Applied Intelligence (2019) 49:125–136
https://doi.org/10.1007/s10489-018-1234-y

Image super-resolution with densely connected convolutional networks

Ping Kuang[1] · Tingsong Ma[1] · Ziwei Chen[1] · Fan Li[1]

Published online: 12 July 2018

Abstract
This paper proposes a new model for single image super-resolution (SR) task by utilizing the design of densely connected convolutional networks (DenseNet). The proposed method is an end-to-end model which is able to learn mapping between low- and high-resolution images. The proposed method takes the low-resolution images as input and generates its high-resolution version. Unlike those conventional methods which adjust each component of convolutional networks separately, our model jointly optimizes all layers. Besides, the proposed model has a lightweight structure and is extensively evaluated on widely adopted data sets. In our experiments, the proposed method outperforms state-of-the-art methods both qualitatively and quantitatively. In addition, we also carried out experiments in terms of different designs and configurations to achieve better balance between reconstruction performance and speed in this paper.

Keywords Image super-resolution · DenseNet · Deep learning

1 Introduction

Nowadays, in the field of image processing, image super-resolution (SR) has become one of the greatest challenges. Image SR aims at reconstructing high-resolution images (HR) by enlarging pixels of low-resolution images (LR), and makes sure that HR images can contain as much high-frequency details as possible. Unlike HR images which can be easily down-sampled into LR images. There are a lot of possible choices existing for the given LR images to become HR images, and we don't know which option can be the right one. Typically, this problem was solved by limiting the solution space through prior information.

✉ Tingsong Ma
290106526@qq.com

Ping Kuang
kuangping@uestc.edu.cn

Ziwei Chen
421525351@qq.com

Fan Li
lifan@uestc.edu.cn

[1] University of Electronic Science and Technology of China, Chengdu, China

The example-based [13] strategy is one of the most prevalent approaches used by computer vision community to learn the prior information. Its basic idea either explore internal connections and similarities of the same images [1, 3, 5, 17, 18], or just learn a mapping function from two different resolution external images pairs [2, 4, 6, 14, 18–23, 27, 28]. According to different training samples, those example-based methods, to some extent are able to be designed as a general-purpose SR method. The sparse-coding-based(SC) method [28, 29], as a widely adopted and publicly accepted example-based SR method, utilizes some special steps to achieve image super-resolution. Firstly, they randomly crop and pre-process a small size of overlapping patches from input. Secondly, the pre-processed patches will be encoded by a low-resolution dictionary. Simultaneously, the sparse coefficients will subsequently be passed into a high-resolution dictionary for creating high-resolution version of patches. Finally, the final output will be produced by aggregating overlapping reconstructed patches. Almost every external example-based method shares the same steps, which requires they to pay more attention to learn to optimize the low/high dictionaries [6, 28, 29] or building mapping functions [17, 18, 21, 22] (Fig. 1).

In the meanwhile, learning-based methods have caught more and more attention of people who interested in image

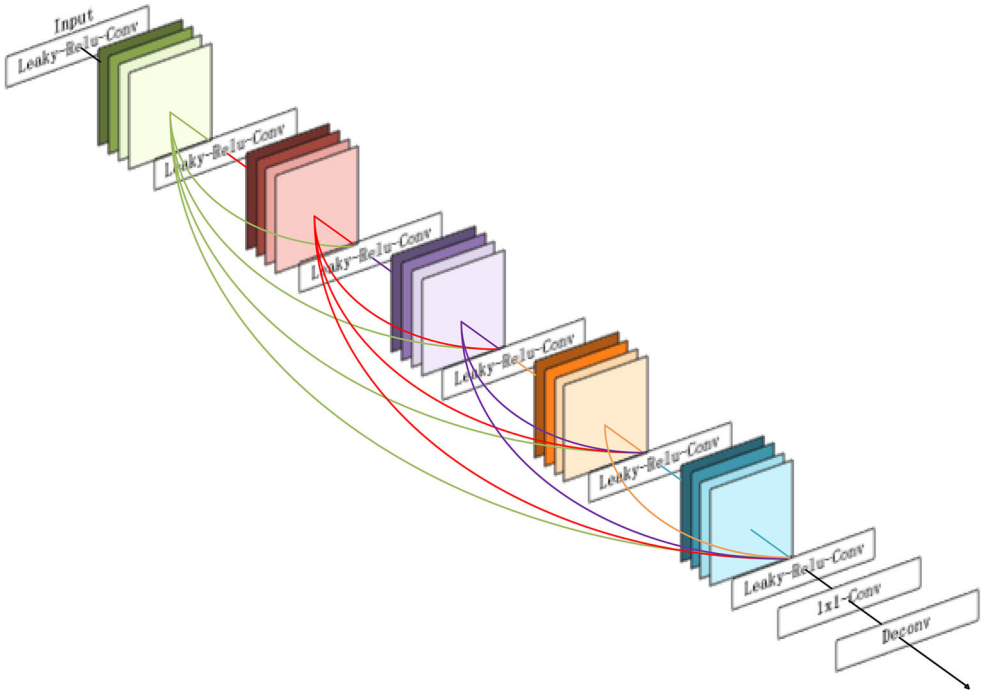


Fig. 1 A DenseNet model with 5-layer dense block, followed by a 1×1 convolutional layer and one deconvolution layer

process and achieved better performance in image SR. Unlike the example-based methods, it focuses on learning a proper mapping function from the connections between LR images and HR images by using auxiliary data which could be collected either from input LR image itself directly [1, 3, 7], or from fruitful natural images indirectly [8–11]. Multifarious machine learning algorithms, e.g., regression trees or forests [12, 15, 16], sparse coding [10, 21], anchored neighbor [3, 4, 7, 19, 24], have already been adopted to learn the mapping function. Although these methods have achieved significant progresses, their successes are based on hand-designed features, which means the data must be pre-processed and people can easily make mistakes and further affect the whole prediction results. In these methods, their solution pipeline heavily relies on hand-designed features to describe LR images, which are not originally learned from the models. Moreover, their learning abilities are limited, because of adopting shallow models. Therefore, the performance of these methods is influenced by its shallow structure and limited learning capacity.

Lately, deep neural networks(DNNs), specifically, deep convolutional neural networks(CNNs), has already achieved breaking performance in various image processing tasks, including image classification, object detection, semantic segmentation, etc. In addition, this kind of neural network has been applied in image SR. Most of these initial attempts constitute two steps. In the first step, Bicubic interpolation is applied to LR image to upscale the LR image to the size of its HR counterpart. Then the CNNs take the up-scaled LR images as input and reconstruct its HR version. Compared with example-based methods, CNNs have a much deeper structure which can learn more information and ensure a stronger learning capacity and more accurate prediction of HR images. Furthermore, CNNs could also learn rich feature hierarchies without using hand-designed features, which is more suitable for image SR task.

