RECONSTRUCTED DENSENETS FOR IMAGE SUPER-RESOLUTION

Lingfeng Wang, Linwei Qiu, Wei Sui, and Chunhong Pan

NLPR, Institute of Automation, Chinese Academy of Sciences
{lfwang,wsui,chpan}@nlpr.ia.ac.cn and qiulinwei@buaa.edu.cn

ABSTRACT

Deep learning has been successfully applied to single image super-resolution problem due to its high data fitting ability. However, the trending of deeper layers and wider receptive field to acquire better performance brings high computation complexity and serious information vanishing. To address this problem, we proposed a new *Reconstructed DenseNets* model for super-resolution. The basic idea behind *Reconstructed DenseNets* is to improve the recent *DenseNets* model by modifying the two core modules, *dense blocks* and *transition blocks*, so that the *Reconstructed DenseNets* can emphasize the quality of data reconstruction. Specifically, on the one hand, the batch normalization layers in *dense blocks* is ignored to overcome the data shift risk. One the other hand, the pooling layers in *transition blocks* is also ignored to ensure the ability to reconstruct. Based on the above two improvements, the new *DenseNets* is named as *Reconstructed DenseNets*. Extensive experiments evaluate the effectiveness of our model, showing the outperforming of the state-of-the-art approaches.

Index Terms— Super-Resolution, Deep Learning, DenseNets, Residual Learning

1. INTRODUCTION

Super-resolution (SR) is an important task for various computer vision applications, such as, object detection and recognition, video compression and communication, and has wide applications, such as HDTV and satellite imaging, and medical imaging. SR is an inherently ambiguous and highly ill-posed problem for the reason that multiple HR image patches could correspond to the same LR image patches.

Numerous methods have been applied to this problem trying to overcome this difficulty to some extent, such as reconstruction methods [1, 2, 3], learning methods [4, 5, 6, 7]. Recently, most of them fall into the example based learning framework, which try to learn an explicit or implicit prior mapping from LR and HR patches. Representative methods consist of neighbor embedding [8, 9], random forest [10, 11], and the famous deep convolutional neural network (CNN).

This work is supported by the National Natural Science Foundation of China (Grant No. 61773377 and 91646207).

To the best of our knowledge, Dong *et al.*[12] first proposed a CNN based SR method, called SRCNN, to directly learn an end-to-end mapping. Motivated by it, the VDSR [13] first introduced a very deep residual network by increasing the depth from 3 to 20 layers. As reported in [13], the SR performance increases as the depth adds in most cases. However, this trending is crude in considering computational cost and memory utilization. Furthermore, it also occurs that the information about the input and gradient may vanish when it passes through deep layers. Especially for the ill-posed SR problem, the information of the input images may hardly be reconstructed, if it vanishes through the former layers.

Recently, there are many networks in the fields of computer vision that deal with different computer vision problems, such as classification and detection. For example, recently proposed network named *DenseNets* get important improvements over the state-of-the-art on most of the highly competitive object recognition benchmark tasks, whilst shorting connections between layers close to the input and the output [14]. *DenseNets* combine the observation that convolutional networks may be substantially deeper, more accurate, and easy to train if they contain shorter connections between layers [14]. In other word, each layer of this certain special network receives additional information from all preceding layers as inputs. The above characteristic makes *DenseNets* can be very deep. Unfortunately, it is very difficult to directly apply the *DenseNets* to the SR problem.

Based on the above two observations, we propose a novel model derived from *DenseNets* and encourage their strength to solve the highly ill-posed SR problem. We modify the two basic blocks of *DenseNets*, called *dense blocks* and *transition block*, so that the improved *DenseNets* can be the improved networks which may be appropriately applied to the reconstruction problem. Thereby, the improved *DenseNets* is named as *Reconstructed DenseNets*. Besides, we combine some tricks of residual learning [15] and sub-pixel layers [16] with our *Reconstructed DenseNets* to reduce computational complexity. Furthermore, we train the networks with L_1 loss function rather than L_2 loss function for the scales of 2, 3, 4, respectively. We evaluate our model on the standard benchmark datasets, and the results show that it gains state-of-the-art performances on all datasets in terms of peak signal-to-noise ratio (PSNR) and structural similarity (SSIM).

