PEIQIN LI

1 Introduction:

Deep convolutional neural networks have led to a series of breakthroughs for single image super resolution (SISR). Since Dong et al. (2014) used a 3 layers convolutional neural network SRCNN [3] to achieve a relatively good resolution performance, how to make use of CNN to achieve better super resolution performance has been a hot topic.

One generally agreement is that deeper neural structure well provide better result [4] [6] [7]. And with the help of residual network [1] [4], the problem of degradation arises form deep neural structures can be eased, which make very deep neural network possible to train. It has been the mainstream to use deep residual network for SISR [5] [6] [8] [9] [11] [12].

In this report, I introduction a multiblock neural network which is trained by multi-step procedure. This algorithm is inspired by one of the state-of-art structure algorithm RCAN [11] and try to provide an alternative and effective way for SISR task. The aim of this report is trying to demonstrate the potential power of multiblock structure and making the network of image super resolution even deeper as well as trainable.

2 Related work:

2.1 dataset and RCAN structure

Most of the existing SISR algorithms are trained and evaluated on simulated datasets, where the low-resolution images are generated by applying a simple down sampling and blurring degradation (i.e., bicubic down sampling and simulated noise) to their high-resolution (HR) counterparts [5]. However, the problem we are facing in real world is more complicated. In this report, instead using simulated dataset, I use the real world paired low-resolution (LR) and High-resolution (HR) image by Cai et al. [5].

The network structure is inspired by "Residual Channel Attention Networks" (RCAN) [11] which is show below:

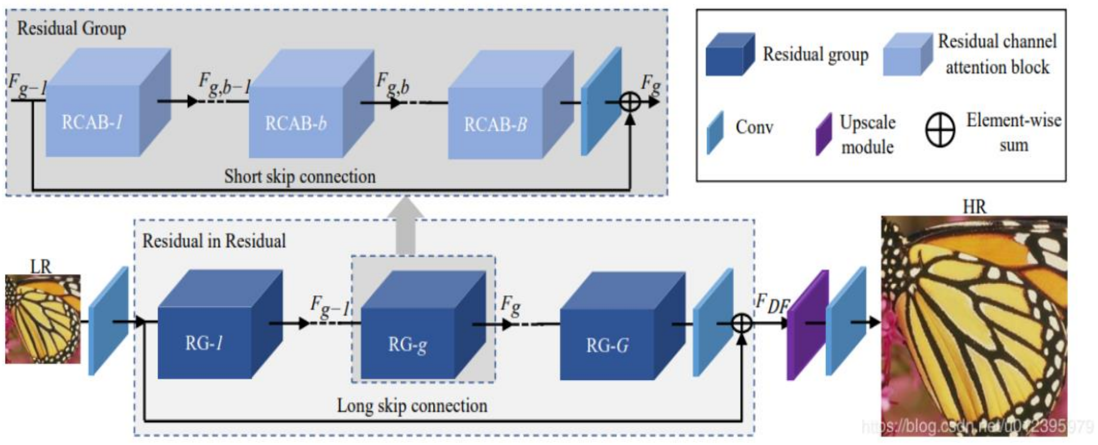


Fig1 the structure of RCAN [11]

The network consists of 10 residual groups (RG in the Fig 1), and 20 residual blocks for each RG (RCAB in Fig 1). For each residual block, there is a channel attention mechanism and the exact structure for RCAB is shown below:

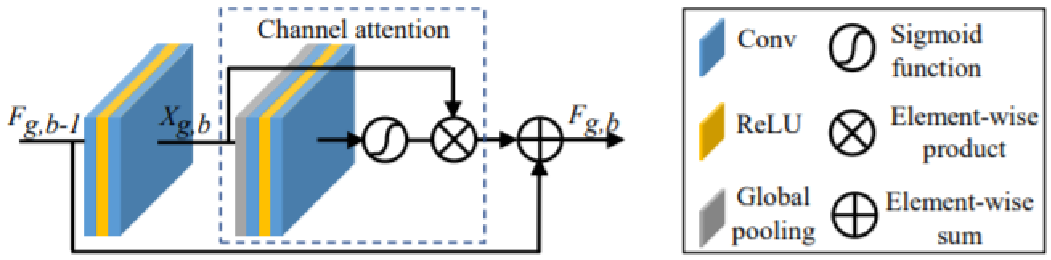


Fig2 the structure of RCAB [11]

In general, this is a very deep neural network has 400 layers depth and contain 200 channel attention layers which don't increase the depth.

2.2 receptive field

Receptive field is rarely discussed in super resolution field. This term represents the size of input image which has influence on output pixel.

