A Seminar Report on

Image Super-Resolution Using Deep Convolutional Networks

undergone at

National Institute of Technology Karnataka Surathkal

under the guidance of

Ms. Mariet P. Furtado
Mrs. Uma Priya

Submitted by

Bairi Sandhya Rani
14CO205
VII Sem B.Tech (CSE)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER ENGINEERING

Department of Computer Science & Engineering Technology
National Institute of Technology Karnataka, Surathkal.
March 2017

ABSTRACT

This report deals with a deep learning method for single image super-resolution (SR). Our method directly learns an end-to-end mapping between the low/high-resolution images. The mapping is represented as a deep convolutional neural network (CNN) that takes the low-resolution image as the input and outputs the high-resolution one. Traditional sparse-coding-based SR methods can also be viewed as a deep convolutional network. But unlike traditional methods that handle each component separately, deep CNN method jointly optimizes all layers. This deep CNN has a lightweight structure, yet demonstrates state-of-the-art restoration quality, and achieves fast speed for practical on-line usage. Different network structures and parameter settings are explored to achieve trade-offs between performance and speed. This also includes extension of the network to cope with three color channels simultaneously, and show better overall reconstruction quality.

INDEX

1. TECHNOLOGIES USED

This is primarily implemented in MATLAB using the concepts of Deep Learning and Neural Networks. It can also be implemented using Caffe and Cuda to observe the similar performance.

2. WORK DONE

This method of implementation involves a fully feed forward convolutional neural network.

Given a single low-resolution image as input, we first upscale it to the desired size using bicubic interpolation, which is the only pre-processing that is performed. Let the interpolated image be Y. Output must be an image F(Y) that is as similar as possible to the ground truth high resolution image X. For clarity, Y is considered a “low-resolution” image.

The mapping F 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.

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

Where W1 and B1 represent the filters and biases respectively, and ‘*’ denotes the convolution operation. Here, W1 corresponds to n1 filters of support c X f1 X f1, where c is the number of channels in the input image, f1 is the spatial size of a filter. Intuitively, W1 applies n1 convolutions on the image, and each convolution has a kernel size c X f1 X f1. The output is composed of n1 feature maps. B1 is an n1-dimensional vector, whose each element is associated with a filter. We apply the ReLU (max (0; x)) on the filter responses.

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.

$$F_2(\mathbf{Y}) = \max(0, W_2 * F_1(\mathbf{Y}) + B_2).$$

Here W2 contains n2 filters of size n1 X f2 X f2, and B2 is n2-dimensional. Each of the output n2-dimensional vectors is conceptually a representation of a high-resolution patch that will be used for reconstruction.
It is possible to add more convolutional layers to increase the non-linearity. But this can increase the complexity of the model (n2 X f2 X f2 X n2 parameters for one layer), and thus demands more training time.

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.

$$F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3.$$

Here W3 corresponds to c filters of a size n2 X f3 X f3, and B3 is a c-dimensional vector.

Training

Learning the end-to-end mapping function F requires the estimation of network parameters Θ = {W1, W2, W3, B1, B2, B3}. This is achieved through minimizing the loss between the reconstructed images F(Y; Θ) and the corresponding ground truth high-resolution images X. Given a set of high-resolution images {Xi} and their corresponding low-resolution images {Yi}, we use mean squared error (MSE) as the loss function:

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i; \Theta) - \mathbf{X}_i||^2,$$

Where n is the number of training samples. Using MSE as the loss function favors a high PSNR.

A relatively small training set that consists of 91 images, and a large training set that consists of 395,909 images from the ILSVRC 2013 ImageNet detection training partition are used for training. The size of training sub-images is f_{sub} = 33. Thus the 91-image dataset can be decomposed into 24,800 sub-images, which are extracted from original images with a stride of 14. Whereas the ImageNet provides over 5 million sub-images even using a stride of 33. We use the basic network settings, i.e., f1 = 9, f2 = 1, f3 = 5, n1 = 64, and n2 = 32. We use the Set5 as the validation set. Similar trend is observed if we use the larger Set14 set. The upscaling factor is 3.

