Image Super-Resolution
SRCNN

Astitva Gupta
2018101085

Aryamaan Jain
2019121002

Ayan Biswas
2019121009

Trusha Sakharkar
2018101093

Abstract

This project is the implementation of the paper titled ***Image Super-Resolution Using Deep Convolutional Networks*** *by Chao Dong et al. The implementation is done in* ***Python3*** *and the library used for Learning is* ***Pytorch****. Apart from implementation, we have done may experiments on the hyper-parameters such as scale, blur radius, number of layers, number of kernels per layer etc.*

This project also includes the implementation of ***Image super-resolution via sparse representation*** *by Yang J. et al. which is a non deep-learning approach for Image super-resolution. The project also compares our SRCNN and Sparse Coding implementaion results with several other Classical Methods as well as Bayesian Model results.*

1. Introduction

Single image super-resolution (SR), which aims at recovering a high-resolution image from a single low-resolution image, is a classical problem in computer vision. This problem is inherently ill-posed since a multiplicity of solutions exist for any given low-resolution pixel. In other words, it is an under determined inverse problem, of which solution is not unique. Such a problem is typically mitigated by constraining the solution space by strong prior information. To learn the prior, recent state-of-the-art methods mostly adopt the example-based strategy. These methods either exploit internal similarities of the same image, or learn mapping functions from external low and high-resolution exemplar pairs. The external example-based methods can be formulated for generic image super-resolution, or can be designed to suit domain specific tasks, i.e., face hallucination, according to the training samples provided.

The sparse-coding-based method is one of the representative external example-based SR methods. This method involves several steps in its solution pipeline. First, overlapping patches are densely cropped from the input image and pre-processed (e.g.,subtracting mean and normalization). These patches are then encoded by a low-resolution dictionary. The sparse coefficients are passed into a high-resolution dictionary for reconstructing high-resolution patches. The overlapping re-constructed patches are aggregated (e.g., by weighted averaging) to produce the final output. This pipeline is shared by most external example-based methods, which pay particular attention to learning and optimizing the dictionaries or building efficient mapping functions. However, the rest of the steps in the pipeline have been rarely optimized or considered in an unified optimization framework.

1.1. SRCNN Formulation

Formulation Consider a single low-resolution image, we first upscale it to the desired size using bicubic interpolation, which is the only pre-processing we perform 3 . Let us denote the interpolated image as Y. Our goal is to recover from Y an image F(Y) that is as similar as possible to the ground truth high-resolution image X. For the ease of presentation, we still call Y a “low-resolution” image, although it has the same size as X. We wish to learn a mapping F, which conceptually consists of three operations:

1) Patch extraction and representation: this operation extracts (overlapping) patches from the low- resolution image Y and represents each patch as a high-dimensional vector. These vectors comprise a set of feature maps, of which the number equals to the dimensionality of the vectors.

2) Non-linear mapping: this operation nonlinearly maps each high-dimensional vector onto another high-dimensional vector. Each mapped vector is conceptually the representation of a high-resolution patch. These vectors comprise another set of feature maps.

3) Reconstruction: this operation aggregates the above high-resolution patch-wise representations to generate the final high-resolution image. This image is expected to be similar to the ground truth X.

