DEEP LEARNING LAB(EEE4423-01) Week 5 : Accelerating the Super-Resolution Convolutional Neural Network

Hyeon Si Eun[1], 2019142214
[1] Department of Electric and Electronic Engineering, Yonsei University

1 Introduction

The paper "Accelerating the Super-Resolution Convolutional Neural Network" [2] proposes a new method to accelerate the training of super-resolution convolutional neural networks (SRCNNs[1]), which are commonly used to increase the resolution of low-resolution images. Deep-learning methods are mainly used to solve SR problems, and among them, SRCNN has attracted attention for its simple structure and good performance. Compared to other previous learning-based methods, SRCNN is already fast enough, but its speed on large images is not satisfactory. After examining the network structure, The authors found two causes of speed limiting. First, bicubic upsampling is applied to the LR image and fed into the network as input. If the image size is increased n times, the amount of computation increases by the power of n. For example, When upsampling a 240 X 240 image to 3 times the size, SRCNN shows a throughput of 1.32 fps, but it needs to reach 24 fps, which is about 17 times the real-time performance. Therefore, if the input image is used as it is without upsampling, it will be faster by the power of n.

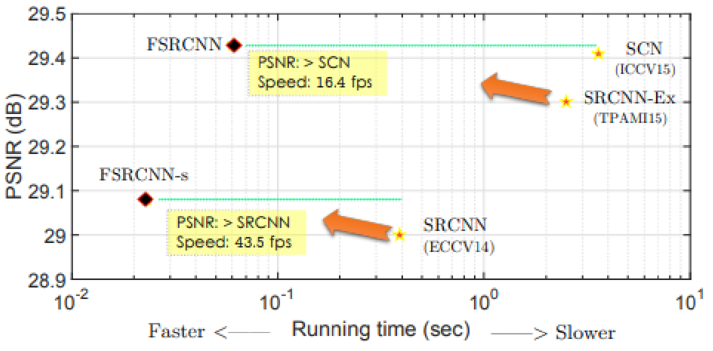

Figure 1: The proposed FSRCNN networks achieve better super-resolution quality than existing methods, and are tens of times faster. Especially, the FSRCNN-s can run in real-time (> 24 fps) on a generic CPU.

FSRCNN is 40 times faster than the existing SRCNN-Ex (larger version with 6 times more parameters than SRCNN) while showing higher performance. Similarly, FSRCNN-s, a small version of FSRCNN, is 17.36 times faster than SRCNN while achieving almost the same performance. Since this is the speed at which the CPU can process 24 fps, it can be said that real-time serving is possible.

Second, in SRCNN, input patches are projected onto high-dimensional LR feature space and HR feature space. The higher the number of parameters, the slower the speed, but at the same time the better the performance. The problem is 'how to reduce the size of the network while maintaining the existing performance'. FSRCNN was designed by solving the two problems mentioned above as follows. To solve the first problem, we changed the bicubic interpolation to a deconvolution layer. To solve the second problem, shrinking and expanding layers were added at the beginning and end of the mapping layer.

2 Related Work

1. Deep Learning for SR

After SRCNN was first proposed to solve the SR problem, more and more deep structures are emerging. However, all other models require bicubic interpolation in the pre-processing process. The proposed FSRCNN does not only perform on the original LR image, but also contains a simpler but more efficient mapping layer. Furthermore, the previous methods have to train a totally different network for a specific upscaling factor, while the FSRCNN only requires a different deconvolution layer. This also provides us a faster way to upscale an image to several different sizes.

2. CNNs Acceleration

Many studies have been conducted to increase the speed of CNNs in areas such as object detection or image classification. However, the studies are designed for high-level vision problems. Since deep models for SR problems do not have fully-connected layers, the role of convolution filters is important.

3 Fast Super-Resolution by CNN

In this paper, the authors briefly describe SRCNN and explain how it was changed. SRCNN aims at learning an end-to-end mapping function F between the bicubic-interpolated LR image Y and the HR image X. The network contains all convolution layers, thus the size of the output is the same as that of the input image. As depicted in Figure 2, the overall structure consists of three parts that are analogous to the main steps of the sparse-coding-based methods.

