Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks

Wei-Sheng Lai, Jia-Bin Huang, *Member, IEEE*,
Narendra Ahuja, and Ming-Hsuan Yang, *Senior Member, IEEE*

Abstract—Convolutional neural networks have recently demonstrated high-quality reconstruction for single image super-resolution. However, existing methods often require a large number of network parameters and entail heavy computational loads at runtime for generating high-accuracy super-resolution results. In this paper, we propose the deep Laplacian Pyramid Super-Resolution Network for fast and accurate image super-resolution. The proposed network progressively reconstructs the sub-band residuals of high-resolution images at multiple pyramid levels. In contrast to existing methods that involve the bicubic interpolation for pre-processing (which results in large feature maps), the proposed method directly extracts features from the low-resolution input space and thereby entails low computational loads. We train the proposed network with deep supervision using the robust Charbonnier loss functions and achieve high-quality image reconstruction. Furthermore, we utilize the recursive layers to share parameters across as well as within pyramid levels, and thus drastically reduce the number of parameters. Extensive quantitative and qualitative evaluations on benchmark datasets show that the proposed algorithm performs favorably against the state-of-the-art methods in terms of run-time and image quality.

Index Terms—Single-image super-resolution, deep convolutional neural networks, Laplacian pyramid

1 Introduction

Single image super-resolution (SR) aims to reconstruct a high-resolution (HR) image from one single low-resolution (LR) input image. Example-based SR methods have demonstrated the state-of-the-art performance by learning a mapping from LR to HR image patches using large image datasets. Numerous learning algorithms have been applied to learn such a mapping function, including dictionary learning [1], [2], local linear regression [3], [4], and random forest [5], to name a few.

Convolutional Neural Networks (CNNs) have been widely used in vision tasks ranging from object recognition [6], segmentation [7], optical flow [8], to super-resolution. In [9], Dong et al. propose a Super-Resolution Convolutional Neural Network (SRCNN) to learn a nonlinear LR-to-HR mapping function. This network architecture has been extended to embed a sparse coding model [10], increase network depth [11], or apply recursive layers [12], [13]. While these models are able to generate high-quality SR images, there remain three issues to be addressed. First, these methods use a pre-defined upsampling operator, e.g., bicubic interpolation, to upscale an input LR image to the desired spatial resolution *before* applying a network for predicting the details (Fig. 1a). This pre-upsampling step increases unnecessary computational cost and does not provide additional high-frequency information for reconstructing HR images. Several algorithms accelerate the SRCNN by extracting features directly from the input LR images (Fig. 1b) and replacing the pre-defined upsampling operator with sub-pixel convolution [14] or transposed convolution [15] (also named as deconvolution in some literature). These methods, however, use relatively small networks and cannot learn complicated mappings well due to the limited model capacity. Second, existing methods optimize the networks with an $\mathcal{L}_2$ loss (i.e., mean squared error loss). Since the same LR patch may have multiple corresponding HR patches and the $\mathcal{L}_2$ loss fails to capture the underlying multi-modal distributions of HR patches, the reconstructed HR images are often over-smoothed and inconsistent to human visual perception on natural images. Third, existing methods mainly reconstruct HR images in *one upsampling step*, which makes learning mapping functions for large scaling factors (e.g., $8\times$) more difficult.

To address these issues, we propose the deep Laplacian Pyramid Super-Resolution Network (LapSRN) to progressively reconstruct HR images in a coarse-to-fine fashion. As shown in Fig. 1c, our model consists of a feature extraction branch and an image reconstruction branch. The feature extraction branch uses a cascade of convolutional layers to extract non-linear feature maps from LR input images. We then apply a transposed convolutional layer for upsampling the feature maps to a finer level and use a convolutional layer to predict the sub-band residuals (i.e., the differences between the

- *W.-S. Lai and M.-H. Yang are with the Department of Electrical and Engineering and Computer Science, University of California, Merced, CA 95340. E-mail: {wlai24, mhyang}@ucmerced.edu.*
- *J.-B. Huang is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24060. E-mail: jbhuang@vt.edu.*
- *N. Ahuja is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL 61801. E-mail: n-ahuja@illinois.edu.*

*Manuscript received 3 Oct. 2017; revised 9 July 2018; accepted 7 Aug. 2018. Date of publication 12 Aug. 2018; date of current version 10 Oct. 2019. (Corresponding author: Ming-Hsuan Yang.)
Recommended for acceptance by Y. Wu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2018.2865304*

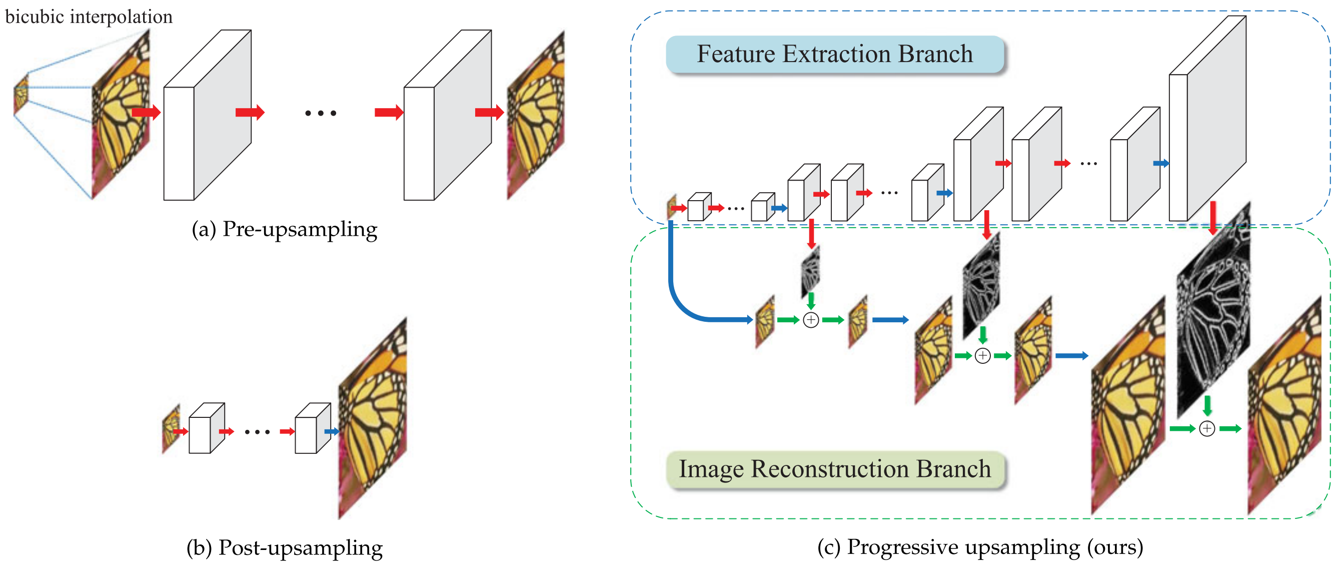

bicubic interpolation
(a) Pre-upsampling
(b) Post-upsampling
Feature Extraction Branch
Image Reconstruction Branch
(c) Progressive upsampling (ours)

Fig. 1. *Comparisons of upsampling strategies in CNN-based SR algorithms.* Red arrows indicate convolutional layers. Blue arrows indicate transposed convolutions (upsampling), and green arrows denote element-wise addition operators. (a) Pre-upsampling based approaches (e.g., SRCNN [9], VDSR [11], DRCN [12], DRRN [13]) typically use the bicubic interpolation to upscale LR input images to the target spatial resolution before applying deep networks for prediction and reconstruction. (b) Post-upsampling based methods directly extract features from LR input images and use sub-pixel convolution [14] or transposed convolution [15] for upsampling. (c) Progressive upsampling approach using the proposed Laplacian pyramid network reconstructs HR images in a coarse-to-fine manner.

upsampled image and the ground truth HR image at the respective pyramid level). The image reconstruction branch upsamples the LR images and takes the sub-band residuals from the feature extraction branch to efficiently reconstruct HR images through element-wise addition. Our network architecture naturally accommodates deep supervision (i.e., supervisory signals can be applied simultaneously at each level of the pyramid) to guide the reconstruction of HR images. Instead of using the $\mathcal{L}_2$ loss function, we propose to train the network with the robust Charbonnier loss functions to better handle outliers and improve the performance. While both feature extraction and image reconstruction branches have multiple levels, we train the network in an end-to-end fashion without stage-wise optimization.

Our algorithm differs from existing CNN-based methods in the following three aspects:

1) *Accuracy*. Instead of using a pre-defined upsampling operation, our network jointly optimizes the deep convolutional layers and upsampling filters for both images and feature maps by minimizing the Charbonnier loss function. As a result, our model has a large capacity to learn complicated mappings and effectively reduces the undesired artifacts caused by spatial aliasing.
2) *Speed*. Our LapSRN accommodates both fast processing speed and high capacity of deep networks. Experimental results demonstrate that our method is faster than several CNN-based super-resolution models, e.g., VDSR [11], DRCN [12], and DRRN [13]. The proposed model achieves real-time performance as FSRCNN [15] while generating significantly better reconstruction accuracy.
3) *Progressive reconstruction*. Our model generates multiple intermediate SR predictions in *one* feed-forward pass through progressive reconstruction. This characteristic renders our method applicable to a wide range of tasks that require resource-aware adaptability. For example, the same network can be used to enhance the spatial resolution of videos depending on the available computational resources. For scenarios with limited computing resources, our $8\times$ model can still perform $2\times$ or $4\times$ SR by simply bypassing the computation of residuals at finer levels. Existing CNN-based methods, however, do not offer such flexibility.

In this work, we make the following extensions to improve our early results [16] substantially:

1) *Parameter sharing*. We re-design our network architecture to share parameters *across* pyramid levels and *within* the feature extraction sub-network via recursion. Through parameter sharing, we reduce 73 percent of the network parameters while achieving better reconstruction accuracy on benchmark datasets.
2) *Local skip connections*. We systematically analyze three different approaches for applying local skip connections in the proposed model. By leveraging proper skip connections to alleviate the gradient vanishing and explosion problems, we are able to train an 84-layer network to achieve the state-of-the-art performance.
3) *Multi-scale training*. Unlike in the preliminary work where we train three different models for handling $2\times$, $4\times$ and $8\times$ SR, respectively, we train one *single* model to handle *multiple* upsampling scales. The multi-scale model learns the inter-scale correlation and improves the reconstruction accuracy against single-scale models. We refer to our multi-scale model as MS-LapSRN.