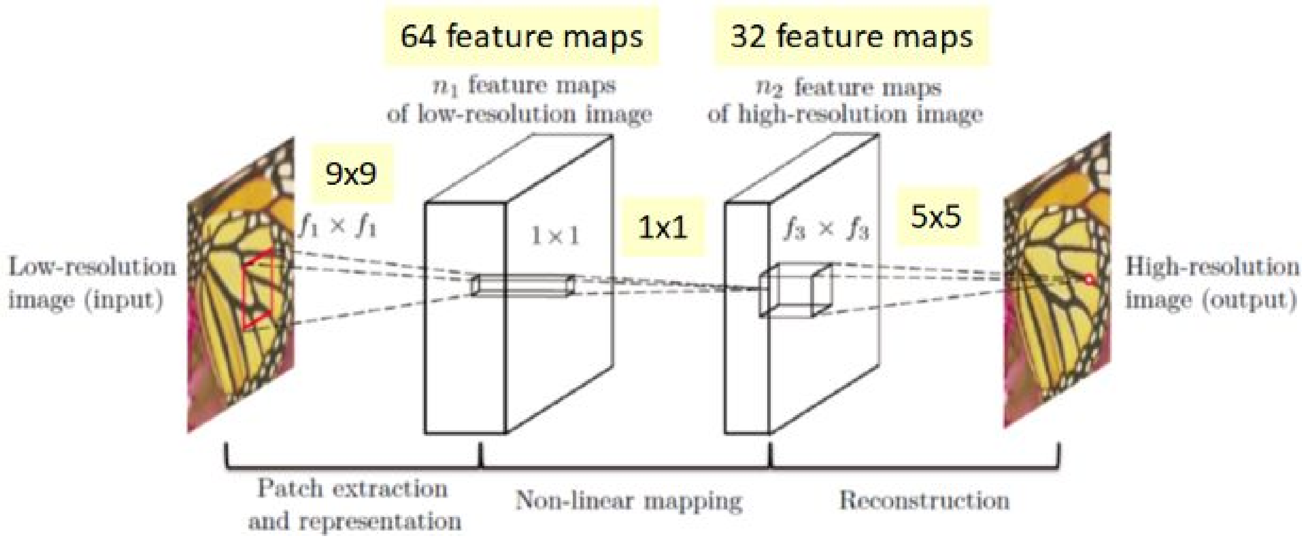

Figure 1. Given a low-resolution image Y, the first convolutional layer of the SRCNN extracts a set of feature maps. The second layer maps these feature maps nonlinearly to high-resolution patch representations. The last layer combines the predictions within a spatial neighbourhood to produce the final high-resolution image F(Y)

1.2. Sparse coding formulation

1) Basic Ideas: To be more precise, let $D \in R^n \times K$ be an overcomplete dictionary of K atoms $(K < n)$, and suppose a signal $x \in R^n$ can be represented as a sparse linear combination with respect to D. That is, the signal x can be written as $x = D\ \alpha_0$ where where $\alpha_0 \in R^K$ is a vector with very few $(<< n)$ nonzero entries. In practice, we might only observe a small set of measurements y of x:

$$y = Lx = LD\alpha_0$$

where $L \in \mathbb{R}^{k \times n}\ with\ k\ \ n$ is a projection matrix.

2) Reconstruction constraint: The observed low-resolution image Y is a blurred and downsampled version of the high resolution image X:

$$Y = SHX$$

Here, H represents a blurring filter, and S the down sampling operator.

3) Sparsity prior: The patches x of the high-resolution image X can be represented as a sparse linear combination in a dictionary D_h trained from high-resolution patches sampled from training images:

$$x \approx D_h \alpha$$

for some $\alpha\ \epsilon\ R^K\ with\ ||\alpha_0||\ <<\ K$

4) Ideal formulation For each input low-resolution patch y, we find a sparse representation with respect to D_l. The corresponding high- resolution patch bases D h will be combined according to these coefficients to generate the output high-resolution patch x. The problem of finding the sparsest representation of y can be formulated as:

$$min\ ||\alpha_0||$$

subject to

$$||FD_l\ \alpha\ -\ Fy||_2^2\ \leq \epsilon$$

5) Final formulation

$$min\ ||\alpha||_1$$

subject to

$$||FD_l\alpha - Fy||_2^2 \leq \epsilon_1$$

$$||PD_h\alpha - w||_2^2 \leq \epsilon_2$$

1.3. Relation between SRCNN and SC

In the sparse-coding-based methods, let us consider that an f1 × f1 low-resolution patch is extracted from the input image. Then the sparse coding solver, like Feature-Sign, will first project the patch onto a (low-resolution) dictionary. If the dictionary size is n_1 , this is equivalent to applying n_1 linear filters (f1 × f1) on the input image (the mean subtraction is also a linear operation so can be absorbed)

The sparse coding solver will then iteratively process the n_1 coefficients. The outputs of this solver are n_2 coefficients, and usually $n_2 = n_1$ in the case of sparse coding.