For example, if the neural network is so shallow that it only contains one 3×3 convolution layer, then for each output pixel, there exist a 3×3 window in the input image which has influence on that pixel.

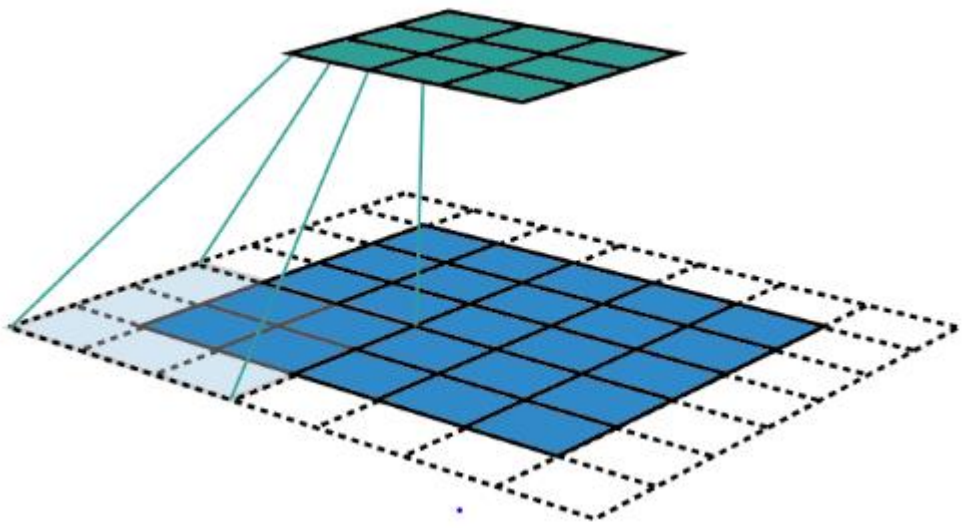


*Fig 3 the receptive field for single 3*3 layer [13]*

Similarly, if there are two 3×3 convolution layers, there exists a 5×5 window in the input image which has influence on that pixel.

Thus, if we build a 400-layer depth convolution network like RCAN, the receptive filed is near 1000×1000. Zero padding is a common way to control the size of input and output [5] [6] [8] [9] [11] [12], however a very deep neural network trained on very small patch is questionable (numerical test can be seen in section 5.1), and RCAN is trained on 48×48 patch due to the computation resource limit [11].

People put lots of focus on how to design a deep, complex and smart network for image resolution task on certain dataset. But there is less discussion about how to avoid local minimum as well as make the algorithm converge faster.

Multiblock Channel Attention Network is designed for dealing with the problem arises from receptive field and make the network easy to convergent.

3 Multiblock Channel Attention Network:

3.1 Multiblock network: an initialization method

To better solve the degradation of deep neural network, and also keep the complexity and depth of the total structure, one natural way is separate a deep neural network into several "not so deep neural network". Unlike other computer version test, such as image recognition whose input is image and output is label, and image style transformation, which doesn't have a clearly ground truth, SISR has its uniqueness that the input and output are of the same form and there exists a clear ground truth.

Thus basically for all residual network structure, by adding an additional n to 3 convolution layer, we can separate a deep network N into several not-so-deep network block $N_1, N_2, .., N_m$ where $N(input) = N_m(..\left(N_2\left(N_1(input)\right)\right))$. We train $N_1, N_2, .., N_m$ in an forward manner using backpropagation, and every time the input for N_i is the patch after $N_1, .., N_{i-1}$, that is $N_{i-1}(..\left(N_1\left(raw_{input}\right)\right))$.

Since each time the size of trainable parameters is $\frac{1}{n}$ of total parameters, it is easier for network to converge and can avoid degradation. At the end of the procedure, we can train the whole network by making all parameter trainable, just like what we usually do. This is the end of Multiblock procedure.

I want to use a toy model to illustrate my idea: There are two network blocks N_1, N_2 with the same structure, and a combined deeper network $N(input) = N_2(N_1(input))$. For each block, it contains 6 convolution layers with 3×3 kernel and a residual mechanism. The stride for convolution layers is 1 and zero padding strategy is applied. The Initialization method for all layers is the normal initializer with 0 mean and 0.1 standard deviation.

