Image Super-Resolution using Deep Convolutional Neural Networks

1st Kushagra Mishra
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Mumbai, India
200010042@iitb.ac.in

2nd Advait Mehla
Department of Physics
Indian Institute of Technology, Bombay
Mumbai, India
advaitmehla@iitb.ac.in

3rd Advait Risbud
Department of Materials Science
Indian Institute of Technology, Bombay
Mumbai, India
advaitrisbud@gmail.com

***Abstract*—A deep convolutional neural network architecture is used to super-resolve images. This technique used neural networks to learn the difference between images of high and poor resolution, the network can then be applied to new images of poor quality to super-resolve them. The primary metrics used for gauging image quality are the peak signal-to-noise ratio, mean-squared error, and structural similarity index.**

***Index Terms*—Neural networks, image processing, super-resolution.**

I. Introduction

Why improve the resolution of images? Images with better resolution essentially give us better quality and more detail. This can be helpful in various fields such as forensic analysis, microstructural analysis of materials, art restoration etc. A natural solution here would be to use better cameras. This is not as straightforward a solution as it seems because the cost of any camera installation scales with the number of cameras used and their quality. Take, for example, a CCTV setup in a large corporate headquarters. In such a case, there may be thousands of cameras required, and high-quality recording in each of them would lead to a large capital expenditure and a large operating cost in the form of storage space for high-resolution video.

An innovative solution to this problem is image super-resolution, a long-standing computer vision problem. This problem is ill-posed as multiple possible super-resolved images exist for a given low-resolution image. A CNN is used to learn a complete mapping between low and high-resolution images with little need for pre-processing.

II. Convolutional Neural Networks

A. Convolution

A convolution is a mathematical operation between two functions and is analytically represented as follows:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau. \tag{1}$$

To understand how this extends to matrices, let us consider the discrete convolution first.

$$x[n] * h[n] = \sum_{-\infty}^{\infty} x[k] \cdot h[n-k] \tag{2}$$

This can be extended to 2-D matrices. The expression for the extracted matrix is:

$$y[m,n] = x[m,n]*h[m,n] = \sum_{j=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} x[i,j]\cdot h[m-i,n-j] \tag{3}$$

We choose an appropriate kernel matrix which we convolve with a given input matrix, this extract features from the input matrix based on the values inside the kernel. An image can also be represented as a matrix. For black and white images each entry in the matrix ranges from 0 to 255 for an 8-bit depth (some image formats such as DICOM and some medical image storage formats, use 16-bit depth for greater accuracy). Different kernels can extract different types of information based on their entries called weights. Some kernels can find the whether there is an edge at a particular location or if there is a certain kind of structural feature in a given space.

How do we decide the weights? Perfectly chosen weights will always be able to identify whether a certain feature is present in a particular image or not. We can train our model by comparing the output and the target at each iteration. Thus optimizing the weights of our kernel.

B. Comparing images

A wide variety of metrics are present for comparing the difference between images We will discuss three of these: (1) mean square error, (2) structural similarity index, and (3) peak signal-to-noise ratio. Let us discuss each of them in turn.

a) Mean-square error: In the context of scalars this is given by,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2. \tag{4}$$

Here t_i is the target value, y_i is the predicted value, and n is the number of samples. We want to reduce the distance between the prediction and the target, hence our aim is to minimise the MSE. In the context of matrices, we modify the above equation but only slightly.

$$\text{MSE} = \frac{1}{nm}\sum_{i=1}^{m}\sum_{j=1}^{n}(t[i,j] - y[i,j])^2 \tag{5}$$

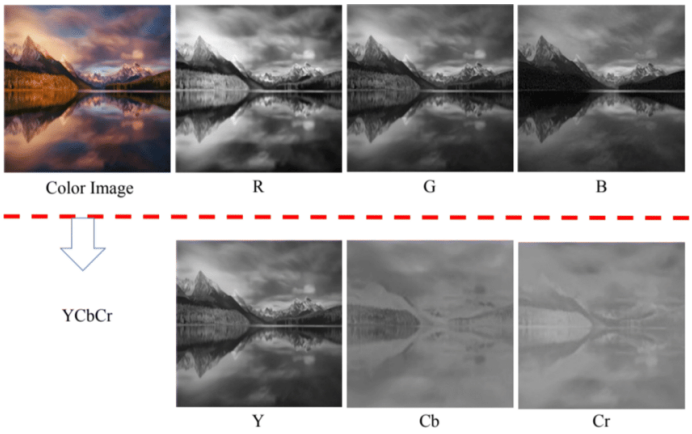

Fig. 1. An example illustrating the importance of the Y channel, showing how all the details is concentrated in it