TABLE 1
Feature-by-Feature Comparisons of CNN-based SR Algorithms

Method	Input	Reconstruction	Depth	Filters	Parameters	GRL	LRL	Multi-scale training	Loss function
SRCNN [9]	LR + bicubic	Direct	3	64	57 k				$\mathcal{L}_2$
FSRCNN [15]	LR	Direct	8	56	12 k				$\mathcal{L}_2$
ESPCN [14]	LR	Direct	3	64	20 k				$\mathcal{L}_2$
SCN [10]	LR + bicubic	Progressive	10	128	42 k				$\mathcal{L}_2$
VDSR [11]	LR + bicubic	Direct	20	64	665 k	✓		✓	$\mathcal{L}_2$
DRCN [12]	LR + bicubic	Direct	20	256	1775 k	✓			$\mathcal{L}_2$
DRRN [13]	LR + bicubic	Direct	52	128	297 k	✓	✓	✓	$\mathcal{L}_2$
MDSR [17]	LR	Direct	162	64	8000 k		✓	✓	Charbonnier
LapSRN [16]	LR	Progressive	24	64	812 k	✓			Charbonnier
MS-LapSRN (ours)	LR	Progressive	84	64	222 k	✓	✓	✓	Charbonnier

Methods with direct reconstruction performs one-step upsampling from the LR to HR space, while progressive reconstruction predicts HR images in multiple upsampling steps. Depth represents the number of convolutional and transposed convolutional layers in the longest path from input to output for $4\times$ SR. Global residual learning (GRL) indicates that the network learns the difference between the ground truth HR image and the upsampled (i.e., using bicubic interpolation or learned filters) LR images. Local residual learning (LRL) stands for the local skip connections between intermediate convolutional layers.

2 Related Work

Single-image super-resolution has been extensively studied in the literature. Here we focus our discussion on recent example-based and CNN-based approaches.

2.1 SR Based on Internal Databases

Several methods [18], [19], [20] exploit the self-similarity property in natural images and construct LR-HR patch pairs based on the scale-space pyramid of the LR input image. While internal databases contain more relevant training patches than external image datasets, the number of LR-HR patch pairs may not be sufficient to cover large textural appearance variations in an image. Singh et al. [21] decompose patches into directional frequency sub-bands and determine better matches in each sub-band pyramid independently. In [22], Huang et al. extend the patch search space to accommodate the affine and perspective deformation. The SR methods based on internal databases are typically slow due to the heavy computational cost of patch searches in the scale-space pyramid. Such drawbacks make these approaches less feasible for applications that require computational efficiency.

2.2 SR Based on External Databases

Numerous SR methods learn the LR-HR mapping with image pairs collected from external databases using supervised learning algorithms, such as nearest neighbor [23], manifold embedding [24], [25], kernel ridge regression [26], and sparse representation [1], [2], [27]. Instead of directly modeling the complex patch space over the entire database, recent methods partition the image set by K-means [4], sparse dictionary [3], [28] or random forest [5], and learn locally linear regressors for each cluster. While these approaches are effective and efficient, the extracted features and mapping functions are hand-designed, which may not be optimal for generating high-quality SR images.

2.3 Convolutional Neural Networks Based SR

CNN-based SR methods have demonstrated state-of-the-art results by jointly optimizing the feature extraction, non-linear mapping, and image reconstruction stages in an end-to-end manner. The VDSR network [11] shows significant improvement over the SRCNN method [9] by increasing the network depth from 3 to 20 convolutional layers. To facilitate training a deeper model with a fast convergence speed, the VDSR method adopts the global residual learning paradigm to predict the differences between the ground truth HR image and the bicubic upsampled LR image instead of the actual pixel values. Wang et al. [10] combine the domain knowledge of sparse coding with a deep CNN and train a cascade network (SCN) to upsample images progressively. In [12], Kim et al. propose a network with multiple recursive layers (DRCN) with up to 16 recursions. The DRRN approach [13] further trains a 52-layer network by extending the local residual learning approach of the ResNet [6] with deep recursion. We note that the above methods use bicubic interpolation to pre-upsample input LR images *before* feeding into the deep networks, which increases the computational cost and requires a large amount of memory.

To achieve real-time speed, the ESPCN method [14] extracts feature maps in the LR space and replaces the bicubic upsampling operation with an efficient sub-pixel convolution (i.e., pixel shuffling). The FSRCNN method [15] adopts a similar idea and uses a hourglass-shaped CNN with transposed convolutional layers for upsampling. As a trade-off of speed, both ESPCN [14] and FSRCNN [15] have limited network capacities for learning complex mappings. Furthermore, these methods upsample images or features in *one upsampling step* and use only one supervisory signal from the target upsampling scale. Such a design often causes difficulties in training models for large upsampling scales (e.g., $4\times$ or $8\times$). In contrast, our model progressively upsamples input images on *multiple* pyramid levels and use *multiple* losses to guide the prediction of sub-band residuals at each level, which leads to accurate reconstruction, particularly for large upsampling scales.

All the above CNN-based SR methods optimize networks with the $\mathcal{L}_2$ loss function, which often leads to over-smooth results that do not correlate well with human perception. We demonstrate that the proposed deep network with the robust Charbonnier loss function better handles outliers and improves the SR performance over the $\mathcal{L}_2$ loss function. Most recently, Lim et al. [17] propose a multi-scale deep SR model (MDSR) by extending ESPCN [14] with three branches for scale-specific upsampling but sharing most of the parameters

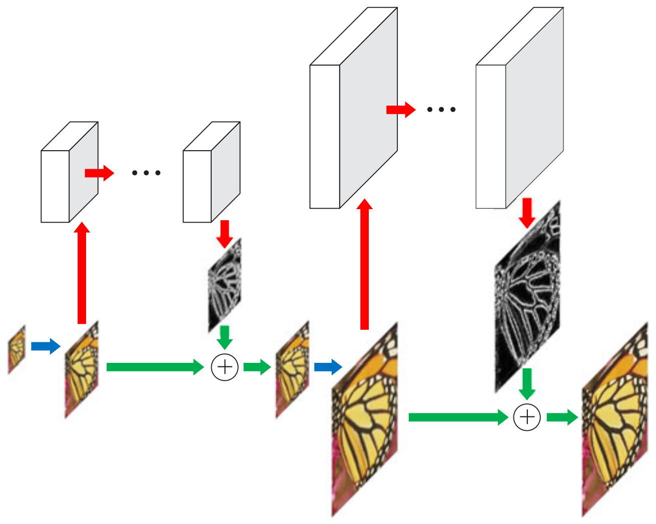


Fig. 2. *Generative network of LAPGAN* [34]. The LAPGAN first upsamples the input images before applying convolution for predicting residuals at each pyramid level.

across different scales. The MDSR method is trained on a high-resolution DIV2K [29] dataset (800 training images of 2k resolution), and achieves the state-of-the-art performance. Table 1 shows the main components of the existing CNN-based SR methods. The preliminary method of this work, i.e., LapSRN [16], and the proposed MS-LapSRN algorithm are listed in the last two rows.

2.4 Laplacian Pyramid

The Laplacian pyramid has been widely used in several vision tasks, including image blending [30], texture synthesis [31], edge-aware filtering [32] and semantic segmentation [33]. Denton et al. [34] propose a generative adversarial network based on a Laplacian pyramid framework (LAPGAN) to generate realistic images, which is the most related to our work. However, the proposed LapSRN differs from LAPGAN in two aspects.

First, the *objectives* of the two models are different. The LAPGAN is a generative model which is designed to synthesize diverse natural images from random noise and sample inputs. On the contrary, the proposed LapSRN is a super-resolution model that predicts a particular HR image based on the given LR image and upsampling scale factor. The LAPGAN uses a cross-entropy loss function to encourage the output images to respect the data distribution of the training datasets. In contrast, we use the Charbonnier penalty function to penalize the deviation of the SR prediction from the ground truth HR images.

Second, the differences in *architecture designs* result in disparate inference speed and network capacities. As shown in Fig. 2, the LAPGAN upsamples input images *before* applying convolution at each level, while our LapSRN extracts features directly from the LR space and upscales images at the end of each level. Our network design effectively alleviates the computational cost and increases the size of receptive fields. In addition, the convolutional layers at each level in our LapSRN are *connected* through multi-channel transposed convolutional layers. The residual images at a higher level are therefore predicted by a deeper network with shared feature representations at lower levels. The shared features at lower levels increase the non-linearity at finer convolutional layers to learn complex mappings.

2.5 Adversarial Training

The Generative Adversarial Networks (GANs) [35] have been applied to several image reconstruction and synthesis problems, including image inpainting [36], face completion [37], and face super-resolution [38]. Ledig et al. [39] adopt the GAN framework for learning natural image super-resolution. The ResNet [6] architecture is used as the generative network and train the network using the combination of the $\mathcal{L}_2$ loss, perceptual loss [40], and adversarial loss. The SR results may have lower PSNR but are visually plausible. Note that our LapSRN can be easily extended to incorporate adversarial training. We present experimental results on training with the adversarial loss in Section 5.6.

3 Deep Laplacian Pyramid Network for SR

In this section, we describe the design methodology of the proposed LapSRN, including the network architecture, parameter sharing, loss functions, multi-scale training strategy, and details of implementation as well as network training.

3.1 Network Architecture

We construct our network based on the Laplacian pyramid framework. Our model takes an LR image as input (rather than an upscaled version of the LR image) and progressively predicts residual images on the $\log_2 S$ pyramid levels, where S is the upsampling scale factor. For example, our network consists of 3 pyramid levels for super-resolving an LR image at a scale factor of 8. Our model consists of two branches: (1) feature extraction and (2) image reconstruction.

3.1.1 *Feature Extraction Branch*

As illustrated in Figs. 1c and 3, the feature extraction branch consists of (1) a feature embedding sub-network for transforming high-dimensional non-linear feature maps, (2) a transposed convolutional layer for upsampling the extracted features by a scale of 2, and (3) a convolutional layer ($\mathrm{Conv}_{\mathrm{res}}$) for predicting the sub-band residual image. The first pyramid level has an additional convolutional layer ($\mathrm{Conv}_{\mathrm{in}}$) to extract high-dimensional feature maps from the input LR image. At other levels, the feature embedding sub-network directly transforms features from the upscaled feature maps at the previous pyramid level. Unlike the design of the LAPGAN in Fig. 2, we do not *collapse* the feature maps into an image before feeding into the next level. Therefore, the feature representations at lower levels are connected to higher levels and thus can increase the non-linearity of the network to learn complex mappings at the finer levels. Note that we perform the feature extraction at the *coarse* resolution and generate feature maps at the *finer* resolution with only one transposed convolutional layer. In contrast to existing networks (e.g., [11], [13]) that perform all feature extraction and reconstruction at the finest resolution, our network design significantly reduces the computational complexity.

3.1.2 *Image Reconstruction Branch*

At level s, the input image is upsampled by a scale of 2 with a transposed convolutional layer, which is initialized with a

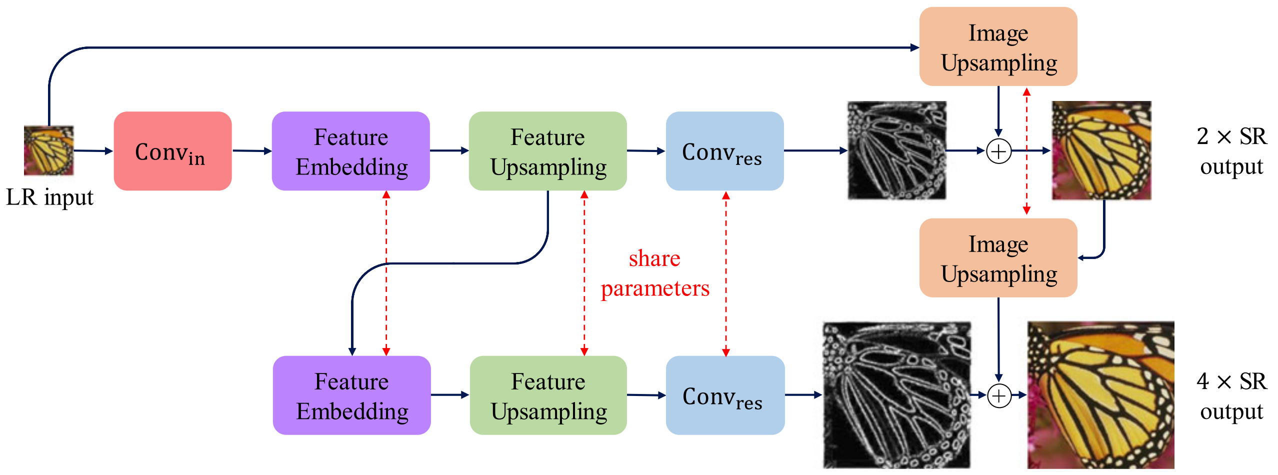


Fig. 3. *Detailed network architecture of the proposed LapSRN.* At each pyramid level, our model consists of a feature embedding sub-network for extracting non-linear features, transposed convolutional layers for upsampling feature maps and images, and a convolutional layer for predicting the sub-band residuals. As the network structure at each level is highly similar, we share the weights of those components across pyramid levels to reduce the number of network parameters.