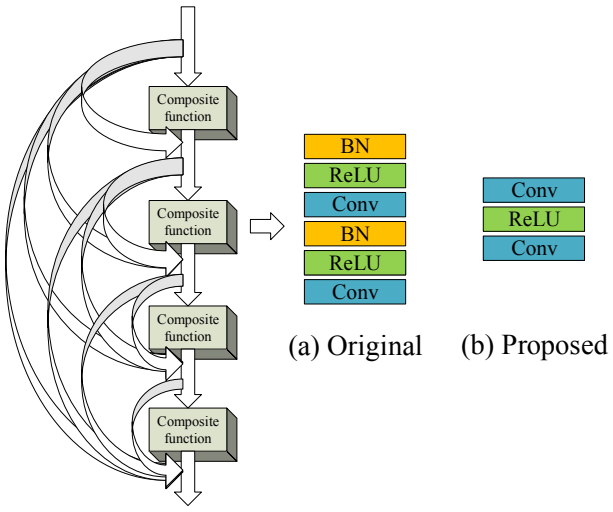

Fig. 1. Reconstructed DenseNets.

Specifically, the main contributions and details of this paper are highlighted in three aspects as follows:

1. The *DenseNets* is first introduced to make very deep network can be used in SR problem reasonably.

2. To ensure the reconstruction ability of *DenseNets*, the *dense blocks* and *transition blocks* are modified by ignoring batch normalization and pooling layers. As a result, the new *Reconstructed DenseNets* is suitable to SR problem.

3. To reduce the computational complexity, the *Reconstructed DenseNets* based SR model is further improved by introducing residual learning and sub-pixel layer.

2. RECONSTRUCTED DENSENETS MODEL FOR SR

2.1. Reconstructed DenseNets

The core of *DenseNets* is dense connectivity, which refers that the l^{th} layer uses the information of all preceding layers. That is, denoting by $X_0, \ldots, X_{l-1}$, as input, we get that

$$X_l = f_l([X_0, X_1, \ldots, X_{l-1}]), \tag{1}$$

where $[X_0, X_1, \ldots, X_{l-1}]$ denotes the concatenation of the results produced in the layers $0, \ldots, l-1$. *DenseNets* consist of *dense blocks* and *transition blocks*, which are two neighboring blocks [14]. In this paper, we improve the structure of two blocks, so that the new *Reconstructed DenseNets* can be well used of in the field of SR.

Reconstructed Dense Blocks. In Fig. 1, we compare the *dense block* from *DenseNets* to *Reconstructed DenseNets*. In traditional DenseNets, $f_l(*)$ in the Eqn. (1) is defined as a composite function of four consecutive operations: bottleneck layer (the first three layers of original in Fig. 1), followed by a batch normalization (**BN**), a rectified linear unit (**ReLU**) layer and a 1×1 convolution (**Conv**). We remove the first batch normalization layer and its following rectified linear unit (**ReLU**) and the fourth batch normalization layer. The main reason is that batch normalization layers normalize the features but get rid of the range flexibility from networks reported in [17]. The depth of our dense blocks is 4, which is the same as *DenseNets* shown in Fig. 1.

Reconstructed Transition Blocks. Transition blocks are the composes of two special layers, such as convolution and pooling [14]. A normal transition block is composed of a batch normalization layer and an 1×1 convolutional layer followed by a 2×2 average pooling layer [14]. A pooling layer may loss some information, which is of great importance in SR. Thus, we only put an 1×1 convolutional layer in our model. We preserve this special part for compression purpose which will be discussed later.

Growth Rate. It is obvious to know that the l^{th} layer has $k \times (l-1) + k_0$ input feature-maps if k feature-maps are produced as output by each composite function, where k_0 is the number of channels in the input image. The parameter k is referred to the growth rate of the network [14]. It is significant factor to prevent the network from growing too widely and to improve the parameter efficiency. However, it is well-known that increasing the feature-maps can be helpful to enhance model performance. The value of k is set as 48 on the consideration of the aspects discussed above.

Bottleneck Layers and Compression. As shown in the original sub-figure, to reduce the number of input feature-maps, the original structures introduce a 1×1 convolution as *bottleneck layer* before each 3×3 convolution. In our model, we do not have to introduce this layers. Hence, we change the kernel size of the *bottleneck layer* from 1×1 to 3×3. At transition layers, compression is also needed to improve model compactness. If a dense block contains m feature-maps, the following transition layer will generate $\theta \times m$ output feature-maps [14]. We set $\theta = 0.5$ in our experiment.