Once again, $t[i,j]$ and $y[i,j]$ are the target and prediction, respectively.

b) Structural-similarity index:

$$\text{SSIM(x,y)} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{6}$$

Here:

- μ_x, μ_y are the sample mean of images x and y respectively.
- σ_x, σ_y, σ_{xy} are the variances of standard deviation of x, y, and the covariance of x and y.
- c_1 and c_2 are variables used to stabilise the weak denominator. They only depend on the number of bits per pixel.

c) Peak signal-to-noise ratio::

$$\text{PSNR} = 20 \cdot \log_{10}\left(\frac{2^B - 1}{\sqrt{\text{MSE}}}\right) \tag{7}$$

Here B is the bit-depth of the image, and MSE is the mean square error.

C. Colour spaces

In the case of multi-channel images, like RGB, the images are converted to the YCbCr colour space. It consists of three colour components: Y (luma or brightness), Cb (blue-difference chroma), and Cr (red-difference chroma). The Y channel represents the brightness or luminance of the image and is often used alone in image processing tasks. This is because human vision is more sensitive to changes in brightness than changes in colour. By separating the brightness information from the colour information, we can perform operations such as brightness adjustments, contrast enhancements, and image thresholding without affecting the colour information.

For super-resolution tasks as well, it is common to isolate the Y channel and only operate on it as it contains most of the detail and structural information of the image. Moreover, it is less susceptible to noise and artefacts like chroma noise, chromatic aberrations, demosaicking artefacts etc. It is thus useful to enhance the Y channel and combine it with the original Cb and Cr channels to restore a higher-resolution version of the original image with the colours intact.

III. CNNs for Super-Resolution

We start with a single low-resolution image and resize it to the desired size using linear interpolation. This is the only pre-processing performed. Let this interpolated image be denoted by $\mathbf{Y}$. The low-resolution image we started with is in turn a downscaled version of the ground truth, $\mathbf{X}$. The goal of the SRCNN is to recover from $\mathbf{Y}$ an image $F(\mathbf{Y})$, which is as close as possible to the original $\mathbf{X}$. Note that $\mathbf{Y}$ is called the 'low-resolution' image, even though it's technically the same size as $\mathbf{X}$. The mapping F can be boiled down to three sequential steps, as explained below.

1) **Patch extraction:** This is the process of extracting overlapping patches from $\mathbf{Y}$ and representing each as a higher dimensional vector. These vectors essentially form feature maps, and their dimensionality equals the number of maps.
2) **Non-linear mapping:** This operation involves nonlinear mapping of each of the higher dimensional vectors to another set of vectors. Each vector is a representation of a high-resolution patch of the image.
3) **Reconstruction:** This operation combines the above patch-wise representation of the image to generate the final high-resolution image that is intended to be similar to the ground truth.

We will now show that these operations can be expressed as a CNN as depicted in Figure 2.

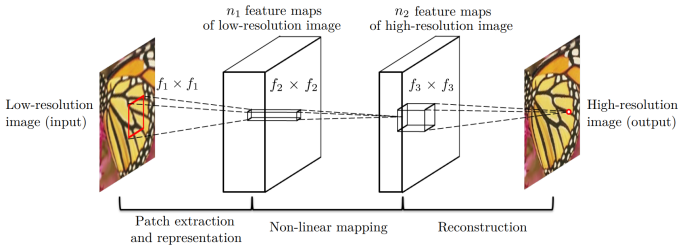

Fig. 2. The SRCNN's initial convolutional layer takes a low-resolution image Y, and generates a collection of feature maps. The subsequent layer then applies a non-linear mapping to these feature maps, resulting in high-resolution patch representations. Finally, the last layer combines these representations in a spatially local neighbourhood to produce the final high-resolution image $F(\mathbf{Y})$. Figure credits [1]

A. Patch Extraction

In this method, the image is split into patches, and each patch is convolved with a set of filters which form a basis (PCA, Haar etc.). The optimization of these bases is involved in the optimization of the network. The first layer can thus be represented by the relation

$$F_1(\mathbf{Y}) = \text{ReLu}\left(W_1 * \mathbf{Y} + B_1\right)$$

where W_1 and B_1 represent the filters and biases respectively. W_1 consists of n_1 filters that are of size $c \times f_1 \times f_1$,

where c represents the number of channels in the input image and f_1 represents the input dimensions. Therefore, the image undergoes n_1 convolutions using kernels of size $c \times f_1 \times f_1$, which results in n_1 feature maps in the output. The biases, denoted by B_1, form an n_1 dimensional vector with each element corresponding to one of the filter kernels. The activation function used is the Rectified Linear Unit (ReLu) on the filter responses. Consequently, this layer generates an n_1 dimensional feature for every patch of the input image.

