The Visual Computer (2021) 37:925–938
https://doi.org/10.1007/s00371-020-01843-3

ORIGINAL ARTICLE

Back-projection-based progressive growing generative adversarial network for single image super-resolution

Tingsong Ma[1] · Wenhong Tian[1]

Published online: 30 April 2020

Abstract
Recent advanced deep learning studies have shown the positive role of feedback mechanism in image super-resolution task. However, current feedback mechanism only calculates residual errors of images with the same resolution without considering the useful features that may be carried by different resolution features. In this paper, to explore the potential of feedback mechanism, we design a new network structure (progressive up- and downsampling back-projection units) to construct a generative adversarial network for single image super-resolution and use progressive growing methodologies to train it. Unlike previous feedback structure, we use progressively increasing scale factor to build up- and down-projection units, which aims to learn fruitful features across scales. This method allows us to get more meaningful information from early feature maps. Additionally, we train our network progressively; in the process of training, we start from single layer network structure and add new layers as the training goes on. By this mean, the training process can be greatly accelerated and stabilized. Experiments on benchmark dataset with the state-of-the-art methods show that our network achieves 0.01 dB, 0.11 dB, 0.13 dB and 0.4 dB better PSNR results than that of RDN+, MDSR, D-DBPN and EDSR on 8× enlargement, respectively, and also achieves favorable performance against the state-of-the-art methods on 2× and 4× enlargement.

Keywords Back-projection · Progressive growing · Generative adversarial networks · Single image super-resolution

1 Introduction

Machine vision, as one of the most significant researching field of deep learning has already obtained substantial achievements [1–6]. Recently, this trend has also occurred in image super-resolution (SR) field, where many extraordinary works have shown up [7–12]. Our design draws significant lessons from those works, making good use of their ideas, theory and methodology to make a new generative network that achieves even better performance on image super-resolution task.

Single image super-resolution (SISR) is designed to generate visually pleasing high-resolution (HR) images through its degraded low-resolution (LR) input. SISR is used for variety of computer vision tasks such as medical [13] and security image [14]. The biggest problem of SISR is that there are tremendous possible choices existing for the given LR image to generate a HR image, and we do not know which option is right. To address this problem, plenty of image SR resolution have been proposed, some of which are based on interpolation [15] and some are based on reconstruction [16]. But the most popular and best approaches are learning-based methods [10,17–23]. These learning-based approaches compute a sequence of feature maps from LR image, increasing resolution by going through one or multiple upsampling layers and finally generate the HR image.

Christian et al. [24] firstly introduce a generative adversarial network for single image super-resolution problem and achieve significant improvement over conventional methods. After this, Kim et al. propose DRCN [25] and VDSR [10] by increasing the network's depth and using gradient clipping, RNN, skip connection and recursive supervision to ease the difficulty of training deep network. DRCN also learns the difference between a HR image and a LR image, that is, to restore the high-frequency portion of the image, which helps DRCN get better performance. Lim et al. propose a very wide network EDSR [26], which uses residual blocks and scaling.

✉ Wenhong Tian
tian_wenhong@uestc.edu.cn

Tingsong Ma
201811090811@std.uestc.edu.cn

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

They also propose a very deep network MDSR [27]. Furthermore, Yulun et al. propose a novel residual dense network RDN [28], which uses residual dense block to extract local features through dense connected convolutional layers, and utilize global feature fusion to jointly and adaptively learn global hierarchical features in a holistic way. All of these methods are purely feed forward. However, some researches [29,30] have shown that human visual system is likely to use a feedback connection to complete the task. Based on this theory, Muhammad et al. propose a feedback-based network DBPN [31], which is an end-to-end trainable architecture based on the idea of iterative up- and downsampling. The core idea of DBPN is to compute the reconstruction error of every up- and downsampling layers, and then fuse the errors back to the output to improve image quality.

Inspired by DBPN [31], we find that in every up- and down-projection units, DBPN uses the same scaling factor, which is set according to a hyper parameter (e.g., 2, 4 and 8). As a result, some of the low-resolution features might be missed, which could lead a negative impact on the final output. Therefore, according to this observation, in this paper, we propose a new CNN-based method for SISR which is able to exploit more information and feature maps from different scaling contexts. To achieve this goal, the back-projection module is improved by using a progressive amplification structure, instead of taking all the same scaling factor. And the progressive training strategy is adopted for fast training and convergence. The experiments on several benchmarks verified the effectiveness of proposed method. Our work provides the following contributions.

Progressive projection units We propose new up- and down-projection units, where their scaling factors are not affected by hyper parameter setting and can calculate both up- and down-projection errors in different scaling contexts. Detailed explanations can be seen in Sect. 3.1. Furthermore, our network takes projection block as basic unit, and every block contains several projection units with different scaling factors. This structure allows the block to produce output with target size by gathering features from different scaling contexts. Besides that, each block is connected by dense connection, and by which mean, the network can share features from each block, making training smooth and improve network's performance.

Progressive growing For the generator network, we use progressive growing methodology to train it. The training progress can be seen in Fig. 3, where our generator's training process starts from only one projection block, and then, the network's structure progressively grows via adding new blocks. Before adding a new block into the network, we use a similar strategy as in [32] that inserts the new block smoothly, in case it disrupts the previous well-trained blocks.

The remainder of this paper is organized as follows: In Sect. 2, we review some related state-of-the-art deep learning-based approaches. The proposed method is described in Sect. 3. Section 4 shows experimental results, and Sect. 5 concludes this paper.

2 Related work

2.1 GAN for image SR

SRGAN [24] is the first proposed generative adversarial network for single image super-resolution, where a deep convolutional network with residual blocks is introduced as its generator, and another deep convolutional network is regarded as its discriminator. SRGAN utilizes a perceptual loss function to generate more human visual system oriented HR pictures. SRGAN performs outstanding results compared to conventional network, and after that, more and more GAN-based SR methods are proposed [33–35]. Among them, ESRGAN [34] obtains the best results by replacing batch normalization (BN) layers of generator in SRGAN with residual dense block and using new loss function and training methodology.

2.2 Deep back-projection networks

Deep back-projection network (DBPN) [31] is a super-resolution networks which is based on the theory of iterative back-projection (IBP) [36]. According to IBP algorithm, the efficient iterative procedure can minimize the reconstruction error and improve the quality of SR image.

Usually, deep super-resolution networks learn a feature first and then map low-resolution (LR) image space to high-resolution (HR) image space with the help of its powerful nonlinear mapping ability. But in DBPN, the author thinks that the conventional operation may not be effective for mining the mutual dependencies between LR and HR images. Therefore, the author constructs a network with continuous up- and downsampling stages for error feedback. Its idea is similar to IBP algorithm, and its output achieves state-of-the-art standard.

Let $I_h \in \mathbb{R}^{M \times N}$ and $I_l \in \mathbb{R}^{M' \times N'}$ be HR and LR images, where $M' < M$, $N' < N$. DBPN's main building block is the projection unit, which is trained to map either an LR feature map to an HR map or an HR map to an LR map. The up-projection unit is defined as follows:

$$\begin{cases} \text{scaleup: } H_0^t = (L^{t-1} * p_t) \uparrow_s, \\ \text{scaledown: } L_0^t = (H_0^t * g_t) \downarrow_s, \\ \text{residual: } e_t^l = L_0^t - L^{t-1}), \\ \text{scaleresidualup: } H_1^t = (e_t^l * q_t) \uparrow_s, \\ \text{outputfeaturemap: } H^t = H_0^t + H_1^t, \end{cases} \tag{1}$$

where $*$ is the spatial convolution operator, $\uparrow_s$ and $\downarrow_s$ are the up- and downsampling operators with scaling factor s, respectively, and p_t, g_t, q_t are the (de)convolutional layers at stage t.

The up-projection unit tries to make the generated HR map back to LR map and then calculates the residual error between the original input LR map and the back-projected LR map. Subsequently, it uses the residual error to generate another HR map. Finally, the output is obtained by summing the two intermediate HR maps.

As for down-projection unit, it has a similar definition, which is shown as follows:

$$\begin{cases} \text{scaledown: } L_0^t = (H^{t-1} * g_t') \downarrow_s, \\ \text{scaleup: } H_0^t = (L_0^t * p_t') \uparrow_s, \\ \text{residual: } e_t^h = h_0^t - h^t), \\ \text{scaleresidualdown: } L_1^t = (e_t^h * g_t') \downarrow_s, \\ \text{outputfeaturemap: } L^t = L_0^t + L_1^t, \end{cases} \tag{2}$$

To further acquire improved features, DBPN adopts dense connections between those up- and down-projection units. In the improved version of DBPN (D-DBPN), up-projection unit has connection with each down-projection unit, and down-projection unit has connection with each up-projection unit.