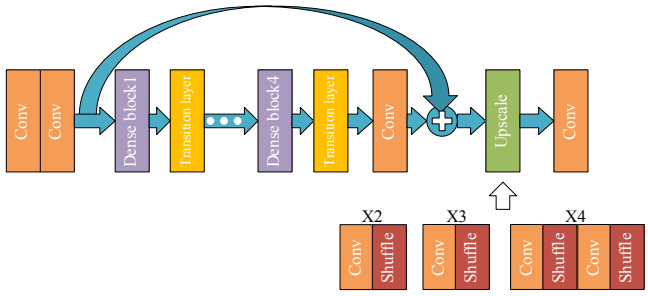

Fig. 2. Reconstructed DenseNets Model (DenseSR).

Dataset	Scale	Bicubic	A+	SRCNN	VDSR	DenseSR(ours)
Set 5	×2	33.66/0.9299	36.54/0.9544	36.66/0.9542	37.53/0.9587	38.08/0.9602
	×3	30.39/0.8682	32.58/0.9088	32.75/0.9090	33.66/0.9213	34.53/0.9271
	×4	28.43/0.8104	30.28/0.8603	30.48/0.8628	31.35/0.8838	32.25/0.8939
Set 14	×2	30.24/0.8688	32.28/0.9056	32.42/0.9063	33.03/0.9124	33.70/0.9182
	×3	27.55/0.7742	29.13/0.8188	29.28/0.8209	29.77/0.8314	30.37/0.8432
	×4	26.00/0.7027	27.32/0.7491	27.49/0.7503	28.01/0.7674	28.58/0.7816
B100	×2	29.56/0.8431	31.21/0.8863	31.36/0.8879	31.90/0.8960	32.27/0.9005
	×3	27.21/0.7385	28.29/0.7835	28.41/0.7863	28.82/0.7976	29.18/0.8071
	×4	25.96/0.6675	26.82/0.7087	26.90/0.7101	27.29/0.7251	27.62/0.7376
DIV2K validation	×2	31.01/0.9393	32.89/0.9570	33.05/0.9581	33.66/0.9625	34.89/0.9687
	×3	28.22/0.8906	29.50/0.9116	29.64/0.9138	30.09/0.9208	31.12/0.9321
	×4	26.66/0.8521	27.70/0.8736	27.78/0.8753	28.17/0.8841	29.04/0.8978

Table 1. Average **PSNR/SSIM** for scale factor ×2,×3,and ×4 on datasets *S*et5, *S*et14, *B*100 and *DIV2K validation*. Red color indicates the best performance and blue color indicates the second best performance.

2.2. DenseSR Model

Our DenseSR model with our reconstructed dense blocks and reconstructed transition layers is shown in Fig. 2. We also combine residual learning and sub-pixel layers.

Feature Extraction. In our model, the first two convolutional layers play the role of feature extraction. If the number of *feature channels* (the width of a CNN architecture) is bigger, the model performance will be better. However, it leads to heavier computation time and memory at the same time. We set the number of *feature channels* as 64 for less computation and memory in our experiment.

Residual Learning. Since our network is deep, gradient exploding and computational complexity are mitigated by learning residuals only, which proves a traditional but efficient approach used in deep networks, such as VDSR [13]. Residual learning [15] skill adds a skip connection that passes the non-linear transformations with an identity function:

$$X_l = f_l(X_{l-1}) + X_{l-1}. \tag{2}$$

Upscale Layer. To reduce computational complexity and achieve real-time performance, the ESPCN network [16] is studied to extract feature maps in the LR space, which uses an efficient sub-pixel convolution. We use this sub-pixel convolution layer that learns an array of upscaling filters to upscale the final LR feature maps into HR output so that computation can be alleviated as much as possible. When the scale is from 2 to 4, we train the different upsample layers accordingly which are showed in Fig. 2. The upscale layer is a function which can be presented as follows:

$$I^{SR} = f^L(I^{LR}) = \mathit{RE}(W_L * f^{L-1}(I^{LR}) + b_L), \tag{3}$$

where $f^{L-1}(I^{LR})$ means the result of the layer $L-1$, which is the layer before the upscale layer. $W_L * f^{L-1}(I^{LR}) + b_L$ represents the function of the layer in front of the *shuffle layer*, where W_L is the weight of the convolution, and b_L is the bias of the convolution. **RE** is an periodic shuffling operator that rearranges the elements of a $H \times W \times C \times r^2$ tensor into a tensor of shape $rH \times rW \times C$, in which H is the height and W is the width of the LR image, and r is of course the scale which varies from 2 to 4. Both the LR image I^{LR} and the HR image I^{HR} can have C color channels.

