (LR) image, commonly referred as single image super-resolution (SISR) [12], [8], [9]. SISR is widely used in computer vision applications ranging from security and surveillance imaging to medical imaging where more image details are required on demand.

Many SISR methods have been studied in the computer vision community. Early methods include interpolation such as bicubic interpolation and Lanczos resampling [7] more powerful methods utilizing statistical image priors [20, 13] or internal patch recurrence [9].

Currently, learning methods are widely used to model a mapping from LR to HR patches. Neighbor embedding [4, 15] methods interpolate the patch subspace. Sparse coding [25, 26, 21, 22] methods use a learned compact dictionary based on sparse signal representation. Lately, random forest [18] and convolutional neural network (CNN) [6] have also been used with large improvements in accuracy.

Among them, Dong et al. [6] has demonstrated that a CNN can be used to learn a mapping from LR to HR in an end-to-end manner. Their method, termed SRCNN, does not require any engineered features that are typically necessary in other methods [25, 26, 21, 22] and shows the state-of-the-art performance.

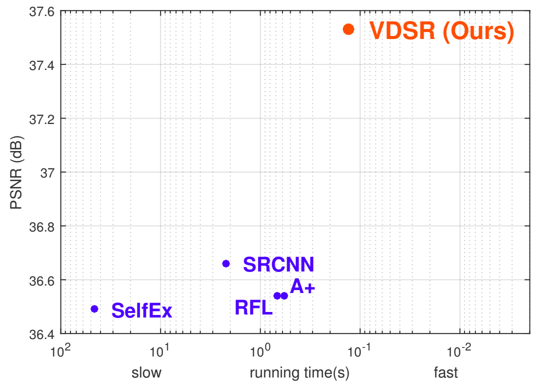


Figure 1: Our VDSR improves PSNR for scale factor $\times 2$ on dataset Set5 in comparison to the state-of-the-art methods (SRCNN uses the public slower implementation using CPU). VDSR outperforms SRCNN by a large margin (0.87 dB).

While SRCNN successfully introduced a deep learning technique into the super-resolution (SR) problem, we find its limitations in three aspects: first, it relies on the context of small image regions; second, training converges too slowly; third, the network only works for a single scale.

In this work, we propose a new method to practically resolve the issues.

Context We utilize contextual information spread over very large image regions. For a large scale factor, it is often the case that information contained in a small patch is not sufficient for detail recovery (ill-posed). Our very deep network using large receptive field takes a large image context into account.

Convergence We suggest a way to speed-up the training: residual-learning CNN and extremely high learning rates. As LR image and HR image share the same information to a large extent, explicitly modelling the residual image, which is the difference between HR and LR images, is advantageous. We propose a network structure for efficient learning when input and output are highly correlated. Moreover, our initial learning rate is 10^4 times higher than that of SRCNN [6]. This is enabled by residual-learning and gradient clipping.

Scale Factor We propose a single-model SR approach. Scales are typically user-specified and can be arbitrary including fractions. For example, one might need smooth zoom-in in an image viewer or resizing to a specific dimension. Training and storing many scale-dependent models in preparation for all possible scenarios is impractical. We find a single convolutional network is sufficient for multi-scale-factor super-resolution.

Contribution In summary, in this work, we propose a highly accurate SR method based on a very deep convolutional network. Very deep networks converge too slowly if small learning rates are used. Boosting convergence rate with high learning rates lead to exploding gradients and we resolve the issue with residual-learning and gradient clipping. In addition, we extend our work to cope with multi-scale SR problem in a single network. Our method is relatively accurate and fast in comparison to state-of-the-art methods as illustrated in Figure 1.

2. Related Work

SRCNN is a representative state-of-art method for deep learning-based SR approach. So, let us analyze and compare it with our proposed method.

2.1. Convolutional Network for Image Super-Resolution

Model SRCNN consists of three layers: patch extraction/representation, non-linear mapping and reconstruction. Filters of spatial sizes 9×9, 1×1, and 5×5 were used respectively.

In [6], Dong et al. attempted to prepare deeper models, but failed to observe superior performance after a week of training. In some cases, deeper models gave inferior performance. They conclude that deeper networks do not result in better performance (Figure 9).

However, we argue that increasing depth significantly boosts performance. We successfully use 20 weight layers (3×3 for each layer). Our network is very deep (20 vs. 3 [6]) and information used for reconstruction (receptive field) is much larger (41×41 vs. 13×13).

Training For training, SRCNN directly models high-resolution images. A high-resolution image can be decomposed into a low frequency information (corresponding to low-resolution image) and high frequency information (residual image or image details). Input and output images share the same low-frequency information. This indicates that SRCNN serves two purposes: carrying the input to the end layer and reconstructing residuals. Carrying the input to the end is conceptually similar to what an auto-encoder does. Training time might be spent on learning this auto-encoder so that the convergence rate of learning the other part (image details) is significantly decreased. In contrast, since our network models the residual images directly, we can have much faster convergence with even better accuracy.