2.3 Progressive upsampling

LapSRN [11] is a recently SISR approach which utilizes progressive upsampling method to reconstruct HR image from a given LR input. LapSRN progressively reconstructs several LR images with different scales and then concatenates all the LR images to generate HR result. However, LapSRN is just a simple feed-forward network, and only limited features and information from LR input are exploited. Due to this fact, we adopt the progressive upsampling idea and apply it to back-projection background, with the purpose of getting more features from different scaling contexts. Then, we can make good use of them to calculate residual error from output of each projection unit. Finally, the residual error will play a positive role in the final output of our network.

2.4 Progressive growing of GANs

The author of PG-GAN [32] uses a progressive growing method to train their generative adversarial networks, in order to both speed up the training and greatly stabilize it. Their core idea is to start with a shallow network which could only generate low-resolution images and then progressively add new layers to the network to increase the resolution. During this phase, each new layer will be faded in smoothly to make good use of previous training results and avoid sudden shocks to the already well-trained, smaller-resolution layers. Other than this, PG-GANs use standard deviation of feature as a measure to estimate variation and give new normalization method to stabilize training and reduce collapse situation. Furthermore, it also balances the generator and discriminator by adding noise to real sample to ease the mode collapse. The author proposes the following way of adding noise:

$$\text{noisestrength} = 0.2 * max(0,\ d_t - 0.5)^2 \tag{3}$$

where $d_t = 0.9 * d_{t-1} + 0.1 * d$ is an exponential moving average of the discriminator output d.

2.5 Other recent works

Juncheng et al. [23] first make deep insight into some classical SR models, such as SRCNN, EDSR and SRResNet, according to their observation, and they claim that these models are: (a) hard to reproduce (training skills have a great impact); (b) insufficient function utilization (LR image features are not fully utilized, and they gradually disappear with thee increase of depth); (c) poor expansibility (it is difficult to adapt to any upgrade, only tiny adjustment can be made). Thus, they propose a multi-scale residual network (MSRN), which includes two parts: (a) multi-scale feature fusion; (b) and local residual learning. In their work, different bypasses use different convolution kernels to adaptively detect image features of different scales. Additionally, they have shown that using residual learning makes the network more efficient. Yunlun et al. [22] propose a residual channel attention networks (RCAN) which allows researchers to build deeper CNN (includes 10 residual groups and 20 residual channel blocks). The authors claim that depth of network matters, although EDSR and MSDR have made a splash, simply stacking of residual blocks to build a deeper network cannot get better improvement. Therefore, they propose residual-in-residual structure. And another highlight in their work is the channel attention mechanism. This channel attention mechanism can adaptively readjust the characteristics of each channel and focus on more useful channels. Li et al. [37] propose an image super-resolution feedback network to refine low-level representations with high-level information. They utilize hidden states in a RNN with constraints to achieve such feedback manner. The feedback block is designed to handle the feedback connections and to generate powerful high-level representations, which makes their model good at reconstructing SR image at early layers. Furthermore, they also propose a channel attention mechanism, which can adaptively scale the features of each channel by modeling the interdependence between feature channels. Experiments show that this mechanism allows the network to focus on more useful channels and enhance discrimination learning ability.

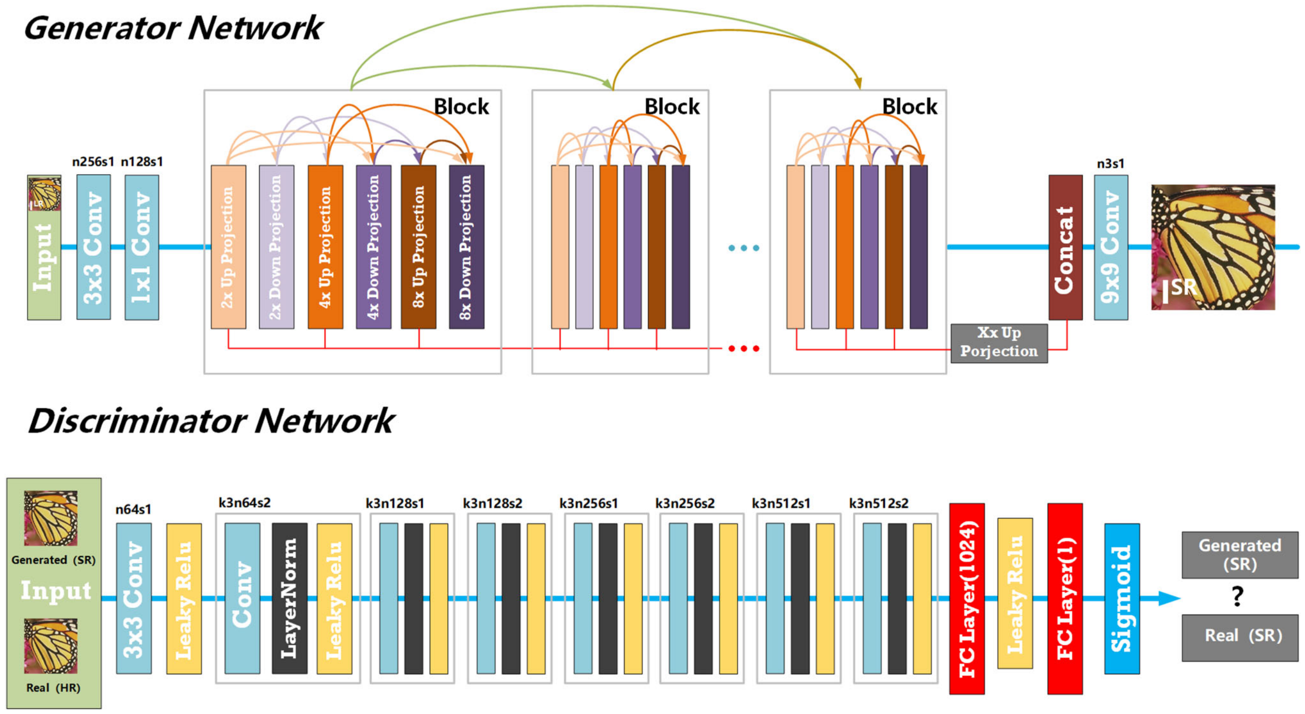

Fig. 1 Architecture of generator G and discriminator D, where k denotes the kernel size, n denotes the number of feature maps, and s denotes the stride for each convolutional layer

3 Method and architecture

In this paper, we aim to propose a generative adversarial network which possesses both back-projection and progressive growing mechanism as well as some other helpful tricks to achieve great image super-resolution (SR) achievement. Our network's architecture is shown in Fig. 1. For convenience, we use $conv(f, n, s)$ to refer to $f \times f$ convolutional layer, n refers to the number of filters and s refers to stride number, and $deconv(f, n, s)$ refers to $f \times f$ deconvolutional layer, n is the number of filter, and s is the stride number.

3.1 Architecture

Feature extraction In generator network, feature extraction phase includes a $conv(3, 256, 1)$ and a $conv(1, n_R, 1)$ layer, where n_R is the number of filters used in each projection unit in the following projection blocks. The output of feature extraction phase will be transmitted to the next series of modules as input (with the size of $w \times h \times n_R$, where w and h denote width and height of input LR image).

Projection units Then, we adopt the same method as [31], using up- and down-projection units to map either a LR feature to HR feature (up-projection) or a HR feature map to a LR map (down-projection). Subsequently, we use the difference between each generated map to update and improve the final output. Up- and down-projection units are defined as Eqs. (1) and (2), respectively.

The difference is that in [31], they use all the same scaling factor to up- and downsampling the feature maps. But in our research, we found that directly up- and downsampling an image to target size with a scaling factor (e.g., 8) could lose plenty of low-resolution details (details in image that is just 2 or 4 times the size of original source). By collecting those information may further help us to improve the final quality of outputs. Therefore, we consider to use a progressive amplification structure, as shown in Fig. 1. In our generator, up- and down-projection units have a fixed scale factor, and the structure of the network is dynamic. For example, if we want to enlarge an image 8 times, the projection block would be just exactly the same as shown in Fig. 1. In addition, if we only want to 4 times enlarge an image, in the block there will only be one up- and down-projection with scaling factor 2 and another one with scaling factor 4. As a result of this structure, we can extract more feature from different scaling contexts.

