Week5 Pre-Report: Accelerating the Super-Resolution Convolutional Neural Network - FSRCNN

Euijin Hong[1]
[1]Department of Electrical and Electronic Engineering, Yonsei University.

1 Introduction and Background

The underlying problem of SRCNN[2][3] was its high computational cost which hampers to achieve real-time applications. The goal of SR(super-resolution) is to recover the resolution of an image from low(LR) to high(HR). The contemporary SR algorithms were learning-based, which learned a mapping between LR and HR image spaces. Although SRCNN was a powerful tool for dealing SR tasks, the speed was insufficient by two innate limits.

The first reason was the requirement of upsampling via bicubic interpolation as a pre-processing step forming the input. This makes the computational cost to increase quadratically; if upscaling factor is given as n, interpolating the LR image becomes n^2 compared to the original one. In other words, we can reduce the computational burden as n^2 times faster by training the network directly with original LR image.

The second reason was by a use of non-linear mapping step, which results additional computational cost. Input images go through SRCNN on a high dimension LR and HR feature space. Adopting a wider mapping layer[3] can improve the mapping accuracy, but invokes running time cost. Thus, shrinking the network scale is another concern to improve speed.

The first problem could solved by adopting a *deconvolution layer* at the end of the network, replacing the bicubic interpolation. This enables computational complexity to depend solely on the spatial size of original LR image. The deconvolution layer is comprised of diverse upsampling kernels which can be learned automatically.

The second problem could be dealt with FSRCNN by shrinking and expanding the layers at the beginning and the end of the mapping layer, separately to restrict mapping in a low-dimensional feature space. Moreover, FSRCNN seperated a single wide mapping layer into several layers with 3×3 filter size fixed. This altogether forms a model to resemble an *hourglass*.

As a result, researchers could attain speed-up via FSRCNN by exceeding $40\times$ with even superior than that in [3]. They also introduce FSRCNN-s which gains similar restoration quality as SRCNN, but shows real time (24 fps) operation. Furthermore, the convolution layers of FSRCNN can be shared, thus only fine-tuning the deconvolution layer for another upscaling factor is needed, with no loss of mapping accuracy. Thus, the contributions can be denoted as follows: a) By introducing FSRCNN with hourglass-shape CNN structure with deconvolution filters, the model could learn end-to-end without preprocessing. b) Achieved explosive speed up that can perform on real time. c) Transferring the convolution layers of the network across different upscaling factors, with no loss of restoration quality.

2 Related Works

Deep Learning for SR: The pioneer work of applying SR is SRCNN proposed in [2][3]. There were some further attempts such as replacing a mapping layer by a set of sparse coding sub-networks also known as SCN(sparse coding based network)[9], but has problems of shrinking the network since it mostly mimics the sparse-coding solver. Plus, these models have to process the bicubic-upscaled LR images. The proposed network FSRCNN performs on the original LR image, contains more efficient layer, and requires only some different deconvolution layers.

Acceleration of CNN: There were numerous studies investigating the acceleration of CNN such as in [1], [11]. The FSRCNN model aims to accelerate CNNs, but in a different approach. While the studies focused on approximating the existing well-trained models, researchers of FSRCNN reformulated the prior model. Also, the models dealt with studies are proposed for high-level vision tasks(e.g. object detection), but FSRCNN is for low-level vision problems. Since models for SR does not contain FC layers, the approximation of conv. layers will thoroughly influence the performance.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.

3 Theory and Architecture

3.1 SRCNN

The goal of SRCNN is to learn an end-to-end mapping functon F between the LR image Y formed from bicubic interpolation and the HR image X. The size of the output is same as that of the input image, because it contains all conv. layers. The structure consists of three parts that resembles that of sparse-coding-based methods[10], as shown in Figure 1.

Computational complexity of SRCNN is calculated as follows:

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

Here, $\{f_i\}_{i=1}^3$ and $\{n_i\}_{i=1}^3$ are the filter size and filter number of three layers, respectively, while S_{HR} is the size of HR image. Thus complexity is proportional to the size of HR image, while the middle layer contributes the most among three layers.

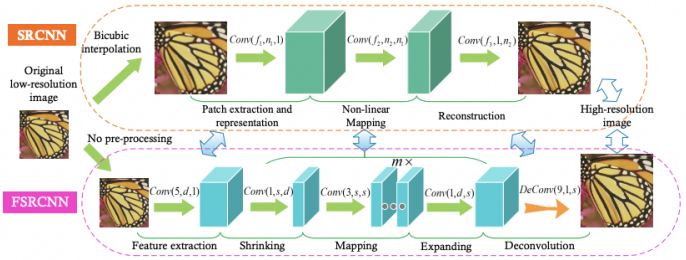

Figure 1: The network structures of SRCNN and FSRCNN. First, an original LR image without bicubic interpolation serves as an input in FSRCNN, while a deconvolution layer is introduced at the end of the network to perform upsampling. Second, the shrinking, mapping, and expanding steps replaces the non-linear mapping step in SRCNN. Third, FSRCNN uses layers with smaller filter sizes and a deeper network structure.