4×4 bilinear kernel. We then combine the upsampled image (using element-wise summation) with the predicted residual image to generate a high-resolution output image. The reconstructed HR image at level s is then used as an input for the image reconstruction branch at level $s+1$. The entire network is a cascade of CNNs with the same structure at each level. We jointly optimize the upsampling layer with all other layers to learn better a upsampling function.

3.2 Feature Embedding Sub-Network

In our preliminary work [16], we use a stack of multiple convolutional layers as our feature embedding sub-network. In addition, we learn distinct sets of convolutional filters for feature transforming and upsampling at different pyramid levels. Consequently, the number of network parameters increases with the depth of the feature embedding sub-network and the upsampling scales, e.g., the $4\times$ SR model has about twice number of parameters than the $2\times$ SR model. In this work, we explore two directions to reduce the network parameters of LapSRN.

3.2.1 Parameter Sharing Across Pyramid Levels

Our first strategy is to share the network parameters *across* pyramid levels as the network at each level shares the same structure and the task (i.e., predicting the residual images at $2\times$ resolution). As shown in Fig. 3, we share the parameters of the feature embedding sub-network, upsampling layers, and the residual prediction layers across all the pyramid levels. As a result, the number of network parameters is independent of the upsampling scales. We can use a single set of parameters to construct multi-level LapSRN models to handle different upsampling scales.

3.2.2 Parameter Sharing within Pyramid Level

Our second strategy is to share the network parameters *within* each pyramid level. Specifically, we extend the feature embedding sub-network using deeply recursive layers to effectively increase the network depth without increasing the number of parameters. The design of recursive layers has been adopted by several recent CNN-based SR approaches. The DRCN method [12] applies a *single* convolutional layer repeatedly up to 16 times. However, with a large number of filters (i.e., 256 filters), the DRCN is memory-demanding and slow at runtime. Instead of reusing the weights of a single convolutional layer, the DRRN [13] method shares the weights of a *block* (2 convolutional layers with 128 filters). In addition, the DRRN introduces a variant of local residual learning from the ResNet [6]. Specifically, the identity branch of the ResNet comes from the output of the *previous* block, while the identity branch of the DRRN comes from the input of the *first* block. Such a local skip connection in the DRRN creates multiple short paths from input to output and thereby effectively alleviates the gradient vanishing and exploding problems. Therefore, DRRN has 52 convolutional layers with only 297 k parameters.

In the proposed LapSRN, the feature embedding sub-network has R recursive blocks. Each recursive block has D distinct convolutional layers, which controls the number of parameters in the entire model. The weights of the D convolutional layers are shared among the recursive blocks. Given an upsampling scale factor S, the depth of the LapSRN can be computed by

$$\text{depth} = (D \times R + 1) \times L + 2, \tag{1}$$

where $L = \log_2 S$. The 1 within the parentheses represents the transposed convolutional layers, and the 2 at the end of (1) represents the first convolutional layer applied on input images and the last convolutional layer for predicting residuals. Here we define the depth of a network as the longest path from input to output.

3.2.3 Local Residual Learning

As the gradient vanishing and exploding problem are common issues when training deep models, we explore three different methods of local residual learning in our feature embedding sub-network to stabilize our training process:

1) *No skip connection*: A plain network without any local skip connection. We denote our LapSRN without skip connections as $\text{LapSRN}_{\textbf{NS}}$.

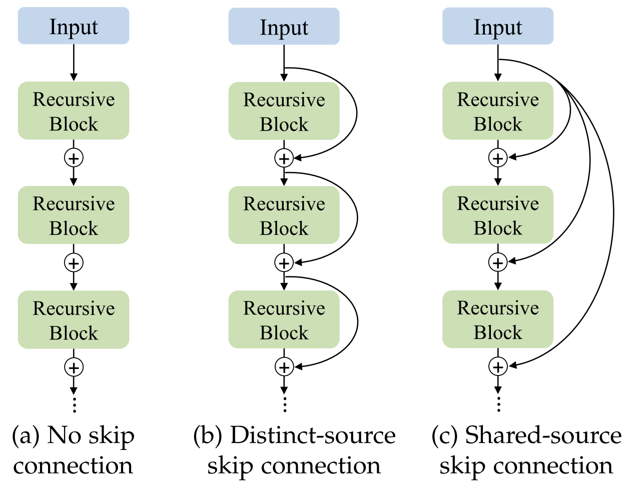


Fig. 4. *Local residual learning.* We explore three different ways of local skip connection in the feature embedding sub-network of the LapSRN for training deeper models.

2) *Distinct-source skip connection*: The ResNet-style local skip connection. We denote our LapSRN with such skip connections as $\text{LapSRN}_{\textbf{DS}}$.
3) *Shared-source skip connection*: The local skip connection introduced by DRRN [13]. We denote our LapSRN with such skip connections as $\text{LapSRN}_{\textbf{SS}}$.

We illustrate the three local residual learning methods in Fig. 4 and the detailed structure of our recursive block in Fig. 5. We use the pre-activation structure [41] without the batch normalization layer in our recursive block.

3.3 Loss Function

Let x be the input LR image and θ be the set of network parameters to be optimized. Our goal is to learn a mapping function f for generating an HR image $\hat{y} = f(x;\theta)$ that is as similar to the ground truth HR image y as possible. We denote the residual image at level l by $\hat{r}_l$, the upscaled LR image by x_l and the corresponding HR images by $\hat{y}_l$. The desired output HR images at level l is modeled by $\hat{y}_l = x_l + \hat{r}_l$. We use the bicubic downsampling to resize the ground truth HR image y to y_l at each level. Instead of minimizing the mean square errors between $\hat{y}_l$ and y_l, we propose to use a robust loss function to handle outliers. The overall loss function is defined as

$$\begin{aligned}\mathcal{L}_S(y,\hat{y};\theta) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L}\rho\left(y_l^{(i)} - \hat{y}_l^{(i)}\right)\\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L}\rho\left(\left(y_l^{(i)} - x_l^{(i)}\right) - \hat{r}_l^{(i)}\right),\end{aligned} \tag{2}$$

where $\rho(x) = \sqrt{x^2+\epsilon^2}$ is the Charbonnier penalty function (a differentiable variant of $\mathcal{L}_1$ norm) [42], N is the number of training samples in each batch, S is the target upsampling scale factor, and $L = \log_2 S$ is the number of pyramid levels in our model. We empirically set ϵ to $1e-3$.

In the proposed LapSRN, each level s has its own loss function and the corresponding ground truth HR image y_s. This multi-loss structure resembles the deeply-supervised networks for classification [43] and edge detection [44]. The deep multi-scale supervision guides the network to reconstruct HR images in a coarse-to-fine fashion and reduce spatial aliasing artifacts.

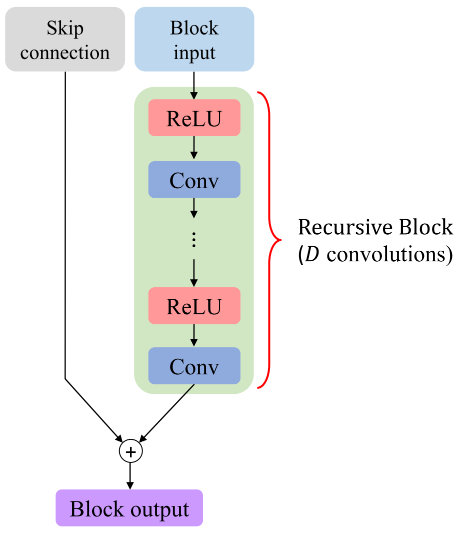


Fig. 5. *Structure of our recursive block.* There are D convolutional layers in a recursive block. The weights of convolutional layers are distinct within the block but shared among all recursive blocks. We use the pre-activation structure [41] without the batch normalization layer.

3.4 Multi-Scale Training

The multi-scales SR models (i.e., trained with samples from multiple upsampling scales simultaneously) have been shown more effective than single-scale models as SR tasks have inter-scale correlations. For pre-upsampling based SR methods (e.g., VDSR [11] and DRRN [13]), the input and output of the network have the same spatial resolution, and the outputs of different upsampling scales are generated from the *same* layer of the network. In the proposed LapSRN, samples of different upsampling scales are generated from *different* layers and have different spatial resolutions. In this work, we use $2\times$, $4\times$, and $8\times$ SR samples to train a multi-scale LapSRN model. We construct a 3-level LapSRN model and minimize the combination of loss functions from three different scales

$$\mathcal{L}(y,\hat{y};\theta) = \sum_{S\in\{2,4,8\}}\mathcal{L}_S(y,\hat{y};\theta). \tag{3}$$

We note that the pre-upsampling based SR methods could apply scale augmentation for arbitrary upsampling scales, while in our LapSRN, the upsampling scales for training are limited to $2^n\times$ SR where n is an integer.

3.5 Implementation and Training Details

In the proposed LapSRN, we use 64 filters in all convolutional layers except the first layer applied on the input LR image, the layers for predicting residuals, and the image upsampling layer. The filter size of the convolutional and transposed convolutional layers are 3×3 and 4×4, respectively. We pad zeros around the boundaries before applying convolution to keep the size of all feature maps the same as the input of each level. We initialize the convolutional filters using the method of He et al. [45] and use the leaky rectified linear units (LReLUs) [46] with a negative slope of 0.2 as the non-linear activation function.

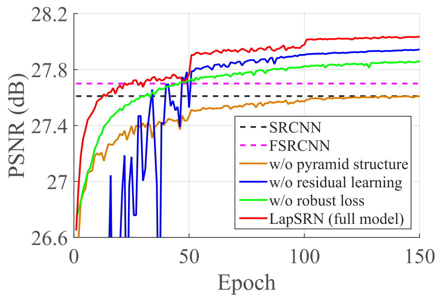


Fig. 6. *Convergence analysis.* We analyze the contributions of the pyramid structures, loss functions, and global residual learning by replacing each component with the one used in existing methods. Our full model converges faster and achieves better performance.