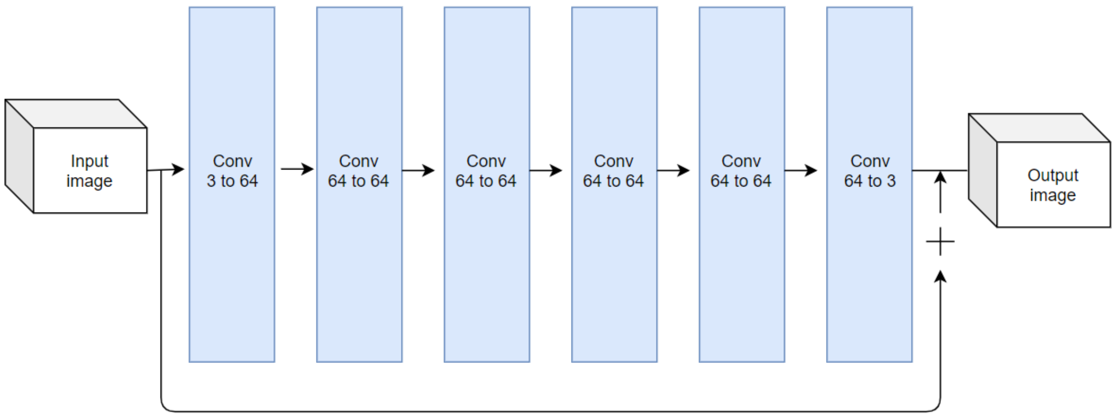


Fig 4 structure of N_i

I trained two blocks in the *cannon 2 scale* data set [5] which contains 71 image pairs, set the image patch size 61×61 and batch size 64. Also, I apply the random 90, 180 270 and horizontal flipping to the original dataset in order to enlarge the sample size. The validation part was complemented on the provided 15 test image pairs, and the test images' size is restricted less than 1500×1500. The optimization algorithm used here is ADAM [14] with β1 = 0.9, β2 = 0.999, and ε = 10−8. And L1 norm is used as the Loss function.

For the two blocks, I trained each of them using 1000 iterations with learning rate 0.001 in a forward manner and using iteration 500 iteration with learning rate 0.0001 to train the combined network (making all parameter trainable). For the deeper Network N, I used 1500 iterations with learning rate 0.001 in the first 1000 iterations and decay to 0.0001 in the last 500 iterations. The initialization method for all trainable convolution parameters is normal initializer with 0 mean and 0.1 standard deviation. In addition, L1 loss is applied for each training phase.

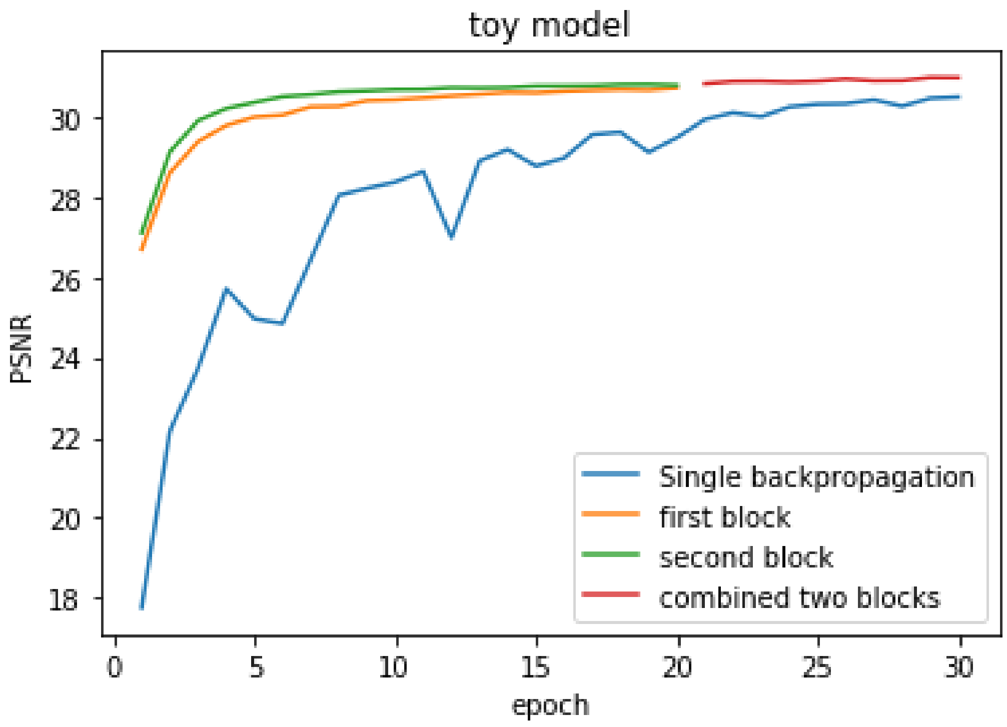


Fig 5, the result of toy model within 1500 iteration (50 iteration per epoch)