Scale As in most existing SR methods, SRCNN is trained for a single scale factor and is supposed to work only with the specified scale. Thus, if a new scale is on demand, a new model has to be trained. To cope with multiple scale SR (possibly including fractional factors), we need to construct individual single scale SR system for each scale of interest.

However, preparing many individual machines for all possible scenarios to cope with multiple scales is inefficient and impractical. In this work, we design and train a single network to handle multiple scale SR problem efficiently. This turns out to work very well. Our single machine is compared favorably to a single-scale expert for the given sub-task. For three scales factors ($\times 2, 3, 4$), we can reduce the number of parameters by three-fold.

In addition to the aforementioned issues, there are some minor differences. Our output image has the same size as the input image by padding zeros every layer during training whereas output from SRCNN is smaller than the input. Finally, we simply use the same learning rates for all layers while SRCNN uses different learning rates for different layers in order to achieve stable convergence.

3. Proposed Method

3.1. Proposed Network

For SR image reconstruction, we use a very deep convolutional network inspired by Simonyan and Zisserman [19]. The configuration is outlined in Figure 2. We use d layers where layers except the first and the last are of the same type: 64 filter of the size $3 \times 3 \times 64$, where a filter operates on 3×3 spatial region across 64 channels (feature maps). The first layer operates on the input image. The last layer, used for image reconstruction, consists of a single filter of size $3 \times 3 \times 64$.

The network takes an interpolated low-resolution image (to the desired size) as input and predicts image details. Modelling image details is often used in super-resolution methods [21, 22, 15, 3] and we find that CNN-based methods can benefit from this domain-specific knowledge.

In this work, we demonstrate that explicitly modelling image details (residuals) has several advantages. These are further discussed later in Section 4.2.

One problem with using a very deep network to predict dense outputs is that the size of the feature map gets reduced every time convolution operations are applied. For example, when an input of size $(n+1) \times (n+1)$ is applied to a network

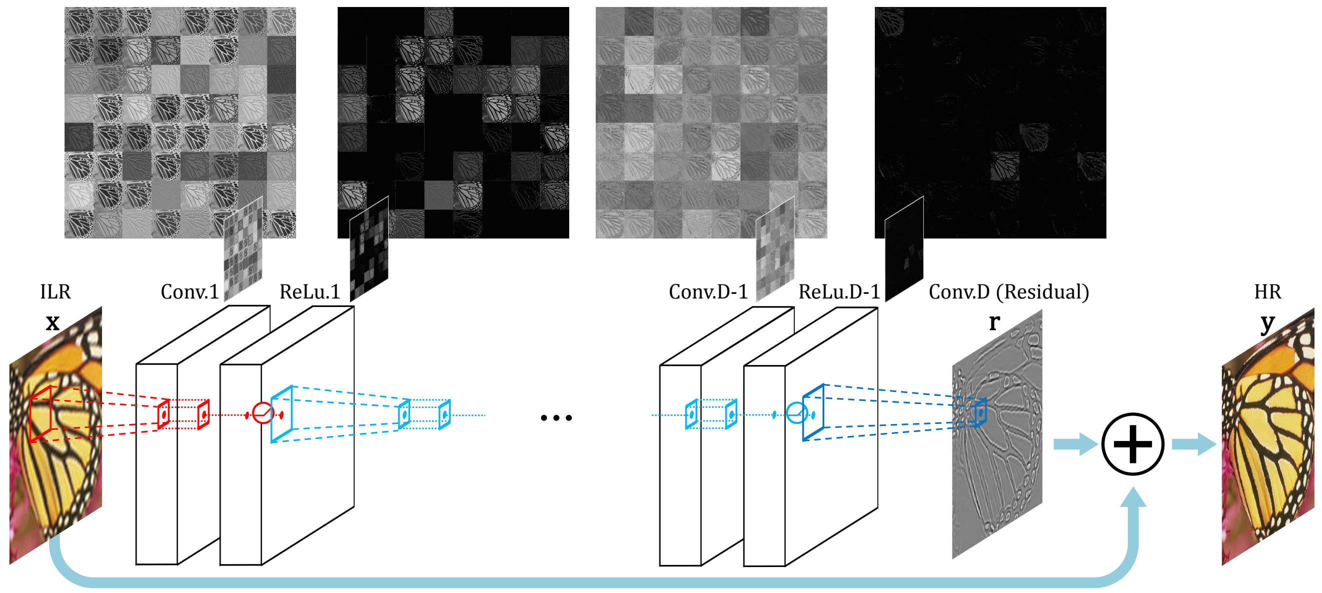


Figure 2: Our Network Structure. We cascade a pair of layers (convolutional and nonlinear) repeatedly. An interpolated low-resolution (ILR) image goes through layers and transforms into a high-resolution (HR) image. The network predicts a residual image and the addition of ILR and the residual gives the desired output. We use 64 filters for each convolutional layer and some sample feature maps are drawn for visualization. Most features after applying rectified linear units (ReLu) are zero.

with receptive field size $n \times n$, the output image is 1×1.

This is in accordance with other super-resolution methods since many require surrounding pixels to infer center pixels correctly. This center-surround relation is useful since the surrounding region provides more constraints to this ill-posed problem (SR). For pixels near the image boundary, this relation cannot be exploited to the full extent and many SR methods crop the result image.

