Deeply-Recursive Convolutional Network for Image Super-Resolution

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 propose an image super-resolution method (SR) using a deeply-recursive convolutional network (DRCN). Our network has a very deep recursive layer (up to 16 recursions). Increasing recursion depth can improve performance without introducing new parameters for additional convolutions. Albeit advantages, learning a DRCN is very hard with a standard gradient descent method due to exploding/vanishing gradients. To ease the difficulty of training, we propose two extensions: recursive-supervision and skip-connection. Our method outperforms previous methods by a large margin.

1. Introduction

For image super-resolution (SR), receptive field of a convolutional network determines the amount of contextual information that can be exploited to infer missing high-frequency components. For example, if there exists a pattern with smoothed edges contained in a receptive field, it is plausible that the pattern is recognized and edges are appropriately sharpened. As SR is an ill-posed inverse problem, collecting and analyzing more neighbor pixels can possibly give more clues on what may be lost by downsampling.

Deep convolutional networks (DCN) succeeding in various computer vision tasks often use very large receptive fields (224x224 common in ImageNet classification [13, 24]). Among many approaches to widen the receptive field, increasing network depth is one possible way: a convolutional (conv.) layer with filter size larger than a 1×1 or a pooling (pool.) layer that reduces the dimension of intermediate representation can be used. Both approaches have drawbacks: a conv. layer introduces more parameters and a pool. layer typically discards some pixel-wise information.

For image restoration problems such as super-resolution and denoising, image details are very important. Therefore, most deep-learning approaches for such problems do not use pooling. Increasing depth by adding a new weight layer basically introduces more parameters. Two problems can arise. First, overfitting is highly likely. More data are now required. Second, the model becomes too huge to be stored and retrieved.

To resolve these issues, we use a deeply-recursive convolutional network (DRCN). DRCN repeatedly applies the same convolutional layer as many times as desired. The number of parameters do not increase while more recursions are performed. Our network has the receptive field of 41 by 41 and this is relatively large compared to SRCNN [5] (13 by 13). While DRCN has good properties, we find that DRCN optimized with the widely-used stochastic gradient descent method does not easily converge. This is due to exploding/vanishing gradients [1]. Learning long-range dependencies between pixels with a single weight layer is very difficult.

We propose two approaches to ease the difficulty of training (Figure 3(a)). First, all recursions are supervised. Feature maps after each recursion are used to reconstruct the target high-resolution image (HR). Reconstruction method (layers dedicated to reconstruction) is the same for all recursions. As each recursion leads to a different HR prediction, we combine all predictions resulting from different levels of recursions to deliver a more accurate final prediction. The second proposal is to use a skip-connection from input to the reconstruction layer. In SR, a low-resolution image (input) and a high-resolution image (output) share the same information to a large extent. Exact copy of input, however, is likely to be attenuated during many forward passes. We explicitly connect the input to the layers for output reconstruction. This is particularly effective when input and output are highly correlated.

Contributions In summary, we propose an image super-resolution method deeply recursive in nature. It utilizes a very large context compared to previous SR methods with only a single recursive layer. We improve the simple recursive network in two ways: recursive-supervision and skip-connection. Our method demonstrates state-of-the-art performance in common benchmarks.

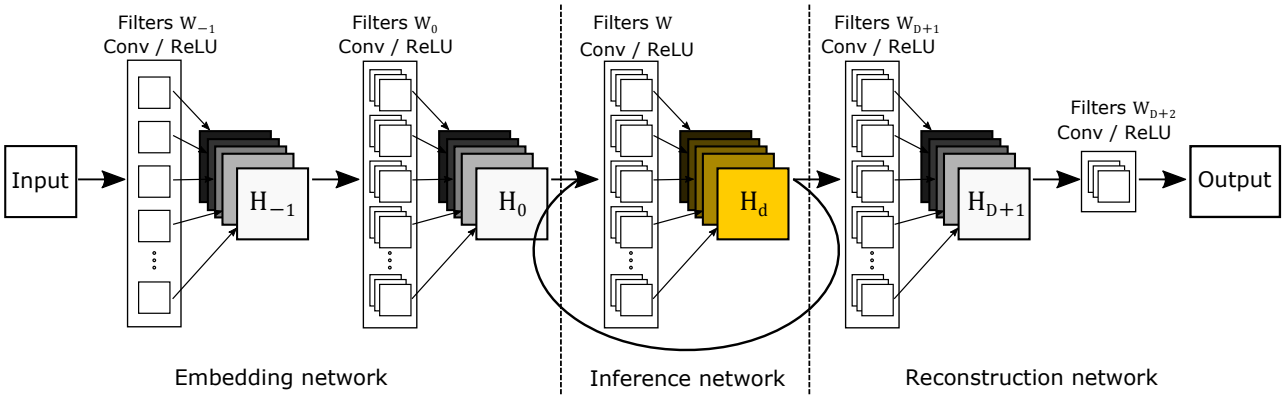


Figure 1: Architecture of our basic model. It consists of three parts: embedding network, inference network and reconstruction network. Inference network has a recursive layer and its unfolded version is in Figure 2.