As we can see in Fig 5, even though the deeper (12 layers) network N has more complexity to general high-resolution images, where using the blue line to demonstrate, it cannot converge as fast as shallower network blocks N_1, N_2 in the first 20 epoch with learning rate 0.001, which is demonstrated by orange and green line separately. In addition, the block-combined training method can further improve the performance of multiblock structure, as demonstrated as red line. In the end, using single backpropagation to train a 12-layer network achieved 30.505 dB in 30 epochs while using multi-block structure can achieve 30.987 dB in 30 epochs, nearly 0.5 dB improvement. These two methods used exactly the same computational resource, each parameter will eventually go through 1500 backpropagations, but the result is quite different.

3.2 channel attention network and phased training

The toy model demonstrated doesn't solve the receptive field problem I mentioned before, the respect field still get large even we make part of the net structure untrainable. Thus, for deeper structure contain channel attention mechanism, I further introduce the phased training technique.

There I involve 3 basic neural net blocks: N_1, N_2, N_3, with the same structure as shown in Fig 6. After the single 2D convolution layer, I add 16 residual blocks inspired by RCAN [11]. Each residual block contains 2 convolution layers and an attention channel mechanism (CA). By using global pooling, we can get the channel-wise global spatial information into a 1D vector $X = [X_1, X_2, \ldots, X_{64}]$, and to fully learn the aggregated information from pooling, I build the double 1D convolution layers structure [15]: $\mathrm{Ca} = \mathrm{Sigmoid}(W_U\left(\mathrm{ReUL}(\,(W_D(\mathrm{X}))\right))$, Where W_D is a 1D convolution with input of dimension 64 and output of dimension 16 and W_U is a 1D convolution with input of dimension 16 and output of dimension 64. At the end we multiply the Ca with the whole channel to get the residual and add the input back as the result for single residual block. All the 2D convolution uses 3×3 kernel and zero padding strategy with stride size 1.

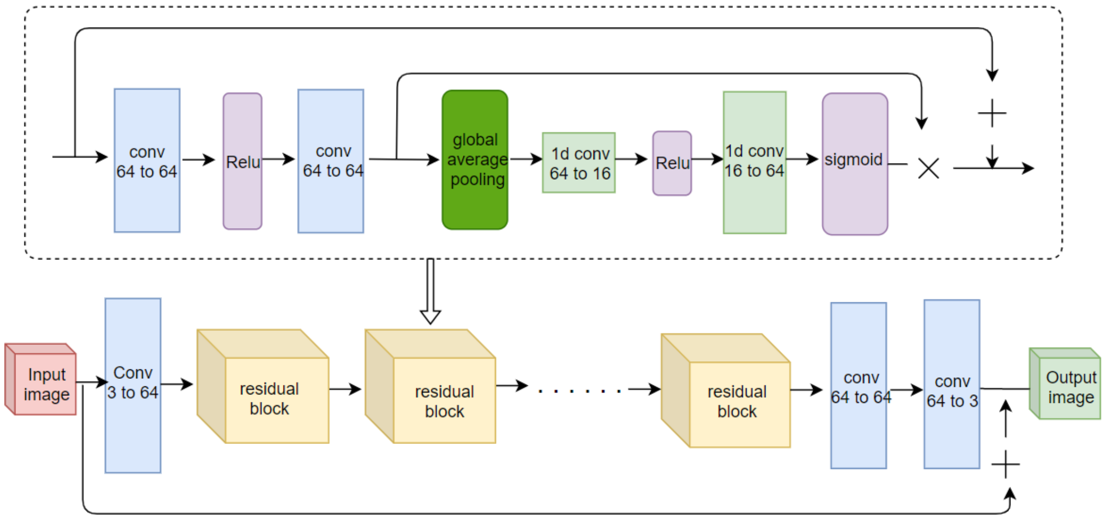


Fig 6 N_i neural block structure

The depth of each network block N_i is 35 where 32 1D convolution layers are not include. Thus, the size of receptive filed is 71×71 which is much less than the origin RCAN network (over 800×800). Moreover, I don't pass the input data directly into N_1, N_2, N_3 as what I did in the toy model, but separate the learning procedure into 4 different phases:

Phase1-1: using LR and HR images to train N_1

Phase1-2: using pretrained N_1 *and LR to general a new low-resolution data set LR_1*

Phase 2-1: using LR_1 and HR images to train N_2

Phase 2-2: using pretrained N_2 *and LR_1 to general a new low-resolution data set LR_2*