3. EXPERIMENTS

In this section, we evaluate the performance of our model on several benchmark datasets. First, we introduce our training and testing datasets. Next, we describe our training details briefly. Eventually, we evaluate our model and compare our proposed model with several state-of-the-art methods.

3.1. Datasets

We use the 800 training images of DIV2K dataset [18] to train our model, which is a newly proposed high-quality (2K resolution) image dataset for image restoration tasks. We compare the performance on the DIV2K validation dataset and three standard benchmark datasets: Set5 [19], Set14 [20] and B100 [21]. The LR images are acquired by bicubic interpolation first. Then they will be up-sampled to desired size to generate LR-HR image pairs for both training and testing. Zero padding and 1×1 stride are applied in all convolutional layers to keep the size of output maps as the same as input.

3.2. Training Details

The size of RGB input patches and target patches is 48×48. And we train our model with ADAM optimizer [22] with the settings of $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The batch size is 16 and the learning rate is initialized as 10^{-4} which will be halved at every 200 epochs.

Almost all of the CNN-based SR methods optimize networks with an L_2 loss function or mean squared error (MSE).

Fig. 3. SR results of "108005" (in B100) with scale factor $\times 3$. DenseSR recovers the sharp lines on the body of tiger.

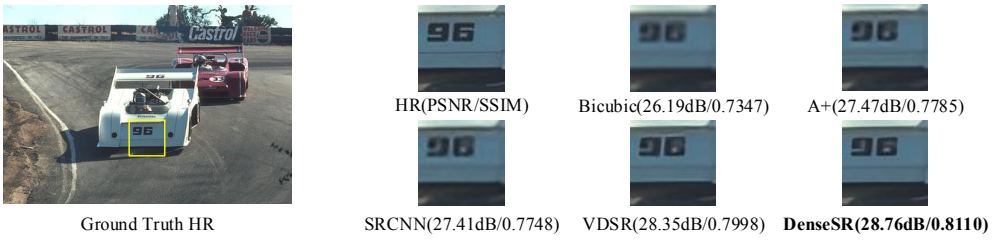

Fig. 4. SR results of "21077" (in B100) with scale factor $\times 3$. The number on the automobile is clear enough to recognize.

However, [23] demonstrated that the L_2 loss is less effective for learning and predicting sparse residuals. Zhao *et al* [24] trained the network with L_1 loss and achieved improved performance compared with the network trained with L_2. We use L_1 loss instead of L_2 for the simple reason that L_1 loss has better convergence than L_2. L_1 loss function is like: $L_1 = \frac{1}{n}\sum_{i=1}^{n} \|F_{x_i}(\theta) - y_i\|$, where F_{x_i} is the reconstructed HR image from LR image x_i, y_i is the ground truth image and θ denotes all parameters for training. Torch 7 framework is implemented for our networks. It takes no more than two days to train the *DenseSR* model using GTX 980TiGPU.

3.3. Geometric Self-ensemble

Self-ensemble strategy is a method to maximize the performance of the SR model [24]. We take use of it to enhance our model performance. This has the advantage that it does not need train and just operate on the images. Although it does not change the parameters of the model, it gains slight enhance on the PSNR and SSIM.

The flow of the method is shown as follows:

STEP_1: Using the input images I^{LR} to generate 7 other images $I_i^{LR} = T_i(I^{LR})$ with 7 different geometric transformations, such as horizontal flips and 90 rotations;

STEP_2: Generating corresponding eight super-resolved images $\{I_1^{SR}, I_2^{SR}, \ldots, I_8^{SR}\}$, including the original input image and 7 different geometric transformed images;

STEP_3: Applying the corresponding inverse transform to the 7 output images $\tilde{I}_i^{SR} = T_i^{-1}(I_i^{SR})$;

STEP_4: Averaging the 8 images together by the following equation $I^{SR} = \frac{1}{8}\sum_{i=1}^{8} \tilde{I}_i^{SR}$.