2. Related Work

2.1. Single-Image Super-Resolution

We apply DRCN to single-image super-resolution (SR) [11, 7, 8]. Many SR methods have been proposed in the computer vision community. Early methods use very fast interpolations but yield poor results. Some of the more powerful methods utilize statistical image priors [27, 12] or internal patch recurrence [8, 10]. Recently, sophisticated learning methods have been widely used to model a mapping from LR to HR patches. Many methods have paid attention to find better regression functions from LR to HR images. This is achieved with various techniques: neighbor embedding [4, 19], sparse coding [31, 32, 28, 29], convolutional neural network (CNN) [5] and random forest [23].

Among several recent learning-based successes, convolutional neural network (SRCNN) [5] demonstrated the feasibility of an end-to-end approach to SR. One possibility to improve SRCNN is to simply stack more weight layers as many times as possible. However, this significantly increases the number of parameters and requires more data to prevent overfitting. In this work, we seek to design a convolutional network that models long-range pixel dependencies with limited capacity. Our network recursively widens the receptive field without increasing model capacity.

2.2. Recursive Neural Network in Computer Vision

Recursive neural networks, suitable for temporal and sequential data, have seen limited use on algorithms operating on a single static image. Socher et al. [25] used a convolutional network in a separate stage to first learn features on RGB-Depth data, prior to hierarchical merging. In these models, the input dimension is twice that of the output and recursive convolutions are applied only two times. Similar dimension reduction occurs in the recurrent convolutional

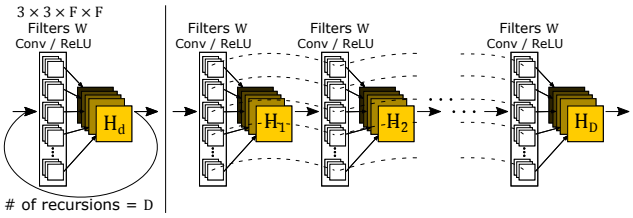


Figure 2: Unfolding inference network. **Left**: A recursive layer **Right**: Unfolded structure. The same filter W is applied to feature maps recursively. Our model can utilize very large context without adding new weight parameters.

neural networks used for semantic segmentation [22]. As SR methods predict full-sized images, dimension reduction is not allowed.

In Eigen et al. [6], recursive layers have the same input and output dimension, but recursive convolutions resulted in worse performances than a single convolution due to overfitting. To overcome overfitting, Liang and Hu [17] uses a recurrent layer that takes feed-forward inputs into all unfolded layers. They show that performance increases up to three convolutions. Their network structure, designed for object recognition, is the same as the existing CNN architectures.

Our network is similar to the above in the sense that recursive or recurrent layers are used with convolutions. We further increase the recursion depth and demonstrate that very deep recursions can significantly boost the performance for super-resolution. We apply the same convolution up to 16 times (the previous maximum is three).