Phase 3: using LR_2 and HR images to train N_3

Phase 4 (combined training): initialize the total net work N *with pretrained network block* N_1, N_2, N_3 *and use backpropagation to further modify the parameters.*

It is shown that compared to simply apply single backpropagation procedure on the deep network N, the multiblock and multiphase training manner can achieve better result. The computation cost for this strategy is that we need additional backpropagation iterations for each network block N_1, N_2, N_3. However, the good news is that it may not necessary to put too much computation resource for each network block if we can involve the combined training procedure in phase 4. That is, we only need to learn roughly correct things for each block. For example, if we want to do 10^5 iterations for a depth network structure, we may do 10^4 additional iterations for each network blocks.

4 Implement detail:

4.1 Orthogonal initialization for the first block

I use the real image dataset from Cai [5], all the image pairs are taken from a Cannon camera (scale two dataset). There are 71 image pairs for training and 15 image pairs for testing. Different from what we used to do in the toy model demonstration, here I use orthogonal initializer [16] since N_1, N_2, N_3 all contain the residual mechanism where the input image will be eventually added back at the end of network. Therefore, I want the absolute value of residual learned after 16 residual blocks small enough and at the same time represent the low frequency information correctly. Orthogonal initialization is widely use in RNN to solve the situation where one matrix after hundreds of matrix multiplication becomes exponential in the number of layers – either become too large or converge to zero. Thus, this initialization method is also suitable for our case: after many residual blocks, the output tensor will be neither too large nor nearly zeros.

4.2 inheriting initialization for the second and third blocks.

Once I get the trained N_1, it is not necessary to training N_2 from script. I directly pass the parameters of N_1 to N_2 as the initial value. Similar, after trained N_2, I pass the parameters of N_2 to N_3 and the initial value.

4.3 N_i *and* N_i^+

Since I use phased training trick here (see 3.2 for detail), After every network block finishes backward propagation, it has to generate a predicted high-resolution image set depend on the given low-resolution image set. However, the problem I faced there is that the range of output tensor is not strictly [0,1], which is questionable to converge into [0,255] RGB dimension. Unfortunately, simply add the restriction $\begin{cases} x = 1, & when\, x > 1 \\ x = 0, & when\, x < 0 \end{cases}$ at the end of network block didn't preform well since this restriction drop too much information. For example, 2 and 1.5 is quite different but they eventually become 1. As a result, the network block become hard to converge at the beginning.

Therefore, firstly I trained N_i without restriction condition for 2000 iteration, with Adadelta optimizer [17] with lr=1, rho=0.9, eps=1e-06, weight-decay=0. Compare to Adam, Adadelta can find a roughly satisfying position very fast and there is no need to set learning rate. Then I take the parameters of N_i as the initial value, add the restriction function $\begin{cases} x = 1, & when\, x > 1 \\ x = 0, & when\, x < 0 \end{cases}$ in the end to form a new network block $N_i^+ \; where\; i = 1, 2,3$. Adam is again applied there, 3000 iterations are implemented with learning rate 0.0001, and then the learning rate is decay to 0.00001 for 10000 iterations. Thus, the total backpropagation needed for each phase 15000 times.

To coincide with RCAN, data augmentation is performed on the 71 images pairs, which are randomly rotated by 90◦, 180◦, 270◦ and flipped horizontally. 16 LR color patches with the size of 48×48 are extracted as inputs. In addition, the of parameters of Adam optimizer is: β1 = 0.9, β2 = 0.999, and ε = 10−8.