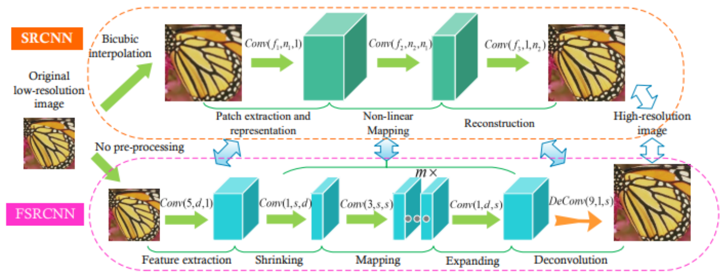

Figure 2: This figure shows the network structures of the SRCNN and FSRCNN.

The computation complexity of the network can be calculated as follows,

$$O\left\{(f_1^2 n_1 + n_1 f_2^2 n_2 + n_2 f_3^2) S_{HR}\right\} \tag{1}$$

where $\{f_i\}_{i=1}^3$ and $\{n_i\}_{i=1}^3$ are the filter size and filter number of the three layers, respectively. S_{HR} is the size of the HR image. If you want to know detailed model structures, we recommend viewing this paper.

4 FSRCNN

FSRCNN can be divided into 5 parts: feature extraction, shrinking, mapping, expanding, deconvolution. The first four are convolution layers, and the last one is a deconvolution layer. From now on, the convolution layer is expressed as $Conv(f_i, n_i, c_i)$, and the deconvolution layer is expressed as $DeConv(f_i, n_i, c_i)$ where the variables (f_i, n_i, c_i) represent the filter size, the number of filters and the number of channels, respectively.

1) Feature Extraction

The filter size of the first layer of SRCNN is 9. Let LR be Y_s and interpolated Y_s be Y. Since most of the pixels in Y are interpolated in Y_s, a 5×5 patch in Y_s can cover as much as a 9×9 patch in Y. Therefore, we set the first layer as $Conv$(5,d,1).

This is an extended abstract. The full paper is available at the this paper.

2) Shrinking

In the mapping step of SRCNN, high-dimensional LR features are directly mapped to the HR feature space. At this time, since dimension d is usually very large, the amount of calculation increases. Therefore, to reduce d, a shrinking layer was added after the feature extraction step. The second layer is $Conv(1,s,d)$. (At this time, s « d) This process greatly reduces the number of parameters.

3) Non-linear Mapping

Non-linear mapping is an important step that greatly affects SR performance. The most influential here are width (the number of filters in a layer) and depth (the number of layers). Performance was better when using 5×5 layers than when using 1×1 layers in SRCNN. Based on this experiment, a more effective mapping layer was devised. Since there is a trade-off between network size and performance, we set $f_3 = 3$. And to maintain good performance in SRCNN, we used several 3×3 layers. In this way, the non-linear mapping layer is $m \times Conv(3,s,s)$.

4) Expanding

The expanding layer does the exact opposite of the shrinking layer. The shrinking process reduces the LR feature dimension for calculation, but the performance is poor when creating HR images with such low-dimensional features. So They added an expanding layer. They used a 1×1 filter to keep it consistent with the shrinking layer. Contrary to the shrinking layer being $Conv(1,s,d)$, the expanding layer is $Conv(1,d,s)$.

5) Deconvolution

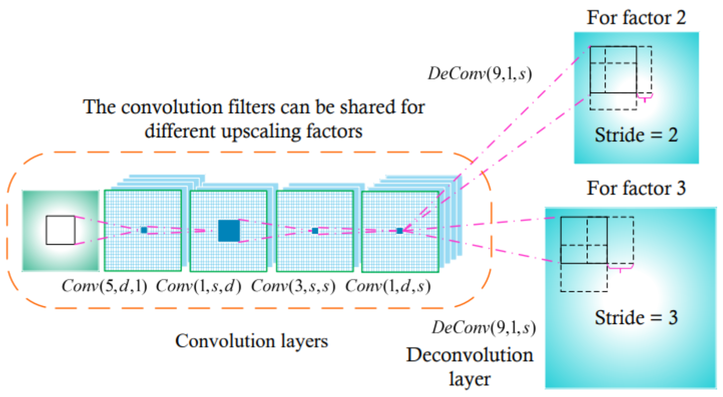
For factor 2
$DeConv(9,1,s)$
The convolution filters can be shared for different upscaling factors
Stride = 2
For factor 3
$Conv(5,d,1)$ $Conv(1,s,d)$ $Conv(3,s,s)$ $Conv(1,d,s)$
Stride = 3
$DeConv(9,1,s)$
Convolution layers
Deconvolution layer

Figure 3: The FSRCNN consists of convolution layers and a deconvolution layer. The convolution layers can be shared for different upscaling factors. A specific deconvolution layer is trained for different upscaling factors.