We use 91 images from Yang et al. [2] and 200 images from the training set of the Berkeley Segmentation Dataset [47] as our training data. The training dataset of 291 images is commonly used in the state-of-the-art SR methods [5], [11], [13], [16]. We use a batch size of 64 and crop the size of HR patches to 128×128. An epoch has 1,000 iterations of back-propagation. We augment the training data in three ways: (1) *Scaling*: randomly downscale images between $[0.5, 1.0]$; (2) *Rotation*: randomly rotate image by 90°, 180°, or 270°; (3) *Flipping*: flip images horizontally with a probability of 0.5. Following the training protocol of existing methods [9], [11], [13], we generate the LR training patches using the bicubic downsampling. We use the MatConvNet toolbox [48] and train our model using the Stochastic Gradient Descent (SGD) solver. In addition, we set the momentum to 0.9 and the weight decay to $1e-4$. The learning rate is initialized to $1e-5$ for all layers and decreased by a factor of 2 for every 100 epochs.

4 Discussions and Analysis

In this section, we first validate the contributions of different components in the proposed network. We then discuss the effect of local residual learning and parameter sharing in our feature embedding sub-network. Finally, we analyze the performance of multi-scale training strategy.

4.1 Model Design

We train a LapSRN model with 5 convolutional layers (without parameters sharing and the recursive layers) at each pyramid level to analyze the performance of pyramid network structure, global residual learning, robust loss functions, and multi-scale supervision.

4.1.1 *Pyramid Structure*

By removing the pyramid structure, our model falls back to a network similar to the FSRCNN but with the global residual learning. We train this network using 10 convolutional layers in order to have the same depth as our LapSRN. Fig. 6 shows the convergence curves in terms of PSNR on the Set14 for $4\times$ SR. The quantitative results in Table 2 and Fig. 7 show that the pyramid structure leads to considerable performance improvement (e.g., 0.7 dB on Set5 and 0.4 dB on Set14), which validates the effectiveness of our Laplacian pyramid network design.

TABLE 2
Ablation Study of LapSRN

GRL	Pyramid	Loss	Set5	Set14
✓		Charbonnier	30.58	27.61
	✓	Charbonnier	31.10	27.94
✓	✓	$\mathcal{L}_2$	30.93	27.86
✓	✓	Charbonnier	**31.28**	**28.04**

Our full model performs favorably against several variants of the LapSRN on both Set5 and Set14 for 4× SR.

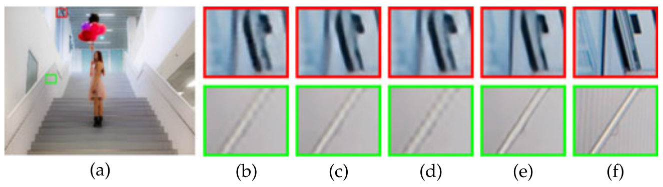


Fig. 7. *Contribution of different components in LapSRN.* (a) Ground truth HR image (b) without pyramid structure (c) without global residual learning (d) without robust loss (e) full model (f) HR patch.

4.1.2 *Global Residual Learning*

To demonstrate the effectiveness of global residual learning, we remove the image reconstruction branch and directly predict the HR images at each level. In Fig. 6, the performance of the non-residual network (blue curve) converges slowly and fluctuates significantly during training. Our full LapSRN model (red curve), on the other hand, outperforms the SRCNN within 10 epochs.

4.1.3 *Loss Function*

To validate the effectiveness of the Charbonnier loss function, we train the proposed network with conventional $\mathcal{L}_2$ loss function. We use a larger learning rate ($1e-4$) since the gradient magnitude of the $\mathcal{L}_2$ loss is smaller. As illustrated in Fig. 6, the network optimized with the $\mathcal{L}_2$ loss (green curve) requires more iterations to achieve comparable performance with SRCNN. In Fig. 7, we show that the SR images reconstruct by our full model contain relatively clean and sharp details.

4.1.4 *Multi-scale Supervision*

As described in Section 3.3, we use the multiple loss functions to supervise the intermediate output at each pyramid level. We show the intermediate output images at each pyramid scale in Fig. 8. The model without the multi-scale supervision (i.e., only applying supervision at the finest scale) cannot reduce the spatial aliasing artifacts well, while our LapSRN progressively reconstructs clear and sharp straight lines.

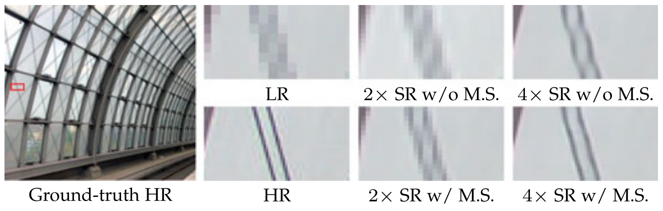


Fig. 8. *Contribution of multi-scale supervision (M.S.).* The multi-scale supervision guides the network training to progressively reconstruct the HR images and help reduce the spatial aliasing artifacts.

TABLE 3
Parameter Sharing in LapSRN

Model	#Parameters	BSDS100	URBAN100
LapSRN [16]	812k	**27.32**	**25.21**
LapSRN$_{\mathbf{NS}}$-D10R1	407k	**27.32**	25.20
LapSRN$_{\mathbf{NS}}$-D5R2	222k	27.30	25.16
LapSRN$_{\mathbf{NS}}$-D2R5	112k	27.26	25.10

We reduce the number of network parameters by sharing the weights between pyramid levels and applying recursive layers in the feature embedding sub-network.

4.2 Parameter Sharing

In this section, we reduce the network parameters in our LapSRN by sharing weights *across* and *within* pyramid levels and discuss the performance contribution.

4.2.1 Parameter Sharing Across Pyramid Levels

Our preliminary LapSRN $4\times$ model [16] has 812k parameters as each pyramid level has distinct convolutional and transposed convolutional layers. By sharing the weights across pyramid levels as shown in Fig. 3, we reduce the number of parameters to 407k. Such model has 10 convolutional layers, 1 recursive block, and does not use any local residual learning strategies. We denote this model by LapSRN$_{\mathbf{NS}}$-D10R1. We compare the above models on the BSDS100 and URBAN100 datasets for $4\times$ SR. Table 3 shows that the LapSRN$_{\mathbf{NS}}$-D10R1 achieves comparable performance with the LapSRN [16] while using only half of the network parameters.

4.2.2 Parameter Sharing within Pyramid Levels

We further reduce the network parameters by decreasing the number of convolutional layers (D) and increasing the number of recursive blocks (R). We train another two models: LapSRN$_{\mathbf{NS}}$-D5R2 and LapSRN$_{\mathbf{NS}}$-D2R5, which have 222 and 112 k parameters, respectively. As shown in Table 3, while the LapSRN$_{\mathbf{NS}}$-D5R2 and LapSRN$_{\mathbf{NS}}$-D2R5 have fewer parameters, we observe the performance drop, particularly on the challenging URBAN100 dataset.

4.3 Training Deeper Models

In Section 4.2, we show that we can achieve comparable performance to the preliminary LapSRN by using only half or 27 percent of parameters. Next, we train deeper models to

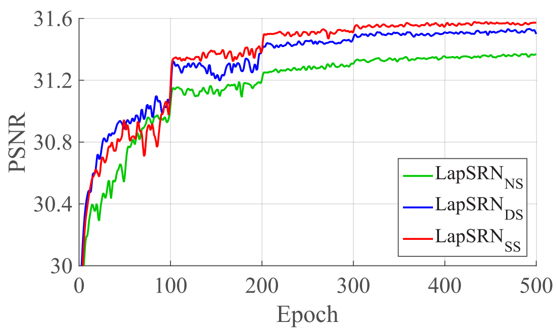


Fig. 9. *Comparisons of local residual learning*. We train our LapSRN-D5R5 model with three different local residual learning methods as described in Section 3.2.3 and evaluate on the SET5 for $4\times$ SR.

TABLE 4
Quantitative Evaluation of Local Residual Learning

Model	Depth	LapSRN$_{\mathbf{NS}}$	LapSRN$_{\mathbf{DS}}$	LapSRN$_{\mathbf{SS}}$
D5R2	24	25.16	25.22	**25.23**
D5R5	54	25.18	25.33	**25.34**
D5R8	84	25.26	25.33	**25.38**

We compare three different local residual learning methods on the URBAN100 dataset for $4\times$ SR. Overall, the shared local skip connection method (LapSRN$_{\mathbf{SS}}$) achieves superior performance for deeper models.

improve the performance without increasing the number of the network parameters.

4.3.1 Local Residual Learning

We increase the number of recursive blocks in our feature embedding sub-network to increase the depth of network but keep the number of parameters the same. We test three LapSRN models: D5R2, D5R5, and D5R8, which have 5 distinct convolutional layers with 2, 5 and 8 recursive blocks, respectively. We train the models with three different local residual learning methods as described in Section 3.2.3. We plot the convergence curves of the LapSRN-D5R5 in Fig. 9 and present the quantitative evaluation in Table 4. Overall, the shared-source local skip connection method (LapSRN$_{\mathbf{SS}}$) performs favorably against other alternatives, particularly for deeper models (i.e., more recursive blocks).

4.3.2 Study of D and R

Our feature embedding sub-network consists of R recursive blocks, and each recursive block has D distinct convolutional layers which are shared among all the recursive blocks. Here we extensively evaluate the contributions of R and D to the reconstruction accuracy. We use D $= 2, 4, 5, 10$ to construct models with different network depth. We use the shared-source local skip connection for all the evaluated models. We show the quantitative evaluation in Table 5 and visualize the performance over the network depth in Fig. 10. While the D2R5, D5R2, and D10R1 models perform comparably,

TABLE 5
Quantitative Evaluation of the Number of Recursive Blocks **R** and the Number of Convolutional Layers **D** in Our Feature Embedding Sub-Network

Model	#Parameters	Depth	BSDS100	URBAN100
D2R5	112 k	24	27.33	25.24
D2R12	112 k	52	27.35	**25.31**
D2R20	112 k	84	**27.37**	**25.31**
D4R3	185 k	28	27.33	25.25
D4R6	185 k	52	**27.37**	25.34
D4R10	185 k	84	**27.37**	**25.35**
D5R2	222 k	24	27.32	25.23
D5R5	222 k	54	27.38	25.34
D5R8	222 k	84	**27.39**	**25.38**
D10R1	407 k	24	27.33	25.23
D10R2	407 k	44	27.36	25.27
D10R4	407 k	84	**27.38**	**25.36**

We build LapSRN with different network depth by varying the values of D and R and evaluate on the BSDS100 and URBAN100 datasets for $4\times$ SR.

