Image Super-Resolution Via SRCNN

Mingfei Sun
Electrical Engineering
Columbia University
New York, United States
ms5898@columbia.edu

Zhuoxu Duan
Electrical Engineering
Columbia University
New York, United States
zd2235@columbia.edu

Yifeng Deng
Electrical Engineering
Columbia University
New York, United States
yd2505@columbia.edu

Zihan Yang
Electrical Engineering
Columbia University
New York, United States
Yang.zihan@columbia.edu

***Abstract*—We used two methods to do image super-resolution (SR). The first one is sparse representation (ScSR), and the second one is super-resolution convolutional neural network (SRCNN), an end-to-end algorithm to recover the high-resolution images from low algorithm via convolution neural network (CNN). We compare the results of these methods and then get the advantages and disadvantages of these two methods.**

Keywords—super-resolution, SRCNN, SCSR, CNN

I. INTRODUCTION

Recovering a high-resolution image from a low image super-resolution is a classical problem. Thus, in our project, we are interested in super-resolution. We used two methods to recover high-resolution images from blurred images. As we mentioned, these two methods we used are ScSR and SRCNN.

A. ScSR Method

ScSR uses sparse representation to recover high-resolution images. Because image patches can be well-represented as a sparse linear combination of elements from an chosen dictionary, we can use this representation to generate the high-resolution image. We also need to train dictionaries for the low-resolution image patches and high-resolution image –patches, and then we could get the similarity of the sparse representations between low-resolution image patches and high-resolution image patches.

B. SRCNN Method

SRCNN is an deep learning method based on convolution neural network for SR. SRCNN is an effective and simple method to do SR, because it learns the end-to-end mapping from the low-resolution images to high-resolution images when we train the CNN models. After we constructed the model, we can get the high-resolution images as output from the model.

C. Our Works

We complete our own software part to train the models and get the high-resolution images. As to ScSR, we also built our software codes to operate these procedure, but we used the model provided by [2], instead of training the model by ourselves. We also note which part of ScSR is from the original paper in our code.

Then we compared the results of ScSR and SRCNN, and analyzed the advantages and disadvantages of these methods.

You can check our codes on Github[4].

II. RELATED WORK

In this section, we will discuss two main original papers of our references. Reference [1] proposed the SRCNN, and [2] proposed ScSR. We would focus on the parts we implemented of these papers.

SRCNN is a deep learning method for super-resolution via convolution neural network. Reference [1] compared the results of different models to get the best performance. It concludes that the performance can be further gained by exploring more and larger filters, but it cannot be improved by more layers. Actually, the results of different models shows that more layers used could result in even worse performance, but the reason behinds is still unclear. [1] So we chose the three layer of the paper in our project to compare this deep learning method with ScSR.

Reference [2] proposed ScSR which can uses sparse representation to recover the high-resolution images. The sparse representation based on the dictionaries trained by low-resolution image patches and high-resolution image patches, and the part of training dictionaries is a little similar to the SRCNN model training part.

III. SUPER-RESOLUTION CONVOLUTIONAL NEURAL NETWORK

A. Dataset

Reference [1] shows the training set can choose 91 images or ImageNet, and ImageNet performs better than 91 images when increasing the number of epoch and backpropagation. However, Training on ImageNet needs much more time than on 91 images but only get 0.2dB increasement of performance. So, we choose SRCNN training on 91 images.

B. Algorithm and Implemention

SRCNN needs data processing before training. The network in [1] is designed to input 33×33 images and output 21×21 images, so data preprocessing should first convert high definition images into low definition images by bicubic interpolation. The low-resolution images were then clipped into 33×33 images as a training set for network input, and this is the only preprocessing.[1] Now we got the low-resolution images which can be represent by $\boldsymbol{Y}$, and then we need to find a mapping from $\boldsymbol{Y}$ to high-resolution images.

$$F(\boldsymbol{Y}) = \boldsymbol{X} \tag{1}$$

$\boldsymbol{X}$ represent the original high-resolution images, and $F(\boldsymbol{Y})$ denote the high-resolution image we recovered from $\boldsymbol{Y}$. $F(\boldsymbol{Y})$

should be as similar as possible to $\boldsymbol{X}$. Therefore, the whole process is to find the mapping F.[1]

Now we divide this into three parts.