Although impressive performance has been demonstrated, existing deep learning based SR methods have several drawbacks. Firstly, when CNNs become increasingly deep, there is a serious problem. During training phase, input or gradient can vanish by the time it reaches the end or beginning of the network, because they go through too many layers. That means a large number of features will be lost through deep network, and that could further affect learning capacity and may loss key information of LR image. Secondly, the exploration of training deeper networks for SR is still very limited. It is widely known that deeper networks with more complex architectures are more possible to show great performance, at the same time, however, make the training process more challenging. Most prior methods avoid this issue with relatively shallow CNNs (no more than five layers) [25, 26, 30, 33]. The research in [31] proposes to train very deep networks for SR and good performance has been achieved.

2 Related work

At present, the research of image super-resolution is generally concluded as three main categories: interpolation-based [32, 34], reconstruction-based [35–37] and learning-based methods [3, 7, 15]. This paper mainly focuses on

learning-based methods. Its fundamental principle is to use training dataset to create a learning model, which could be utilized to reconstruct HR images from LR images. In other words, the original purpose of learning-based method is to learn a mapping function between LR and HR images. The mapping function could be trained and optimized in a supervised learning process. For example, Baker et al. [38] first proposed a method based on the recognition of prior knowledge. Their basic idea is to learn and train a certain category by algorithm, and apply the attained prior knowledge to image SR. Motivated by this idea, Freeman W T [9] proposed an example-based method, whose core solution is to learn details of high- resolution images which correspond to different regions of low resolution images in the image library by using the Markov network. Recently, Yang et al. [10] put forward a new measure based on sparse signal recovery to single-image SR. In this method, each patch of low-resolution input is constrained in a sparse representation and it utilize coefficients of these representations to generate HR output. And in [22], the primary goal is to learn a mapping function between LR patches and HR patches. To further accelerate the training speed and improve computational efficiency, Yang and Yang [7] divided the feature space into multiple subspaces and collected samples to learn priors of each subspace. It is noteworthy that most of existing image SR methods prefer to learn regressor to predict residual between HR and LR, because the LR and HR images are highly correlated. By utilizing this method, the training process can be more reliable.

Recently, more and more researchers focus their attention on applying convolutional neural networks(CNNS) to image SR, which indeed achieved good results. For example, in [26], the researcher started to utilize the advantage of deep learning to solve SR problem, their underlying idea is to directly learn an end-to-end mapping between low/high-resolution images. Other than that, Liu et al. [33] showed deep networks based on sparse coding could further improve training efficiency and deliver promising results. However, there are still many potentials of convolutional networks can be exploited.

Figure 2 shows that structure of convolutional networks is one of the most crucial factors that can directly affect performance of image SR. Shallow models with few layers are not able to acquire enough high-frequency contents to reconstruct HR images and its learning capacity is limited. Therefore, DNNs becomes a more suitable model for image SR. However, it is not easy to apply a very deep network to image SR. Because too many layers make the whole network difficult to train and could further lose many low-frequency contents. However, Kim et al. [31] proposed a new method using a very deep convolutional networks, they resolve the difficult-training problem by learning residual between HR images and LR images.

3 Our method

Compared with [31], we replace the deep convolution networks with relatively Shallow DenseNet (Fig. 1) and few normal convolution layers, where the DenseNet seek abundant high-frequency contents to ensure accurate HR reconstruction and the convolution layers stabilize our training. Although our work gets a similar spirit with prior DNNs, the proposed method significantly differs from them in three aspects: i) adopts a relatively shallow network; ii) unlike the previous work which need hand-design to create labels, our network will automatically extract small patches

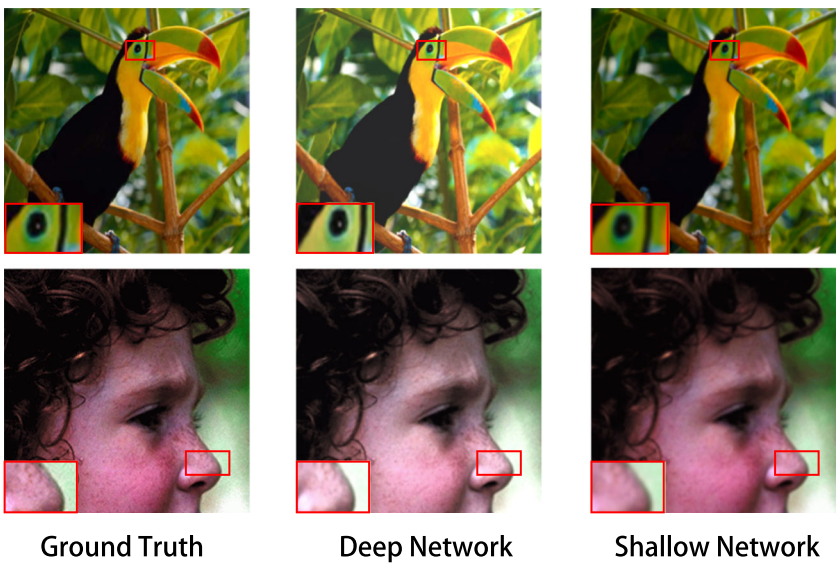


Fig. 2 SR results of different networks with an upscaling factor 3. The discrepancy between results of ground truth, deep network and the shallow network varies across different images. It is notable that shallow network fail to restore high-frequency details. On the other hand, deep network has better results and could restore more high-frequency contents

from training images and utilize those patches to generate labels while training; iii) instead of using bicubic to upscale images, we adopt a deconvolution layer to achieve this without augmenting the number of parameters.

3.1 Architecture

In this section, we introduce our method for image SR. Figure 3 overviews the architecture of the network which consists of 28 layers in total, and can be further divided into four parts: data enhancement, feature extraction, up-sampling and reconstruction (data enhancement are not counted in the number of layers). As the first step of our network, we randomly crop certain number of fixed size fragments from the training images, and each label will be generated from those fragments. The feature extraction part consists of 20 densely connected convolutional layers (5 dense blocks) and 6 normal convolution layers and followed by a 1×1 convolution layer and a deconvolution layer. The complex composition enables the network to perform greatly, at the same time, however, makes the training process more challenging. Usually, more than 28 layers are adopted in a DenseNet, here, a relatively shallow DenseNet is much easier to converge, which aims at stabilizing the training process.