Details of projection units are shown in Fig. 2. $\left[L^1, \ldots, L^{t-1}\right]$ and $\left[H^1, \ldots, H^{t-1}\right]$ denote the feature maps of all preceding down- and up-projection units, respectively. The first $conv(1, n_R, 1)$ is used to reduce the dimension from $n_c = t - 1 * n_R$ to n_R before entering projection step, where n_R is the number of filters used in each projection unit, and $t - 1 \times n_R$ is the sum of channels from all preceding units,

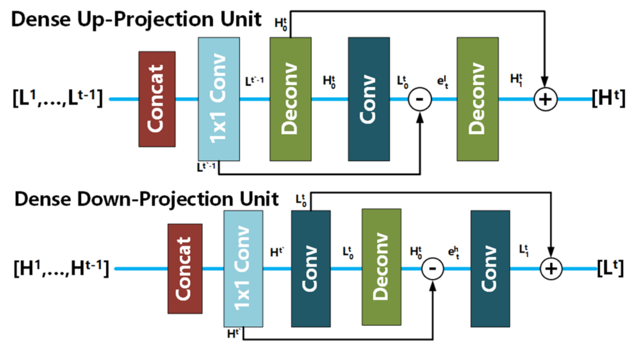

Fig. 2 Structure of densely connected up- and down-projection units

t is serial number of current units. In each projection unit, it uses $deconv(f, n_R, s)$ to upsample the input to target size, where s refers to scaling factor, and then uses $conv(f, n_R, s)$ to downsample the output of preceding $deconv$ layer to LR size again. For up-projection unit, it gets input with size of $w \times h \times n_c$ (where w and h denote width and height of input LR image, and $n_c = t - 1 * n_R$), and outputs $W \times H \times n_R$ (where W and H refer to width and height of HR image). The shape of down-projection unit's input and output is exactly the opposite of that of up-projection unit's ($W \times H \times n_R$ for input and $w \times h \times n_c$ for output).

Projection blocks Besides, as shown in Fig. 1, we take every one ($2\times$ up- and down-projection) or two ($2\times$ and $4\times$ up- and down-projection) or three ($2\times$, $4\times$ and $8\times$ up- and down-projection) couple of up- and down-projection units as a projection block. And dense structure happens in every block. In each dense block, every up-projection unit is associated with all subsequent down-projection unit. In addition, before the concatenation of every up-projection unit, there is a X up-projection unit, where X represents the magnified multiple the output of each preceding up-projection unit needs to hit the target size, since up-projection unit has multiple scale factors.

Reconstruction Finally, the target HR image is reconstructed as $I^{SR} = R([H^1, H^2, \ldots, H^t])$, where R uses $conv(9, 3, 1)$ as reconstruction and $[H^1, H^2, \ldots, H^t]$ refers to the concatenation of the feature maps produced by each up-projection unit.

Implementation details The filter size of $conv$ layer in every projection unit is various with respect to the scaling factor. For $2\times$ enlargement, we set $f = 6$, $n_R = 32$, $s = 2$ and $w = h = 96$. For $4\times$ enlargement, we set $f = 8$, $n_R = 64$, $s = 4$ and $w = h = 48$. Finally, the $8\times$ enlargement task uses $f = 12$, $n_R = 128$, $s = 8$ and $w = h = 24$. The discriminator is identical to SRGAN [24], and the architecture is shown in Fig. 1. It contains 8 convolutional layers with an increasing number of 3×3 filter kernels, increasing by a factor of 2 from 64 to 512 kernels. As suggested by WGAN-GP [38], we adopt LayerNorm after every convolutional layer instead of Batchnorm and use LeakyReLU activation ($\alpha = 0.1$) and avoid max-pooling throughout the network. The output of final convolutional layer with 512 feature maps is followed by two dense layers and a final sigmoid activation function to obtain a probability for sample classification.

3.2 Loss function

For generator network, we define loss function L_G as:

$$L_G = l_{\text{mse}} + 10^{-3} l_{\text{Gen}} \tag{4}$$

where l_{mse} is a pixel-wise MSE loss, which is formulated as:

$$l_{\text{mse}} = \frac{1}{r^2 WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I^{\text{HR}}_{x,y} - G_{\theta_G}(I^{\text{LR}})_{x,y})^2 \tag{5}$$

where W and H denote the width and height of image, r is the scaling factor when we apply a Gaussian filter and downsampling operator to HR image I^{HR}, by which mean we generate LR image I^{SR} as input. G_{θ_G} is our CNN generator G parameterized by θ_G, where $\theta_G = \{W_{1:L}; b_{1:L}\}$, which denotes the weights and biases of the Lth layer.

And l_{Gen} is calculated by:

$$l_{\text{Gen}} = \text{CE}(Sig(D_{\theta_D}(I^{\text{SR}})),\ O(D_{\theta_D}(I^{\text{SR}}))) \tag{6}$$

where CE is cross-entropy function, Sig is sigmoid function and $D_{\theta D}$ is our discriminator parameterized by θ_D, where $\theta_D = \{W_{1:L}; b_{1:L}\}$, which denotes the weights and biases of the Lth layer; $I^{\text{SR}} = G_{\theta_G}(I^{\text{LR}})$; $O(D_{\theta_D}(I^{\text{SR}})$ is a matrix which has the same shape as $D_{\theta_D}(I^{\text{SR}})$ with all elements set to 1.

For discriminator, the loss function L_D is defined as:

$$\begin{aligned} L_D = {} & \text{CE}(D_{\theta D}(I^{\text{HR}}),\ E(D_{\theta D}(I^{\text{HR}}))) \\ & + \text{CE}(D_{\theta D}(I^{\text{SR}}),\ Z(D_{\theta D}(I^{\text{SR}}))) \end{aligned} \tag{7}$$

where $Z(D_{\theta D}(I^{\text{SR}}))$ is a matrix which has the same shape as $D_{\theta D}(I^{\text{SR}})$ with all elements set to 0.

Here, we do not adopt a VGG loss as SRGAN [24] does, because our model utilizes back-projection units, which helps the model acquire deeper features to reconstruct numerous LR and HR features, and finally guides the reconstruction for making better results. As a result of that, our loss is not a perceptual loss function, since we do not get SR image and HR image features from a pre-trained VGG network. Therefore, the output of our model is not that comfortable from the perspective of human, but it gets higher PSNR and SSIM results, and its visual effect is just slightly poorer, and we believe this is an acceptable trade-off.

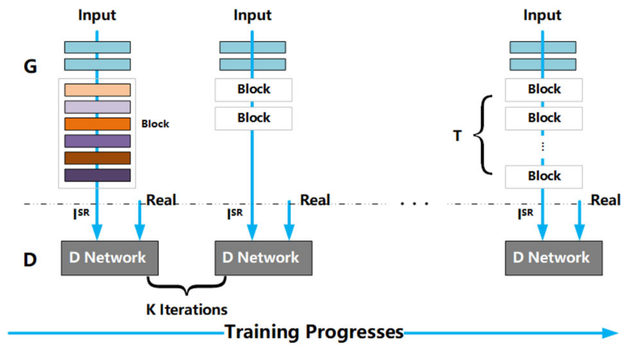

Fig. 3 The procedure of progressive training

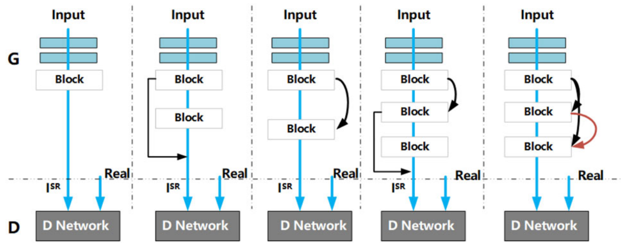

Fig. 4 The process of how we add new layer smoothly

3.3 Progressive growing

3.4 Model analysis

In our research, we also adopt a new training methodology from PG-GAN [32], with the purpose of improving training efficiency, avoiding to learn all layers simultaneously. One of the biggest differences from PG-GAN is that this training method is only applied to generator, instead of both generator and discriminator, since our goal is to generate super-resolution images, we need the discriminator to guide the generator to train in the right direction. Otherwise, the discriminator may not be powerful enough to provide generator with proper guidance in early stage of training process.

Another difference is that at each time, we do not just insert a new layer but a complete projection block as visualized in Fig. 3. As the training advances, we progressively add a new projection block to generator and keep discriminator structure unchanged. Before the training, there is a hyper parameter T, which decides when the training progress needs to stop adding new block to generator. Other than that, parameter K means during training, the network will add a new block every K training iterations. When T blocks are in place, the training will turn to be an ordinary training.