This methodology, however, is not valid if the required surround region is very big. After cropping, the final image is too small to be visually pleasing.

To resolve this issue, we pad zeros before convolutions to keep the sizes of all feature maps (including the output image) the same. It turns out that zero-padding works surprisingly well. For this reason, our method differs from most other methods in the sense that pixels near the image boundary are also correctly predicted.

Once image details are predicted, they are added back to the input ILR image to give the final image (HR). We use this structure for all experiments in our work.

3.2. Training

We now describe the objective to minimize in order to find optimal parameters of our model. Let $\mathbf{x}$ denote an interpolated low-resolution image and $\mathbf{y}$ a high-resolution image. Given a training dataset $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, our goal is to learn a model f that predicts values $\hat{\mathbf{y}} = f(\mathbf{x})$, where $\hat{\mathbf{y}}$ is an estimate of the target HR image. We minimize the mean squared error $\frac{1}{2}||\mathbf{y} - f(\mathbf{x})||^2$ averaged over the training set is minimized.

Residual-Learning In SRCNN, the network must preserve all input detail since the image is discarded and the output is generated from the learned features alone. With many weight layers, this becomes an end-to-end relation requiring very long-term memory. For this reason, the vanishing/exploding gradients problem [2] can be critical. We can solve this problem simply with residual-learning.

As the input and output images are largely similar, we define a residual image $\mathbf{r} = \mathbf{y} - \mathbf{x}$, where most values are likely to be zero or small. We want to predict this residual image. The loss function now becomes $\frac{1}{2}||\mathbf{r} - f(\mathbf{x})||^2$, where $f(\mathbf{x})$ is the network prediction.

In networks, this is reflected in the loss layer as follows. Our loss layer takes three inputs: residual estimate, network input (ILR image) and ground truth HR image. The loss is computed as the Euclidean distance between the reconstructed image (the sum of network input and output) and ground truth.

Training is carried out by optimizing the regression objective using mini-batch gradient descent based on backpropagation (LeCun et al. [14]). We set the momentum parameter to 0.9. The training is regularized by weight decay (L_2 penalty multiplied by 0.0001).

High Learning Rates for Very Deep Networks Training deep models can fail to converge in realistic limit of time. SRCNN [6] fails to show superior performance with more than three weight layers. While there can be various reasons, one possibility is that they stopped their training procedure before networks converged. Their learning rate 10^{-5} is too small for a network to converge within a week on a common GPU. Looking at Fig. 9 of [6], it is not easy to say their deeper networks have converged and their performances were saturated. While more training will eventually resolve the issue, but increasing depth to 20 does not seems practical with SRCNN.

It is a basic rule of thumb to make learning rate high to boost training. But simply setting learning rate high can also lead to vanishing/exploding gradients [2]. For the reason, we suggest an adjustable gradient clipping for maximal boost in speed while suppressing exploding gradients.

Adjustable Gradient Clipping Gradient clipping is a technique that is often used in training recurrent neural networks [17]. But, to our knowledge, its usage is limited in training CNNs. While there exist many ways to limit gradients, one of the common strategies is to clip individual gradients to the predefined range $[-\theta, \theta]$.

With clipping, gradients are in a certain range. With stochastic gradient descent commonly used for training, learning rate is multiplied to adjust the step size. If high learning rate is used, it is likely that θ is tuned to be small to avoid exploding gradients in a high learning rate regime. But as learning rate is annealed to get smaller, the effective gradient (gradient multiplied by learning rate) approaches zero and training can take exponentially many iterations to converge if learning rate is decreased geometrically.

For maximal speed of convergence, we clip the gradients to $[-\frac{\theta}{\gamma}, \frac{\theta}{\gamma}]$, where γ denotes the current learning rate. We find the adjustable gradient clipping makes our convergence procedure extremely fast. Our 20-layer network training is done within 4 hours whereas 3-layer SRCNN takes several days to train.

Multi-Scale While very deep models can boost performance, more parameters are now needed to define a network. Typically, one network is created for each scale factor. Considering that fractional scale factors are often used, we need an economical way to store and retrieve networks.

For this reason, we also train a multi-scale model. With this approach, parameters are shared across all predefined scale factors. Training a multi-scale model is straightforward. Training datasets for several specified scales are combined into one big dataset.

Data preparation is similar to SRCNN [5] with some differences. Input patch size is now equal to the size of the receptive field and images are divided into sub-images with no overlap. A mini-batch consists of 64 sub-images, where sub-images from different scales can be in the same batch.

We implement our model using the *MatConvNet*[1] package [23].

[1]http://www.vlfeat.org/matconvnet/

Epoch	10	20	40	80
Residual	36.90	36.64	37.12	37.05
Non-Residual	27.42	19.59	31.38	35.66
Difference	9.48	17.05	5.74	1.39

(a) Initial learning rate 0.1

Epoch	10	20	40	80
Residual	36.74	36.87	36.91	36.93
Non-Residual	30.33	33.59	36.26	36.42
Difference	6.41	3.28	0.65	0.52

(b) Initial learning rate 0.01

Epoch	10	20	40	80
Residual	36.31	36.46	36.52	36.52
Non-Residual	33.97	35.08	36.11	36.11
Difference	2.35	1.38	0.42	0.40

(c) Initial learning rate 0.001

Table 1: Performance table (PSNR) for residual and non-residual networks ('Set5' dataset, $\times 2$). Residual networks rapidly approach their convergence within 10 epochs.