3.2 Data enhancement

The network loads the complete image and then intercept it into small pieces. Here we tried two kinds of image segmentation methods. The first one is sequential cropping, by using this method we cut pieces every few pixels. That means we can get 10000 patches from one 100×100 image, when we cut patches every one pixels. As for other larger images, the number of patches would be extremely huge, which is difficult to train. Hence we utilize another method: random cropping, by which means the amount of patches could be controlled and make our network much easier to train. In the training phase, we randomly intercepted a certain number of patches. The size of patches we set into $p \times p$. Moreover in our configuration, we set the input size to $l \times l$ and the scaling is S. Therefore the size of label is $h \times h$(h= $l \times S$). Again we randomly intercept an $h \times h$ image from every patch as label (high resolution image), then apply random flip, random brightness and random contrast to the label (data enhancement). In order to get network input, we need to down-sampling the labels. We provide three methods to achieve this purpose: nearest neighbor, bicubic and area. Every time we randomly select one of these algorithms to down-sampling the images to obtain $l \times l$ low resolution(LR) images as input. Here we abandon the bilinear, because bilinear can be seen as a powerful low frequency filter. That means it could lose a large amount of high-frequency content. In addition to above data enhancements, we can also add noise into the LR images, for example, the jpeg noise.

3.3 Feature extraction

In order to extract local features of high-frequency contents, prior shallow networks implement extraction process by calculating the first and second order gradients of image patches, which is equivalent to filtering the input image with hand-designed high-pass filters. Instead of manually

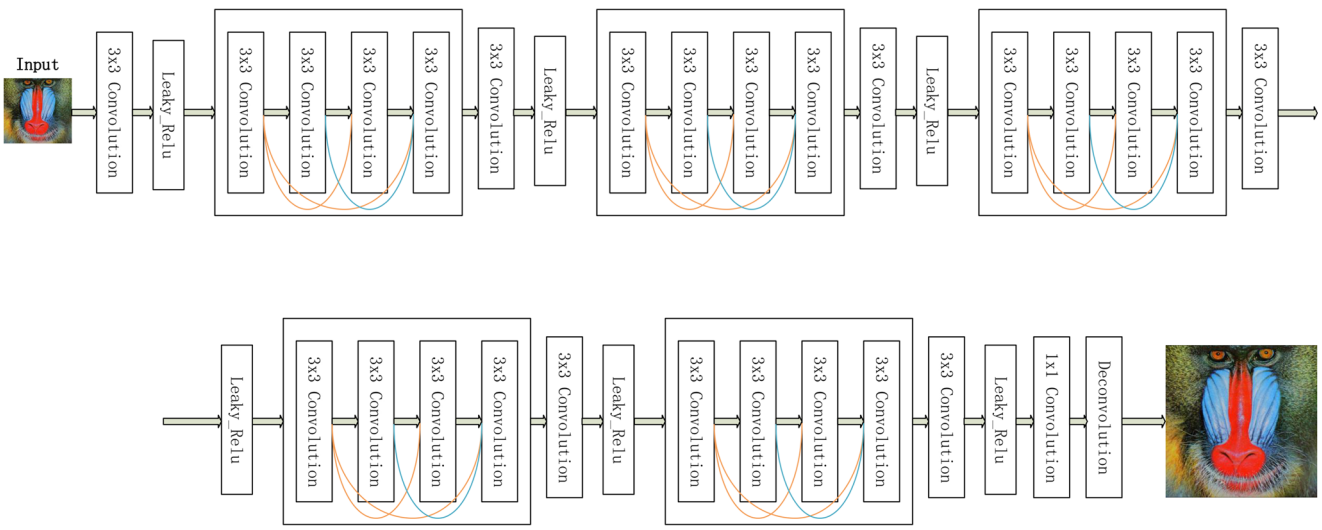


Fig. 3 Network architecture: Our network consists of 6 normal convolution layers followed by a leaky-Relu and 5-layer dense blocks where contains 4 densely connected convolution layers

designing these filters, deep learning based methods automatically learn the filters from training data. However, most previous works, no matter adopting shallow or deep models, extract features from low precision HR images, which are generated by up-sampling the LR images to the HR size with bicubic interpolation. A lot of previous studies have shown that bicubic interpolation is not the best approach [30], because it can be integrated into the network and regarded as part of the training (or the process of amplifying). According to this theory, our method adopts an alternative strategy which performs feature extraction directly on the original LR images with convolution layers and one deconvolution layer.

Our feature extraction module consists of 20 densely connected convolutional layers in total (separate into 5 dense blocks, each block has 4 layers), 7 convolution layers and 1 deconvolution layer. In densely connected part, every convolutional layer is followed by a rectified linear unit (ReLU) and adopts zero padding to preserve the spatial size of the output feature maps. The other convolutional layers are followed by leaky rectified linear unit (Leaky ReLU). Each convolutional layer can be expressed as:

$$F_l = \max(0, W_l \times F_{l-1}) \tag{1}$$

Where W_l denotes the l-th convolutional layer's kernel's filter; F_l presents the output feature map of the l-th layer, especially, F_0 denoting the original LR images. All convolution layers (including densely connected convolution layers) have the same kernel size of 3×3, the growth rate of densely connected layer is set to 12. In other words, if every function produces feature-maps as output, it follows that the $l + 1$-th layer has $k \times (l - 1) + k_0$ input feature-maps, where the k_0 is the channel numbers of input.

3.4 Upsampling

The output of feature extraction phase is utilized to upscale to the target HR size. Recent studies have noticed that bicubic interpolation is not necessary and can be integrated into the network by using deconvolution and unpooling operations. Thus, this kind of upsampling method is learning based and could give rise to an end-to-end trainable system. The reason why we abundant unpooling operation in our network is that the unpooling operation with an upscaling factor replaces each entry in the input feature maps with a $s \times s$ block, where the top left element in the block is set to the value of the input entry and the others to zero. That means the output of unpooling layers will enlarge feature maps of output as well as make them more sparse. By comparison, the deconvolution operation up-scales the input feature maps by $s - fold$ through reversing forward and backward propagation of convolutional layers with an output stride of s. Although unpooling and deconvolution have different implementations, they are similar in upscaling feature maps and both are suitable to image SR task. Therefore, we adopt the deconvolution layer and achieve a promising performance.

The upsampling module plays a key role in our image SR method. During our experiments, the results show that the size of deconvolution kernel has a great effect on the upsampling quality, which further enhances the final performance. This may be attributed to the fact that bigger kernel size offers the up-sampling operation a bigger view that enables it to process a larger input neighborhood and better enforces spatial consistency. In order to preserve the spatial size, deconvolution layer adopts zero padding on each side of the output feature maps. In addition, one 1×1 convolutional layer is implemented before the up-sampling to reduce the computational complexity, where the 1×1 convolutional layer maps $d - channel$ input feature map to $3 - channel$ output feature map for up-sampling.

3.5 Reconstruction

As it is mentioned in the Data Enhancement section that we don't use a complete image as an input for training, instead, we randomly intercepted a certain number of patches from a complete image and used those patches to generate their own labels and inputs. In other words, we do not reconstruct the whole picture, because we only trained randomly copped patches in training phase, thus, we only reconstruct high-resolution patches and put all the high-resolution patches together in testing phase to generate complete high-resolution picture. In order to reconstruct a complete HR image, testing images will be divided into small patches sequentially, and all the patches will be input into our network to generate corresponding high resolution patches. Finally, HR images will be reconstructed by splicing those patches in order. Unlike [10] who proposed a HR reconstruction module which consists of 7 trainable layers, we do not utilize any trainable layers to fulfill this task. An obvious reason is our unique learning method, where the input data is subsequently selected from original LR images, thus there is a potential rule here for ours to put their corresponding output together to generate another complete image. Another consideration is that the network is too deep already, adding additional layers in the network might make training results unstable and more challenging.