The last part, deconvolution, upsamples and collects the previous features with the deconvolution layer. Deconvolution can be seen as the opposite operation of convolution operation. When convolution is performed with a stride k, the output is $\frac{1}{k}$ the size of the input. If you swap the input and output, the output will be k times the input as shown above. Thinking of the network in reverse like this, the reversed network takes the HR image and makes the LR come out. As such, the deconvolution layer extracts features from the HR image, so a 9×9 filter was used. The deconvolution layer is denoted by $DeConv(9, 1, d)$. Unlike interpolation kernels, the deconvolution layer learns an upsampling kernel.

6) PReLU

PReLU was used to eliminate 'dead features' created from zero gradients in ReLU. The expression of PReLU is as follows:

$$f(x_i) = max(x_i, 0) + a_i min(0, x_i) \tag{2}$$

where x_i is the input signal of the activation f on the i-th channel, and a_i is the coefficient of the negative part. The parameter a_i is fixed to be zero for ReLU, but is learnable for PReLU.

7) Overall Structure

The complete FSRCNN network architecture is as follows :
$Conv(5,d,1) \rightarrow PReLU \rightarrow Conv(1,s,d) \rightarrow PReLU \rightarrow m \times Conv(3,s,s)$
$\rightarrow PReLU \rightarrow Conv(1,d,s) \rightarrow PReLU \rightarrow Deconv(9,1,d)$
The computational complexity of FSRCNN is as follows:

$$O(25d + sd + 9ms^2 + ds + 81d)S_{LR} = O(9ms^2 + sd + 106d)S_{LR} \tag{3}$$

8) Cost Function

They adopt the mean square error (MSE) as the cost function. The optimization objective is represented as

$$\min_{\theta} \frac{1}{n} \sum_{n}^{i=1} \left\| F(Y_i^s; \theta) - X^i \right\|_2^2 \tag{4}$$

where Y_s^i and X^i are the i-th LR and HR sub-image pair in the training data, and $F(Y_i^s; \theta)$ is the network output for Y_s^i with parameters θ.

5 Differences Against SRCNN: From SRCNN to FSRCNN

	SRCNN-Ex	Transition State 1	Transition State 2	FSRCNN (56,12,4)
First part	Conv(9,64,1)	Conv(9,64,1)	Conv(9,64,1)	**Conv(5,56,1)**
Mid part	Conv(5,32,64)	Conv(5,32,64)	**Conv(1,12,64)-4Conv(3,12,12)-Conv(1,64,12)**	**Conv(1,12,56)-** 4Conv(3,12,12) **-Conv(1,56,12)**
Last part	Conv(5,1,32)	**DeConv(9,1,32)**	**DeConv(9,1,64)**	**DeConv(9,1,56)**
Input size	S_{HR}	S_{LR}	S_{LR}	S_{LR}
Parameters	57184	58976	17088	12464
Speedup	1×	8.7×	30.1×	41.3×
PSNR (Set5)	32.83 dB	32.95 dB	33.01 dB	33.06 dB

Figure 4: the transitions from SRCNN to FSRCNN.

6 SR for Different Upscaling Factors

Another advantage of FSRCNN over the previous learning-based methods is that FSRCNN could achieve fast training and testing across different upscaling factors. This is also proved by experiments, of which the convolution filters are almost the same for different upscaling factors. With this property, we can transfer the convolution filters for fast training and testing. The speed is increased by using a deconvolution layer in this model. we only fine-tune the deconvolution layer for another upscaling factor and leave the convolution layers unchanged.

7 Experiments

Experimental results on various benchmark datasets demonstrate that the FSRCNN outperforms existing state-of-the-art methods in terms of both quantitative metrics and visual quality. Furthermore, the proposed method is computationally efficient, allowing for real-time applications.

If you want to see detailed experimental results, we recommend viewing the full paper.

8 Conclusion

the proposed Fast Super-Resolution Convolutional Neural Network (FSRCNN) provides a more efficient and effective approach to single image super-resolution tasks compared to existing methods. The network is designed to reduce the number of parameters and operations required for super-resolution while maintaining high reconstruction quality. In this paper, research was conducted for deep learning-based super-resolution that restores faster and more accurately. The overall structure of SRCNN was reorganized and the name was changed to FSRCNN. As a result, it produces results about 40 times faster, bringing it one step closer to real-time super-resolution.

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[2] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 391–407. Springer, 2016.