The first part is to extract the features from the low-resolution images $\boldsymbol{Y}$, so we uses a set of filters of our first layer to convolve $\boldsymbol{X}$ to get these feature map.

$$F_1(\boldsymbol{Y}) = \max(0, W_1 * \boldsymbol{Y} + B_1) \quad (2)$$

W and B represent filters and biases. W_1 denotes n_1 filters of which size is $c \times f_1 \times f_1$, where c represents channels and f_1 is the size of filters.[1]

Then the second part is non-linear mapping, we need to use the second layer to get another vector from the output of the first layer.

$$F_2(\boldsymbol{Y}) = \max(0, W_2 * F_1(\boldsymbol{Y}) + B_2) \quad (3)$$

W and B represent the same variables as we mentioned above, and W_2 consists of n_2 filters of which size is $n_1 \times f_2 \times f_2$.[1]

The last part is reconstruction, we use the third layer to produce the high-resolution images.

$$F_3(\boldsymbol{Y}) = \max(0, W_3 * F_2(\boldsymbol{Y}) + B_3) \quad (4)$$

W and B represent filters and bias, and W_3 denote c filters of which size is $n_2 \times f_3 \times f_3$.[1]

Because we want to recover the high-resolution images, the output of the third layer is images, thus c also represents how many filters in the layer. Therefore, we could get high-resolution images with c channels.

C. *Network Structure*

According to [1], There are many options for network architecture: 9-5-5; 9-3-5; 9-1-5. We chose the 9-1-5 network, which means the first layer filter size is 9×9, the second filter size is 1×1 and the final size is 5×5. The 9-5-5 network has a better effect and can achieve a higher PSNR value. The number of parameters of 9-1-5, 9-3-5, and 9-5-5 is 8,032, 24,416, and 57,184 respectively.[1] Although 9-5-5 network can get about 0.3dB increase of performance than the others, it is about 6 more times complicated than 9-1-5 network. For that reason we choose 9-1-5 network and an optimization of approximately 3dB is obtained.

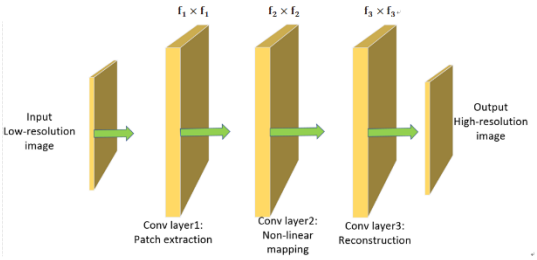


Fig. 1. SRCNN Structure

D. *Expected Results*

After training the network, the low resolution image is taken as input, while the neural network output is high resolution image. However, because of the neural network, the output of high-resolution images will lose some information at image edges.

E. *Data Augment*

When we training our model, 91 pictures were cut into approximately 20,000 pictures as the training set. In the case of a small training set, we did data augment. After data augment, we can see that the network converges faster. The reference SRCNN method required 12 hours of training with 15000 epochs, but we were able to achieve the same effect with 50 epochs.

F. *Scaling Effect*

Because of the scaling of the clipping and neural networks, the resulting image we generated is smaller than the original low-resolution image. However, the SCSR method does not lose the image size.

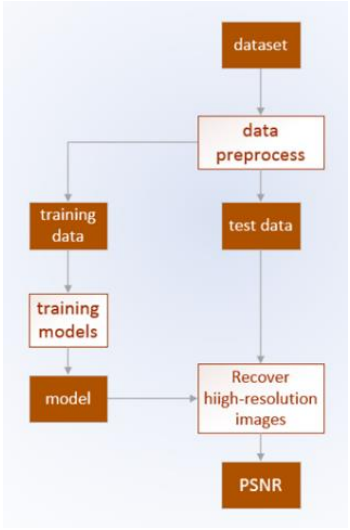


Fig. 2. Flow chart of super-resolution convolutional neural network

IV. SR VIA SPARSE REPRESENTATION

A. *Basic Ideas* [2]

The basic idea from sparse representation is quite straight forward. Given a low-resolution signal, its relationship with the original high dimensional one can be recovered precisely from an over completed dictionary.