TABLE 6
Quantitative Evaluation of State-of-the-Art SR Algorithms

Algorithm	Scale	SET5	SET14	BSDS100	URBAN100	MANGA109
		PSNR / SSIM / IFC	PSNR / SSIM / IFC	PSNR / SSIM / IFC	PSNR / SSIM / IFC	PSNR / SSIM / IFC
Bicubic		33.69 / 0.931 / 6.166	30.25 / 0.870 / 6.126	29.57 / 0.844 / 5.695	26.89 / 0.841 / 6.319	30.86 / 0.936 / 6.214
A+ [3]		36.60 / 0.955 / 8.715	32.32 / 0.906 / 8.200	31.24 / 0.887 / 7.464	29.25 / 0.895 / 8.440	35.37 / 0.968 / 8.906
RFL [5]		36.59 / 0.954 / 8.741	32.29 / 0.905 / 8.224	31.18 / 0.885 / 7.473	29.14 / 0.891 / 8.439	35.12 / 0.966 / 8.921
SelfExSR [22]		36.60 / 0.955 / 8.404	32.24 / 0.904 / 8.018	31.20 / 0.887 / 7.239	29.55 / 0.898 / 8.414	35.82 / 0.969 / 8.721
SRCNN [9]		36.72 / 0.955 / 8.166	32.51 / 0.908 / 7.867	31.38 / 0.889 / 7.242	29.53 / 0.896 / 8.092	35.76 / 0.968 / 8.471
FSRCNN [15]		37.05 / 0.956 / 8.199	32.66 / 0.909 / 7.841	31.53 / 0.892 / 7.180	29.88 / 0.902 / 8.131	36.67 / 0.971 / 8.587
SCN [10]	2×	36.58 / 0.954 / 7.358	32.35 / 0.905 / 7.085	31.26 / 0.885 / 6.500	29.52 / 0.897 / 7.324	35.51 / 0.967 / 7.601
VDSR [11]		37.53 / 0.959 / 8.190	33.05 / 0.913 / 7.878	31.90 / 0.896 / 7.169	30.77 / 0.914 / 8.270	37.22 / 0.975 / 9.120
DRCN [12]		37.63 / 0.959 / 8.326	33.06 / 0.912 / 8.025	31.85 / 0.895 / 7.220	30.76 / 0.914 / 8.527	37.63 / 0.974 / 9.541
LapSRN [16]		37.52 / 0.959 / 9.010	33.08 / 0.913 / 8.501	31.80 / 0.895 / 7.715	30.41 / 0.910 / 8.907	37.27 / 0.974 / 9.481
DRRN [13]		37.74 / 0.959 / 8.671	33.23 / 0.914 / 8.320	**32.05** / 0.897 / 7.613	**31.23** / **0.919** / 8.917	**37.92** / **0.976** / 9.268
MS-LapSRN-D5R2 (ours)		37.62 / **0.960** / 9.038	33.13 / 0.913 / 8.539	31.93 / 0.897 / 7.776	30.82 / 0.915 / 9.081	37.38 / 0.975 / 9.434
MS-LapSRN-D5R5 (ours)		37.72 / **0.960** / 9.265	33.24 / 0.914 / 8.726	32.00 / **0.898** / 7.906	31.01 / 0.917 / 9.334	37.71 / 0.975 / 9.710
MS-LapSRN-D5R8 (ours)		**37.78** / **0.960** / **9.305**	**33.28** / **0.915** / **8.748**	**32.05** / **0.898** / **7.927**	31.15 / **0.919** / **9.406**	37.78 / **0.976** / **9.765**
Bicubic		30.41 / 0.869 / 3.596	27.55 / 0.775 / 3.491	27.22 / 0.741 / 3.168	24.47 / 0.737 / 3.661	26.99 / 0.859 / 3.521
A+ [3]		32.62 / 0.909 / 4.979	29.15 / 0.820 / 4.545	28.31 / 0.785 / 4.028	26.05 / 0.799 / 4.883	29.93 / 0.912 / 4.880
RFL [5]		32.47 / 0.906 / 4.956	29.07 / 0.818 / 4.533	28.23 / 0.782 / 4.023	25.88 / 0.792 / 4.781	29.61 / 0.905 / 4.758
SelfExSR [22]		32.66 / 0.910 / 4.911	29.18 / 0.821 / 4.505	28.30 / 0.786 / 3.923	26.45 / 0.810 / 4.988	27.57 / 0.821 / 2.193
SRCNN [9]		32.78 / 0.909 / 4.682	29.32 / 0.823 / 4.372	28.42 / 0.788 / 3.879	26.25 / 0.801 / 4.630	30.59 / 0.914 / 4.698
FSRCNN [15]		33.18 / 0.914 / 4.970	29.37 / 0.824 / 4.569	28.53 / 0.791 / 4.061	26.43 / 0.808 / 4.878	31.10 / 0.921 / 4.912
SCN [10]	3×	32.62 / 0.908 / 4.321	29.16 / 0.818 / 4.006	28.33 / 0.783 / 3.553	26.21 / 0.801 / 4.253	30.22 / 0.914 / 4.302
VDSR [11]		33.67 / 0.921 / 5.088	29.78 / 0.832 / 4.606	28.83 / 0.799 / 4.043	27.14 / 0.829 / 5.045	32.01 / 0.934 / 5.389
DRCN [12]		33.83 / 0.922 / 5.202	29.77 / 0.832 / 4.686	28.80 / 0.797 / 4.070	27.15 / 0.828 / 5.187	32.31 / 0.936 / 5.564
LapSRN [16]		33.82 / 0.922 / 5.194	29.87 / 0.832 / 4.662	28.82 / 0.798 / 4.057	27.07 / 0.828 / 5.168	32.21 / 0.935 / 5.406
DRRN [13]		34.03 / **0.924** / **5.397**	29.96 / 0.835 / **4.878**	**28.95** / 0.800 / **4.269**	**27.53** / 0.764 / **5.456**	**32.74** / **0.939** / **5.659**
MS-LapSRN-D5R2 (ours)		33.88 / 0.923 / 5.165	29.89 / 0.834 / 4.637	28.87 / 0.800 / 4.040	27.23 / 0.831 / 5.142	32.28 / 0.936 / 5.384
MS-LapSRN-D5R5 (ours)		34.01 / **0.924** / 5.307	29.96 / **0.836** / 4.758	28.92 / 0.801 / 4.127	27.39 / 0.835 / 5.333	32.60 / 0.938 / 5.559
MS-LapSRN-D5R8 (ours)		**34.06** / **0.924** / 5.390	**29.97** / **0.836** / 4.806	28.93 / **0.802** / 4.154	27.47 / **0.837** / 5.409	32.68 / **0.939** / 5.621
Bicubic		28.43 / 0.811 / 2.337	26.01 / 0.704 / 2.246	25.97 / 0.670 / 1.993	23.15 / 0.660 / 2.386	24.93 / 0.790 / 2.289
A+ [3]		30.32 / 0.860 / 3.260	27.34 / 0.751 / 2.961	26.83 / 0.711 / 2.565	24.34 / 0.721 / 3.218	27.03 / 0.851 / 3.177
RFL [5]		30.17 / 0.855 / 3.205	27.24 / 0.747 / 2.924	26.76 / 0.708 / 2.538	24.20 / 0.712 / 3.101	26.80 / 0.841 / 3.055
SelfExSR [22]		30.34 / 0.862 / 3.249	27.41 / 0.753 / 2.952	26.84 / 0.713 / 2.512	24.83 / 0.740 / 3.381	27.83 / 0.866 / 3.358
SRCNN [9]		30.50 / 0.863 / 2.997	27.52 / 0.753 / 2.766	26.91 / 0.712 / 2.412	24.53 / 0.725 / 2.992	27.66 / 0.859 / 3.045
FSRCNN [15]		30.72 / 0.866 / 2.994	27.61 / 0.755 / 2.722	26.98 / 0.715 / 2.370	24.62 / 0.728 / 2.916	27.90 / 0.861 / 2.950
SCN [10]	4×	30.41 / 0.863 / 2.911	27.39 / 0.751 / 2.651	26.88 / 0.711 / 2.309	24.52 / 0.726 / 2.860	27.39 / 0.857 / 2.889
VDSR [11]		31.35 / 0.883 / 3.496	28.02 / 0.768 / 3.071	27.29 / 0.726 / 2.627	25.18 / 0.754 / 3.405	28.83 / 0.887 / 3.664
DRCN [12]		31.54 / 0.884 / 3.502	28.03 / 0.768 / 3.066	27.24 / 0.725 / 2.587	25.14 / 0.752 / 3.412	28.98 / 0.887 / 3.674
LapSRN [16]		31.54 / 0.885 / 3.559	28.19 / 0.772 / 3.147	27.32 / 0.727 / 2.677	25.21 / 0.756 / 3.530	29.09 / 0.890 / 3.729
DRRN [13]		31.68 / 0.888 / 3.703	28.21 / 0.772 / 3.252	27.38 / 0.728 / **2.760**	25.44 / 0.764 / 3.700	29.46 / 0.896 / 3.878
MS-LapSRN-D5R2 (ours)		31.62 / 0.887 / 3.585	28.16 / 0.772 / 3.151	27.36 / 0.729 / 2.684	25.32 / 0.760 / 3.537	29.18 / 0.892 / 3.750
MS-LapSRN-D5R5 (ours)		**31.74** / 0.888 / 3.705	28.25 / 0.773 / 3.238	27.42 / **0.731** / 2.737	25.45 / 0.765 / 3.674	29.48 / 0.896 / 3.888
MS-LapSRN-D5R8 (ours)		**31.74** / **0.889** / **3.749**	**28.26** / **0.774** / **3.261**	**27.43** / **0.731** / 2.755	**25.51** / **0.768** / **3.727**	**29.54** / **0.897** / **3.928**
Bicubic		24.40 / 0.658 / 0.836	23.10 / 0.566 / 0.784	23.67 / 0.548 / 0.646	20.74 / 0.516 / 0.858	21.47 / 0.650 / 0.810
A+ [3]		25.53 / 0.693 / 1.077	23.89 / 0.595 / 0.983	24.21 / 0.569 / 0.797	21.37 / 0.546 / 1.092	22.39 / 0.681 / 1.056
RFL [5]		25.38 / 0.679 / 0.991	23.79 / 0.587 / 0.916	24.13 / 0.563 / 0.749	21.27 / 0.536 / 0.992	22.28 / 0.669 / 0.968
SelfExSR [22]		25.49 / 0.703 / 1.121	23.92 / 0.601 / 1.005	24.19 / 0.568 / 0.773	21.81 / 0.577 / 1.283	22.99 / 0.719 / 1.244
SRCNN [9]		25.33 / 0.690 / 0.938	23.76 / 0.591 / 0.865	24.13 / 0.566 / 0.705	21.29 / 0.544 / 0.947	22.46 / 0.695 / 1.013
FSRCNN [15]		25.60 / 0.697 / 1.016	24.00 / 0.599 / 0.942	24.31 / 0.572 / 0.767	21.45 / 0.550 / 0.995	22.72 / 0.692 / 1.009
SCN [10]	8×	25.59 / 0.706 / 1.063	24.02 / 0.603 / 0.967	24.30 / 0.573 / 0.777	21.52 / 0.560 / 1.074	22.68 / 0.701 / 1.073
VDSR [11]		25.93 / 0.724 / 1.199	24.26 / 0.614 / 1.067	24.49 / 0.583 / 0.859	21.70 / 0.571 / 1.199	23.16 / 0.725 / 1.263
DRCN [12]		25.93 / 0.723 / 1.192	24.25 / 0.614 / 1.057	24.49 / 0.582 / 0.854	21.71 / 0.571 / 1.197	23.20 / 0.724 / 1.257
LapSRN [16]		26.15 / 0.738 / 1.302	24.35 / 0.620 / 1.133	24.54 / 0.586 / 0.893	21.81 / 0.581 / 1.288	23.39 / 0.735 / 1.352
DRRN [13]		26.18 / 0.738 / 1.307	24.42 / 0.622 / 1.127	24.59 / 0.587 / 0.891	21.88 / 0.583 / 1.299	23.60 / 0.742 / 1.406
MS-LapSRN-D5R2 (ours)		26.20 / 0.747 / 1.366	24.45 / 0.626 / 1.170	24.61 / 0.590 / 0.920	21.95 / 0.592 / 1.364	23.70 / 0.751 / 1.470
MS-LapSRN-D5R5 (ours)		**26.34** / 0.752 / 1.414	**24.57** / **0.629** / 1.200	**24.65** / 0.591 / 0.938	**22.06** / 0.597 / 1.426	23.85 / 0.756 / 1.538
MS-LapSRN-D5R8 (ours)		**26.34** / **0.753** / **1.435**	**24.57** / **0.629** / **1.209**	**24.65** / **0.592** / **0.943**	**22.06** / **0.598** / **1.446**	**23.90** / **0.759** / **1.564**

We report the average PSNR/SSIM/IFC for $2\times$, $3\times$, $4\times$ and $8\times$ SR. ***Red*** *and blue indicate the best and the second best performance, respectively. Both LapSRN [16] and the proposed MS-LapSRN do not use any $3\times$ SR images for training. To generate the results of $3\times$ SR, we first perform $4\times$ SR on input LR images and then downsample the output to the target resolution.*

the D5R8 method achieves the best reconstruction accuracy when the network depth is more than 80.