3.4. Comparisons with the state-of-the-arts

The quantitative evaluation results of PSNR and SSIM on several datasets are available in Table 1. We compare our models with the state-of-the-art methods such as A+ [9], SRCNN [12] and VDSR[13]. Our models outperform the previous methods in these datasets.

In Fig. 3 and 4, we compare our methods with previous methods qualitatively. Our models can recover the sharp lines on the body of the tiger in Fig. 3 and reconstruct the number on the automobile in Fig. 4. It is obvious that our results are both sharper and clearer than other results. Our proposed one can reconstruct the details of HR images because our model structures may reserve the information and avoid inputs and gradients information vanishing to some extent.

4. CONCLUSIONS

In this paper, we proposed a new model, called *Reconstructed DenseNets* for SR problem. We first introduce and improve the *DenseNets*, which attain significant improvements over the state-of-the-art in the fields of object recognition. And then, we modify the two basic parts of *DenseNets* and get the reconstructed *dense blocks* and *transition blocks*. Based on the reconstructed dense and transition blocks, a novel model is proposed with the thought of residual learning and sub-pixel layer. Experiment results show that our models perform favorably against the state-of-the-art approaches.

5. REFERENCES

[1] Shengyang Dai, Mei Han, Wei Xu, Ying Wu, and Yihong Gong, "Soft edge smoothness prior for alpha channel super resolution," in *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, 2007.

[2] Hussein H. Aly and Eric Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Trans. Image Processing*, vol. 14, no. 10, pp. 1647–1659, 2005.

[3] Qi Shan, Zhaorong Li, Jiaya Jia, and Chi-Keung Tang, "Fast image/video upsampling," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 153:1–153:7, 2008.

[4] Haijun Wang, Xinbo Gao, Kaibing Zhang, and Jie Li, "Single-image super-resolution using active-sampling gaussian process regression," *IEEE Trans. Image Processing*, vol. 25, no. 2, pp. 935–948, 2016.

[5] Kaibing Zhang, Xinbo Gao, Dacheng Tao, and Xuelong Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE Trans. Image Processing*, vol. 21, no. 11, pp. 4544–4556, 2012.

[6] William T. Freeman, Thouis R. Jones, and Egon C. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[7] Daniel Glasner, Shai Bagon, and Michal Irani, "Super-resolution from a single image," in *IEEE 12th International Conference on Computer Vision, ICCV*, 2009, pp. 349–356.

[8] Hong Chang, Dit-Yan Yeung, and Yimin Xiong, "Super-resolution through neighbor embedding," in *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, 2004, pp. 275–282.

[9] Radu Timofte, Vincent De Smet, and Luc J. Van Gool, "A+: adjusted anchored neighborhood regression for fast super-resolution," in *Computer Vision - ACCV 2014 - 12th Asian Conference on Computer Vision*, 2014, pp. 111–126.

[10] Samuel Schulter, Christian Leistner, and Horst Bischof, "Fast and accurate image upscaling with super-resolution forests," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2015, pp. 3791–3799.

[11] Jordi Salvador and Eduardo Perez-Pellitero, "Naive bayes super-resolution forest," in *2015 IEEE International Conference on Computer Vision, ICCV*, 2015, pp. 325–333.

[12] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision - ECCV 2014 - 13th European Conference*, 2014, pp. 184–199.

[13] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 1646–1654.

[14] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017, pp. 2261–2269.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 770–778.

[16] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 1874–1883.

[17] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, 2017, pp. 1132–1140.

[18] Radu Timofte and Eirikur Agustsson, "NTIRE 2017 challenge on single image super-resolution: Methods and results," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, 2017, pp. 1110–1121.

[19] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *British Machine Vision Conference, BMVC*, 2012, pp. 1–10.

[20] Roman Zeyde, Michael Elad, and Matan Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces - 7th International Conference*, 2010, pp. 711–730.

[21] David R. Martin, Charless C. Fowlkes, Doron Tal, and Jitendra Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001, pp. 416–425.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[23] W. Lai, J. Huang, N. Ahuja, and M. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 00, pp. 5835–5843.

[24] Radu Timofte, Rasmus Rothe, and Luc Van Gool, "Seven ways to improve example-based single image super resolution," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 1865–1873.