$$\boldsymbol{Y} = L\boldsymbol{X} = L\boldsymbol{D}\boldsymbol{\alpha} \quad (5)$$

where $\boldsymbol{Y}$ denotes the input low resolution image of $k \times k$, $\boldsymbol{X}$ is the high resolution original image of $n \times n$ $(k < n)$; $\boldsymbol{L}$ represents the linear relationship of the input output signals; When the dictionary, $\boldsymbol{D}(n, K)$ where $K > n$, is over completed, solution for the coefficient alpha varies. But the sparsest

solution of alpha will be unique. Moreover, there will also be variety of solutions for $\boldsymbol{X}$ recovered from $\boldsymbol{Y}$.[2]

Two kinds of reconstruction constraints (optimization problem) are proposed to limit and find optimal solutions of $\boldsymbol{\alpha}$ and $\boldsymbol{X}$

$$\boldsymbol{\alpha} = \arg\min_{\alpha} \|\boldsymbol{D\alpha} - \boldsymbol{Y}\|^2 + \lambda\|\boldsymbol{\alpha}\|_1 \quad (6)$$

$$\boldsymbol{X}^* = \arg\min_{X} \|\boldsymbol{LX} - \boldsymbol{Y}\|_2^2 + c\|\boldsymbol{X} - \boldsymbol{X_0}\|_2^2 \quad (7)$$

where λ and c are Lagrange coefficients of constraint criteria; $\boldsymbol{X_0}$ is a first solution derived from

$$\boldsymbol{X_0} = \boldsymbol{D} * \boldsymbol{\alpha} \quad (8)$$

while $\boldsymbol{X}^*$ is the optimal solution at last when taking $\boldsymbol{X_0}$ as the input of the second criteria.[2]

However, these criteria are not explicitly illustrated in the article thus not implementable for us by hand. For example, the dictionary needs to be split into $\boldsymbol{D_l}$ and $\boldsymbol{D_h}$ and times two different matrices F and P. Where F denotes feature of input image but not explained what the feature is and how to extract it; P denotes region of overlap between target patch and image which is obviously much smaller than $\boldsymbol{D_h}$ that we cannot apply matrix multiplication here.

So, we finally chose to use the codes for solving these optimization problems provided by the author (which are also different from what was illustrated in the article).

B. Content of Implementation

We realized the algorithm 1 (SR via Sparse Representation) based on the illustrated structure and idea in the article with some different points. To match the method of producing low resolution image in SRCNN part above, we took an input image as the original high resolution one and do a simple zooming to it to generate a new image with lower resolution. This new image will be the input image of the algorithm while its zoomed (bicubic, with a reversed scale) version is applied to make a comparison as before.

The pseudo code of the re-implemented algorithm is as shown in Fig 1. Note that we took a pre-trained dictionary of zoom factor 2 and patch size 5 by 5 to achieve more efficient computing speed and recover to original image size for better comparison. While in the article, the input image was divided into patches of 3 by 3 and no extra zooming effect is applied. Besides, we applied new methods from author for the optimization solving as well as the back-projection part.

Algorithm 1 SR via sparse representation

Input: Input low resolution image Y
Pre-trained dictionary D_l, D_h
Optimization parameter λ

1: Normalize dictionary $D \leftarrow D_{norm}$
2: Zoom Y to output scale
3: **for** 5×5 patch $y \in Y$ **do**
4: extract feature of y, Fy
5: Update $y \leftarrow y - \bar{y}$
6: Update $\beta \leftarrow -D_l^T \cdot Fy$
7: $\alpha^* \leftarrow L1QPFeatureSign(\lambda, D_l^T \cdot D_l, \beta)$
8: high resolution patch $x \leftarrow D_h alpha^* + \bar{y}$
9: put x into corresponding position in X_0
10: **end for**
11: IF $x_i \in X_0 == 0$
12: $x_i \leftarrow bicubic(y_i)$
13: Apply back projection: $X^* \leftarrow \arg\min_X \|SHX - Y\|_2^2 + c\|X - X_0\|_2^2$

Output: Super resolution image X^*

Fig. 3. Pseudo code of the re-implemented algorithm

V. EXPERIMENTS

We tested both SRCNN and ScSR model to compare the performance difference between them.