3.6 Training

During the training phase, we provide numbers of high-resolution images Y_i and their corresponding low-resolution version X_i, the proposed networks are learned by minimiz-

ing the mean square error(MSE) loss between the predicted HR image $Y^{'}$ and ground truth Y:

$$L(\Theta) = \frac{1}{n}\sum \| F(Y;\Theta) - X_i \|^2 \tag{2}$$

Where n is the number of training samples, $F(Y;\Theta)$ is the predicted HR image $Y^{'}$ of Y_i. The optimization is implemented by mini-batch stochastic gradient descent method with AdagradOptimizer. In our configuration, the batch size is set to 8. Our experiment empirically shows that large batch size leads to huge loss and makes it difficult to converge. Moreover, in this task, the gradient competition in each sample of batch is intense, which slow down the convergence rate. Thus relatively small batch size could benefit efficiency of our training and perform better results. To initialization, every filter in our model (including convolutional layers and deconvolutional layers) will be randomly initialized by zero-mean Gaussian distribution, where the standard deviation is calculated by:

$$Stddev = 2.0/(f \times f \times c) \tag{3}$$

where f denotes the kernel size and c is number of input channels, and the kernels' size of each convolutional layer are shown in Fig. 3. The learning rate is initially set to 0.001 and decreased by a factor of 1.5 every 10000 epochs, and we train one million epochs in total. In addition, we utilize Peak Signal to Noise Ratio(PSNR) to evaluate our predicted results. The PSNR is an objective criterion that can be utilized to evaluate image's restoration quality, to some extent, PSNR is also perceptual quality related.

4 Expreriments

We firstly investigated how the size of training dataset can affect our model's reconstruction performance. Secondly, we build several model versions with different structures and study the relation between reconstruction performance and different design of network. Subsequently, in order to fairly compare our method with other existing approaches, we will use the same training datasets and test datasets that are widely adopted. For facilitating comparison, we only use a relatively small training datasets [27] which merely contains 91 images, and because we have applied several data augmentation techniques during data enhancement phase, the actual number of training data could be much bigger than 91. For each upscaling factor (i.e., 2, 3, or 4), the size of LR image is set to 28×28. The size of ground truth HR example is set to 56×56, 84×84, and 112×112 respectively, and these HR examples are all cropped from 120×120 patches, which are randomly cropped from original images. Especially, the LR training samples are generated by downsampling those HR examples. The proposed model is trained by using Tensorflow framework on a workstation with Intel 4 GHz and a GTX1080 GPU. We use Set5 [41], Set14 [19] and BSD100 [40] as the test datasets, which contain 5, 14 and 100 images respectively. Furthermore, we utilize PSNR and SSIM metrics to compare our method with recent state-of-the-art both quantitatively and qualitatively, which are widely used in image SR literature.

4.1 Analysis

In order to obtain an insight of our contributions, we conduct additional evaluations on different variants of the proposed method. Because the SR results tend to be similar with different up-scaling factors, we only report the results for the upscaling factor of 2 and 3. Moreover, the experiments we present next are internal comparison and analysis of the network itself. Thus, we only want to get an average performance of every version of our model. And there is no need to spend a lot of time to excavate every network's potentials, because we just want to prove that our design and strategy have positive effect on final results.

Training data It is widely known that, one of the simplest ways to improve a neural network's performance is to expand its training datasets. Here we will find out to what degree the size of training is related to the final performance. For comparison, we utilize a relatively small training set [10, 24] which contains 91 images, and a much larger training set that comprises 395,909 images from the ILSVRC2013 ImageNet detection training partition. Because of our data augmentation techniques, the 91-image dataset could be decomposed into more than 30,000 patches while the ImageNet will provide over 5 million patches. Next we will give specific parameter settings, i.e., size of patches is 120×120, batch size is 4, input size is 28×28. The learning rate is initially set to 0.001 and decreased by a factor of 1.25 every 5000 epochs, and we train 30,000 epochs in total. We use Set5 to validate the training effect. Similar results can be seen if we use larger validate set like Set14. The upscaling factor is set to 2. We use SC method [10] as our base line, which achieves an average PSNR value of 31.42 dB.

The performance of our method trained on different training sets are shown in Fig. 4. In accordance with this figure, with the same number of backpropagations, the performance of our method trained on ImageNet is better than on 91-image dataset. This results obviously prove that our method performance may be further boosted by using a larger training set. The reason why we don't use ImageNet as our training set is mainly because that

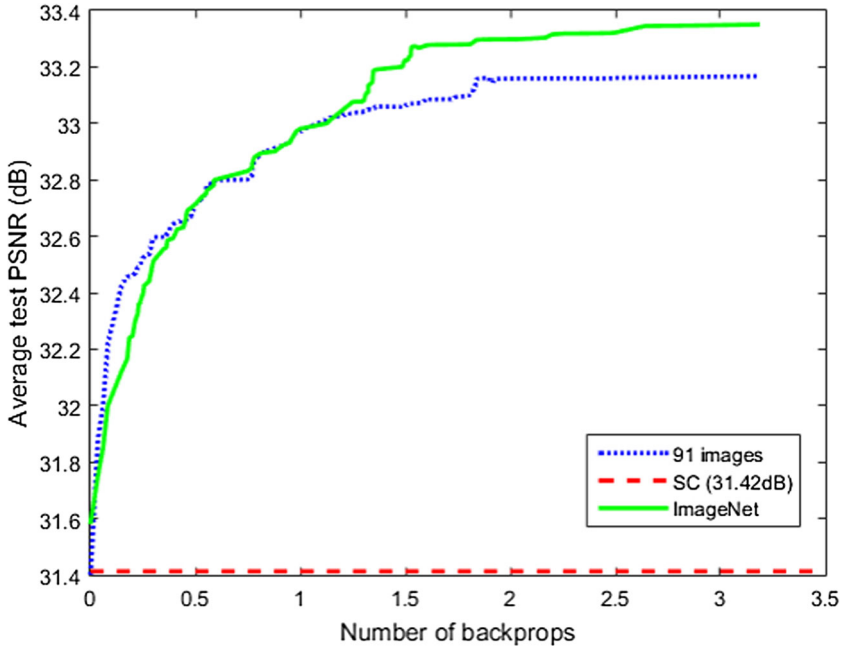


Fig. 4 Performance by using 91-image dataset and ImageNet dataset

our method is a relatively small and shallow network, which could not exhaustively absorb all the information that ImageNet dataset consists of, and 91-image dataset has already captured sufficient variability of natural images.