Each time we add a new block in the network, in order to avoid shocks to the already well-trained layers, we also fade them in smoothly like what PG-GAN does as illustrated in Fig. 4. The difference is that after each smooth fading operation, we create dense connections between each projection blocks; as a result, the output from all previous blocks can be concatenated, enabling us to generate the feature maps effectively.

4 Experiment

4.1 Implementation and training details

As shown in Fig. 1, some details have already been illustrated. Before entering the project block, we apply one 3×3 convolution layer (stride is 1 and the number of filters is 256) and one 1×1 convolution layer (stride is 1 and the number of filters is n_R, where n_R is the number of filters used in each projection unit in the following projection blocks) to initially extract features and reduce the dimension. In the proposed network, the kernel size of convolution and deconvolution layer in each projection unit is various with respect to the scaling factor, like what they do in [31]. For $2\times$ enlargement, kernel size is 6×6 with 2 striding and $n_R = 32$. $4\times$ enlargement projection unit has 8×8 filter size and 4 striding and $n_R = 64$. Then, $8\times$ enlargement uses 12×12 convolutional layer with 8 striding and $n_R = 128$. Besides, we also set different numbers of projection blocks (T) according to different scaling factor, which are set to 8 ($2\times$), 6 ($4\times$) and 4 ($8\times$), respectively.

We initialize the weight using a zero-mean Gaussian distribution, where $std = \sqrt{\frac{2}{n_l}}$ [39]. Setting $n_l = f_t^2 n_t$, f_t is the filter size, and n_t is the number of filters. The bias is initialized to zero. Every convolutional and deconvolutional layers in projection units are followed by parametric rectified linear units (PReLUs).

In model analysis phase, to make a quick comparison we train our method by using images from DIV2K [40] (800 images) and part of ImageNet [41] (about 65K high-resolution images) without any augmentation. When compared with other methods, we use DIV2K [40] and about 125K pictures collected from ImageNet [41] to train and test them on Set5 [42], Set14 [43], BSD100 [44], Urban100 [45] and Manga109 [46]. We apply a Gaussian filter and downsampling operator to HR image I^{HR} to obtain LR image I^{SR} as input with size 32×32. We use batch size of 32 for $2\times$ enlargement task, 16 for $4\times$ enlargement and 8 for $8\times$, and learning rate is set to 1e$-$3 for all layers and decreased by a factor of 10 for every 4×10^4 iterations for total 2×10^5 epochs. For optimization, we use Adam with momentum to 0.99 and weight decay to 0.001. All experiments were conducted using Tensorflow (version 1.10.0) and MATLAB R2016a on a workstation with 32GB memory, a NVIDIA GTX 1080Ti GPU and a 3.70GHz i7-8700K CPU.

Depth analysis To have a deep insight into the structure we adopt for $2\times$, $4\times$ and $8\times$ super-resolution tasks,

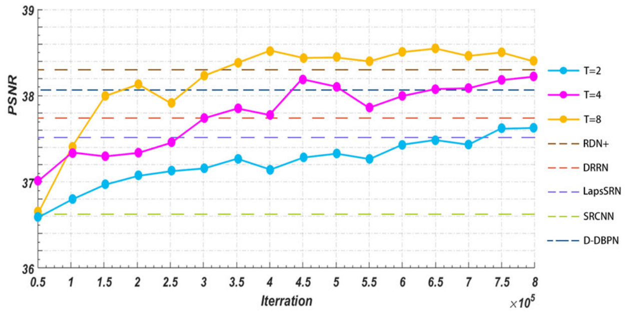

Fig. 5 The depth analysis of proposed network compared to other networks (D-DBPN [31], SRCNN [8], LapSRN [11], DRRN [12] and RDN+ [28]) on Set5 dataset for 2× enlargement

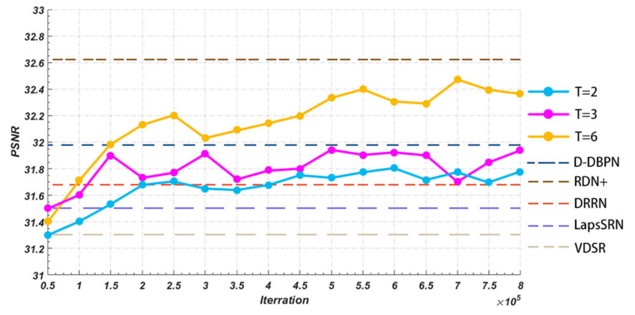

Fig. 6 The depth analysis of proposed network compared to other networks (VDSR [10], LapSRN [11], DRRN [12], RDN+ [28] and D-DBPN [31]) on Set5 dataset for 4× enlargement

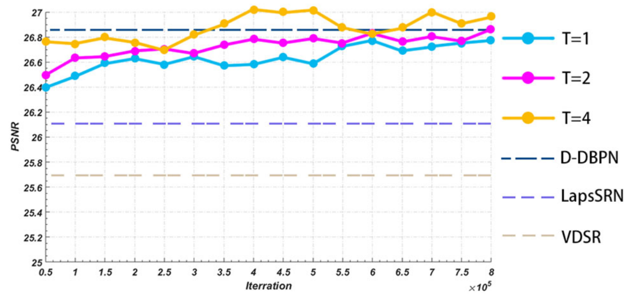

Fig. 7 The depth analysis of proposed network compared to other networks (VDSR [10], LapSRN [11], and D-DBPN [31]) on Set5 dataset for 8× enlargement

we construct multiple networks structures for those tasks, respectively. And it should be noted that only normal training method is used in this experiment, and we use T to refer to the number of projection block we use in network structure. For 2× task, $T = 2, 4, 8$. As for 4× and 8× enlargement, $T = 2, 3, 6$ and $T = 1, 2, 4$, respectively. According to different number of T, the network will be referred as S, M and L, respectively. Before the first projection block, we use $conv(3, 256)$, followed by $conv(1, 128)$. Then, we use $conv(9, 9)$ for the reconstruction. The input and output image are luminance only.

The results on 2× enlargement are shown in Fig. 5. The proposed network outperforms the state-of-the-art methods. The S $(T = 2)$ network performs better PSNR results than that of SRCNN and LapSRN and also gives similar performance to DRRN. The S network only uses 16 convolutional layers, much less than LapSRN and DRRN. In addition, with extra convolutional layers, the proposed network could obtain even better results. The M and L network both have shown excellent reconstruction effects. M $(T = 4)$ network achieves 38.22 dB at best performance, which is 0.12 dB, 0.48 dB, 0.7 dB and 1.56 dB better than that of D-DBPN, DRRN, LapSRN and SRCNN. And M network totally uses 32 convolutional layers, slightly more than LapSRN (24 convolutional layers) but much less than DRRN (56 convolutional layers). As for the L $(T = 8)$ network, its performance exceeds all four existing state-of-the-art methods (D-DBPN, SRCNN, LapSRN, DRRN and RDN+). At the best performance, L network can achieve 38.40 dB which is 0.1 dB, 0.3dB, 0.66 dB, 0.88 dB and 1.74 dB better than that of RDN+, D-DBPN, DRRN, LapSRN and SRCNN, respectively.

The results on 4× enlargement are shown in Fig. 6. We do not achieve the best performance at 4× enlargement task. At the best performance, the L $(T = 6)$ network can achieve 32.37 dB, 0.38 dB, 0.69 dB, which are 0.83 dB and 1.07 dB better than that of D-DBPN, DRRN, LapSRN and VDSR, but 0.24 dB less than that of RDN+. However, RDN+ uses over 120 convolutional layers to get the best performance, the L network only uses 96 convolutional layers, which shows the effectiveness of our proposed network.

The results on 8× enlargement are shown in Fig. 7. S, M and L networks have similar best performance (26.73 dB, 26.87 dB, respectively) to D-DBPN (26.86 dB). S and M have already exceed most of the current state-of-the-art approaches (VDSR and LapSRN), while they only use 24 and 48 convolutional layers, respectively, compared to LapSRN (24 conv layers) and D-DBPN (48 conv layers). And L network's best performance is up to 27.03 dB, 0.17 dB and 1.34 dB better than that of D-DBPN and VDSR. However, the L network is only 0.3 dB and 0.16 dB better than that of S and M network, and the gap among these three networks narrows significantly. Therefore, as for 8× enlargement task, the benefit obtained from extra projection blocks is limited and extra technology is needed to boost its performance.