4.4 Multi-Scale Training

We train our LapSRN using the multi-scale training strategy (Section 3.4). As our pyramid network design only accounts for training with $2^n\times$ samples, we train our $\text{LapSRN}_{\text{SS}}$-D5R8 model with the following scale combinations: $\{2\times\}$, $\{4\times\}$, $\{8\times\}$, $\{2\times, 4\times\}$, $\{2\times, 8\times\}$, $\{4\times, 8\times\}$ and $\{2\times, 4\times, 8\times\}$. During training, we equally split a batch of samples for every upsampling scale. Note that all these models have the same numbers of parameters due to parameter sharing. We evaluate the above models for $2\times$, $4\times$ and $8\times$ SR by constructing LapSRN with the corresponding pyramid levels. We also evaluate $3\times$ SR using our 2-level LapSRN and resizing the network output to the desired spatial resolution. From our

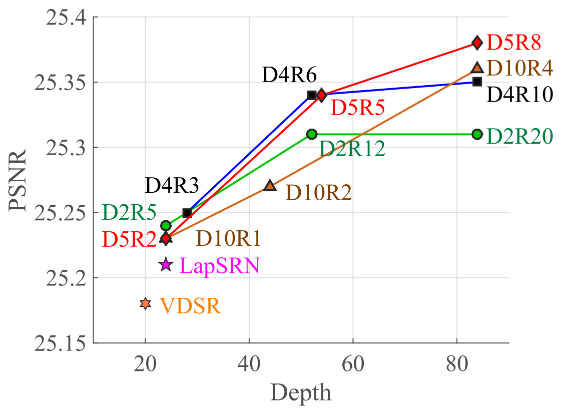


Fig. 10. *PSNR versus network depth*. We test the proposed model with different D and R on the Urban100 dataset for $4\times$ SR.

experimental results, the model trained with $2\times, 4\times$ and $8\times$ samples has the capacity to handle multiple upsampling scales and generalizes well to the unseen $3\times$ SR examples. Furthermore, the multi-scale models perform favorably against the single-scale models, particularly on the Urban100 dataset. Due to the space limit, we present complete quantitative and visual comparisons in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2018.2865304.

5 Experiment Results

In this section, we compare the proposed LapSRN with several state-of-the-art SR methods on benchmark datasets. We present the quantitative evaluation, qualitative comparison, runtime, and parameters comparisons. We then evaluate our method on real-world photos, compare with the LAPGAN [34], and incorporate the adversarial training. In addition, we conduct a human subject study using the pairwise comparison and provide the analysis in the supplementary material, available online. Finally, we discuss the limitation of the proposed method. We provide our source code and SR results generated by all the evaluated methods on our project website at http://vllab.ucmerced.edu/wlai24/LapSRN.

5.1 Comparisons with State-of-the-arts

We compare the proposed method with 10 state-of-the-art SR algorithms, including dictionary-based methods (A+ [3] and RFL [5]), self-similarity based method (SelfExSR [22]), and CNN-based methods (SRCNN [9], FSRCNN [15], SCN [10], VDSR [11], DRCN [12], DRRN [13] and our preliminary approach [16]). We carry out extensive experiments on five public benchmark datasets: Set5 [24], Set14 [27], BSDS100 [47], Urban100 [22] and manga109 [49]. The Set5, Set14 and BSDS100 datasets consist of natural scenes; the Urban100 set contains challenging urban scenes images with details in different frequency bands; and the manga109 is a dataset of Japanese manga.

We evaluate the SR results with three widely used image quality metrics: PSNR, SSIM [50], and IFC [51] and compare performance on $2\times$, $3\times$, $4\times$ and $8\times$ SR. We re-train existing methods for $8\times$ SR using the source code (A+ [3], RFL [5], SRCNN [9], FSRCNN [15], VDSR [11], and DRRN [13]) or our own implementation (DRCN). Both the SelfExSR and SCN methods can naturally handle different scale factors using progressive reconstruction. We use $2\times, 4\times$ and $8\times$ SR

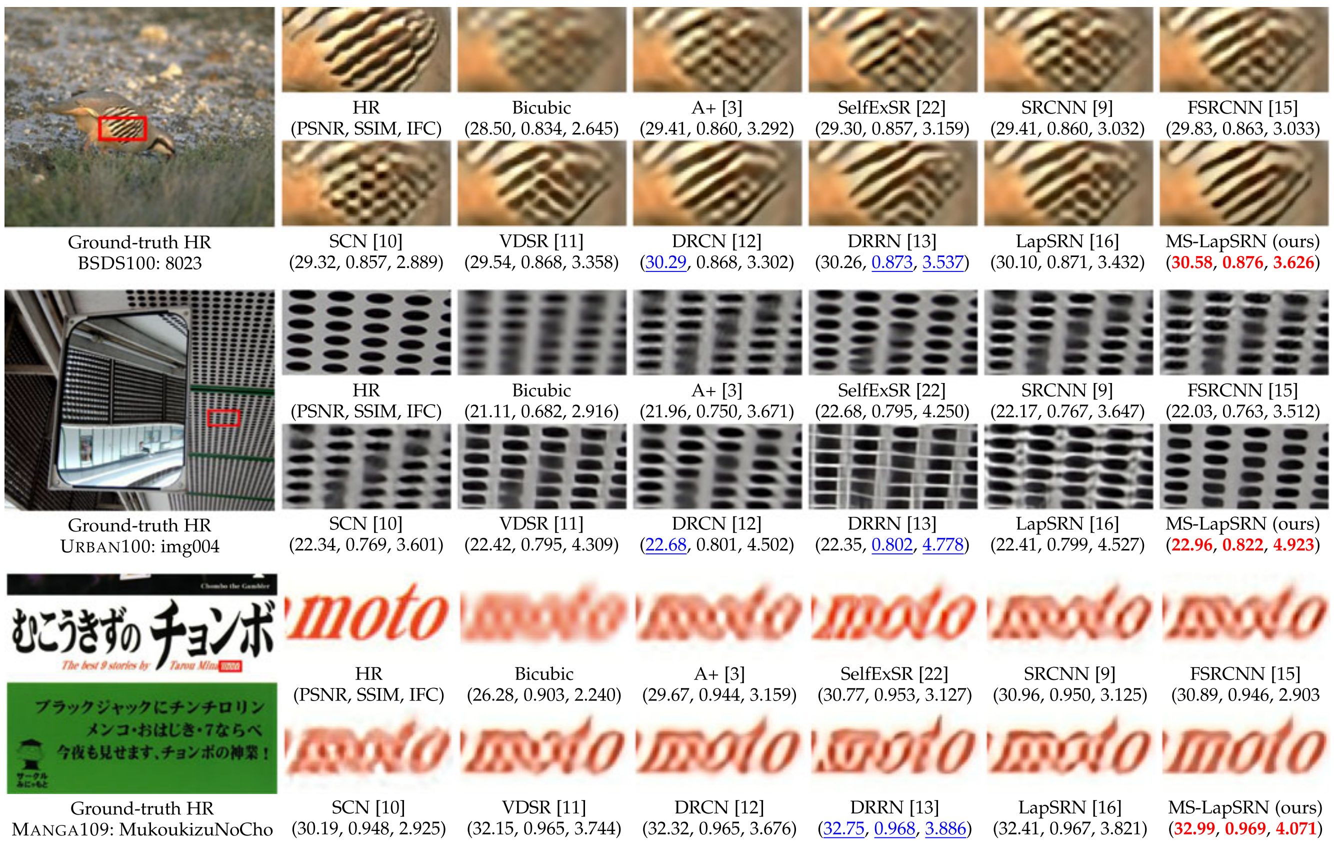


Fig. 11. Visual comparison for $4\times$ SR on the BSDS100, Urban100 and Manga109 datasets.