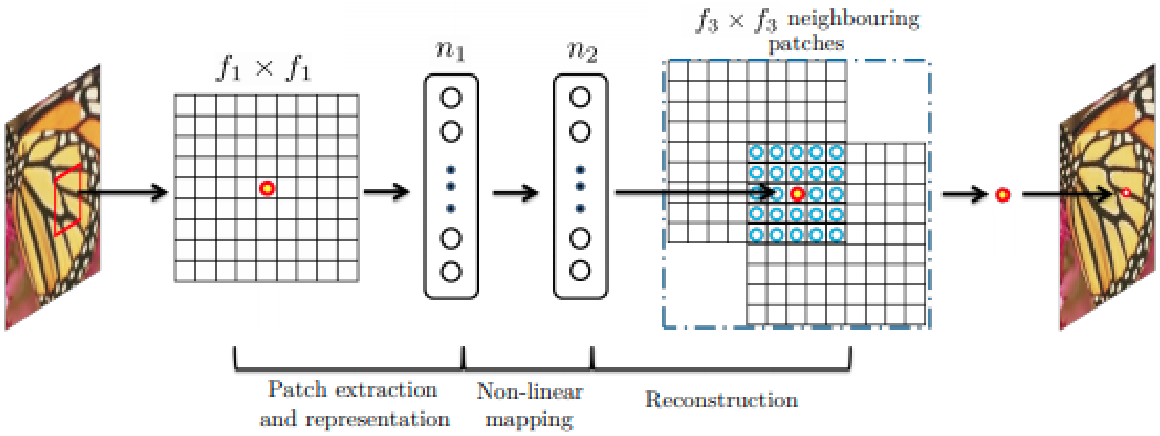

Figure 2. An illustration of sparse-coding-based methods in the view of a convolutional neural network. This can be seen as the link between SC and SRCNN.

These n_2 coefficients are the representation of the high-resolution patch. In this sense, the sparse coding solver behaves as a special case of a non-linear mapping operator, whose spatial support is 1×1. However, the sparse coding solver is not feed-forward, i.e.,it is an iterative algorithm. On the contrary, our non-linear operator is fully feed-forward and can be computed efficiently. If we set f2 = 1, then our non-linear operator can be considered as a pixel-wise fully-connected layer. It is worth noting that "the sparse coding solver" in SRCNN refers to the first two layers, but not just the second layer or the activation function (ReLU). Thus the nonlinear operation in SRCNN is also well optimized through the learning process. The above n_2 coefficients (after sparse coding) are then projected onto another (high-resolution) dictionary to produce a high-resolution patch. The overlapping high-resolution patches are then averaged. As discussed above, this is equivalent to linear convolutions on the n_2 feature maps. If the high-resolution patches used for reconstruction are of size f3 × f3 , then the linear filters have an equivalent spatial support of size f3 × f3.

2. Experiments

We perform various experiments here. These include varying time, scale, hyperparameters of neural network like network size, filter size, etc.