Loss analysis. In order to verify that we should use cross-entropy (CE) in our loss function, we compare the performance of the model with different loss functions. One of the experimental object is our proposed method, and its detail can be seen in Sect. 3.2. As for the contrast object, we remove CE from the proposed loss function. Therefore, the new function is formulated as $l_{\text{Gen}} = -\log D_{\theta_D}(I^{\text{SR}})$, and $L_D = -\log D_{\theta_D}(I^{\text{HR}}) + l_{\text{Gen}}$. The experimental results can be observed in Fig. 10. We can easily observe that PSNR and SSIM results drop rapidly when we remove CE from

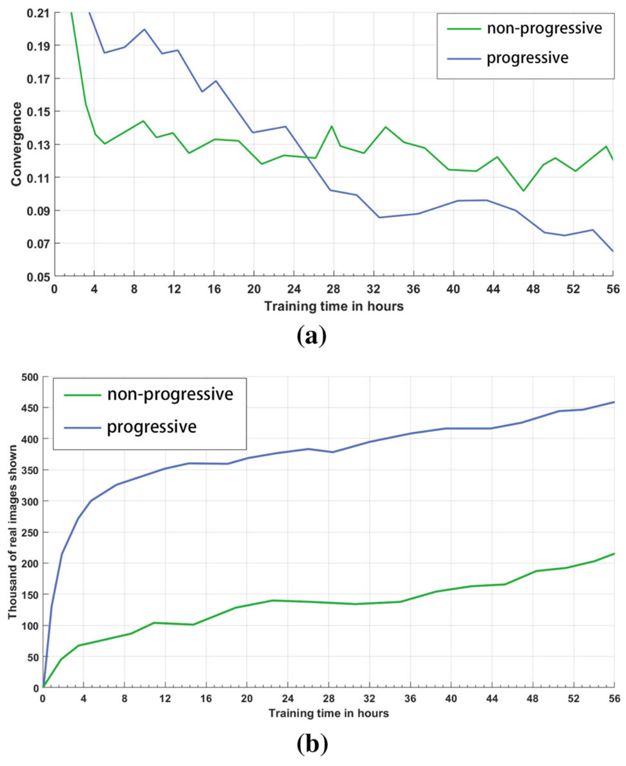

Fig. 8 **a** The impact of progressive growing on training convergence and speed; **b** training progress, measured in number of real images shown to the discriminator

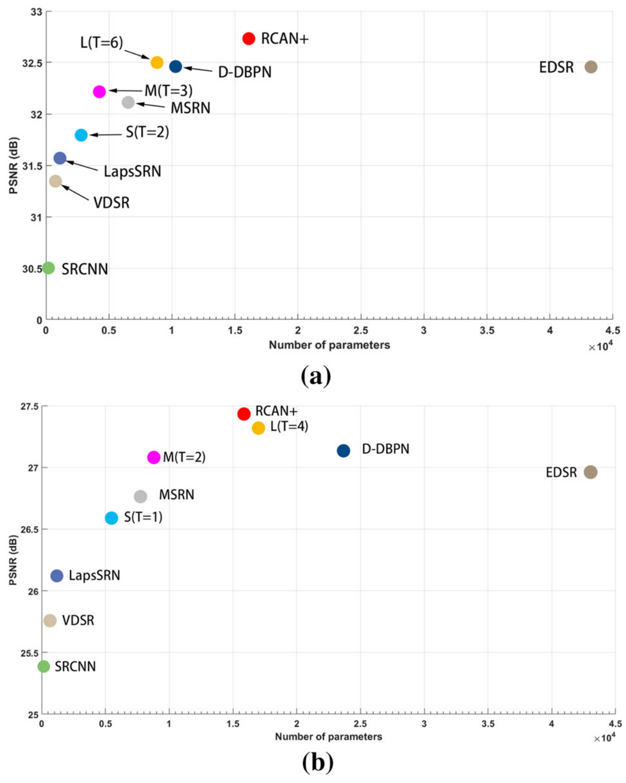

Fig. 9 Set5 dataset is used to test the trade-off between accuracy of 4× (**a**) and 8× (**b**) enlargement reconstruction and the number of parameters they used

our loss function. Besides that, without CE the image generated by our model will contain more blurry boundaries. In the proposed loss function, CE can clearly describe the distance between the model and the ideal model, enhancing the learning ability and thus leading our model to produce more accurate images. Therefore, removing CE from our loss function will have a tremendous impact on the effect of reconstruction.

Progressive training analysis In this experiment, we want to explore the effect of progressive growing on training speed and convergence. The results are shown in Fig. 8. As shown in Fig. 8a, we can observe two primary benefits the progressive training method could bring to the network: (1) It converges way better than that without progressive growing; (2) it also helps the network accelerate training progress. By gradually building the network layers, it can get an increasing learning capacity, which could explain the improved convergence. Other than that, with progressive growing methodology, previous layers are more likely to have converged already; therefore, the network only need to optimize the new introduced layers, which improves training efficiency and reduces training time.

Number of parameters We show the trade-off between performance and number of network parameters from our network and some existing SR methods in Fig. 9. In this experiment, we adopt progressive growing methodology for all the proposed networks. We can observe that both on 4× and 8× enlargement tasks, our *S*, *M* and *L* models have comparable PSNR results. It can be observed that our *S* network has about 56% and 140% more parameters than that of LapSRN on 4× and 8× enlargement, and PSNR results are 0.2 dB and 0.4 dB better, respectively. Additionally, although we do not get the best PSNR performance on 4× and 8× task (about 0.2dB and 0.1dB less) as RCAN+ dose, the parameters our model needs are about 50% less than that of RCAN+ on 4× task, and almost the same on 8×. In particular, RCAN+ consists of 10 residual groups, 20 residual blocks and initial features are set to 64, and uses self-ensemble strategy. Due to residual-in-residual (RIR) structure and channel attention (CA) mechanism RCAN+ adopts, it is able to build a very deep network (up to 400 layers), which is why it needs more parameters than most SR algorithms. Because we adopt similar structure to D-DBPN, parameters we need will increase with magnification factor, and that causes our performance less than that of RCAN+ on 8× enlargement task.

Dense projection blocks Between each projection block, we implement dense connections. In this experiment, we want to show the positive role of dense-connections between projection blocks in our network. Thus, we utilize *L* and *L*-Dense networks as our experimental object which have 4, 3

Table 1 Comparison of L and L-Dense network on $2\times$, $4\times$ and $8\times$ enlargement

Model	Scale	Set5		Set14	
		PSNR	SSIM	PSNR	SSIM
L	2	38.4	0.954	33.80	0.911
L-Dense	2	**38.51**	**0.958**	**34.89**	**0.912**
L	4	32.38	0.897	28.88	0.781
L-Dense	4	**32.51**	**0.899**	**28.98**	**0.783**
L	8	26.85	0.772	24.88	0.641
L-Dense	8	**27.04**	**0.773**	**24.99**	**0.645**

Bold indicates the best performance

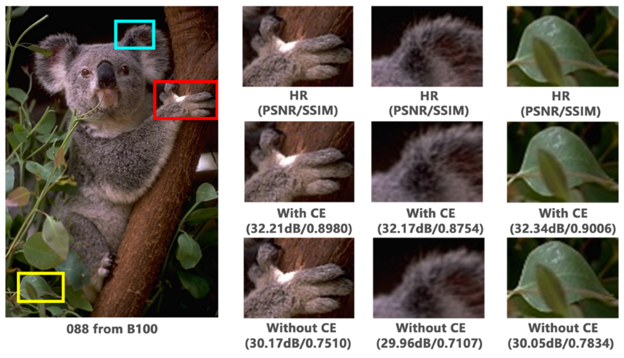

Fig. 10 Compare the effect of reconstructed image using different loss functions [$4\times$ upscaling]. CE refers to cross-entropy function

and 2 projection blocks for $2\times$, $4\times$ and $8\times$ task, respectively. And between L and L-Dense dense-connections is the unique variable. In their structure, before the first projection block, we use $conv(3, 256)$, followed by $conv(1, 128)$. Then, we use $conv(9, 9)$ for the reconstruction layer. The input and output images are luminance only. And training details keep the same as described in Sect. 4.1. The results can be seen in Table 1. We can observe that on $2\times$ enlargement, L-Dense has 0.11 dB and 0.08 dB better than that of L on Set5 and Set14, respectively. And on $4\times$ enlargement, the numbers become 0.13 dB and 0.1 dB. As for $8\times$ enlargement, this gap between L and L-Dense has further expanded, reaching 0.19 dB and 0.11 dB. Thus, we can claim that dense-connections brings positive impact on the network's performance, and this enhancement increases as the scale factor increases (Fig. 10).