3.2 FSRCNN

FSRCNN can be segregated into five parts as shown in Figure 1 - feature extraction, shrinking, mapping, expanding and deconvolution. The former four parts are convolution layers $Conv(f_i, n_i, c_i)$ and the final part is a deconvolution layer $DeConv(f_i, n_i, c_i)$ where f_i: filter size, n_i: number of filters and c_i: number of channels.

Since it is unable to investigate every variables, assigninig the reasonable value to the insensitive variables in advance, while leaving the sensitive variables(slight change can fundamentally influence the performance) unset. The sensitive variables represent important influential factors in SR, which are as follows.

Feature Extraction: By convoluting the small LR input Y_s with the first set of filters, each patch of the input (1-pixel overlapping) is represented as a high-dimensional feature vector. The part is similar to the first part of SRCNN, thus refer to the model for choosing the parameters f_i, n_i, c_i. Specifically, we use a smaller filter size $f_1 = 5$ with small information loss, since most pixels in Y are interpolated from Y_s, 5×5 patch in Y_s can cover almost all information of 9×9 patch in Y. Then, the number of channels is set as $c_1 = 1$ by following SRCNN, and filter number as $n_1 = d$ where d, a number of LR feature dimension, is the first sensitive variable remain unset. As a result, the first layer is represented as $Conv(5, d, 1)$.

Shrinking: In order to save computational cost[6], shinking layer is added after the feature extraction layer to reduce the LR feature dimension d. The filter size is fixed to be $f_2 = 1$, where filters perform like linear combination among LR features. Also, by implementing smaller filter number as $n_2 = s << d$, the LR feature dimension is reduced from d to s, which is the second sensitive variable determining the shrinking level. Eventually, the second layer is represented as $Conv(1, s, d)$, greatly reducing the number of parameters.

Non-linear mapping: This part is the most important step that affects the SR performance, while width and depth of the mapping layer are the most influencing factors. By implementing multiple mapping layers (denoted as m, another sensitive variable) that have a medium filter size $f_3 = 3$, we can maintain the performance of SRCNN and reduce the network scale. Therefore, the non-linear mapping part can be expressed as $m \times Conv(3,s,s)$ where all mapping layers contain the same number of filters $n_3 = s$.

Expanding: This layer acts as the opposite process of shrinking layer. Since directly generating the HR image from low-dimensional features results poor quality, we add an expanding layer after the mapping part to expand the HR feature dimension. The expanding layer is $Conv(1,d,s)$, by adopting 1×1 filters to preserve consistency with shrinking layer.

Deconvolution: This part upsamples and adds up the former features with a set of deconvolution filters. The deconvolution filter deconvolved with the image with stride k obtains output k times of the input, which is the opposite of that of convolution filter. The stride is set to the desired upscaling factor as $k = n$, directly reconstructing the HR image as an output. Filter size can be determined as $f_5 = 9$, by taking the opposite perspective of the network and comparing with the first layer of SRCNN. Thus, the layer can be represented as $DeConv(9,1,d)$.

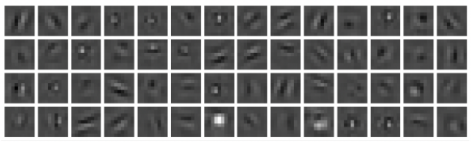

Figure 2: Learned deconvolution layer with 56 channels, upscaling factor 3.

The deconvolution layer learns a set of upsampling kernel for the input feature maps. The learned diverse kernels are significant and unique by themselves, as shown in Figure 2.

PReLU: PReLU(Parametric Rectified Linear Unit) is used as an activation function after each convolution layer to avoid "dead features" caused by zero gradients in ReLU[4]. The function of PReLU can be depicted as $f(x_i) = max(x_i,0) + a_i min(0,x_i)$, where coefficient a_i is non-zero.

Overall structure: A complete FSRCNN is comprised as $Conv(5,d,1) - PReLU - Conv(1,s,d) - PReLU - m \times Conv(3,s,s) - PReLU - Conv(1,d,s) - PReLU - DeConv(9,1,d)$. Thus, the network can be represented as $FSRCNN(d,s,m)$, where d,s,m are three sensitive variables. The computational complexity can be calculated as:

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

Cost function: MSE(mean square error) is adopted as the cost function which can be represented as follows:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} ||F(Y_s^i;\theta) - X^i||_2^2, \quad (3)$$

where Y_s^i and X^i are the i-th LR and HR sub-image pair in the train data, and $F(Y_s^i;\theta)$ is the network output for Y_s^i with parameters θ. SGD is used to optimize all parameters with standard backpropagation.

3.3 Differences between SRCNN and 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 3: Transitions from SRCNN to FSRCNN

We show in Figure 3 the network configurations of SRCNN, FSRCNN and two transition states, and their performances (average PSNR on Set5) trained on 91-image dataset[10].