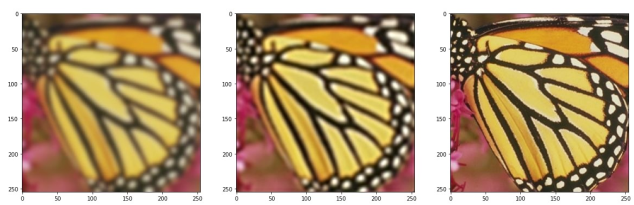

Figure 3. Test image

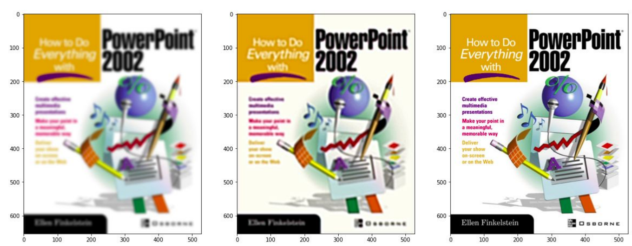

Figure 4. Test image

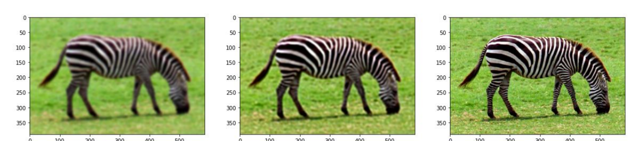

Figure 5. Test image

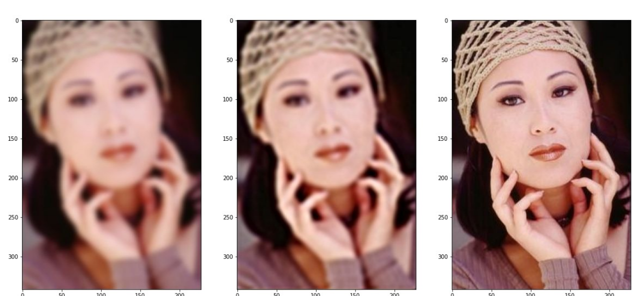

Figure 6. Test image

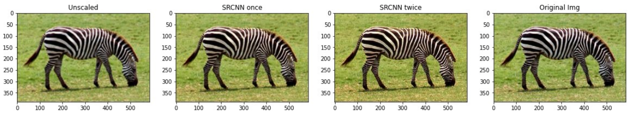

Figure 7. Test image

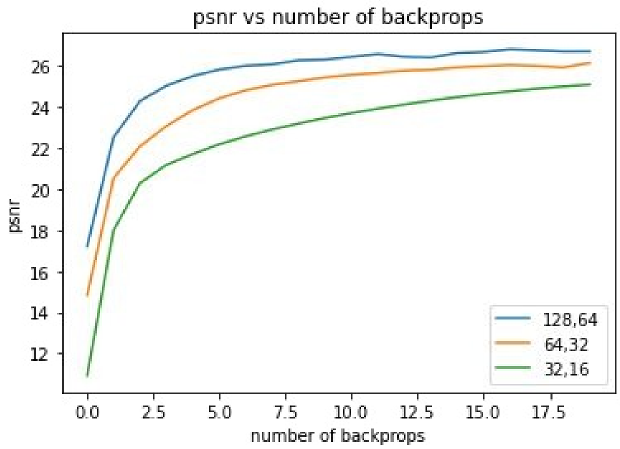