Structure design analysis To clearly demonstrate the effect of network structure we designed (e.g., dense connection in each projection unit, progressive projection unit), we compare our full model with other two special models (one has no dense connection in each projection units and the other one without progressive projection unit). Figure 11 shows the convergence curves in terms of PSNR on the set14 dataset for $8\times$ SR. The performance of the 'non-dense' network converges (pink curve) much harder than the other models and fluctuates significantly. And in Table 2 we can observe that

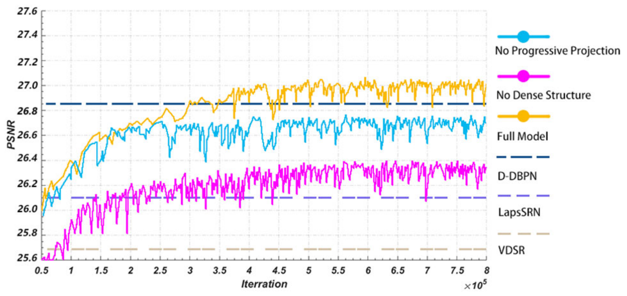

Fig. 11 Convergence analysis on the progressive projection structure and dense connection in every projection block on the Set14 for $8\times$ SR. Our full model converges very fast and achieves improved performance

Table 2 Study of progressive projection units and dense connection in each projection units

Dense	Progressive	Set5	Set14
✓	✓	**27.08**	**25.03**
×	✓	26.83	24.86
✓	×	26.71	24.62
×	×	26.23	24.18

We replace each component with the one used in existing methods, and observe performance (PSNR) drop on both Set5 and Set14 for $8\times$ SR

removing dense connection in each projection unit obviously reduces model's performance; however, due to the progressive design, the network still provides comparable results. Furthermore, as illustrated in Fig. 11, by removing the progressive projection design, our model falls back to a network similar to D-DBPN but with a GAN structure. The quantitative results in Fig. 11 show that the network without progressive projection units (blue curve) although converges faster than the full model (yellow curve), it gives less performance (about o.4 dB less than that of D-DBPN and 0.8 dB less than that of the full model). The same phenomenon can been seen in Table 2, where the PSRN results drop rapidly on both Set5 and Set14 dataset when we remove the progressive projection design from our proposed method.

Besides that, we also give qualitative comparison in Fig. 12, where we can observe that removing either progressive projection design or dense connection in each projection unit, our model suffers from a dramatic performance dropping. What is more, we can find that progressive projection design plays a much more significant role in our method, since after removing dense connection, our model can still contain relatively clear edge and sharp details. On the contrary, Fig. 12d shows that without progressive projection design, the boundary of the image generated by our model will become very blurred, and has more ringing artifacts.

Summary and analysis According to the previous experiments, we have separately shown the positive role played by all parts of the network structure. The core of our network is

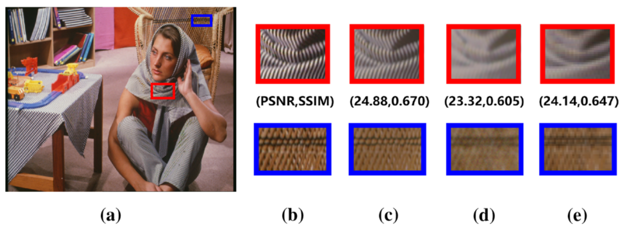

Fig. 12 Contribution of different components in the proposed network. **a** HR image; **b** full model; **c** network without progressive projection structure; **d** network without dense connection in each projection unit

the projection units, where residual errors of reconstructed images at each amplification and narrowing stage are calculated to gradually optimize the final output. What makes our method unique is that we calculate residual errors of reconstructed images with different resolution. Errors obtained by this method will contain both low- and high-frequency features; therefore, abundant useful features can be gathered together to optimize output, generating high-quality images. In terms of overall structure of the proposed network, we adopt the way of dense connection, which alleviates the problem of gradient disappearance, strengthen feature propagation, encourages feature reuse and greatly reduces the number for parameters. Furthermore, densely connected structure also makes the network deeper, more accurate and more effective for training. As for training method, we first train our network with only one projection block, and as the training advances, we gradually add new layers to extract new features. It should be noted that new layers will be added only after the current training effect is stable. Therefore, by using this method our network can be stably trained and produce high-quality images.

4.2 Comparison with state of the art

To precisely evaluate the proposed network, we implement several experiments and analysis to compare our model with other state-of-the-art approaches. The SR algorithms involved in our experiment include A+ [17], SRCNN [8], FSRCNN [47], VDSR [10], EDSR [26], D-DBPN [31], MemNet [48], MDSR [27], RDN+ [28], MSRN [23], SRFBN+ [37] and RCAN+ [22]. In particular, RCAN+ consists of 10 residual groups, 20 residual blocks and initial features are set to 64, and uses self-ensemble strategy. We test all of the models on Set5 [42], Set14 [43], BSD100 [44], Urban100 [45] and Manga109 [46]. We adopt PSNR [49] and SSIM [50] to quantitatively compare the proposed method with other state-of-the-art approaches. It should be noted that the PSNR results of D-DBPN are calculated by dividing the input into four parts.

Our final network uses conv(3, 256) and then conv(1, 128) for the initial feature extraction. For 2× enlargement, $t = 7$ for the number of projection blocks. As for 4× and 8× enlargement task, $t = 6$ and 4, respectively. In the reconstruction phase, we use conv(3, 3). RGB color channels are used for input and output images. It takes about 3 days to train. Some of the method like RDN+ and MemNet did not perform 8× enlargement. Therefore, we retrained the existing networks by using the published codes.

We show the quantitative results in Table 3. Generally speaking, our method outperforms the existing methods on 2×, 4× and 8× enlargement except RCAN+. According to the table, we must admit our model has many shortcomings compared with RCAN+. RCAN+ adopts residual-in-residual structure, where several residual blocks are stacked in each residual group. By this mean, RCAN+ can obtain long short span layer connection, and this linking method in the residual block allows to bypass rich low-frequency information. The advantages of this approach can be seen in the last two columns of Table 3, where the gap between our model and RCAN+ is clearest. Because urban100 and Manga109's images have more sharp edges, which need the model to obtain as many high-frequency information as possible for better reconstruction effect. Due to RCAN+'s structure, it is able to get information much easier than ours; thus, RCAN+'s output can carry more texture and details. And when there are not that many texture or details to reconstruct, our model's performance can be closer to that of RCAN+ (see in the first 6 columns of Table 3). On the other hand, from Fig. 13 we can observe that RCAN+ is slightly faster than our model (about 0.02 second); however, if we reduce the number of layers, the operation speed will be greatly improved, while maintaining a relatively competitive reconstruction performance. In addition, compared with other methods our algorithm has obvious advantages.

Besides that, occasionally our PSNR and SSIM results are worse than that of RDN+. Compared to our dense structure, RDN+ uses local feature fusion to adaptively learn more effective features from preceding and current local features and stabilize the training process of wider network. Furthermore, residual dense blocks in RDN+ concatenate previous features few more times than our dense structures. On the other hand, Table 3 also shows that our proposed models are likely to have better reconstruction effect on Set5, Set14, BSD100 and urban100 than that on Manga109. Because we train our network on ImageNet and DIV2K, where most of the images are natural images. Even though, our performance on Manga109 also has 24.88 dB PSNR which is 0.01 dB and 0.22 dB better than that of RDN+ and MDSR. Eventually, we can observe that the second-best performance we get is just slightly (e.g., 0.03 dB) worse than the best performance.