B. Non-linear mapping

The next layer takes the n_1 feature vectors and maps them to n_2 outputs. This is achieved by convolving the filters over patches of the feature vectors instead of the original image. Hence, the operation of the second layer can be described as follows:

$$F_2(\mathbf{Y}) = \text{ReLu}\left(W_2 * F_1(\mathbf{Y}) + B_2\right)$$

W_2 is composed of n_2 filters that have a size of $n_1 \times f_2 \times f_2$, and B_2 is an n_2 dimensional vector. Each of these vectors represents a high resolution patch, which will be reconstructed from these vectors. If a higher degree of non-linearity is required, deeper networks can also be employed.

C. Reconstruction

Traditional methods typically involve predicting overlapping high-resolution patches, which are then averaged to obtain the complete image. This averaging process can be viewed as a predetermined filter that operates on a collection of feature maps, where each position corresponds to the flattened vector representation of a high-resolution patch. With this in mind, we introduce a convolutional layer that generates the desired high-resolution image:

$$F_1(\mathbf{Y}) = \text{ReLu}\left(W_1 * \mathbf{Y} + B_1\right)$$

The layer represented by W_3 comprises c filters that are of size $n_2 \times f_3 \times f_3$, and B_3 is a c-dimensional vector. When the high-resolution patch representations exist in the image domain (i.e., when we can reshape each representation to create the patch), we anticipate that the filters would function as an averaging filter. Conversely, if the representations of the high-resolution patches exist in a different domain (e.g., as coefficients in some basis), we expect that W_3 would project these coefficients onto the image domain before carrying out the averaging. In either case, W_3 is a collection of linear filters.

As can be seen, all three of these operations can be compactly expressed as a convolutional layer. The three of these can be combined to form a convolutional neural network. Through the training process, all the weights and biases are to be optimized in order to achieve the required goal of image super-resolution.

IV. Training

The MSE is used as the loss for this model, as its minimisation should result in a high PSNR. Although the training is done with the PSNR in mind the output images perform well even when evaluated by SSIM. Stochastic gradient descent is used to minimise the loss. Weights and biases for each layer are calculated using standard backpropagation. The learning rates for each layer are different to ensure that the loss minimisation converges. MSE is a good loss function as it is convex. Other perceptually motivated loss functions may also be used.

V. Results

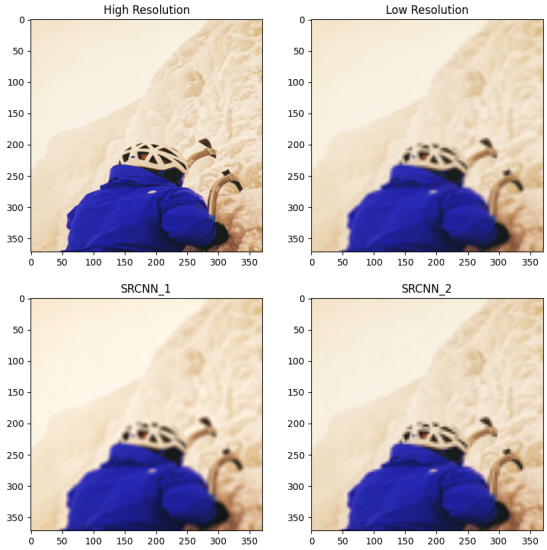

Fig. 3. The SRCNN applied to a sample image. SRCNN_1 corresponds to the weights we trained, while SRCNN_2 uses pretrained weights from [4]

Model	PSNR	MSE	SSIM
SRCNN_1	25.344	189.93	0.67106
SRCNN_2	25.478	184.19	0.68959

TABLE I
METRICS FOR SUPER-RESOLUTION USING THE TRAINED, AND PRE-TRAINED WEIGHTS.

References

[1] C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 2, pp. 295-307, 1 Feb. 2016

[2] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004

[3] A. Horé and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," 2010 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010, pp. 2366-2369

[4] https://github.com/MarkPrecursor/SRCNN-keras