Number of layers A recent study by He and Sun [42] proposed an idea that CNN's performance can be directly improved by moderately deepening its network structure. Here our method jointly trains a deeper version of our model named Dense-13 (13 dense blocks) by adding additional densely connected layers and convolutional layers, and we also trained a shallow one named Dense-3 (3 dense blocks) by removing several dense blocks and normal convolutional layers. We use a 91-image dataset as training set in this experiment, learning rate is set to 0.01 and decreased by a factor of 1.5 every 10000 epochs, and we train one 100,000 epochs in total. The performance of our proposed method (5 dense blocks), Dense-13 method and Dense-3 method during training is measured on a 91-image dataset in Fig. 5 and the upscaling factor is set to 3. Fig. 5

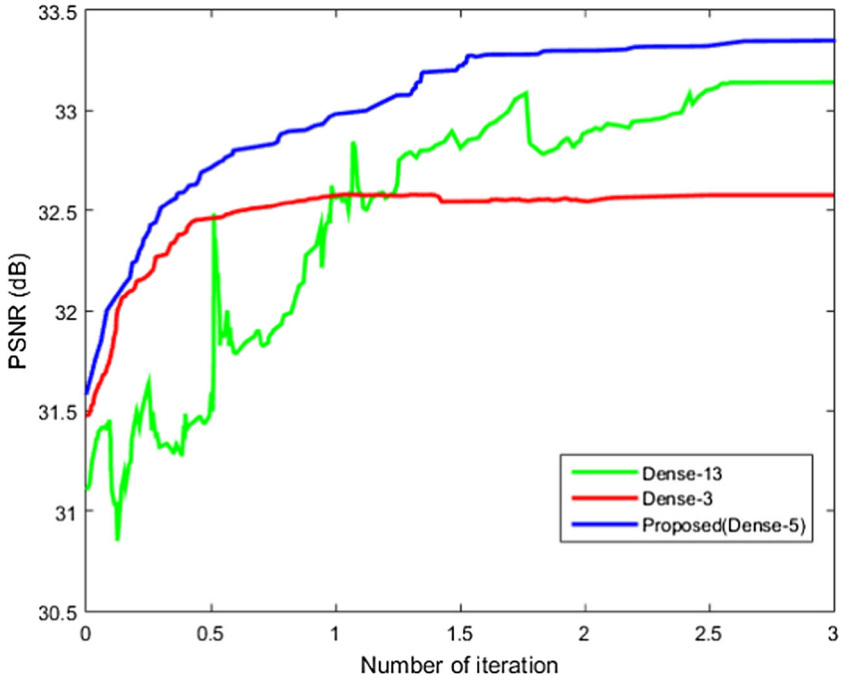


Fig. 5 Performance of different architectures on 91-image dataset with an upsampling factor 3

Table 1 Average PSNR (dB) of different architecture on three datasets with an upscaling factor 3

Model	ND	NWD	NFD	NLD
Set5	33.55	31.86	32.01	32.73
Set14	29.53	28.15	28.57	29.77
BSD100	28.83	27.64	27.88	28.42

depicts the convergence plots of all three models on the 91-image dataset. The Dense-3 method with a shallow network structure converge faster than other methods. However, limited by its building, the final performance of Dense-3 is relatively low in comparison with the Dense-13 and our proposed method (5 dense layers). On the other hand, it could be seen from Fig. 5 that it is more difficult to train Dense-13 method than Dense-3 and our proposed method. Upon convergence, the Dense-13 method achieves higher PSNR than Dense-3. However, the performance of Dense-13 is not stable. The deep architecture of Dense-13 is adequately complex to restore high-frequency details. Whereas, the generated HR images normally suffer from lacking of low-frequency contents and illumination changes compared to the ground truth.

The truth is that deeper structure used in super-resolution task is not able to play a key role, just like it does in image classification field [42]. Moreover, we also found that deeper network does not always outperform the shallow one. Specifically, if we go deeper by adding additional dense blocks, then we have to set a smaller learning rate and take more time to train, but we still do not observe superior performance. Every experiment we have done previously has indicated that "the deeper the better" doesn't work for super-resolution. The difficulty of training network may be the primary reason for this phenomenon. Because of our unique design, our model contains no pooling layers and full-connected layers. As a result of that our model is sensitive to the initialization of learning rate and parameters. When we have a deeper model, it is more difficult to have an appropriate learning rate that is able to guarantee convergence. Even if it converges, the loss value could fall into an extremely bad local optimum. Why deeper structure results lead bad performance is still an open question, which requires more investigations to better understand change of gradients and training dynamics in deep architectures. Because of that, we only adopt a limited number of dense blocks.

Densely connected structure To investigate to what degree densely connected architecture is related to SR performance, we compare the proposed method namely ND (network with densely connected layers) with three variants: NWD (network without densely connected layers). NFD (network with densely connected layers in the first half of the architecture) and NLD (network with densely connected layers in the last half of the architecture). The NWD model is obtained by removing all the densely connected links in the dense blocks in Fig. 3. Similarly, the NFD and NLD models are obtained by removing densely connected links in the last and first 3 dense blocks. All of these three models are trained on 91-image datasets, the learning rate is 0.01 and decreased by a factor of 1.5 every 10000 epochs, and we train one 100,000 epochs in total. Table 1 reports the average PSNR of the compared methods on three test dataset, ND and NLD considerably improves the performance of NWD and NFD respectively.

4.2 Comparison with state-of-the-art

We compare the proposed method with other recent SR methods on all the images in Set5, Set14 and BSD100 for different upscaling factors. The compared methods include traditional bicubic interpolation and other popular learning based methods: EEDS [10], SRCNN [25], SRCNN-L [26], SC [29], CSC [33], CSCN [39], ESPCN [30] and A+ [19]. The results of the compared methods we used in Table 2 are aquired by using avaliable codes the authors published.

Here we collected the best results of every SR method, therefore, in this experiment we have to get the best performance of our model. Through previous experiments we realize how training data, number of network layers and densely connected structures affect final results. Thus, in this experiment we adopt 5 dense block in our model (just the same structure as Dense-5 in Fig. 5) and we use ImageNet as our training set (in order to obtain the beset performance of our model). The learning rate is initially set to 0.001 and decreased by a factor of 2 every 10,000 epochs, and we train one 200,000 epochs in total. In order to have a fair evaluation, we use Set5, Set14 and SSD100 to evaluate each method and use PSNR and SSIM for statistics and comparison.