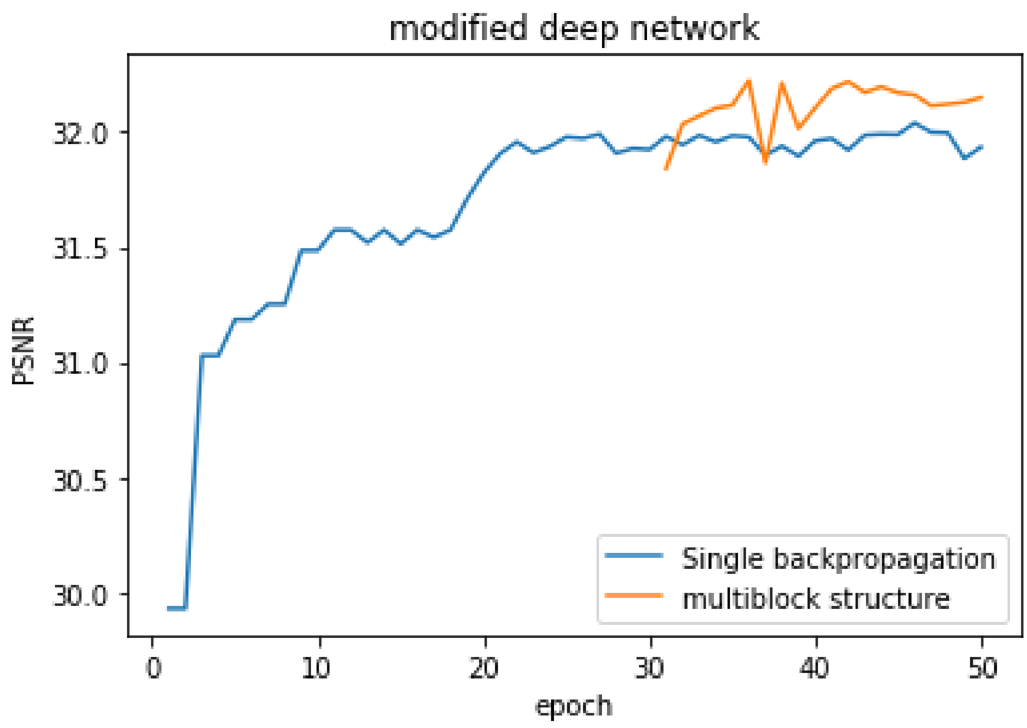


Fig 7, comparison of single backpropagation and multiblock structure

In order to ensure that the computation resource used for two method is the same, for the single back propagation (which is labeled as orange line), we use totally 25000 iterations in 50 epochs. The first 5000 iterations have learning rate 0.0001 and the last 20000 iterations have learning 0.00001. For the combined training phase (phase 4 which is labeled as blue line), only 10000 iteration is used where I set the learning rate as 0.0001 for the first 5000 iterations and 0.00001 for the last 5000 iterations.

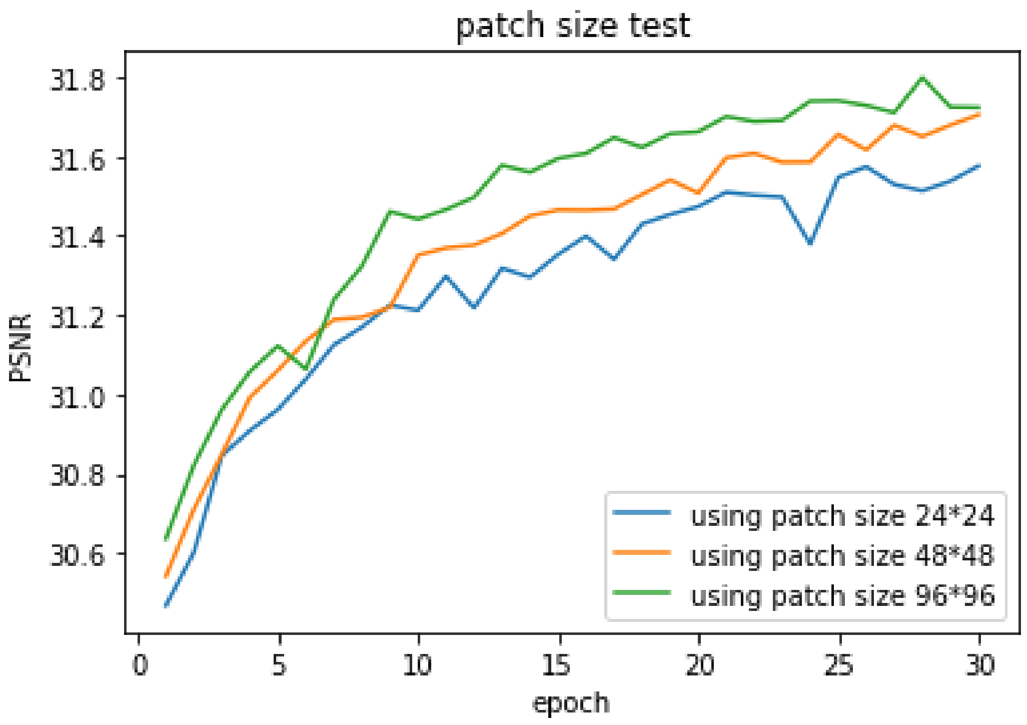
The highest PSNR for multiblock learning structure in 20 epochs is 32.22 while the highest PSNR for single back propagation in 50 epochs is 32.04, nearly 0.2 dB improvement.