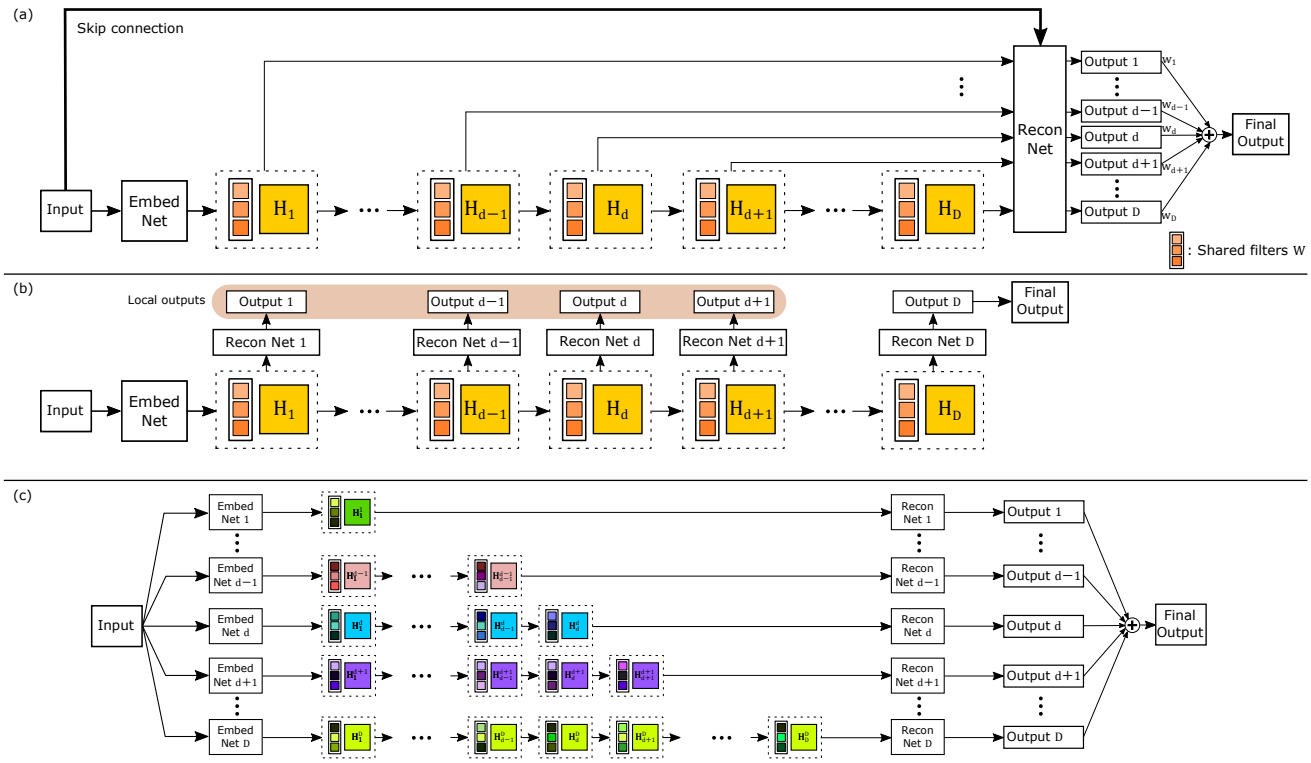


Figure 3: (a): Our final (advanced) model with recursive-supervision and skip-connection. The reconstruction network is shared for recursive predictions. We use all predictions from the intermediate recursion to obtain the final output. (b): Applying deep-supervision [16] to our basic model. Unlike in (a), the model in (b) uses different reconstruction networks for recursions and more parameters are used. (c): An example of expanded structure of (a) without parameter sharing (no recursion). The number of weight parameters is proportional to the depth squared.

3. Proposed Method

3.1. Basic Model

Our first model, outlined in Figure 1, consists of three sub-networks: embedding, inference and reconstruction networks. The embedding net is used to represent the given image as feature maps ready for inference. Next, the inference net solves the task. Once inference is done, final feature maps in the inference net are fed into the reconstruction net to generate the output image.

The **embedding net** takes the input image (grayscale or RGB) and represents it as a set of feature maps. Intermediate representation used to pass information to the inference net largely depends on how the inference net internally represent its feature maps in its hidden layers. Learning this representation is done end-to-end altogether with learning other sub-networks. **Inference net** is the main component that solves the task of super-resolution. Analyzing a large image region is done by a single recursive layer. Each recursion applies the same convolution followed by a rectified linear unit (Figure 2). With convolution filters larger than 1×1, the receptive field is widened with every recursion. While feature maps from the final application of the recursive layer represent the high-resolution image, transforming them (multi-channel) back into the original image space (1 or 3-channel) is necessary. This is done by the **reconstruction net**.

We have a single hidden layer for each sub-net. Only the layer for the inference net is recursive. Other sub-nets are vastly similar to the standard mutilayer perceptrons (MLP) with a single hidden layer. For MLP, full connection of F neurons is equivalent to a convolution with $1 \times 1 \times F \times F$. In our sub-nets, we use $3 \times 3 \times F \times F$ filters. For embedding net, we use 3×3 filters because image gradients are more informative than the raw intensities for super-resolution. For inference net, 3×3 convolutions imply that hidden states are passed to adjacent pixels only. Reconstruction net also takes direct neighbors into account.

Mathematical Formulation The network takes an interpolated input image (to the desired size) as input $\mathbf{x}$ and predicts the target image $\mathbf{y}$ as in SRCNN [5]. Our goal is to learn a model f that predicts values $\hat{\mathbf{y}} = f(\mathbf{x})$, where $\hat{\mathbf{y}}$ is its

estimate of ground truth output $\mathbf{y}$. Let f_1, f_2, f_3 denote subnet functions: embedding, inference and reconstruction, respectively. Our model is the composition of three functions: $f(\mathbf{x}) = f_3(f_2(f_1(\mathbf{x})))$.

Embedding net $f_1(\mathbf{x})$ takes the input vector $\mathbf{x}$ and computes the matrix output H_0, which is an input to the inference net f_2. Hidden layer values are denoted by H_{-1}. The formula for embedding net is as follows:

$$H_{-1} = max(0, W_{-1} * \mathbf{x} + b_{-1}) \quad (1)$$

$$H_0 = max(0, W_0 * H_{-1} + b_0) \quad (2)$$

$$f_1(\mathbf{x}) = H_0, \quad (3)$$

where the operator $*$ denotes a convolution and $max(0, \cdot)$ corresponds to a ReLU. Weight and bias matrices are W_{-1}, W_0 and b_{-1}, b_0.

Inference net f_2 takes the input matrix H_0 and computes the matrix output H_D. Here, we use the same weight and bias matrices W and b for all operations. Let g denote the function modeled by a single recursion of the recursive layer: $g(H) = max(0, W * H + b)$. The recurrence relation is

$$H_d = g(H_{d-1}) = max(0, W * H_{d-1} + b), \quad (4)$$

for $d = 1, ..., D$. Inference net f_2 is equivalent to the composition of the same elementary function g:

$$f_2(H) = (g \circ g \circ \cdots \circ)g(H) = g^D(H), \quad (5)$$

where the operator $\circ$ denotes a function composition and g^d denotes the d-fold product of g.