Table 2 summaries the quantitative performance of compared methods measured by average PSNR and SSIM. It can be seen from Table 2 that our method consistently outperforms all previous methods in both PSNR and SSIM, and with multi-view testing the results can be further improved. As demonstrated in the last three lines of Table 2, our method achieves similar performance with EEDS, despite sometimes the results of ours are not the best. It is worthy to notice that the EEDS method adopts a shallow CNN to restore the overall illumination and also utilizes deep CNN to capture high-frequency details, and that give a lot of help to improve reconstruction quality. Other than that, our model can be further improved by using the same strategy or with more training data. Figures 6 and 7 illustrate some sampled results generated by the compared methods.

Table 2 Average PSNR(SSIM) comparison on three different test datasets among different methods

DataSet	Set5			Set14			BSD100		
Upscaling	x2	x3	x4	x2	x3	x4	x2	x3	x4
Bicubic	33.66 (0.9299)	30.39 (0.8682)	28.42 (0.8104)	30.24 (0.8687)	27.55 (0.7736)	26.01 (0.7019)	29.56 (0.8431)	27.21 (0.785)	25.96 (0.6675)
SC	33.66 (0.9299)	30.39 (0.8821)	28.42 (0.8104)	30.23 (0.8687)	28.31 (0.7954)	26.00 (0.7019)	29.56 (0.8431)	28.34 (0.7954)	25.96 (0.6675)
CSC	36.62 (0.9548)	32.66 (0.9098)	30.36 (0.8607)	32.31 (0.9070)	29.16 (0.8209)	27.30 (0.7499)	31.27 (0.8876)	28.31 (0.7853)	26.83 (0.7101)
SRCNN	36.34 (0.9521)	32.39 (0.9033)	30.09 (0.8530)	32.18 (0.9039)	29.00 (0.8145)	27.21 (0.7413)	31.11 (0.8835)	28.20 (0.7794)	26.70 (0.7018)
SRCNN-L	36.66 (0.9542)	32.75 (0.9090)	30.48 (0.8628)	32.45 (0.9067)	29.30 (0.8215)	27.50 (0.7513)	31.36 (0.8879)	28.41 (0.7863)	26.90 (0.7103)
A+	36.55 (0.9544)	32.59 (0.9088)	30.29 (0.8603)	32.28 (0.9056)	29.13 (0.8188)	27.32 (0.7491)	30.78 (0.8773)	28.18 (0.7808)	26.77 (0.7085)
CNN	36.34 (0.9521)	32.39 (0.9033)	30.09 (0.8530)	32.18 (0.9039)	29.00 (0.8145)	27.20 (0.7413)	31.11 (0.8835)	28.20 (0.7794)	26.70 (0.7018)
CNN-L	36.66 (0.9542)	32.75 (0.9090)	30.49 (0.8628)	32.45 (0.9067)	29.30 (0.8215)	27.50 (0.7513)	31.36 (0.8879)	28.41 (0.7863)	26.90 (0.7103)
ESPCN	36.62 (0.9548)	32.55 (0.9098)	30.36 (0.8607)	32.31 (0.9070)	29.16 (0.8209)	27.30 (0.7499)	31.27 (0.8876)	28.31 (0.7853)	26.83 (0.7101)
CSCN	36.93 (0.9552)	33.10 (0.9144)	30.86 (0.8732)	32.56 (0.9074)	29.41 (0.8238)	27.64 (0.7578)	31.40 (0.8884)	28.50 (0.7885)	27.03 (0.7161)
EEDS	37.29 (0.9579)	33.47 (0.9191)	31.14 (0.8783)	32.81 (0.9105)	29.60 (0.8284)	27.82 (0.7626)	31.64 (0.8928)	28.64 (0.7925)	27.11 (0.7200)
Our Method	37.26 (0.9573)	33.59 (0.9234)	31.16 (0.8788)	32.69 (0.8986)	29.54 (0.8244)	27.85 (0.7644)	31.55 (0.8908)	28.80 (0.7973)	27.08 (0.7090)
Our Gain	−0.03(−0.006)	0.12 (0.043)	0.02 (0.0005)	−0.12(−0.012)	−0.06(−0.004)	0.03 (0.0018)	−0.09(−0.002)	0.16 (0.0048)	−0.03(−0.011)