4. Understanding Properties

In this section, we study three properties of our proposed method. First, we show that large depth is necessary for the task of SR. A very deep network utilizes more contextual information in an image and models complex functions with many nonlinear layers. We experimentally verify that deeper networks give better performances than shallow ones.

Second, we show that our residual-learning network converges much faster than the standard CNN. Moreover, our network gives a significant boost in performance.

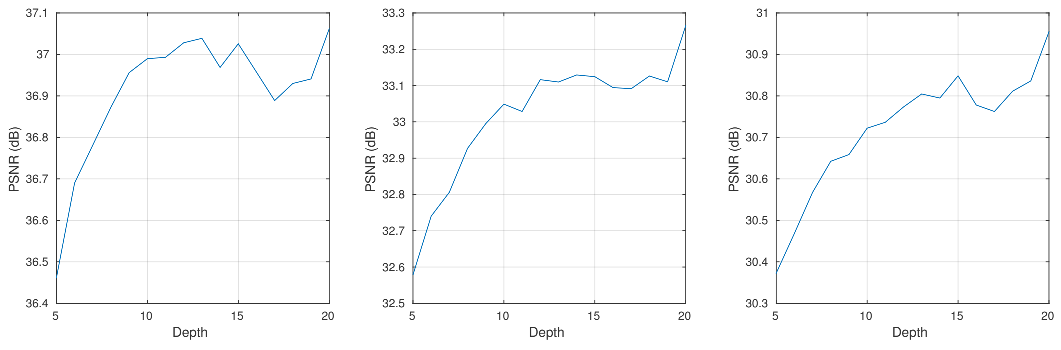
Third, we show that our method with a single network performs as well as a method using multiple networks trained for each scale. We can effectively reduce model capacity (the number of parameters) of multi-network approaches.

4.1. The Deeper, the Better

Convolutional neural networks exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers [1]. In other words, hidden units in layer m take as input a subset of units in layer $m-1$. They form spatially contiguous receptive fields.

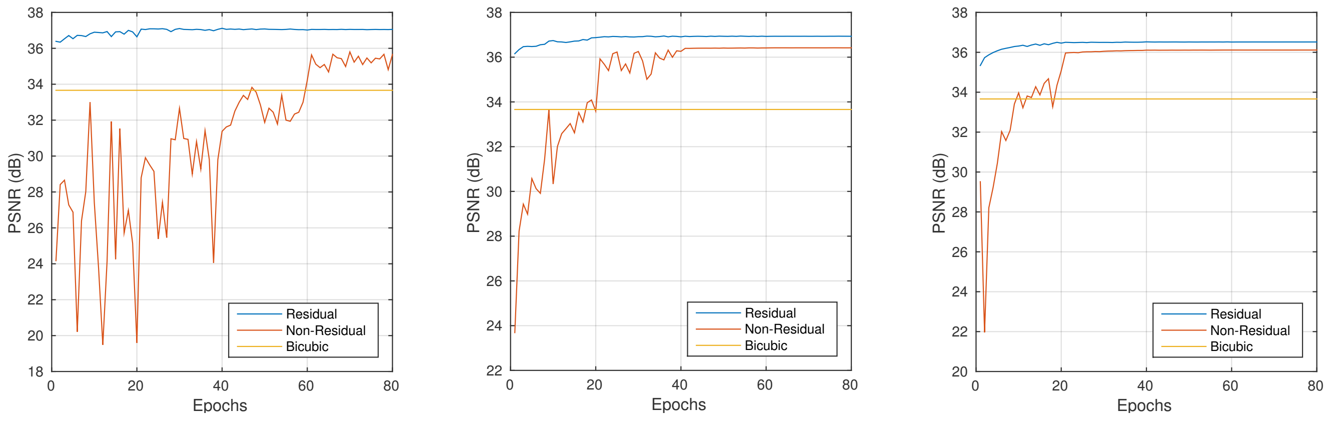
Each hidden unit is unresponsive to variations outside of the receptive field with respect to the input. The architecture thus ensures that the learned filters produce the strongest response to a spatially local input pattern.

However, stacking many such layers leads to filters that become increasingly global (i.e. responsive to a larger region of pixel space). In other words, a filter of very large support can be effectively decomposed into a series of small



(a) Test Scale Factor 2 **(b)** Test Scale Factor 3 **(c)** Test Scale Factor 4

Figure 3: Depth vs Performance



(a) Initial learning rate 0.1 **(b)** Initial learning rate 0.01 **(c)** Initial learning rate 0.001

Figure 4: Performance curve for residual and non-residual networks. Two networks are tested under 'Set5' dataset with scale factor 2. Residual networks quickly reach state-of-the-art performance within a few epochs, whereas non-residual networks (which models high-resolution image directly) take many epochs to reach maximum performance. Moreover, the final accuracy is higher for residual networks.

filters.