5, More Analysis

5.1 simple numerical test about the influence of receptive field

One nature question is that how to choose the proper receptive field size. Clearly that the smallest size 1×1 is not reasonable since there is even no patch learnable. And a every huge receptive field is not proper too (for example 500×500), since there is too much detail for one patch and due to the RAM and GPU limitation, it is not realistic neither.

Here I choose 3 different patch sizes: $24 \times 24, 48 \times 48, 96 \times 96$ and train N_i network block which contain 16 residual blocks. The receptive filed is of size 71×71. I use Adam optimizer with β1 = 0.9, β2 = 0.999, ε = 10−8, and the learning rate is set to 0.0001. 6000 iterations are performed here.



Generally speaking, by using a larger patch size, we can get better results. And this inspire me to change the patch size in further experiment.

5.2 what can be achieved during phase training

I generate the validation result after each phase. The network structure can achieve 31.8557 dB after phase 1, and the PSNR index become 31.9270 after phase 2, and 31.9479 after phase 3

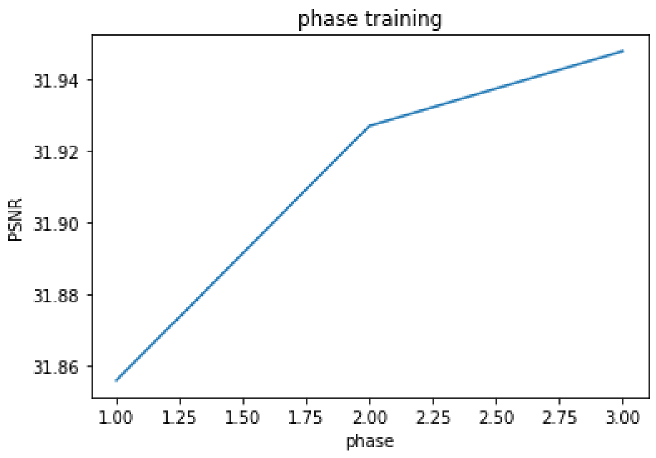


Fig 8, validation for each phase

This result shows that, it is also possible for the network block N_i to learn new things basic on its senior N_{i-1}. Which makes it possible for us to build deeper network structure basic on existing network. For example, in this report I use N_1, N_2, N_3 to form N which contain nearly 100 layers. We can go one step further: using $N_4, N_5..N_9$ to train other two 100-layer network N', N'', and combine $W = N''(N'(N))$ to form a 300-layer deep neural network. A better performance should be expected.

5.3 Inverse analysis

After I trained the deep neural network N by phased training. I tried to do the inverse thing: pass the parameters of pretrained N back to N_1, N_2, N_3.

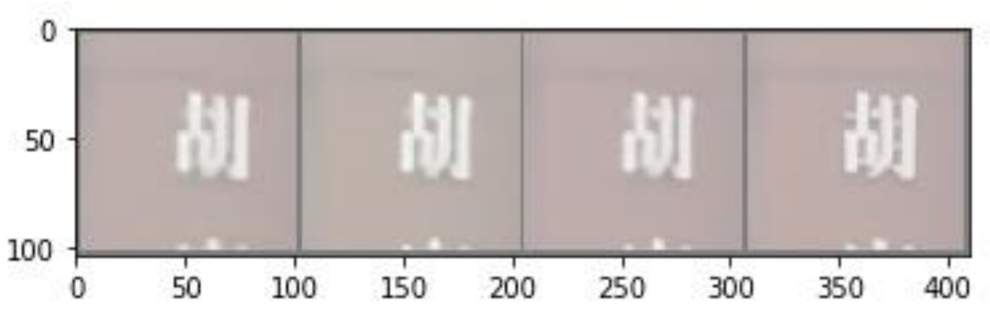
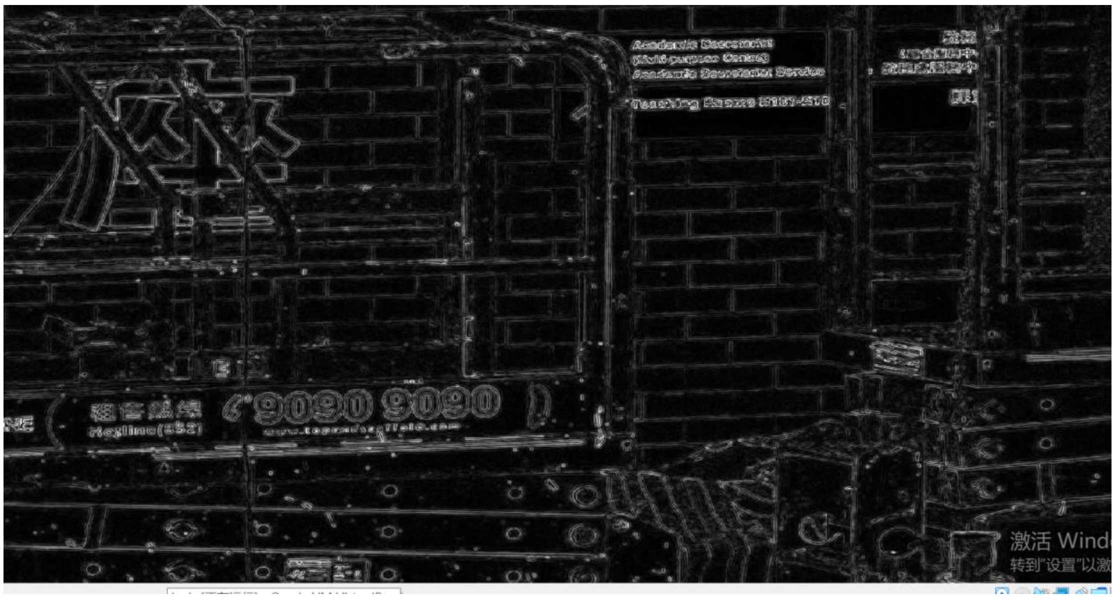