First, we can see that learned deconvolution kernels are better than a single bicubic kernel, increasing performance by about 0.12dB. Second, replacing a single mapping layer to combination of shrinking layer, 4 mapping layers, and an expanding layer decreased the number of parameters, achieved better results (33.01 dB) and accelerated the process - 30.1×. Finally, adopting smaller filter sizes and less filters obtained speedup of 41.3× and 0.05 dB improvement.

3.4 SR for Different Upscaling Factors

Since only the last deconvolution layers contain the information of the upscaling factor, we can only fine-tune the deconvolution layer for another upscaling factor (which is very fast) and maintain the convolution layers unaltered.

4 Evaluation

Implementation Details: The researchers contributed a new General-100 dataset containing 100 bmp-format images, which is used as a training dataset. Training samples are gained by downsampling the training images by scaling factor n to form LR images, and crop them into $f_{sub} \times f_{sub}$-pixel sub-images with stride k. Corresponding HR sub-images with size $(nf_sub)^2$ are also cropped out from GT(ground truth) images. Additional crop by $(n-1)$-pixel borders in HR sub-images is required for deconvolution outputs since $(nf_sub - n + 1)^2$ is generated instead of $(nf_sub)^2$. The researchers also introduced a two-step training tactics for more rapid converging; first by training a network from scratch with 91-image dataset, then adding General-100 dataset for fine-tuning when training is saturated. The learning rate of conv. layers are set to 10^{-3} and that of deconvolution layer to be 10^{-4}, which are divided by two when fine-tuning. Initialization method in [4] is used for initializing the weights of conv. filters, while deconvolution filters are initialized identically as in SRCNN[2, 3].

Evaluation under Different Settings: By controlling the values of three sensitive variables - LR feature dimension $d = 48, 56$, number of shrinking filters $s = 12, 16$, and mapping depth $m = 2, 3, 4$, which shows the output as in Figure 4-(left). The PSNR result shows that deeper depth of m shows better results, while more parameters of d and s does not guarantee a better result. The best parameter-performance tradeoff is at FSRCNN(56,12,4). FSRCNN(48,12,2), the smallest model attaiined 32.87 dB of average PSNR, higher than that of SRCNN-Ex(32.75 dB)[3].

Real-Time SR by FSRCNN: We can calculate the limit of the number of parameters required to satisfy real-time (24fps). Speed of SRCNN in 760×760 input size is 1.32fps, having upscaling factor 3 and 8032 parameters. According to Equation (1) and (2), FSRCNN should have number of parameters less than $8032 \times 1.32/24 \times 3^2 \approx 3976$. This can be achieved with FSRCNN(32,5,1) configuration, having the speed of 24.7fps, and outperforming SRCNN(9-1-5)[2].

Experiments for Different Upscaling Factors: By transferring the convolution filters, FSRCNN is flexible in terms of learning and testing across upscaling factors. The FSRCNN(56,12,4) trained under upscaling factor 3, is then trained for ×2 on the basis of that for ×3. Fine tuning was the only process done during training, compared by that of model trained for ×2 from scratch, where the result is shown in Figure 4-(right).

Comparison with State-of-the-Arts: By using the same training set, FSRCNN(56,12,4) and FSRCNN(32,5,1) was compared with SRF(super-resolution forest)[7], SRCNN[2], SRCNN-Ex[3], and SCN(sparse coding based network)[9], thus achieving ×40 faster speed than SRCNN-Ex, SRF and SCN with factor 3 while outperformiing in terms of PSNR results. Using different training sets and comparing FSRCNN with KK[5] and A+[8] showed exceeding performance of FSRCNN in most upscaling factors and datasets.

Settings	$m = 2$	$m = 3$	$m = 4$
$d = 48, s = 12$	32.87 (8832)	32.88 (10128)	33.08 (11424)
$d = 56, s = 12$	33.00 (9872)	32.97 (11168)	33.16 (12464)
$d = 48, s = 16$	32.95 (11232)	33.10 (13536)	33.18 (15840)
$d = 56, s = 16$	33.01 (12336)	33.12 (14640)	33.17 (16944)

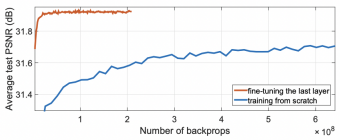

Figure 4: (left): PSNR and parameters of different settings, (right): Conergence curves for different training strategies

5 Conclusion

Introducing FSRCNN could solve the limitations of existing SR models based on deep learning. The team could reform the SRCNN configuration, achieving high running speed(more than 40×) without loss of restoration quality. The experimental results show that the model can generate satisfactory SR results, while maintaining run time advantages. The model can further be adapted for real-time video SR, and influence fast deep models solving other low-level vision tasks.

[1] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[2] 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, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*, pages 184–199. Springer, 2014.

[3] 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.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[5] Kwang In Kim and Younghee Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE transactions on pattern analysis and machine intelligence*, 32(6):1127–1133, 2010.

[6] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[7] Samuel Schulter, Christian Leistner, and Horst Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3791–3799, 2015.

[8] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Computer Vision–ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part IV 12*, pages 111–126. Springer, 2015.

[9] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE international conference on computer vision*, pages 370–378, 2015.

[10] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010.

[11] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38 (10):1943–1955, 2015.