In this work, we use filters of the same size, 3×3, for all layers. For the first layer, the receptive field is of size 3×3. For the next layers, the size of the receptive field increases by 2 in both height and width. For depth D network, the receptive field has size $(2D + 1) \times (2D + 1)$. Its size is proportional to the depth.

In the task of SR, this corresponds to the amount of contextual information that can be exploited to infer high-frequency components. A large receptive field means the network can use more context to predict image details. As SR is an ill-posed inverse problem, collecting and analyzing more neighbor pixels give more clues. For example, if there are some image patterns entirely contained in a receptive field, it is plausible that this pattern is recognized and used to super-resolve the image.

In addition, very deep networks can exploit high nonlinearities. We use 19 rectified linear units and our networks can model very complex functions with moderate number of channels (neurons). The advantages of making a thin deep network is well explained in Simonyan and Zisserman [19].

We now experimentally show that very deep networks significantly improve SR performance. We train and test networks of depth ranging from 5 to 20 (only counting weight layers excluding nonlinearity layers). In Figure 3, we show the results. In most cases, performance increases as depth increases. As depth increases, performance improves rapidly.

4.2. Residual-Learning

As we already have a low-resolution image as the input, predicting high-frequency components is enough for the purpose of SR. Although the concept of predicting residuals has been used in previous methods [21, 22, 26], it has not been studied in the context of deep-learning-based SR framework.

In this work, we have proposed a network structure that learns residual images. We now study the effect of this modification to a standard CNN structure in detail.

First, we find that this residual network converges much faster. Two networks are compared experimentally: the

Test / Train	×2	×3	×4	×2,3	×2,4	×3,4	×2,3,4	Bicubic
×2	37.10	30.05	28.13	37.09	37.03	32.43	37.06	33.66
×3	30.42	32.89	30.50	33.22	31.20	33.24	33.27	30.39
×4	28.43	28.73	30.84	28.70	30.86	30.94	30.95	28.42

Table 2: Scale Factor Experiment. Several models are trained with different scale sets. Quantitative evaluation (PSNR) on dataset 'Set5' is provided for scale factors 2,3 and 4. Red color indicates that test scale is included during training. Models trained with multiple scales perform well on the trained scales.

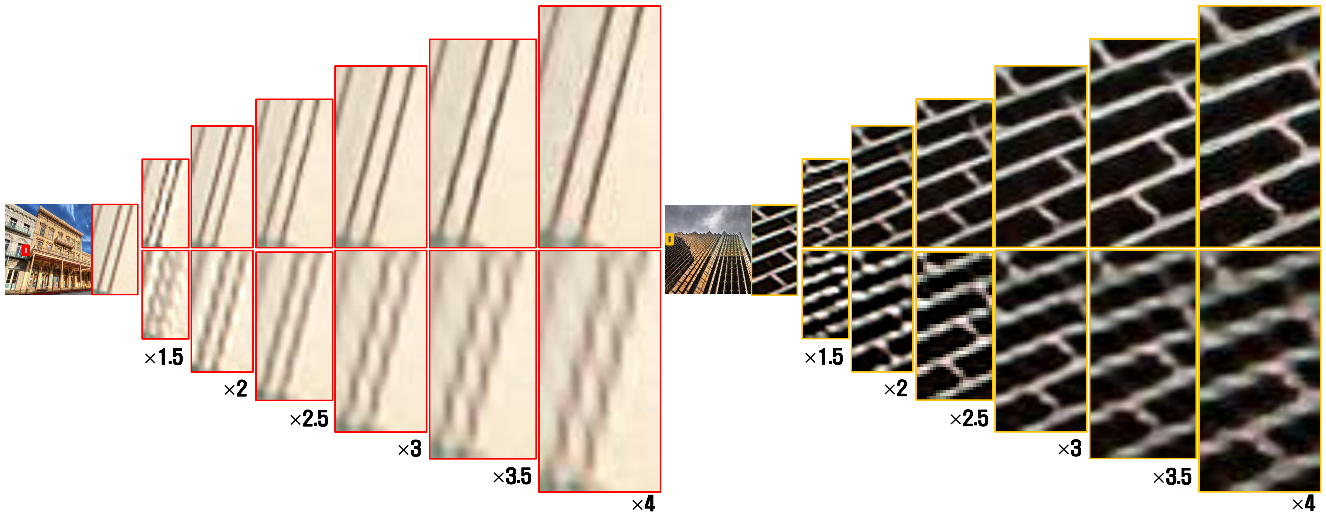


Figure 5: (Top) Our results using a single network for all scale factors. Super-resolved images over all scales are clean and sharp. (Bottom) Results of Dong et al. [5] (×3 model used for all scales). Result images are not visually pleasing. To handle multiple scales, existing methods require multiple networks.

residual network and the standard non-residual network. We use depth 10 (weight layers) and scale factor 2. Performance curves for various learning rates are shown in Figure 4. All use the same learning rate scheduling mechanism that has been mentioned above.