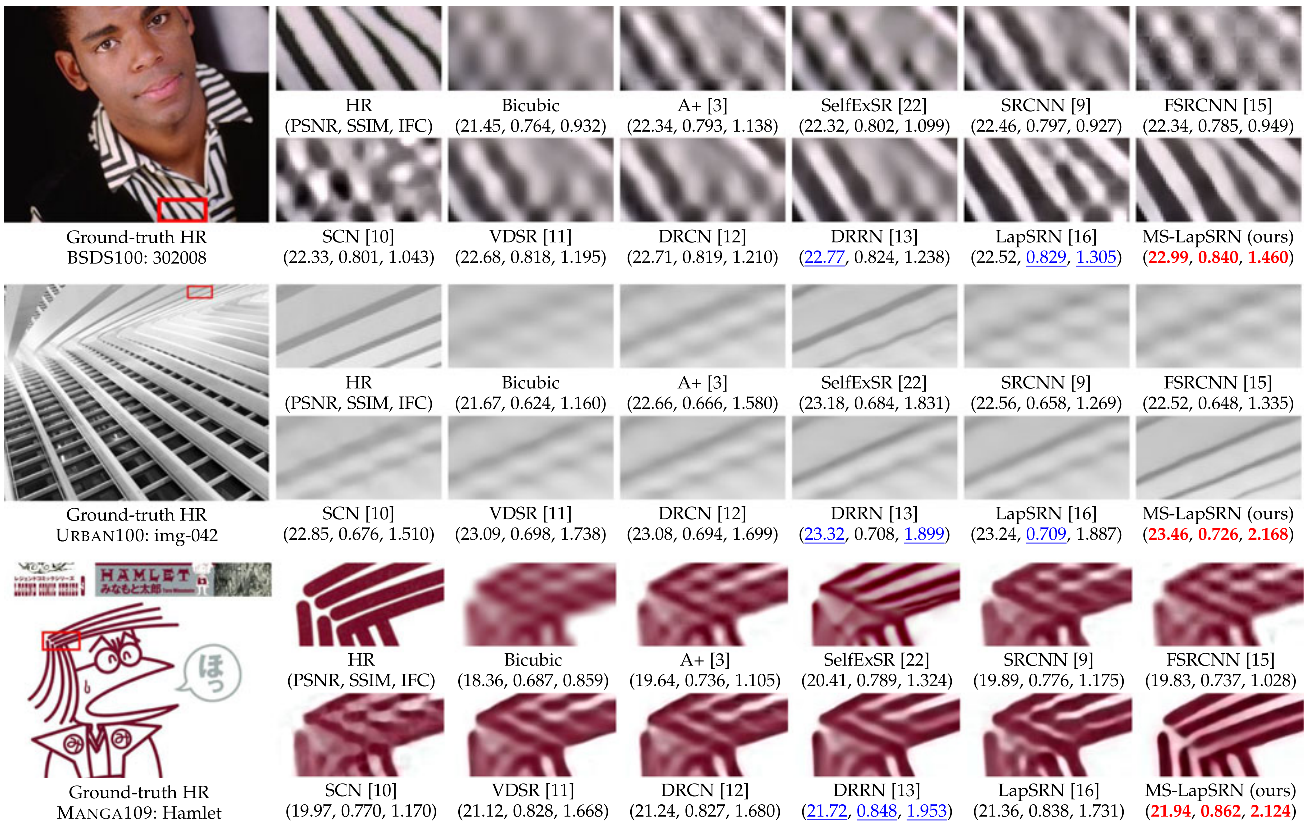

Ground-truth HR
BSDS100: 302008
HR (PSNR, SSIM, IFC)
Bicubic (21.45, 0.764, 0.932)
A+ [3] (22.34, 0.793, 1.138)
SelfExSR [22] (22.32, 0.802, 1.099)
SRCNN [9] (22.46, 0.797, 0.927)
FSRCNN [15] (22.34, 0.785, 0.949)
SCN [10] (22.33, 0.801, 1.043)
VDSR [11] (22.68, 0.818, 1.195)
DRCN [12] (22.71, 0.819, 1.210)
DRRN [13] (22.77, 0.824, 1.238)
LapSRN [16] (22.52, 0.829, 1.305)
MS-LapSRN (ours) (**22.99**, **0.840**, **1.460**)
Ground-truth HR
URBAN100: img-042
HR (PSNR, SSIM, IFC)
Bicubic (21.67, 0.624, 1.160)
A+ [3] (22.66, 0.666, 1.580)
SelfExSR [22] (23.18, 0.684, 1.831)
SRCNN [9] (22.56, 0.658, 1.269)
FSRCNN [15] (22.52, 0.648, 1.335)
SCN [10] (22.85, 0.676, 1.510)
VDSR [11] (23.09, 0.698, 1.738)
DRCN [12] (23.08, 0.694, 1.699)
DRRN [13] (23.32, 0.708, 1.899)
LapSRN [16] (23.24, 0.709, 1.887)
MS-LapSRN (ours) (**23.46**, **0.726**, **2.168**)
Ground-truth HR
MANGA109: Hamlet
HR (PSNR, SSIM, IFC)
Bicubic (18.36, 0.687, 0.859)
A+ [3] (19.64, 0.736, 1.105)
SelfExSR [22] (20.41, 0.789, 1.324)
SRCNN [9] (19.89, 0.776, 1.175)
FSRCNN [15] (19.83, 0.737, 1.028)
SCN [10] (19.97, 0.770, 1.170)
VDSR [11] (21.12, 0.828, 1.668)
DRCN [12] (21.24, 0.827, 1.680)
DRRN [13] (21.72, 0.848, 1.953)
LapSRN [16] (21.36, 0.838, 1.731)
MS-LapSRN (ours) (**21.94**, **0.862**, **2.124**)

Fig. 12. Visual comparison for $8\times$ SR on the BSDS100, URBAN100 and MANGA109 datasets.

samples for training VDSR [11] and DRRN [13] while use only $8\times$ SR samples for other algorithms to follow the training strategies of individual methods.

We compare three variations of the proposed method: (1) LapSRN$_{SS}$-D5R2, which has similar depth as the VDSR [11], DRCN [12] and LapSRN [16], (2) LapSRN$_{SS}$-D5R5, which has the same depth as in the DRRN [13], and (3) LapSRN$_{SS}$-D5R8, which has 84 layers for $4\times$ SR. We train the above three models using the multi-scale training strategy with $2\times, 4\times$ and $8\times$ SR samples and denote our multi-scale models as MS-LapSRN.

We show the quantitative results in Table 6. Our LapSRN performs favorably against existing methods especially on $4\times$ and $8\times$ SR. In particular, our algorithm achieves higher IFC values, which has been shown to be correlated well with human perception of image super-resolution [52]. We note that our method does not use any $3\times$ SR samples for training but still generates comparable results as the DRRN.

We show visual comparisons on the BSDS100, URBAN100 and MANGA109 datasets for $4\times$ SR in Fig. 11 and $8\times$ SR in Fig. 12. Our method accurately reconstructs parallel straight lines, grid patterns, and texts. We observe that the results generated from pre-upsampling based methods [9], [11], [13] still contain noticeable artifacts caused by spatial aliasing. In contrast, our approach effectively suppresses such artifacts through progressive reconstruction and the robust loss function. For $8\times$ SR, it is challenging to predict HR images from bicubic-upsampled input [3], [9], [11] or using one-step upsampling [15]. The state-of-the-art methods do not super-resolve the fine structures well. In contrast, our MS-LapSRN reconstructs high-quality HR images at a relatively fast speed.

We note that the direct reconstruction based methods, e.g., VDSR [11], can also upsample LR images progressively by iteratively applying the $2\times$ SR model. We present detailed comparisons and analysis of such a progressive reconstruction strategy in the supplementary material, available online.

5.2 Execution Time

We use the source codes of state-of-the-art methods to evaluate the runtime on the same machine with 3.4 GHz Intel i7 CPU (32G RAM) and NVIDIA Titan Xp GPU (12G Memory). Since the testing code of the SRCNN [9] and FSRCNN [15] is based on CPU implementation, we rebuild these models in MatConvNet to measure the runtime on GPU. Fig. 13 shows the trade-offs between the runtime and performance

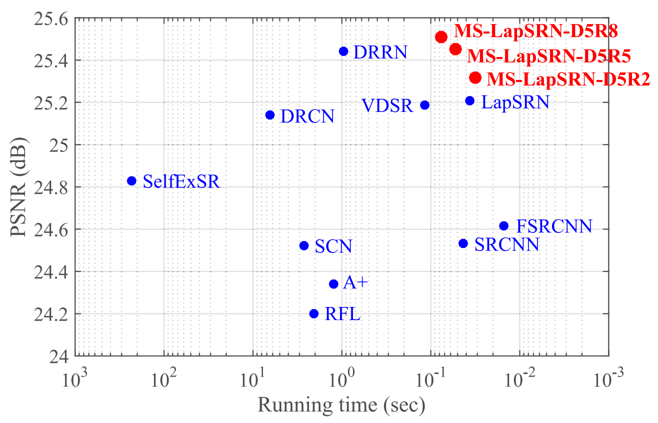


Fig. 13. *Runtime versus performance*. The results are evaluated on the URBAN100 dataset for $4\times$ SR. The proposed MS-LapSRN strides a balance between reconstruction accuracy and execution time.

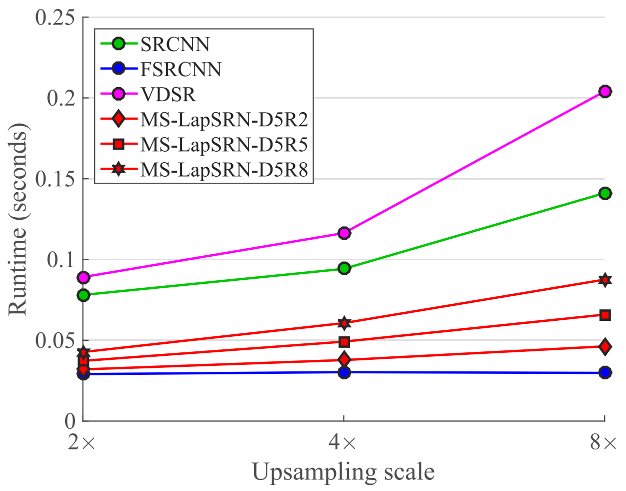


Fig. 14. *Trade-off between runtime and upsampling scales*. We fix the size of input images to 128×128 and perform $2\times$, $4\times$ and $8\times$ SR with the SRCNN [9], FSRCNN [15], VDSR [11] and three variations of MS-LapSRN, respectively.

(in terms of PSNR) on the Urban100 dataset for $4\times$ SR. The speed of our MS-LapSRN-D5R2 is faster than all the existing methods except the FSRCNN [15]. Our MS-LapSRN-D5R8 model outperforms the state-of-the-art DRRN [13] method and is an order of magnitude faster.

Next, we focus on comparisons between fast CNN-based methods: SRCNN [9], FSRCNN [15], VDSR [11], and LapSRN [16]. We take an LR image with a spatial resolution of 128×128, and perform $2\times$, $4\times$ and $8\times$ SR, respectively. We evaluate each method for 10 times and report the averaged runtime in Fig. 14. The FSRCNN is the fastest algorithm since it applies convolution on LR images and has less number of convolutional layers and filters. The runtime of the SRCNN and VDSR depends on the size of *output* images, while the speed of the FSRCNN and LapSRN is mainly determined by the size of *input* images. As the proposed LapSRN progressively upscales images and applies more convolutional layers for larger upsampling scales (i.e., require more pyramid levels), the time complexity slightly increases with respect to the desired upsampling scales. However, the speed of our LapSRN still performs favorably against the SRCNN, VDSR, and other existing methods.

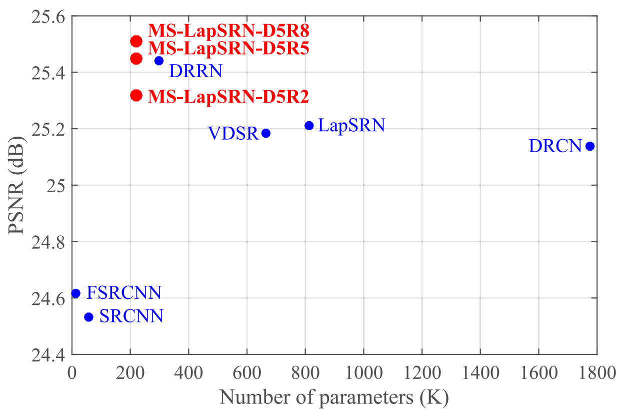


Fig. 15. *Number of network parameters versus performance*. The results are evaluated on the Urban100 dataset for $4\times$ SR. The proposed MS-LapSRN strides a balance between reconstruction accuracy and execution time.

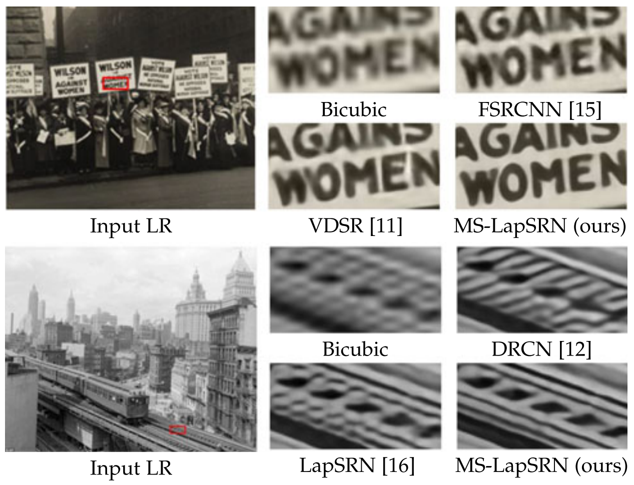


Fig. 16. *Comparison of real-world photos for* $4\times$ *SR*. The ground truth HR images and the blur kernels are not available in these cases. On the top image, our method super-resolves the letter "W" accurately while VDSR incorrectly connects the stroke with the letter "O". On the bottom image, our method reconstructs the rails without the artifacts.