Reconstruction net f_3 takes the input hidden state H_D and outputs the target image (high-resolution). Roughly speaking, reconstruction net is the inverse operation of embedding net. The formula is as follows:

$$H_{D+1} = max(0, W_{D+1} * H_D + b_{D+1}) \quad (6)$$

$$\hat{\mathbf{y}} = max(0, W_{D+2} * H_{D+1} + b_{D+2}) \quad (7)$$

$$f_3(H) = \hat{\mathbf{y}}. \quad (8)$$

Model Properties Now we have all components for our model. The recursive model has pros and cons. While the recursive model is simple and powerful, we find training a deeply-recursive network very difficult. This is in accordance with the limited success of previous methods using at most three recursions so far [17]. Among many reasons, two severe problems are *vanishing* and *exploding gradients* [1, 21].

Exploding gradients refer to the large increase in the norm of the gradient during training. Such events are due to the multiplicative nature of chained gradients. Long term components can grow exponentially for deep recursions. The *vanishing gradients* problem refers to the opposite behavior. Long term components approach exponentially fast to the zero vector. Due to this, learning the relation between distant pixels is very hard. Another known issue is that storing an exact copy of information through many recursions is not easy. In SR, output is vastly similar to input and recursive layer needs to keep the exact copy of input image for many recursions. These issues are also observed when we train our basic recursive model and we did not succeed in training a deeply-recursive network.

In addition to gradient problems, there exists an issue with finding the optimal number of recursions. If recursions are too deep for a given task, we need to reduce the number of recursions. Finding the optimal number requires training many networks with different recursion depths.

3.2. Advanced Model

Recursive-Supervision To resolve the gradient and optimal recursion issues, we propose an improved model. We supervise all recursions in order to alleviate the effect of vanishing/exploding gradients. As we have assumed that the same representation can be used again and again during convolutions in the inference net, the same reconstruction net is used to predict HR images for all recursions. Our reconstruction net now outputs D predictions and all predictions are simultaneously supervised during training (Figure 3 (a)). We use all D intermediate predictions to compute the final output. All predictions are averaged during testing. The optimal weights are automatically learned during training.

A similar but a different concept of supervising intermediate layers for a convolutional network is used in Lee et al [16]. Their method simultaneously minimizes classification error while improving the directness and transparency of the hidden layer learning process. There are two significant differences between our recursive-supervision and deep-supervision proposed in Lee et al. [16]. They associate a unique classifier for each hidden layer. For each additional layer, a new classifier has to be introduced, as well as new parameters. If this approach is used, our modified network would resemble that of Figure 3(b). We would then need D different reconstruction networks. This is against our original purpose of using recursive networks, which is avoid introducing new parameters while stacking more layers. In addition, using different reconstruction nets no longer effectively regularizes the network. The second difference is that Lee et al. [16] discards all intermediate classifiers during testing. However, an ensemble of all intermediate predictions significantly boosts the performance. The final output from the ensemble is also supervised.

Our recursive-supervision naturally eases the difficulty of training recursive networks. Backpropagation goes through a small number of layers if supervising signal goes directly from loss layer to early recursion. Summing all gradients backpropagated from different prediction losses

gives a smoothing effect. The adversarial effect of vanishing/exploding gradients along one backpropagation path is alleviated.

Moreover, the importance of picking the optimal number of recursions is reduced as our supervision enables utilizing predictions from all intermediate layers. If recursions are too deep for the given task, we expect the weight for late predictions to be low while early predictions receive high weights.

By looking at weights of predictions, we can figure out the marginal gain from additional recursions.

We present an expanded CNN structure of our model for illustration purposes in Figure 3(c). If parameters are not allowed to be shared and CNN chains vary their depths, the number of free parameters grows fast (quadratically).

Skip-Connection Now we describe our second extension: skip-connection. For SR, input and output images are highly correlated. Carrying most if not all of input values until the end of the network is inevitable but very inefficient. Due to gradient problems, exactly learning a simple linear relation between input and output is very difficult if many recursions exist in between them.

We add a layer skip [3] from input to the reconstruction net. Adding layer skips is successfully used for a semantic segmentation network [18] and we employ a similar idea. Now input image is directly fed into the reconstruction net whenever it is used during recursions. Our skip-connection has two advantages. First, network capacity to store the input signal during recursions is saved. Second, the exact copy of input signal can be used during target prediction.

Our skip-connection is simple yet very effective. In super-resolution, LR and HR images are vastly similar. In most regions, differences are zero and only small number of locations have non-zero values. For this reason, several super-resolution methods [28, 29, 19, 2] predict image details only. Similarly, we find that this domain-specific knowledge significantly improves our learning procedure.