Figure 8. Evaluation

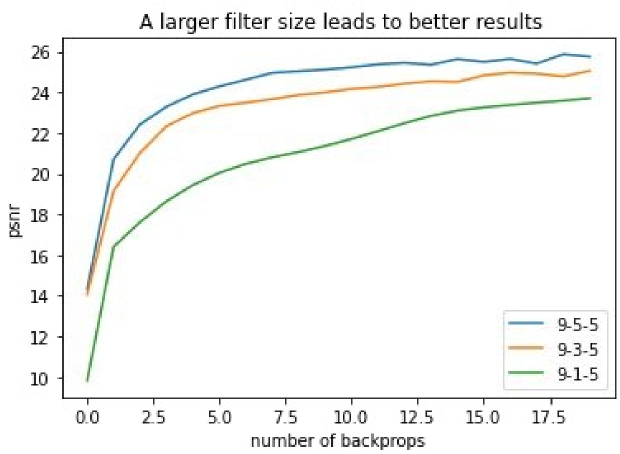

Figure 9. Evaluation

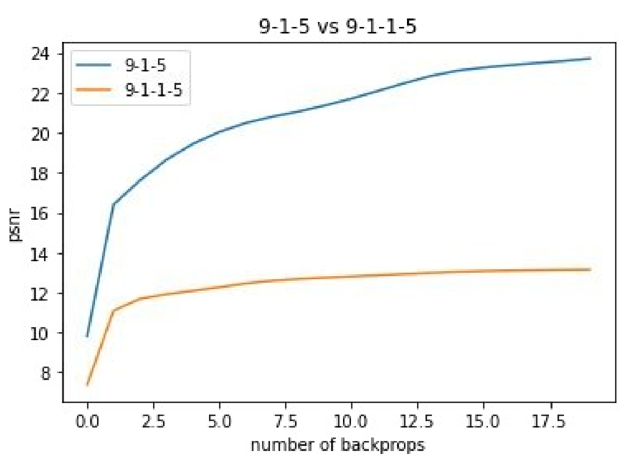

Figure 10. Evaluation

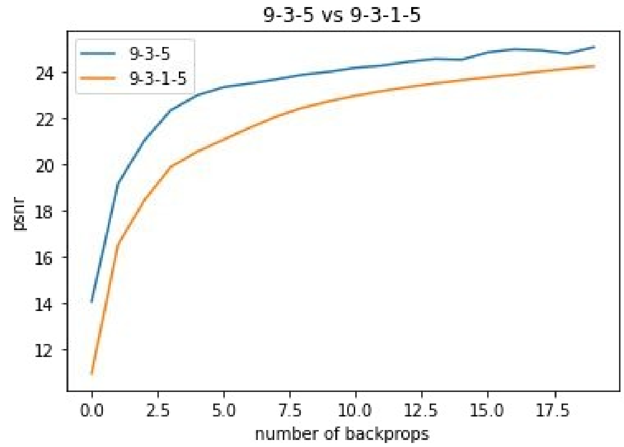

Figure 11. Evaluation

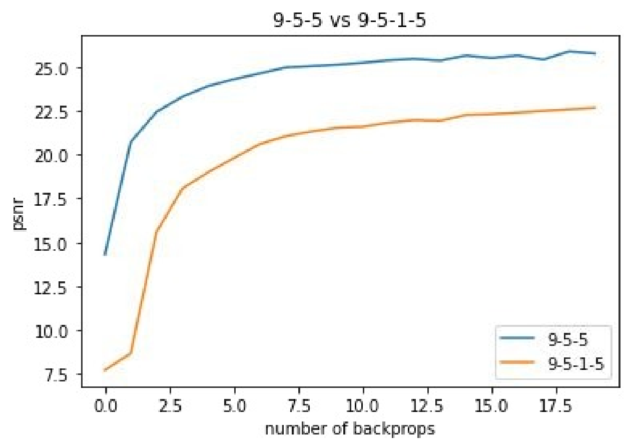

Figure 12. Evaluation

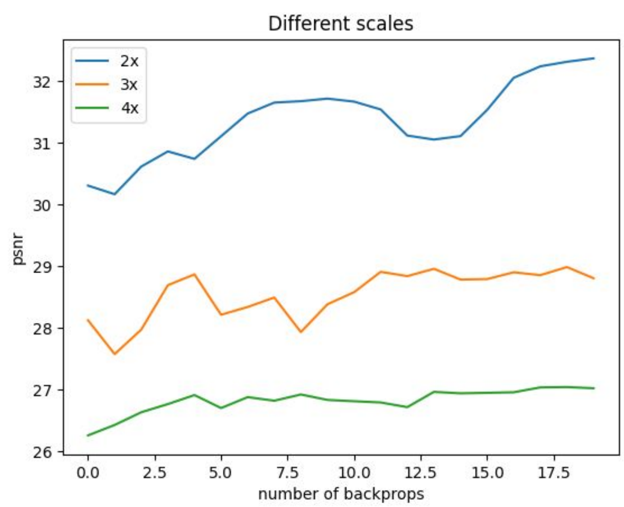