Red and blue indicate the best and the second best performance

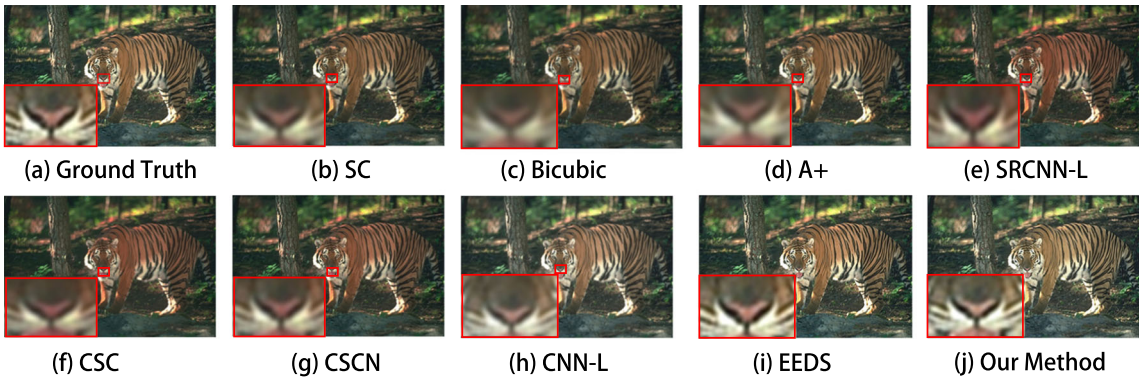


Fig. 6 The 'Tiger' image from BSD100 with an upscaling factor 3

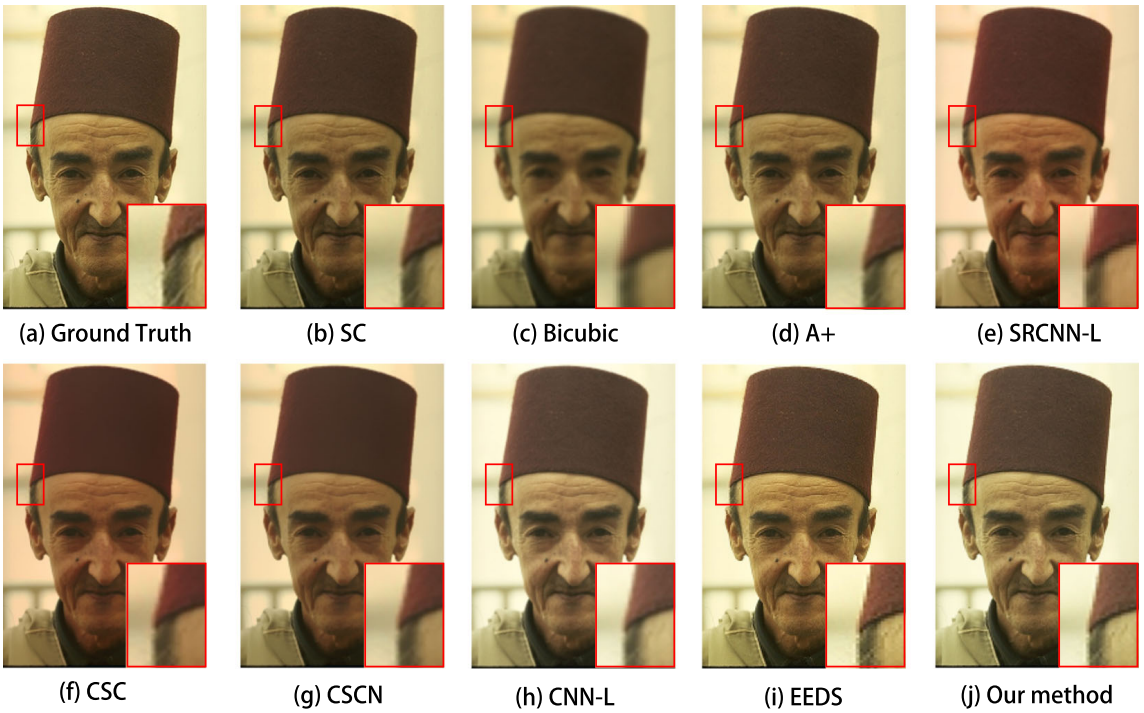


Fig. 7 The 'old man' image from BSD100 with an upscaling factor 3

Table 3 Average PSNR (dB) of every 10,000 training step with an upscaling factor 3

Epoch(thousand)	20	40	60	80	100
Our Model	8.47	13.37	18.61	22.94	26.47
EEDS	9.28	12.75	17.11	23.54	26.04
CNN	6.89	11.33	16.46	21.52	25.86
SRCNN	7.04	14.24	16.33	20.94	25.65
A+	25.06				
SC	25.32				
Bicubic	23.87				

The HR images generated by our method are perceptually more distinct with relatively sharp edges and little artifacts.

4.3 Average comparison with state-of-the-art

In order to further prove that our method averagely performs well in SR task, and can be applied to cope with realistic problems, we additionally collected over 15,000 natural pairs from the internet, among which each pair has two dictionaries for high resolution and low resolution image patches. And, 10,000 of those image pairs are used to train and the remaining images are used for test and validation. Specifically, our model has the same structure as Dense-5 in Fig. 5, every kernel's size is shown in Fig. 3. The learning rate is initially set to 0.001 and decreased by a factor of 1.5 every 10,000 epochs, and we train our model 100,000 epoches in total. The upscaling factor is set to 3. The whole training phase on our workstation(configuration information is shown in Section 4) lasted for approximately 7 hours. The test results are shown in Table 3. Here we show our model's reconstruction performance every 10,000 training epoches, and we use PSNR to make a quantitative comparison. As for example-based method, we give its final results. In order to have fair comparison, the listed method in Table 3 utilize the same training set as ours, and train 100,000 epoches too, other parameters remain unchanged.

Table 3 illustrates that our model's performance regularly improves with training epoches. While the performance is limited by original small training set compared with ImageNet, the reconstruction accuracy is still better than numbers of methods like CNN and SRCNN.

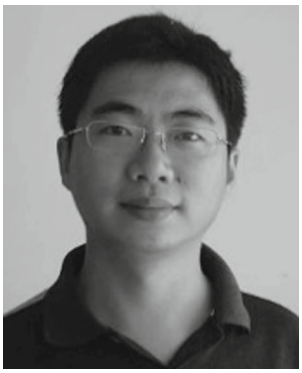
5 Conclusion

This paper presents a new deep learning model with densely connected layers for image super resolution. Our method's core idea is to learn an end-to-end mapping between low and high-resolution images. During the process, all the features are directly extracted and learnt from input, and are used to up-sampling and expand latent feature resolution to target resolution. Furthermore, due to the ability of densely connected layers, our network is able to extract more features which contribute to acquire more high-frequent information, and that information is the main factor for image reconstruction. With lightweight structure, our method has achieved better super-resolution quality than most of current methods. Still, our model's speed can be further improved by applying more advanced network construction methods. For instance, in our future research, we will draw lessons from DeepSCNs network [43], building a deep network by randomized approach. Through this method, network's weights and biases will be constrained and every hidden layer can have a direct connection with output layer. As a result, the network is able to produce more rich representations and speed up the construction of itself. Other than that, we will also adopt PBT [44] (population based training) to shorten our model's training and fine-tuning time.

References

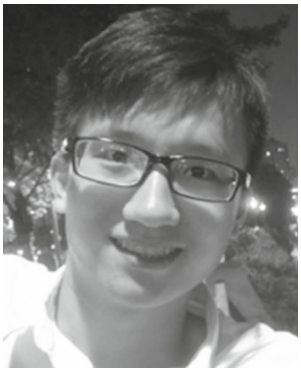
1. Glasner D, Bagon S, Irani M (2009) Super-resolution from a single image. In: Proceedings of IEEE international conference on computer vision. Kyoto, pp 349–356
2. Dai D, Timofte R, Van Gool L (2015) Jointly optimized regressors for image super-resolution. Comput Graph Forum 34(2):95–104
3. Freedman G, Fattal R (2011) Image and video upscaling from local selfexamples. ACM Trans Graphs (TOG) 30(2):12
4. Chang H, Yeung DY, Xiong Y (2004) Super resolution through neighbor embedding. In: Proceedings of the 2004 IEEE Computer society conference on IEEE computer vision and pattern recognition, vol 1, no 1. Washington, DC, pp 1–1
5. Cui Z, Chang H, Shan S, Zhong B, Chen X (2014) Deep network cascade for image super-resolution. European conference on computer vision. Springer, Cham, pp 49–64
6. Bevilacqua M, Roumy A, Guillemot C, Morel MLA (2012) Low-complexity single-image super- resolution based on nonnegative neighbor embedding. In: Proc Brit Mach vis conf, pp 1–10
7. Yang C-Y, Yang M-H (2013) Fast direct super-resolution by simple functions. In: Proceedings of IEEE International conference on computer vision. Sydney, pp 561–568
8. Freeman WT, Pasztor EC, Carmichael OT (2000) Learning lowlevel vision. Int J Comput Vis 40(1):25–47
9. Freeman WT, Jones TR, Pasztor EC (2002) Example-based superresolution. IEEE Comput Graph Appl 22(2):56–65
10. Yang J, Wright J, Huang TS, Ma Y (2010) Image super-resolution via sparse representation. IEEE Trans Image Process 19(11):2861–2873
11. Zeyde R, Elad M, Protter M (2010) On single image scale-up using sparse-representations. In: International conference on curves and surfaces. Springer, Berlin, pp 711–730
12. Schulter S, Lesistner C, Bischof H (2015) Fast and accutate image upscaling with super-resulution forests. In: Proceedings of IEEE Conference on computer vision and pattern recognition, pp 184–199
13. Yang CY, Ma C, Yang MH (2014) Single-image super-resolution: Abenchmark. In: European Conference on computer vision. Springer, Cham, pp 372–386