Second, at convergence, the residual network shows superior performance. In Figure 4, residual networks give higher PSNR when training is done.

Another remark is that if small learning rates are used, networks do not converge in the given number of epochs. If initial learning rate 0.1 is used, PSNR of a residual-learning network reaches 36.90 within 10 epochs. But if 0.001 is used instead, the network never reaches the same level of performance (its performance is 36.52 after 80 epochs). In a similar manner, residual and non-residual networks show dramatic performance gaps after 10 epochs (36.90 vs. 27.42 for rate 0.1).

In short, this simple modification to a standard non-residual network structure is very powerful and one can explore the validity of the idea in other image restoration problems where input and output images are highly correlated.

4.3. Single Model for Multiple Scales

Scale augmentation during training is a key technique to equip a network with super-resolution machines of multiple scales. Many SR processes for different scales can be executed with our multi-scale machine with much smaller capacity than that of single-scale machines combined.

We start with an interesting experiment as follows: we train our network with a single scale factor s_{train} and it is tested under another scale factor s_{test}. Here, factors 2,3 and 4 that are widely used in SR comparisons are considered. Possible pairs (s_{train},s_{test}) are tried for the dataset 'Set5' [15]. Experimental results are summarized in Table 2.

Performance is degraded if $s_{\text{train}} \neq s_{\text{test}}$. For scale factor 2, the model trained with factor 2 gives PSNR of 37.10 (in dB), whereas models trained with factor 3 and 4 give 30.05 and 28.13, respectively. A network trained over single-scale data is not capable of handling other scales. In many tests, it is even worse than bicubic interpolation, the method used for generating the input image.

We now test if a model trained with scale augmentation is capable of performing SR at multiple scale factors. The same network used above is trained with multiple scale factors $s_{\text{train}} = \{2, 3, 4\}$. In addition, we experiment with the cases $s_{\text{train}} = \{2, 3\}, \{2, 4\}, \{3, 4\}$ for more comparisons.

We observe that the network copes with any scale used during training. When $s_{\text{train}} = \{2, 3, 4\}$ ($\times 2, 3, 4$ in Table 2), its PSNR for each scale is comparable to those achieved from the corresponding result of single-scale net-

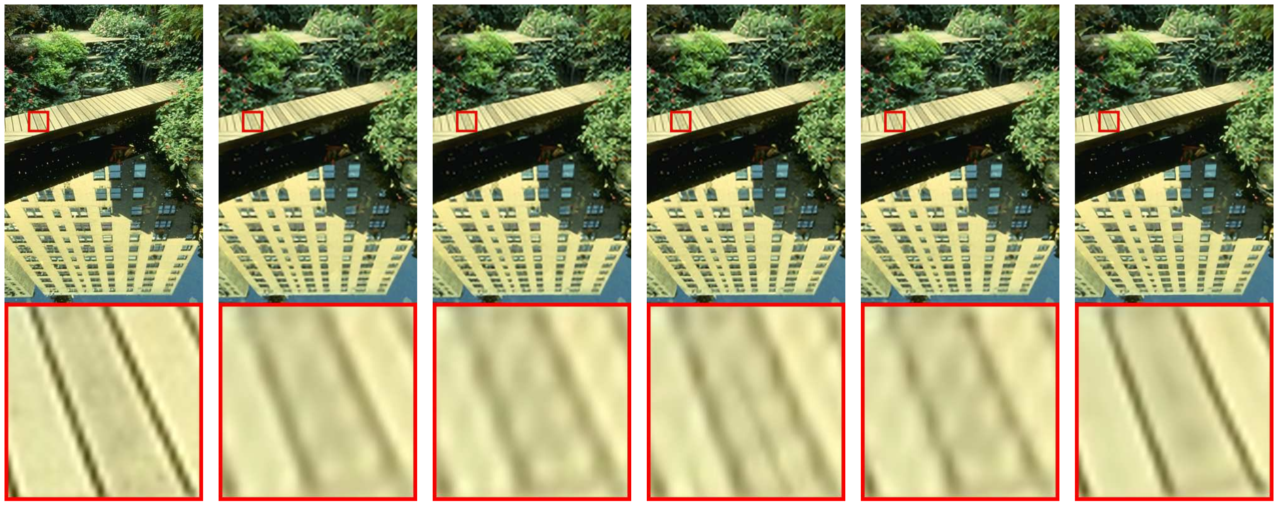

Ground Truth (PSNR, SSIM) | A+ [22] (22.92, 0.7379) | RFL [18] (22.90, 0.7332) | SelfEx [11] (23.00, 0.7439) | SRCNN [5] (23.15, 0.7487) | VDSR (Ours) (23.50, 0.7777)

Figure 6: Super-resolution results of "148026" (*B100*) with scale factor ×3. VDSR recovers sharp lines.