The results of our mode in Figs. 14 and 15 have shown softer pattern compared to D-DBPN, EDSR, RDN and MDSR as well as more harder edge. However, on the other hand, the output of our model is brighter than the ground

Table 3 Average PSNR/SSIM for scale factors $2\times$, $4\times$ and $8\times$

Algorithm	Scale	Set5		Set14		BSD100		urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	2	33.65	0.930	30.34	0.870	29.56	0.844	27.39	0.841	31.05	0.935
A+ [17]	2	36.54	0.954	32.40	0.906	31.22	0.887	29.23	0.894	35.33	0.967
SRCNN [8]	2	36.65	0.954	32.29	0.903	31.36	0.888	29.52	0.895	35.72	0.968
FSRCNN [47]	2	36.99	0.955	32.73	0.909	31.51	0.891	29.87	0.901	36.62	0.971
VDSR [10]	2	37.53	0.958	32.97	0.913	31.90	0.896	30.77	0.914	37.16	0.974
MemNet [48]	2	37.78	0.959	33.28	0.914	32.08	0.8978	31.31	0.919	37.72	0.974
EDSR [26]	2	38.11	0.960	33.92	0.919	32.32	0.901	32.93	0.935	39.33	0.977
D-DBPN [31]	2	38.09	0.960	33.85	0.919	32.27	0.900	33.02	0.931	39.32	0.978
MSRN [23]	2	38.08	0.961	33.74	0.917	32.23	0.901	32.22	0.933	38.82	**0.987**
MDSR [27]	2	38.11	*0.962*	33.85	0.918	32.29	0.900	32.84	0.934	38.96	0.976
SRFBN+ [37]	2	38.18	0.961	33.90	0.920	32.34	0.902	32.80	0.934	39.28	*0.979*
RDN+ [28]	2	*38.30*	0.961	*34.10*	*0.922*	32.40	*0.902*	33.09	0.937	*39.38*	0.978
RCAN+ [22]	2	**38.33**	*0.962*	**34.23**	**0.923**	**32.46**	**0.903**	**33.54**	**0.940**	**39.61**	0.978
Ours	2	**38.33**	**0.963**	34.05	0.919	*32.44*	0.901	*33.12*	*0.938*	39.33	0.976
Bicubic	4	28.42	0.810	36.10	0.704	25.96	0.669	23.15	0.659	24.92	0.789
A+ [17]	4	30.30	0.859	27.43	0.752	26.82	0.710	24.34	0.720	27.02	0.850
SRCNN [8]	4	30.49	0.862	27.61	0.754	26.91	0.712	24.53	0.724	27.66	0.858
FSRCNN [47]	4	30.71	0.865	27.70	0.756	26.97	0.714	24.61	0.727	27.89	0.859
MemNet [48]	4	31.74	0.889	28.26	0.772	27.40	0.728	25.50	0.763	29.42	0.894
EDSR [26]	4	32.46	0.897	28.80	0.788	27.71	0.742	26.64	0.803	31.02	0.915
D-DBPN [31]	4	32.47	0.898	28.82	0.786	27.72	0.740	*27.08*	0.795	*31.50*	0.914
MSRN [23]	4	32.07	0.890	28.60	0.775	27.52	0.727	26.04	0.789	30.17	0.903
MDSR [27]	4	32.50	0.897	28.72	0.785	27.72	0.741	26.67	0.804	31.11	0.914
SRFBN+ [37]	4	32.56	0.899	28.87	0.788	27.77	0.741	26.73	0.804	31.40	0.918
RDN+ [28]	4	32.61	*0.900*	*28.92*	0.789	*27.80*	*0.743*	26.82	*0.806*	31.39	*0.918*
RCAN+ [22]	4	**32.73**	**0.901**	**28.98**	**0.791**	**27.85**	**0.745**	**27.10**	**0.814**	**31.65**	**0.920**
Ours	4	*32.63*	0.898	28.90	*0.790*	27.77	0.741	26.82	0.805	31.44	0.915
Bicubic	8	24.39	0.657	23.19	0.568	23.67	0.547	20.74	0.516	21.47	0.647
A+ [17]	8	25.52	0.692	23.98	0.597	24.20	0.568	21.37	0.545	22.39	0.680
SRCNN+ [8]	8	25.33	0.689	23.85	0.593	24.13	0.565	21.29	0.543	22.37	0.682
FSRCNN+ [47]	8	25.41	0.682	23.93	0.592	24.21	0.567	21.32	0.537	22.39	0.672
VDSR [10]	8	25.72	0.711	24.21	0.609	24.37	0.576	21.54	0.560	22.83	0.707
LapSRN [11]	8	26.14	0.738	24.44	0.623	24.54	0.586	21.81	0.582	23.39	0.735
EDSR+ [26]	8	26.97	0.775	24.94	0.640	24.80	0.596	22.47	0.620	24.58	0.778
D-DBPN [31]	8	27.21	0.784	25.13	0.648	24.88	0.601	**23.25**	*0.622*	*25.50*	*0.799*
MSRN [23]	8	26.59	0.725	24.88	0.596	24.70	0.541	22.37	0.597	24.28	0.751
MDSR [27]	8	27.23	0.778	25.10	0.644	24.86	0.599	22.54	0.620	24.66	0.780
RDN+ [28]	8	27.33	*0.788*	25.18	0.649	24.90	0.603	22.74	*0.622*	24.87	0.783
RCAN+ [22]	8	**27.47**	**0.791**	**25.40**	**0.655**	**25.05**	**0.607**	*23.22*	**0.652**	**25.58**	**0.809**
Ours	8	*27.34*	0.787	*25.20*	*0.651*	*24.93*	*0.605*	22.75	0.621	24.88	0.786

Bold indicates the best and italic indicates the second-best performance

truth, while RDN and MDSR's outputs' colors are subjectively closer to the ground truth. For the boy image, our model shows similar reconstruction pattern with the ground truth, while others tend to reconstruct some other regular pattern. Additionally, we also can easily conclude that RCAN+ obtains more high-level information and can reconstruct pictures with better visual effects. Compared with that, our model although has similar PSNR and SSIM results, the out-

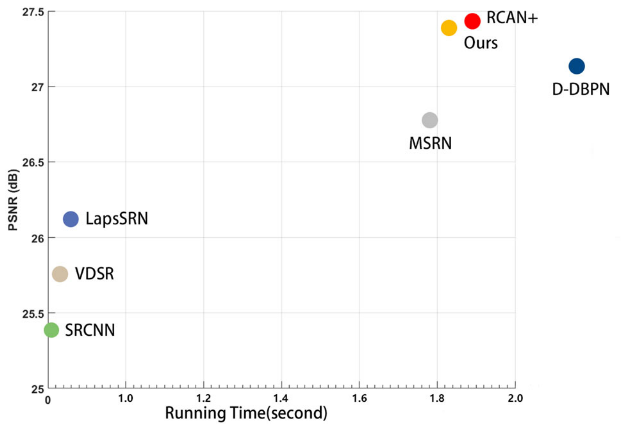

Fig. 13 Speed and accuracy trade-off. The average PSNR and the average inference time for upscaling 8× on Set5

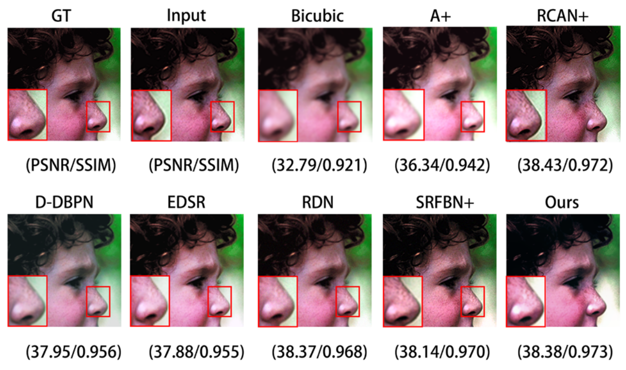

Fig. 14 Qualitative comparison of our models with other state-of-the-art works on 2× super-resolution task

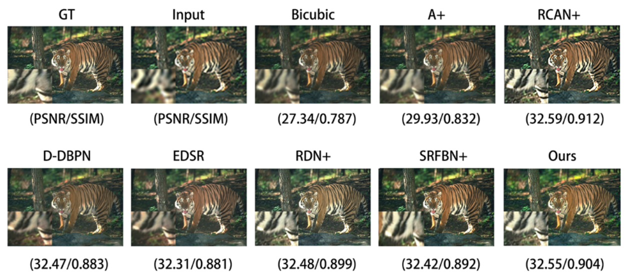

Fig. 15 Qualitative comparison of our models with other state-of-the-art works on 4× super-resolution task

puts are too smooth and lack of sharp boarders. The results of 8× enlargement are visually shown in Fig. 16, and our network outperforms all of the existing methods. In the first line, the image is randomly selected from BSD100 dataset, and the second line comes from urban100 dataset. It shows that either on natural images or urban pictures, our model is able to perform the best reconstruction effects. Also, it shows that our model is good at extracting features and can make good use of LR features with different scaling factor to generate HR components in the case of large scaling factors (e.g., 8× enlargement).