We got our dataset from Github[3], and picked several images for testing.

We inputted testing images in both model, and got outputs as Fig 4 and Fig 5. We can see that the sparse recovery images and SRCNN recovery images have obvious improvement in image quality compared with the input bicubic blur image.

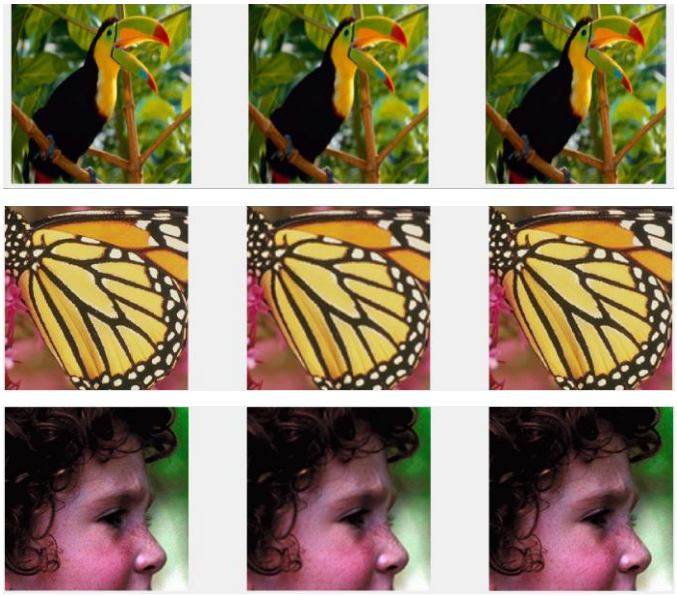


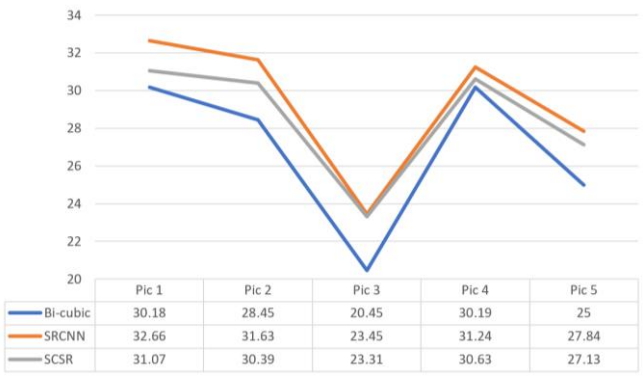
Fig. 4. Three sets of images compare between original image (left), bicubic blur image (middle) and sparse recovery image (right)

Fig. 5. Three sets of images compare between original image (left), bicubic blur image (middle) and SRCNN recovery image (right)

Then we measure the outputs' PSNR via (9) among different methods.

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad (9)$$

We plotted the PSNR curves out as Fig 6. The figure shows SRCNN has the highest PSNR, which means it has the best output quality.



	Pic 1	Pic 2	Pic 3	Pic 4	Pic 5
Bi-cubic	30.18	28.45	20.45	30.19	25
SRCNN	32.66	31.63	23.45	31.24	27.84
SCSR	31.07	30.39	23.31	30.63	27.13

Fig. 6. The PSNR of some tests among methods Bi-cubic, SRCNN and SCSR

There's one thing should notice. When we testing SRCNN method, we used Python with single channel images. But for SCSR, we used Matlab with 3 channel images. This caused the control group, Bi-cubic method, has different PSNR values when compared with SRCNN and SCSR. To cancel this effect, we scaled Bi-cubic/SCSR group's PSNR values down to make the PSNR value of Bi-cubic consistent in both Bi-cubic/SCSR and Bi-cubic/SRCNN testing.

REFERENCES

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image Super-Resolution Using Deep Convolutional Networks" arXiv:1501.00092 [cs.CV]

[2] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma, "Image Super-Resolution Via Sparse Representation" IEEE Transactions on Image Processing, vol. 19, No. 11, November 2010.

[3] tegg89, "SRCNN-Tensorflow" Github https://github.com/tegg89/SRCNN-Tensorflow

[4] ms5898, "ELEN 4810 Digital Signal Processing Project" Github https://github.com/ms5898/DSP-Project