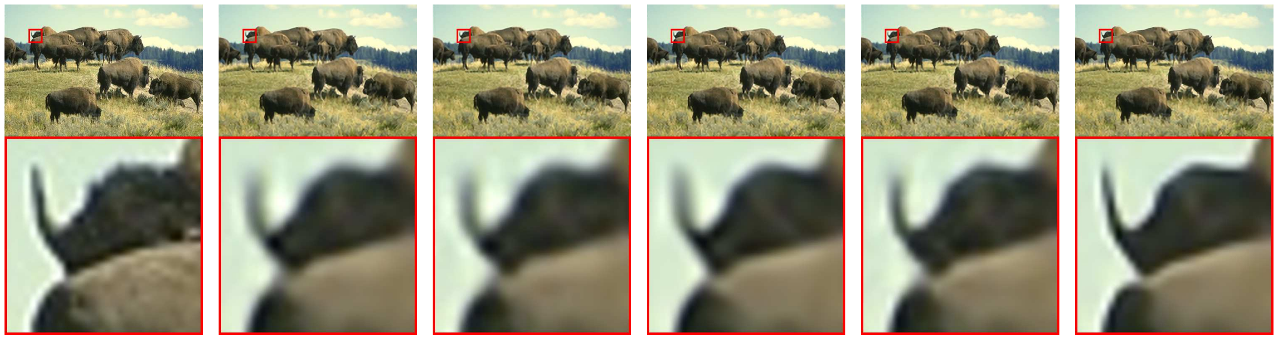

Ground Truth (PSNR, SSIM) | A+ [22] (27.08, 0.7514) | RFL [18] (27.08, 0.7508) | SelfEx [11] (27.02, 0.7513) | SRCNN [5] (27.16, 0.7545) | VDSR (Ours) (27.32, 0.7606)

Figure 7: Super-resolution results of "38092" (*B100*) with scale factor ×3. The horn in the image is sharp in the result of VDSR.

Dataset	Scale	Bicubic PSNR/SSIM/time	A+ [22] PSNR/SSIM/time	RFL [18] PSNR/SSIM/time	SelfEx [11] PSNR/SSIM/time	SRCNN [5] PSNR/SSIM/time	VDSR (Ours) PSNR/SSIM/time
Set5	×2	33.66/0.9299/0.00	36.54/0.9544/0.58	36.54/0.9537/0.63	36.49/0.9537/45.78	36.66/0.9542/2.19	37.53/0.9587/0.13
	×3	30.39/0.8682/0.00	32.58/0.9088/0.32	32.43/0.9057/0.49	32.58/0.9093/33.44	32.75/0.9090/2.23	33.66/0.9213/0.13
	×4	28.42/0.8104/0.00	30.28/0.8603/0.24	30.14/0.8548/0.38	30.31/0.8619/29.18	30.48/0.8628/2.19	31.35/0.8838/0.12
Set14	×2	30.24/0.8688/0.00	32.28/0.9056/0.86	32.26/0.9040/1.13	32.22/0.9034/105.00	32.42/0.9063/4.32	33.03/0.9124/0.25
	×3	27.55/0.7742/0.00	29.13/0.8188/0.56	29.05/0.8164/0.85	29.16/0.8196/74.69	29.28/0.8209/4.40	29.77/0.8314/0.26
	×4	26.00/0.7027/0.00	27.32/0.7491/0.38	27.24/0.7451/0.65	27.40/0.7518/65.08	27.49/0.7503/4.39	28.01/0.7674/0.25
B100	×2	29.56/0.8431/0.00	31.21/0.8863/0.59	31.16/0.8840/0.80	31.18/0.8855/60.09	31.36/0.8879/2.51	31.90/0.8960/0.16
	×3	27.21/0.7385/0.00	28.29/0.7835/0.33	28.22/0.7806/0.62	28.29/0.7840/40.01	28.41/0.7863/2.58	28.82/0.7976/0.21
	×4	25.96/0.6675/0.00	26.82/0.7087/0.26	26.75/0.7054/0.48	26.84/0.7106/35.87	26.90/0.7101/2.51	27.29/0.7251/0.21
Urban100	×2	26.88/0.8403/0.00	29.20/0.8938/2.96	29.11/0.8904/3.62	29.54/0.8967/663.98	29.50/0.8946/22.12	30.76/0.9140/0.98
	×3	24.46/0.7349/0.00	26.03/0.7973/1.67	25.86/0.7900/2.48	26.44/0.8088/473.60	26.24/0.7989/19.35	27.14/0.8279/1.08
	×4	23.14/0.6577/0.00	24.32/0.7183/1.21	24.19/0.7096/1.88	24.79/0.7374/394.40	24.52/0.7221/18.46	25.18/0.7524/1.06

Table 3: Average PSNR/SSIM for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, B100 and Urban100. Red color indicates the best performance and blue color indicates the second best performance.

work: 37.06 vs. 37.10 ($\times 2$), 33.27 vs. 32.89 ($\times 3$), 30.95 vs. 30.86 ($\times 4$).

Another pattern is that for large scales ($\times 3, 4$), our multi-scale network outperforms single-scale network: our model ($\times 2, 3$), ($\times 3, 4$) and ($\times 2, 3, 4$) give PSNRs 33.22, 33.24 and 33.27 for test scale 3, respectively, whereas ($\times 3$) gives 32.89. Similarly, ($\times 2, 4$), ($\times 3, 4$) and ($\times 2, 3, 4$) give 30.86, 30.94 and 30.95 (vs. 30.84 by $\times 4$ model), respectively. From this, we observe that training multiple scales boosts the performance for large scales.