14. Freeman WT, Pasztor EC, Carmichael OT (2000) Learning low-level vision. Int J Comput Vis 40(11):25–47
15. Huang JJ, Siu WC, Liu TR (2015) Fast image interpolation via random forests. IEEE Trans Image Process 24(10):3232–3245
16. Salvador J, PerezPellitero E (2015) Naive Bayes super-resulution forest. In: Proceedings of IEEE International conference on computer vision, pp 325–333
17. Huang JB, Singh A, Ahuja N (2015) Single image super-resolution from transformed self-exemplars. In: Proc IEEE Conf Comput vis pattern recognit, pp 5197–5206
18. Yang J, Lin Z, Cohen S (2013) Fast image super-resolution based on in-place example regression. In: 2013 IEEE Conference on IEEE computer vision and pattern recognition (CVPR), pp 1059–1066
19. Timofte R, De Smet V, Van Gool L (2014) A+: adjusted anchored neighborhood regression for fast super-resolution. In: Asian Conference on computer vision. Springer, Cham, pp 111–126
20. Jia K, Wang X, Tang X (2013) Image transformation based on learning dictionaries across image spaces. IEEE Trans Pattern Anal Mach Intell 35(11):367–380
21. Yang J, Wang Z, Lin Z, Cohen S, Huang T (2012) Coupled dictionary training for image super resolution. IEEE Trans Image Process 21(8):3467–3478
22. Kim KI, Kwon Y (2010) Single-image super-resolution using sparse regression and natural image prior. IEEE Trans Pattern Anal Mach Intell 32(6):1127–1133
23. Schulter S, Leistner C, Bischof H (2015) Fast and accurate image upscaling with super-resolution forests. In: Proc IEEE Conf Comput Vis Pattern Recognit, pp 3791–3799
24. Timofte R, Smet V, Gool L (2013) Anchored neighborhood regression for fast example-based super-resolution. In: Proceedings of IEEE International conference on computer vision, pp 1920–1927
25. Dong C, Loy CC, He K, Tang X (2014) Learning a deep convolutional network for image super-resolution. In: Proceedings of European conference on computer vision, pp 184–199
26. Dong C, Loy CC, He K (2016) Image super-resolution using deep convolutional networks. IEEE Trans Pattern Anal Mach Intell 38(2):295–307
27. Yang J, Wright J, Huang TS, Ma Y (2010) Image super-resolution via sparse representation. IEEE Trans Image Process 19(11):2861–2873
28. Zeyde R, Elad M, Protter M (2012) On single image scale-up using sparse-representations. In: Proc. 7th Int Conf curves surfaces, pp 711–730
29. Yang J, Wright J, Huang T, Ma Y (2008) Image super-resolution as sparse representation of raw image patches. In: Proc IEEE Conf comput vis pattern recog, pp 1–8
30. Shi W, Caballero J, Huszar F, Totz J, Aitken AP, Bishop R, Rueckert D, Wang Z (2016) Real-time single image and video superresolution using an efficient sub-pixel convolutional neural network. In: Proceedings of IEEE Conference on computer vision and pattern recognition, pp 1874–1883
31. Kim J, Lee JK, Lee KM (2016) Accurate image super-resolution using very deep convolutional networks. In: Proceedings of IEEE Conference on computer vision and pattern recognition. arXiv:1511.04587
32. Keys RG (1981) Cubic convolution interpolation for digital image processing. IEEE Trans Acoust Speech Signal Process 29(6):1153–1160
33. Wang Z, Liu D, Yang J, Han W, Huang T (2015) Deep networks for image super-resolution with sparse prior. In: Proceedings of IEEE International conference on computer vision, pp 370–378
34. Duchon CE (1979) Lanczos filtering in one and two dimensions. J Appl Meteorol 18(8):1016–1022
35. Sun J, Sun J, Xu Z, Shum H-Y (2008) Image super-resolution using gradient profile prior. In: Proceedings of IEEE Conference on computer vision and pattern recognition, pp 1–8
36. Protter M, Elad M, Takeda H, Milanfar P (2009) Generalizing the nonlocal-means to super-resolution reconstruction. IEEE Trans Image Process 18(1):36–51
37. Zhang K, Gao X, Tao D, Li X (2012) Single image super-resolution with non-local means and steering kernel regression. IEEE Trans Image Process 21(11):4544–4556
38. Baker S, Kanade T (2000) Limits on super-resolution and how to break them[J]. IEEE Trans Pattern Anal Mach Intell 24(9):1167–1183
39. Gu S, Zuo W, Xie Q, Meng D, Feng X, Zhang L (2015) Convolutional sparse coding for image super-resolution. In: Proceedings of IEEE International conference on computer vision, pp 1823–1831
40. Martin D, Fowlkes C, Tal D, Malik J (2001) A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of IEEE International conference on computer vision, vol 2, pp 416–423
41. Bevilacqua M, Roumy A, Guillemot C, Alberi-Morel M-L (2012) Lowcomplexity single-image super-resolution based on nonnegative neighbor embedding, 1–10
42. He K, Sun J (2015) Convolutional neural networks at constrained time cost. In: Proc IEEE Conf comput vis pattern recog, pp 379–3799
43. Wang D, Li M (2017) Deep stochastic configuration networks with universal approximation property. arXiv:1702.05639
44. Jaderberg M, Dalibard V, Osindero S et al (2017) Population based training of neural networks. arXiv:1711.09846

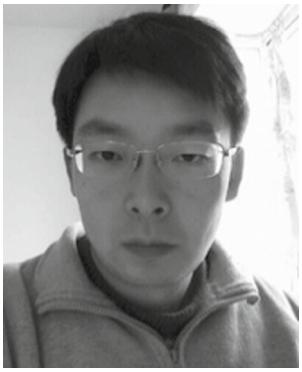


Ping Kuang received the Dr. degree from University of Electronic Science and Technology of China in 2006. He is now an associate professor with School of Information and software engineering. At present, he is mainly engaged in the research of image processing, artificial intelligence and pattern recognition.

Tingsong Ma is currently working toward the M.D. degree in the School of Information and software engineering, University of Electronic Science and Technology of China. His research interests are in artificial intelligence, image processing, machine learning, object detection and medical image.



Ziwei Chen is currently working toward the M.D. degree in the School of Information and software engineering, University of Electronic Science and Technology of China. His research interests are in machine learning, pattern recognition, image processing and algorithms.



Fan Li received the Dr. degree from University of Electronic Science and Technology of China in 2008. More than 10 academic papers have been published as the first author in IEICE, Transactions on Information and System, control and decision making, AI and pattern recognition.