Mathematical Formulation Each intermediate prediction under recursive-supervision (Figure 3(a)) is

$$\hat{\mathbf{y}}_d = f_3(\mathbf{x}, g^{(d)}(f_1(\mathbf{x}))), \tag{9}$$

for $d = 1, 2, \ldots, D$, where f_3 now takes two inputs, one from skip-connection. Reconstruction net with skip-connection can take various functional forms. For example, input can be concatenated to the feature maps H_d. As the input is an interpolated input image (roughly speaking, $\hat{\mathbf{y}} \approx \mathbf{x}$), we find $f_3(\mathbf{x}, H_d) = \mathbf{x} + f_3(H_d)$ is enough for our purpose. More sophisticated functions for merging two inputs to f_3 will be explored in the future.

Now, the final output is the weighted average of all intermediate predictions:

$$\hat{\mathbf{y}} = \sum_{d=1}^{D} w_d \cdot \hat{\mathbf{y}}_d. \tag{10}$$

where w_d denotes the weights of predictions reconstructed from each intermediate hidden state during recursion. These weights are learned during training.

3.3. Training

Objective We now describe the training objective used to find optimal parameters of our model. Given a training dataset $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, our goal is to find the best model f that accurately predicts values $\hat{\mathbf{y}} = f(\mathbf{x})$.

In the least-squares regression setting, typical in SR, the mean squared error $\frac{1}{2}||\mathbf{y} - f(\mathbf{x})||^2$ averaged over the training set is minimized. This favors high Peak Signal-to-Noise Ratio (PSNR), a widely-used evaluation criteria.

With recursive-supervision, we have $D + 1$ objectives to minimize: supervising D outputs from recursions and the final output. For intermediate outputs, we have the loss function

$$l_1(\theta) = \sum_{d=1}^{D} \sum_{i=1}^{N} \frac{1}{2DN} ||\mathbf{y}^{(i)} - \hat{\mathbf{y}}_d^{(i)}||^2, \tag{11}$$

where θ denotes the parameter set and $\hat{\mathbf{y}}_d^{(i)}$ is the output from the d-th recursion. For the final output, we have

$$l_2(\theta) = \sum_{i=1}^{N} \frac{1}{2N} ||\mathbf{y}^{(i)} - \sum_{d=1}^{D} w_d \cdot \hat{\mathbf{y}}_d^{(i)}||^2 \tag{12}$$

Now we give the final loss function $L(\theta)$. The training is regularized by weight decay (L_2 penalty multiplied by β).

$$L(\theta) = \alpha l_1(\theta) + (1 - \alpha) l_2(\theta) + \beta ||\theta||^2, \tag{13}$$

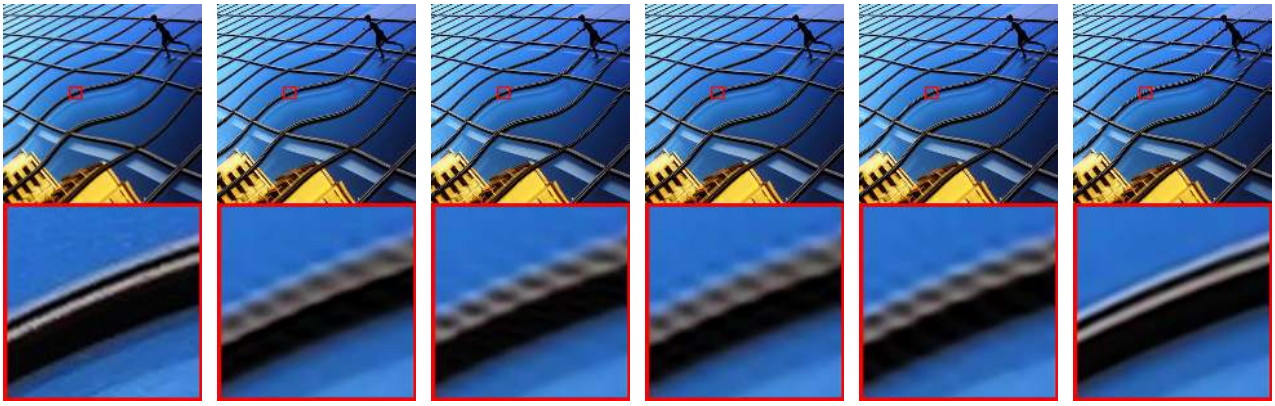
where α denotes the importance of the companion objective on the intermediate outputs and β denotes the multiplier of weight decay. Setting α high makes the training procedure stable as early recursions easily converge. As training progresses, α decays to boost the performance of the final output.

Training is carried out by optimizing the regression objective using mini-batch gradient descent based on back-propagation (LeCun et al. [15]). We implement our model using the *MatConvNet*[1] package [30].