Fig 9, first image is calculated by N_1*, second is calculated by* $N_2(N_1)$*, third is calculated by* $N_3(N_2(N_1))$*, least image is the high-resolution image.*

We can see that the image quality increases with more network block used. And the method has its only strong and clear physical meaning.

6 Further research

1) Our model can't achieve state-of-art performance (about 33.3 dB). One important reason is that this model is not deep enough: its depth is $\frac{1}{4}$ of RCAN. I am planning to build more deeper neural network.
2) According to Blau [18], distortion and perceptual quality are at odds with each other. For this report I only used distortion criterion (L1 Loss) but not consider perceptual quality (GANs). And recent researches either focus on approaching better distortion quality (using L1, L2 loss) or focus on getting higher visualization quality (use perceptual loss and GANs). However, the multiple phased training procedure provide the chance to both consider distortion and perceptual quality since within multiple phased training, various loss function can be applied. This direction needs more investigation.
3) What really need to learn for a "good" neural network? If we calculate the 1st order derivative (only consider the pixel and its 4 neighbors) of the gray scale residuals of high-resolution image and paired low-resolution image. We can get something like this:



The structure of the picture is still very clear, which means that the pixel value between HR and LR change a lot in the boundary. And making the 1st order derivative picture unclear sounds like a proper way to get good high-resolution prediction. However, I fail to use this new loss (sum of 1st order derivate value) and the result can't converge. I am still thinking how to combine this new idea with multiple phased training procedure and can't coming up a clever way to implement this.

Reference

[1] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015). "Deep Residual Learning for Image Recognition". arXiv:1512.03385

[2] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, Yun Fu (2018). "Image Super-Resolution Using Very Deep Residual Channel Attention Networks". arXiv:1807.02758

[3] Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang. Learning a Deep Convolutional Network for Image Super-Resolution, in Proceedings of European Conference on Computer Vision (ECCV), 2014

[4] Jiwon Kim, Jung Kwon Lee, Kyoung Mu Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks", arXiv:1511.04587

[5] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, Lei Zhang, "Toward Real-World Single Image Super-Resolution: A New Benchmark and A New Model", arXiv:1904.00523

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

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015

[8] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi, ” Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, arXiv:1609.04802

[9] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, “Enhanced Deep Residual Networks for Single Image Super-Resolution”, arXiv:1707.02921

[10] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, Yun Fu, “Residual Dense Network for Image Super-Resolution”, arXiv:1802.08797

[11] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, Yun Fu, “Image Super-Resolution Using Very Deep Residual Channel Attention Networks”, arXiv:1807.0275

[12] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, Xiaoou Tang, ” ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks”, arXiv:1809.00219

[13] “A guide to receptive field arithmetic for Convolutional Neural Networks” https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

[14] Diederik P. Kingma, Jimmy Ba, “Adam: A Method for Stochastic Optimization”, arXiv:1412.6980

[15] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507

[16] Andrew M. Saxe, James L. McClelland, Surya Ganguli,” Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”, arXiv:1312.6120

[17] Matthew D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method”, arXiv:1212.5701

[18] Yochai Blau, Tomer Michaeli, “The Perception-Distortion Tradeoff”, arXiv:1711.06077