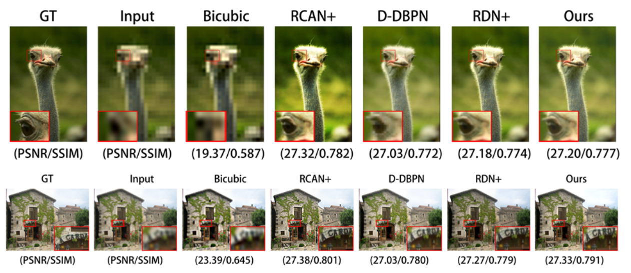

Fig. 16 Qualitative comparison of our models with other state-of-the-art works on 8× super-resolution task

5 Conclusions

We have proposed new generative adversarial networks using back-projection and progressive growing methodologies for single image super-resolution. Unlike the previous back-projection image SR network (DBPN) which uses the same scaling factor in every up- and downsampling stage. We build the projection units with progressive scaling factors, with the purpose of making good use of differently scaled features. Besides, we use multiple up- and down-projection units to combine into one projection block, and dense connection happens in every projection unit and block. And that structure helps us to learn richer features and improve model's efficiency. Other than that, we adopt progressive growing methodology to train our model, which stabilizes and accelerates the training progress. The results show the effectiveness of the proposed network compared to other state-of-the-art methods. Even on large scaling factors such as 8× enlargement, our model can achieve the best performance compared to other existing approaches. However, some recent researches have shown even better results than ours (e.g., RCAN+), and in our future work, we will take these methods as our goal and strive to catch up with them.

Acknowledgements This work was supported partially by National Key Research and Development Program (ID 2018AAA003203)

Compliance with ethical standards

Conflict of interest Author Tingsong Ma declares that he has no conflict of interest. Author Wenhong Tian declares that he has no conflict of interest.

References

1. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceeding s of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for imagerecognition. arXiv preprint arXiv:1512.03385 (2015)
3. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a Laplacian pyramid of adversarial networks. In:

Proceedings of the 28th International Conference on Neural Information Processing Systems, pp. 1486–1494 (2015)
4. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from simuilated and unsupervised images through adversarial training. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
5. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
6. Ulg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: flownet. Evolution of optical flow estimation with deep networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
7. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision, pp. 694–711. Springer (2016)
8. Dong, C., Loy, C.C., He, K., Tang, X.: Image super resolution using deep convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell. **38**(2), 295–307 (2016)
9. Haris, M., Widyanto, M.R., Nobuhara, H.: Inception learning super-resolution. Appl. Opt. **56**(22), 6043–6048 (2017)
10. Kim, J., KwonLee, J., MuLee, K.: Accurate image super resolution using very deep convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1646–1654 (2016)
11. Lai, W.-S., Huang, J.-B., Ahuja, N., Yang, M.-H.: Deep Laplacian pyramid networks for fast and accurate superresolution. In: CVPR (2017)
12. Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: CVPR (2017)
13. Shi, W., Caballero, J., Ledig, C., Zhuang, X., Bai, W., Bhatia, K., de Marvao, A.M.S.M., Dawes, T., ORegan, D., Rueckert, D.: Cardiac image super-resolution with global correspondence using multi-atlas patchmatch. In: MICCAI (2013)
14. Zou, W.W., Yuen, P.C.: Very low resolution face recognition problem. In: TIP (2012)
15. Zhang, L., Wu, X.: An edge-guided image interpolation algorithm via directional filtering and data fusion. In: TIP (2006)
16. Zhang, K., Gao, X., Tao, D., Li, X.: Single image super resolution with non-local means and steering kernel regression. In: TIP (2012)
17. Timofte, R., De Smet, V., Van Gool, L.: A+: adjusted anchored neighborhood regression for fast super-resolution. In: Asian Conference on Computer Vision, pp. 111–126. Springer (2014)
18. Timofte, R., De, V., Gool, L.V.: Anchored neighborhood regression for fast example-based super-resolution. In: ICCV (2013)
19. Peleg, T., Elad, M.: A statistical prediction model based on sparse representations for single image super-resolution. In: TIP (2014)
20. Tong, T., Li, G., Liu, X., Gao, Q.: Image super-resolution using dense skip connections. In: ICCV (2017)
21. Zhang, K., Zuo, W., Zhang, L.: Learning a single convolutional super-resolution network for multiple degradations. In: CVPR (2018)
22. Zhang Y., Li K.: Image super-resolution using very deep residual channel attention network. In: European Conference on Computer Vision (2018)
23. Li, J., Fang, F.: Multi-scale residual network for image super-resolution. In: European Conference on Computer Vision, pp. 527–542 (2018)
24. Ledig, C., Theis, L., Huszar, F., et al.: Photo-realistic single image super-resolution using a generative adversarial network (2016). arXiv:1609.04802v5
25. Kim, J., Lee, J.K., Lee, K.M.: Deeply-recursive convolutional network for image super-resolution (2015). arXiv:1511.04491v2
26. Lim, B., Son, S., Kim, H., Nah, S., Lee, K. M.: Enhanced deep residual networks for single image super-resolution. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
27. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: CVPRW (2017)
28. Zhang, Y., Tian, Y., Kong, Y., et al.: Residual dense network for image super-resolution (2018)
29. Fellemanand, D.J., VanEssen, D.C.: Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex **1**(1), 1–47 (1991)
30. Kravitz, D.J., Saleem, K.S., Baker, C.I., Ungerleider, L.G., Mishkin, M.: The ventral visual pathway: an expanded neural framework for the processing of object quality. Trends Cognit. Sci. **17**(1), 26–49 (2013)
31. Haris, M., Shakhnarovich, G., Ukita, N.: Deep back-projection networks for super-resolution (2018). arXiv:1803.02735
32. Karras, T., Aila, T., Laine, S., et al.: Progressive growing of GANs for improved quality, stability, and variation (2017). arXiv:1710.10196
33. Li, D., Wang, Z.: Face video super-resolution with identity guided generative adversarial networks. In: Chinese Conference on Computer Vision (CCCV), pp. 357–369 (2017)
34. Wang, X., Yu, K., Wu, S., et al.: ESRGAN: enhanced super-resolution generative adversarial networks. In: European Conference on Computer Vision (ECCV), pp. 63–79 (2018)
35. Zhang, D., Jie, S., Gang, H., et al.: Sharp and real image super-resolution using generative adversarial network. In: International Conference on Neural Information Processing, pp. 217–226 (2017)
36. Irani, M., Peleg, S.: Improving resolution by image registration. CVGIP Gr. Models Image Process. **53**(3), 231–239 (1991)
37. Li Z., Yang J.: Feedback Network for Image Super-Resolution. arXiv:1903.09814 (2019)
38. Gulrajani, I., Ahmed, F., Arjovsky, M., et al.: Improved training of wasserstein GANs (2017). arXiv:1704.00028
39. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers surpassing human-level performance on imagenet classification In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
40. Timofte, R., Agustsson, E., Van Gool, L., Yang, M.-H., Zhang, L., Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M., et al.: Ntire 2017 challenge on single image super-resolution: methods and results. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1110–1121. IEEE (2017)
41. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
42. Bevilacqua, M., Roumy, A., Guillemot, C., Morel, M.L.A.: Low-complexity single-image super-resolution based on no nnegative neighbor embedding. In: British Machine Vision Conference (BMVC) (2012)
43. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International Conference on Curves and Surfaces, pp. 711–730 (2010)
44. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 898–916 (2011)
45. Huang, J.B., Singh, A., Ahuja, N.: Single image super resolution from transformed self-exemplars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5197-5206 (2015)
46. Matsui, Y., Ito, K., Aramaki, Y., Fujimoto, A., Ogawa, T., Yamasaki, T., Aizawa, K.: Sketch-based manga retrieval using manga109 dataset. Multimed. Tools Appl. **76**, 1–28 (2016)

47. Dong, C., Loy, C.C., Tang, X.: Accelerating the super resolution convolutional neural network. In: European Conference on Computer Vision, pp. 391–407. Springer (2016)
48. Tai, Y., Yang, J., Liu, X., Xu, C.: Memnet: a persistent memory network for image restoration. In: ICCV (2017)
49. Irani, M., Peleg, S.: Motion analysis for image enhancement: resolution, occlusion, and transparency. J. Vis. Commun. Image Represent. **4**(4), 324–335 (1993)
50. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)

Tingsong Ma is currently working toward the Ph.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 and object detection.

Wenhong Tian Ph.D. graduated from North Carolina State University, professor of University of Electronic Science and Technology of China, Ph.D. supervisor, director of Intelligent Information Technology and Application Laboratory, is a member of the Applied Technology Expert Committee of the Chinese Computer Society (CCF) and senior member of CCF.