Figure 13. Evaluation

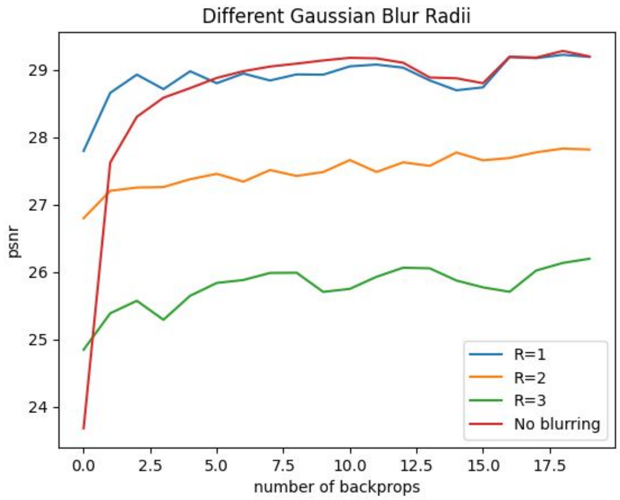

Figure 14. Evaluation

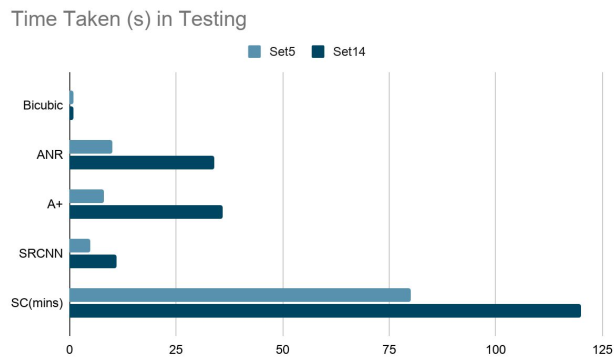

Figure 17. Evaluation

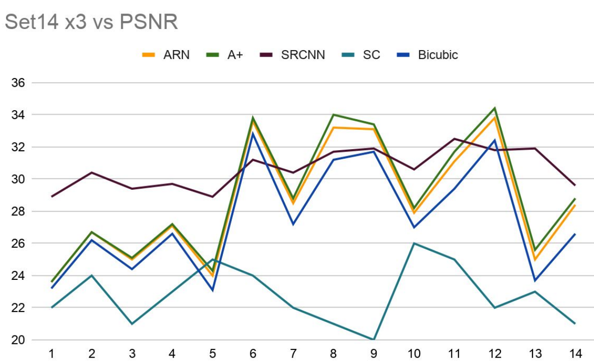

Figure 15. Evaluation

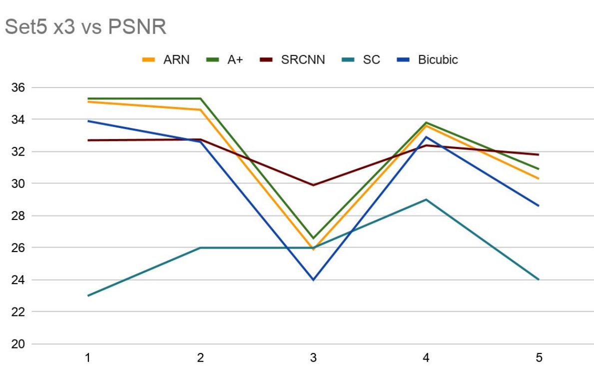

Figure 16. Evaluation

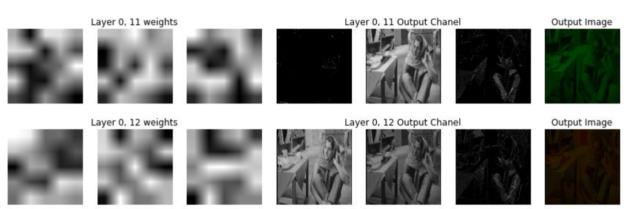

Figure 18. Evaluation