5.3 Model Parameters

We show the reconstruction performance versus the number of network parameters of CNN-based SR methods in Fig. 15. By sharing parameters and using recursive layers, our MS-LapSRN has parameters about 73 percent less than the LapSRN [16], 66 percent less than the VDSR [11], 87 percent less than the DRCN [12], and 25 percent less than the DRRN [13]. While our model has a smaller footprint, we achieve the state-of-the-art performance among these CNN-based methods. Comparing to the SRCNN [9] and FSRCNN [15], our MS-LapSRN-D5R8 has about 0.9 to 1 dB improvement on the challenging Urban100 dataset for $4\times$ SR.

5.4 Super-Resolving Real-world Photos

We demonstrate an application of super-resolving historical photographs with JPEG compression artifacts. In these cases, neither the ground-truth images nor the downsampling kernels are available. As shown in Fig. 16, our method can reconstruct sharper and more accurate images than the state-of-the-art approaches.

5.5 Comparison to LAPGAN

As described in Section 2.4, the target applications of the LAPGAN [34] and LapSRN are different. Therefore, we focus on comparing the *network architectures* for image super-resolution. We train the LAPGAN and LapSRN for $4\times$ and $8\times$ SR with the same training data and settings. We use 5 convolutional layers at each level and optimize both networks with the Charbonnier loss function. We note that in [34] the sub-networks are independently trained. For fair comparisons, we jointly train the entire network for both LAPGAN and LapSRN. We present quantitative comparisons and runtime on the Set14 and BSDS100 datasets in Table 7. Under the same training setting, our method achieves more accurate reconstruction and faster execution speed than that of the LAPGAN.

5.6 Adversarial Training

We demonstrate that our LapSRN can be extended to incorporate the adversarial training [35]. We treat our LapSRN as

TABLE 7
Quantitative Comparisons between the Generative Network of the LAPGAN [34] and Our LapSRN

Method	Scale	SET14		BSDS100	
		PSNR	Seconds	PSNR	Seconds
LAPGAN	4	27.89	0.0446	27.09	0.0135
LapSRN	4	28.04	0.0395	27.22	0.0078
LAPGAN	8	24.30	0.0518	24.46	0.0110
LapSRN	8	24.42	0.0427	24.53	0.0107

Our LapSRN achieves better reconstruction quality and faster processing speed than the LAPGAN.

a generative network and build a discriminative network using the discriminator of the DCGAN [53]. The discriminative network consists of four convolutional layers with a stride of 2, two fully connected layers and one sigmoid layer to generate a scalar probability for distinguishing between real images and generated images from the generative network. We find that it is difficult to obtain accurate SR images by solely minimizing the cross-entropy loss. Therefore, we include the pixel-wise reconstruction loss (i.e., Charbonnier loss) to enforce the similarity between the input LR images and the corresponding ground truth HR images.

We show a visual result in Fig. 17 for $4\times$ SR. The network with the adversarial training generates more plausible details on regions of irregular structures, e.g., grass, and feathers. However, the predicted results may not be faithfully reconstructed with respect to the ground truth high-resolution images. As a result, the accuracy is not as good as the model trained with the Charbonnier loss.

5.7 Limitations

While our model is capable of generating clean and sharp HR images for large upsampling scales, e.g., $8\times$, it does not "hallucinate" fine details. As shown in Fig. 18, the top of the building is significantly blurred in the $8\times$ downscaled LR image. All SR algorithms fail to recover the fine structure except the SelfExSR [22] method which explicitly detects the 3D scene geometry and uses self-similarity to hallucinate the regular structure. This is a common limitation shared by parametric SR methods [3], [9], [11], [12], [13], [15].

6 Conclusions

In this work, we propose a deep convolutional network within a Laplacian pyramid framework for fast and accurate image super-resolution. Our model progressively predicts high-frequency residuals in a coarse-to-fine manner with deeply supervision from the robust Charbonnier loss functions. By sharing parameters *across* as well as *within* pyramid levels, we use 73 percent fewer parameters than our preliminary method [16] and achieve improved performance. We incorporate local skip connections in our network for training deeper models. Furthermore, we adopt the multi-scale training strategy to train a *single* model for handling *multiple* upsampling scales. We present a comprehensive evaluation on various design choices and believe that the thorough analysis benefits our community. We have shown the promise of the proposed LapSRN in the context of image super-resolution. Yet, our network design is general and can potentially be applied to other image transformation and synthesis problems.

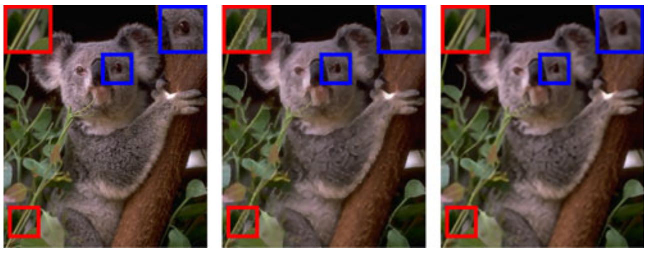


Fig. 17. *Visual comparison for adversarial training*. We compare the results trained with and without the adversarial training on $4\times$ SR.

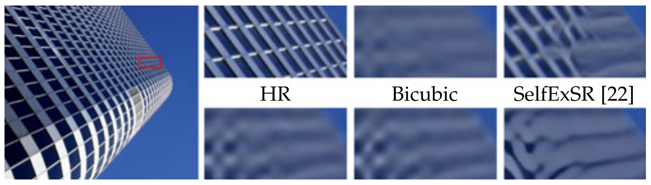


Fig. 18. *Limitation.* A failure case for $8\times$ SR. Our method is not able to hallucinate details if the LR input image does not consist of sufficient amount of structure.

Acknowledgments

This work is supported in part by the NSF CAREER Grant #1149783, NSF Grant No. #1755785, and gifts from Adobe and Nvidia.

References

[1] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
[2] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.
[3] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. Asia Conf. Comput. Vis.*, 2014, pp. 111–126.
[4] C.-Y. Yang and M.-H. Yang, "Fast direct super-resolution by simple functions," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 561–568.
[5] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3791–3799.
[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[8] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.
[9] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.
[10] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 370–378.
[11] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.

[12] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1637–1645.

[13] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2790–2798.

[14] W. Shi, J. Caballero, F. Huszar, J. Totz, A. 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1874–1883.

[15] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 391–407.

[16] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5835–5843.

[17] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1132–1140.

[18] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Trans. Graph.*, vol. 30, no. 2, 2011, Art. no. 12.

[19] C.-Y. Yang, J.-B. Huang, and M.-H. Yang, "Exploiting self-similarities for single frame super-resolution," in *Proc. Asia Conf. Comput. Vis.*, 2010, pp. 497–510.

[20] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 349–356.

[21] A. Singh and N. Ahuja, "Super-resolution using sub-band self-similarity," in *Proc. Asia Conf. Comput. Vis.*, 2014, pp. 552–568.

[22] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5197–5206.

[23] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.*, vol. 22, no. 2, pp. 56–65, Mar./Apr. 2002.

[24] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 1–10.

[25] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2004, pp. I–I.

[26] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1127–1133, Jun. 2010.

[27] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. Curves Surfaces*, 2010, pp. 711–730.

[28] R. Timofte, V. Smet, and L. Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1920–1927.

[29] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "NTIRE 2017 challenge on single image super-resolution: Methods and results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1110–1121.

[30] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. C-31, no. 4, pp. 532–540, Apr. 1983.

[31] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. 22nd Annu. Conf. Comput. Graph. Interactive Techn.*, 1995, pp. 648–651.

[32] S. Paris, S. W. Hasinoff, and J. Kautz, "Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid," *ACM Trans. Graph.*, vol. 30, no. 4, 2011, Art. no. 68.

[33] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 519–534.

[34] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using a Laplacian pyramid of adversarial networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1486–1494.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[36] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2536–2544.

[37] Y. Li, S. Liu, J. Yang, and M.-H. Yang, "Generative face completion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5892–5900.

[38] X. Yu and F. Porikli, "Ultra-resolving face images by discriminative generative networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 318–333.

[39] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 105–114.

[40] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[42] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods," *Int. J. Comput. Vis.*, vol. 61, no. 3, pp. 211–231, 2005.

[43] C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2015, pp. 562–570.

[44] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1395–1403.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[46] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *Workshop*, 2013.

[47] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[48] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. ACM Int. Conf. Multimedia*, 2015, pp. 689–692.

[49] Y. Matsui, K. Ito, Y. Aramaki, T. Yamasaki, and K. Aizawa, "Sketch-based manga retrieval using manga109 dataset," *Multimedia Tools Appl.*, vol. 76, pp. 21811–21838, 2017.

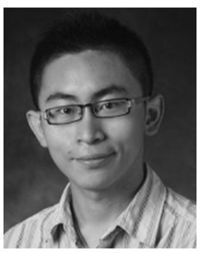
[50] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[51] H. R. Sheikh, A. C. Bovik, and G. De Veciana, "An information fidelity criterion for image quality assessment using natural scene statistics," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2117–2128, Dec. 2005.

[52] C.-Y. Yang, C. Ma, and M.-H. Yang, "Single-image super-resolution: A benchmark," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 372–386.

[53] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2016.

Wei-Sheng Lai received the BS and MS degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2012 and 2014, respectively. He is working toward the PhD degree in Electrical Engineering and Computer Science, University of California, Merced, CA. His research interests include computer vision, computational photography, and deep learning.



Jia-Bin Huang received the BS degree in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan and the PhD degree from the Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, in 2016. He is an assistant professor with the Bradley Department of Electrical and Computer Engineering, Virginia Tech. He is a member of the IEEE.

Narendra Ahuja received the PhD degree from the University of Maryland, College Park, MD, in 1979. He is the Donald Biggar Willet professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL. He is on the Editorial Boards of several journals. He was the founding director of the International Institute of Information Technology, Hyderabad, Hyderabad, India, where he continues to serve as a Director International. He is a fellow of the American Association for Artificial Intelligence, the International Association for Pattern Recognition, the Association for Computing Machinery, the American Association for the Advancement of Science, and the International Society for Optical Engineering.

Ming-Hsuan Yang received the PhD degree in computer science from the University of Illinois at Urbana-Champaign, in 2000. He is a professor in Electrical Engineering and Computer Science, University of California, Merced. He served as an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* from 2007 to 2011, and is an associate editor of the *International Journal of Computer Vision*, the *Image and Vision Computing* and the *Journal of Artificial Intelligence Research*. He received the NSF CAREER award in 2012, the Senate Award for Distinguished Early Career Research at UC Merced in 2011, and the Google Faculty Award in 2009. He is a senior member of the IEEE and the ACM.