4. Experimental Results

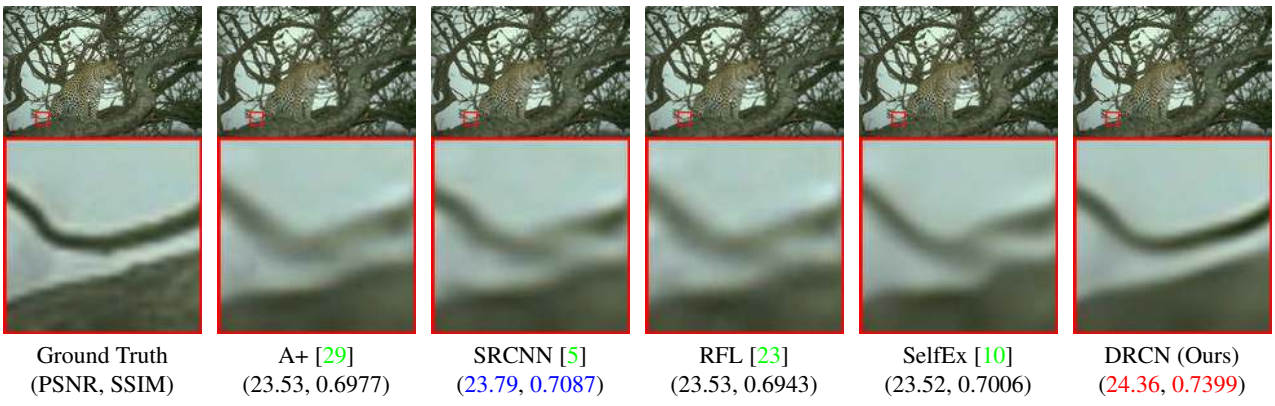
In this section, we evaluate the performance of our method on several datasets. We first describe datasets used

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



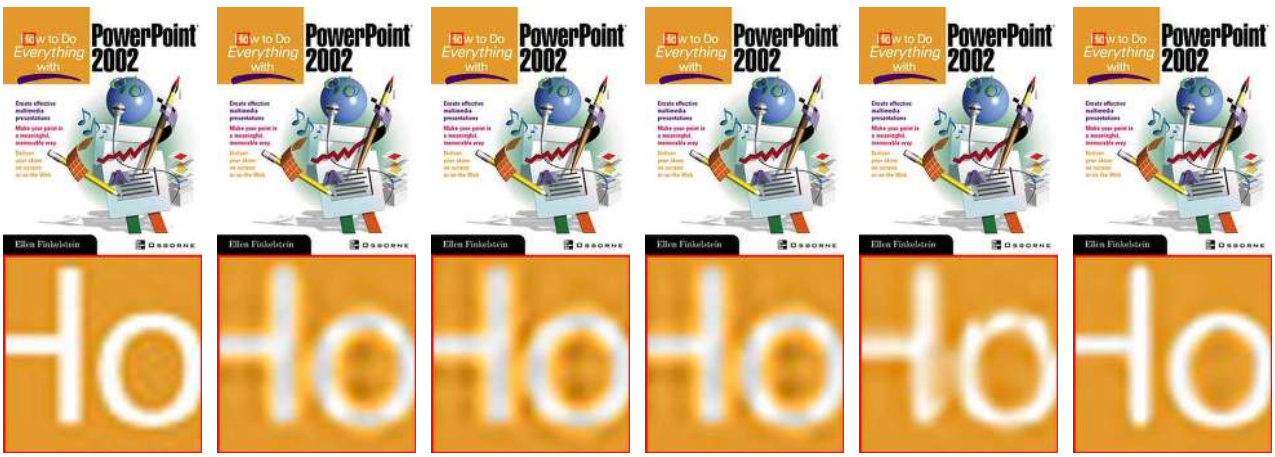
Ground Truth (PSNR, SSIM)	A+ [29] (29.83, 0.9102)	SRCNN [5] (29.97, 0.9092)	RFL [23] (29.61, 0.9026)	SelfEx [10] (30.73, 0.9193)	DRCN (Ours) (32.17, 0.9350)

Figure 4: Super-resolution results of "img082"(*Urban100*) with scale factor $\times 4$. Line is straightened and sharpened in our result, whereas other methods give blurry lines. Our result seems visually pleasing.



Ground Truth (PSNR, SSIM)	A+ [29] (23.53, 0.6977)	SRCNN [5] (23.79, 0.7087)	RFL [23] (23.53, 0.6943)	SelfEx [10] (23.52, 0.7006)	DRCN (Ours) (24.36, 0.7399)

Figure 5: Super-resolution results of "134035" (*B100*) with scale factor $\times 4$. Our result shows a clear separation between branches while in other methods, branches are not well separated.



Ground Truth (PSNR, SSIM)	A+ [29] (26.09, 0.9342)	SRCNN [5] (27.01, 0.9365)	RFL [23] (25.91, 0.9254)	SelfEx [10] (27.10, 0.9483)	DRCN (Ours) (27.66, 0.9608)

Figure 6: Super-resolution results of "ppt3" (*Set14*) with scale factor $\times 3$. Texts in DRCN are sharp while, in other methods, character edges are blurry.