5. Experimental Results

In this section, we evaluate the performance of our method on several datasets. We first describe datasets used for training and testing our method. Next, parameters necessary for training are given.

After outlining our experimental setup, we compare our method with several state-of-the-art SISR methods.

5.1. Datasets for Training and Testing

Training dataset Different learning-based methods use different training images. For example, RFL [18] has two methods, where the first one uses 91 images from Yang et al. [25] and the second one uses 291 images with the addition of 200 images from Berkeley Segmentation Dataset [16]. SRCNN [6] uses a very large ImageNet dataset.

We use 291 images as in [18] for benchmark with other methods in this section. In addition, data augmentation (rotation or flip) is used. For results in previous sections, we used 91 images to train network fast, so performances can be slightly different.

Test dataset For benchmark, we use four datasets. Datasets 'Set5' [15] and 'Set14' [26] are often used for benchmark in other works [22, 21, 5]. Dataset 'Urban100', a dataset of urban images recently provided by Huang et al. [11], is very interesting as it contains many challenging images failed by many of the existing methods. Finally, dataset 'B100', natural images in the Berkeley Segmentation Dataset used in Timofte et al. [22] and Yang and Yang [24] for benchmark, is also employed.

5.2. Training Parameters

We provide parameters used to train our final model. We use a network of depth 20. Training uses batches of size 64. Momentum and weight decay parameters are set to 0.9 and 0.0001, respectively.

For weight initialization, we use the method described in He et al. [10]. This is a theoretically sound procedure for networks utilizing rectified linear units (ReLu).

We train all experiments over 80 epochs (9960 iterations with batch size 64). Learning rate was initially set to 0.1 and then decreased by a factor of 10 every 20 epochs. In total, the learning rate was decreased 3 times, and the learning is stopped after 80 epochs. Training takes roughly 4 hours on GPU Titan Z.

5.3. Benchmark

For benchmark, we follow the publicly available framework of Huang et al. [21]. It enables the comparison of many state-of-the-art results with the same evaluation procedure.

The framework applies bicubic interpolation to color components of an image and sophisticated models to luminance components as in other methods [4], [9], [26]. This is because human vision is more sensitive to details in intensity than in color.

This framework crops pixels near image boundary. For our method, this procedure is unnecessary as our network outputs the full-sized image. For fair comparison, however, we also crop pixels to the same amount.

5.4. Comparisons with State-of-the-Art Methods

We provide quantitative and qualitative comparisons. Compared methods are A+ [22], RFL[18], SelfEx [11] and SRCNN [5]. In Table 3, we provide a summary of quantitative evaluation on several datasets. Our methods outperform all previous methods in these datasets. Moreover, our methods are relatively fast. The public code of SRCNN based on a CPU implementation is slower than the code used by Dong et. al [6] in their paper based on a GPU implementation.

In Figures 6 and 7, we compare our method with top-performing methods. In Figure 6, only our method perfectly reconstructs the line in the middle. Similarly, in Figure 7, contours are clean and vivid in our method whereas they are severely blurred or distorted in other methods.

6. Conclusion

In this work, we have presented a super-resolution method using very deep networks. Training a very deep network is hard due to a slow convergence rate. We use residual-learning and extremely high learning rates to optimize a very deep network fast. Convergence speed is maximized and we use gradient clipping to ensure the training stability. We have demonstrated that our method outperforms the existing method by a large margin on benchmarked images. We believe our approach is readily applicable to other image restoration problems such as denoising and compression artifact removal.

References

[1] Y. Bengio, I. J. Goodfellow, and A. Courville. Deep learning. Book in preparation for MIT Press, 2015. 4

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994. 3, 4

[3] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Morel. Super-resolution using neighbor embedding of back-projection residuals. In *Digital Signal Processing (DSP), 2013 18th International Conference on*, pages 1–8. IEEE, 2013. 2

[4] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *CVPR*, 2004. 1, 8

[5] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*. 2014. 4, 6, 7, 8

[6] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 2015. 1, 2, 3, 4, 8

[7] C. E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology*, 18(8):1016–1022, 1979. 1

[8] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International journal of computer vision*, 40(1):25–47, 2000. 1

[9] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009. 1, 8

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. 8

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

[12] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP: Graphical models and image processing*, 53(3):231–239, 1991. 1

[13] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *TPAMI*, 2010. 1

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3

[15] C. G. Marco Bevilacqua, Aline Roumy and M.-L. A. Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012. 1, 2, 6, 8

[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. 8

[17] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013. 4

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

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2, 5

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

[21] R. Timofte, V. De, and L. V. Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV*, 2013. 1, 2, 5, 8

[22] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014. 1, 2, 5, 7, 8

[23] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014. 4

[24] C.-Y. Yang and M.-H. Yang. Fast direct super-resolution by simple functions. In *ICCV*, 2013. 8

[25] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *TIP*, 2010. 1, 8

[26] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. Springer, 2012. 1, 5, 8