Dataset	Scale	Bicubic PSNR/SSIM	A+ [29] PSNR/SSIM	SRCNN [5] PSNR/SSIM	RFL [23] PSNR/SSIM	SelfEx [10] PSNR/SSIM	DRCN (Ours) PSNR/SSIM
Set5	×2	33.66/0.9299	36.54/0.9544	36.66/0.9542	36.54/0.9537	36.49/0.9537	37.63/0.9588
	×3	30.39/0.8682	32.58/0.9088	32.75/0.9090	32.43/0.9057	32.58/0.9093	33.82/0.9226
	×4	28.42/0.8104	30.28/0.8603	30.48/0.8628	30.14/0.8548	30.31/0.8619	31.53/0.8854
Set14	×2	30.24/0.8688	32.28/0.9056	32.42/0.9063	32.26/0.9040	32.22/0.9034	33.04/0.9118
	×3	27.55/0.7742	29.13/0.8188	29.28/0.8209	29.05/0.8164	29.16/0.8196	29.76/0.8311
	×4	26.00/0.7027	27.32/0.7491	27.49/0.7503	27.24/0.7451	27.40/0.7518	28.02/0.7670
B100	×2	29.56/0.8431	31.21/0.8863	31.36/0.8879	31.16/0.8840	31.18/0.8855	31.85/0.8942
	×3	27.21/0.7385	28.29/0.7835	28.41/0.7863	28.22/0.7806	28.29/0.7840	28.80/0.7963
	×4	25.96/0.6675	26.82/0.7087	26.90/0.7101	26.75/0.7054	26.84/0.7106	27.23/0.7233
Urban100	×2	26.88/0.8403	29.20/0.8938	29.50/0.8946	29.11/0.8904	29.54/0.8967	30.75/0.9133
	×3	24.46/0.7349	26.03/0.7973	26.24/0.7989	25.86/0.7900	26.44/0.8088	27.15/0.8276
	×4	23.14/0.6577	24.32/0.7183	24.52/0.7221	24.19/0.7096	24.79/0.7374	25.14/0.7510

Table 1: Benchmark results. Average PSNR/SSIMs for scale factor ×2, ×3 and ×4 on datasets Set5, Set14, B100 and Urban100. Red color indicates the best performance and blue color refers the second best.

Ground Truth (PSNR, SSIM) | A+ [29] (24.24, 0.8176) | SRCNN [5] (24.48, 0.8267) | RFL [23] (24.24, 0.8137) | SelfEx [10] (24.16, 0.8145) | DRCN (Ours) (24.76, 0.8385)

Figure 7: Super-resolution results of "58060" (*B100*) with scale factor ×2. A three-line stripe in ground truth is also observed in DRCN, whereas it is not clearly seen in results of other methods.

for training and testing our method. Next, our training setup is given. We give several experiments for understanding our model properties. The effect of increasing the number of recursions is investigated. Finally, we compare our method with several state-of-the-art methods.

4.1. Datasets

For training, we use 91 images proposed in Yang et al. [31] for all experiments. For testing, we use four datasets. Datasets *Set5* [19] and *Set14* [32] are often used for benchmark [29, 28, 5]. Dataset *B100* consists of natural images in the Berkeley Segmentation Dataset [20]. Finally, dataset *Urban100*, urban images recently provided by Huang et al. [10], is very interesting as it contains many challenging images failed by existing methods.

4.2. Training Setup

We use 16 recursions unless stated otherwise. When unfolded, the longest chain from the input to the output passes 20 conv. layers (receptive field of 41 by 41). We set the momentum parameter to 0.9 and weight decay to 0.0001. We use 256 filters of the size 3×3 for all weight layers. Training images are split into 41 by 41 patches with stride 21 and 64 patches are used as a mini-batch for stochastic gradient descent.

For initializing weights in non-recursive layers, we use the method described in He et al. [9]. For recursive convolutions, we set all weights to zero except self-connections (connection to the same neuron in the next layer) [26, 14]. Biases are set to zero.

Learning rate is initially set to 0.01 and then decreased by a factor of 10 if the validation error does not decrease for 5 epochs. If learning rate is less than 10^{-6}, the procedure

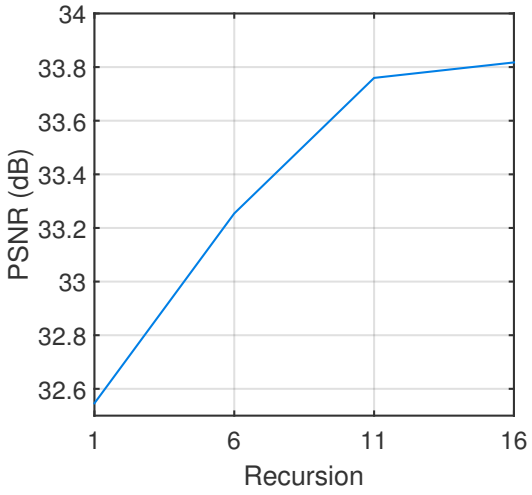


Figure 8: Recursion versus Performance for the scale factor $\times 3$ on the dataset *Set5*. More recursions yielding larger receptive fields lead to better performances.

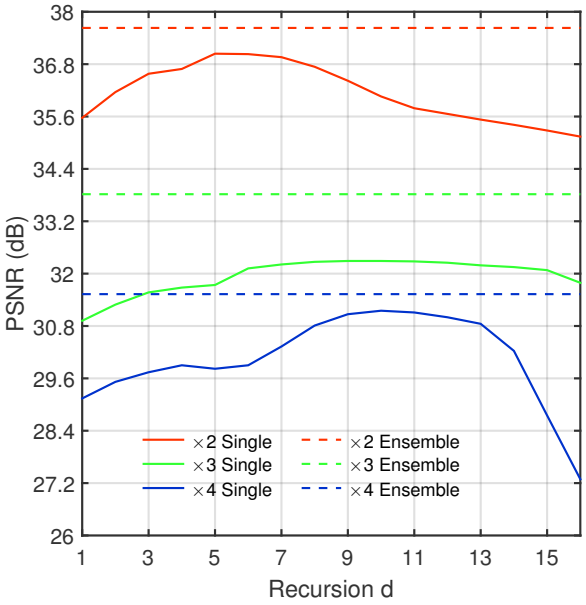


Figure 9: Ensemble effect. Prediction made from intermediate recursions are evaluated. There is no single recursion depth that works the best across all scale factors. Ensemble of intermediate predictions significantly improves performance.

is terminated. Training roughly takes 6 days on a machine using one Titan X GPU.

4.3. Study of Deep Recursions

We study the effect of increasing recursion depth. We trained four models with different numbers of recursions: 1, 6, 11, and 16. Four models use the same number of parameters except the weights used for ensemble. In Figure 8, it is shown that as more recursions are performed, PSNR measures increase. Increasing recursion depth with a larger image context and more nonlinearities boosts performance. The effect of ensemble is also investigated. We first evaluate intermediate predictions made from recursions (Figure 9). The ensemble output significantly improves performances of individual predictions.

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

We provide quantitative and qualitative comparisons. For benchmark, we use public code for A+ [29], SRCNN [5], RFL [23] and SelfEx [10]. We deal with luminance components only as similarly done in other methods because human vision is much more sensitive to details in intensity than in color.

As some methods such as A+ [29] and RFL [23] do not predict image boundary, they require cropping pixels near borders. For our method, this procedure is unnecessary as our network predicts the full-sized image. For fair comparison, however, we also crop pixels to the same amount. PSNRs can be slightly different from original papers as existing methods use slightly different evaluation frameworks. We use the public evaluation code used in [10].

In Table 1, we provide a summary of quantitative evaluation on several datasets. Our method outperforms all existing methods in all datasets and scale factors (both PSNR and SSIM). In Figures 4, 5, 6 and 7, example images are given. Our method produces relatively sharp edges respective to patterns. In contrast, edges in other images are blurred. Our method takes a second to process a 288×288 image on a GPU Titan X.

5. Conclusion

In this work, we have presented a super-resolution method using a deeply-recursive convolutional network. Our network efficiently reuses weight parameters while exploiting a large image context. To ease the difficulty of training the model, we use recursive-supervision and skip-connection. We have demonstrated that our method outperforms existing methods by a large margin on benchmarked images. In the future, one can try more recursions in order to use image-level context. We believe our approach is readily applicable to other image restoration problems such as denoising and compression artifact removal.

References

[1] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2), 1994. 1, 4

[2] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Morel. Super-resolution using neighbor embedding of back-projection residuals. In *International Conference on Digital Signal Processing*, 2013. 5

[3] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006. 5

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

[5] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 2014. 1, 2, 3, 6, 7, 8

[6] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. In *ICLR Workshop*, 2014. 2

[7] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *IJCV*, 2000. 2

[8] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009. 2

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

[10] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution using transformed self-exemplars. In *CVPR*, 2015. 2, 6, 7, 8

[11] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP: Graphical models and image processing*, 53(3), 1991. 2

[12] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *TPAMI*, 2010. 2

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[14] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015. 7

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998. 5

[16] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*, 2014. 3, 4

[17] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *CVPR*, 2015. 2, 4

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014. 5

[19] 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. 2, 5, 7

[20] 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. 7

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

[22] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *Proceedings of The 31st International Conference on Machine Learning*, pages 82–90, 2014. 2

[23] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *CVPR*, 2015. 2, 6, 7, 8

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

[25] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *NIPS*, 2012. 2

[26] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL*, 2012. 7

[27] J. Sun, Z. Xu, and H.-Y. Shum. Image super-resolution using gradient profile prior. In *CVPR*, 2008. 2

[28] R. Timofte, V. De, and L. V. Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV*, 2013. 2, 5, 7

[29] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014. 2, 5, 6, 7, 8

[30] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014. 5

[31] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *TIP*, 2010. 2, 7

[32] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*. Springer, 2012. 2, 7