Training	- on ImageNet
Evaluation	- on the Set5, Set14 dataset.
Model/9-1-5(91 images)	- model parameters of network 9-1-5 trained on 91 images.
Model parameters	- "x2.mat", "x3.mat" and "x4.mat" used for upscaling.
Upscaling factors	- 2, 3 and 4.
Model/9-1-5(ImageNet)	- model parameters of network 9-1-5 trained on ImageNet
Model/9-3-5(ImageNet)	- model parameters of network 9-3-5 trained on ImageNet
Model/9-5-5(ImageNet)	- model parameters of network 9-5-5 trained on ImageNet

3. CODE

In the code, there are mainly five .m files.

demo_SR.m: This is to show the image super resolution and is the main code.

```
close all;
clear all;

%% read ground truth image
```

```
im  = imread('Set5\head_GT.bmp');
%im  = imread('Set14\zebra.bmp');

%% set parameters
up_scale = 3;
model = 'model\9-5-5(ImageNet)\x3.mat';
% up_scale = 3;
% model = 'model\9-3-5(ImageNet)\x3.mat';
% up_scale = 3;
% model = 'model\9-1-5(91 images)\x3.mat';
% up_scale = 2;
% model = 'model\9-5-5(ImageNet)\x2.mat';
% up_scale = 4;
% model = 'model\9-5-5(ImageNet)\x4.mat';

%% work on illuminance only
if size(im,3)>1
    im = rgb2ycbcr(im);
    im = im(:, :, 1);
end
im_gnd = modcrop(im, up_scale);
im_gnd = single(im_gnd)/255;

%% bicubic interpolation
im_l = imresize(im_gnd, 1/up_scale, 'bicubic');
im_b = imresize(im_l, up_scale, 'bicubic');

%% SRCNN
im_h = SRCNN(model, im_b);

%% remove border
im_h = shave(uint8(im_h * 255), [up_scale, up_scale]);
im_gnd = shave(uint8(im_gnd * 255), [up_scale, up_scale]);
im_b = shave(uint8(im_b * 255), [up_scale, up_scale]);

%% compute PSNR
psnr_bic = compute_psnr(im_gnd,im_b);
psnr_srcnn = compute_psnr(im_gnd,im_h);

%% show results
fprintf('PSNR for Bicubic Interpolation: %f dB\n', psnr_bic);
fprintf('PSNR for SRCNN Reconstruction: %f dB\n', psnr_srcnn);

figure, imshow(im_b); title('Bicubic Interpolation');
figure, imshow(im_h); title('SRCNN Reconstruction');

%imwrite(im_b, ['Bicubic Interpolation' '.bmp']);
%imwrite(im_h, ['SRCNN Reconstruction' '.bmp']);
```

SRCNN.m: demo_SR.m uses this to construct the neural network with three convolutional layers. SRCNN.m realizes super resolution given the model parameters.

```
function im_h = SRCNN(model, im_b)

%% load CNN model parameters
load(model);
[conv1_patchsize2,conv1_filters] = size(weights_conv1);
conv1_patchsize = sqrt(conv1_patchsize2);
[conv2_channels,conv2_patchsize2,conv2_filters] =
size(weights_conv2);
conv2_patchsize = sqrt(conv2_patchsize2);
[conv3_channels,conv3_patchsize2] = size(weights_conv3);
conv3_patchsize = sqrt(conv3_patchsize2);
[hei, wid] = size(im_b);

%% conv1
weights_conv1 = reshape(weights_conv1, conv1_patchsize,
conv1_patchsize, conv1_filters);
conv1_data = zeros(hei, wid, conv1_filters);
for i = 1 : conv1_filters
    conv1_data(:,:,i) = imfilter(im_b, weights_conv1(:,:,i), 'same',
'replicate');
    conv1_data(:,:,i) = max(conv1_data(:,:,i) + biases_conv1(i), 0);
end

%% conv2
conv2_data = zeros(hei, wid, conv2_filters);
for i = 1 : conv2_filters
    for j = 1 : conv2_channels
        conv2_subfilter = reshape(weights_conv2(j,:,i), conv2_patchsize,
conv2_patchsize);
        conv2_data(:,:,i) = conv2_data(:,:,i) + imfilter(conv1_data(:,:,j),
conv2_subfilter, 'same', 'replicate');
    end
    conv2_data(:,:,i) = max(conv2_data(:,:,i) + biases_conv2(i), 0);
end

%% conv3
conv3_data = zeros(hei, wid);
for i = 1 : conv3_channels
    conv3_subfilter = reshape(weights_conv3(i,:), conv3_patchsize,
conv3_patchsize);
```

```
    conv3_data(:,:) = conv3_data(:,:) + imfilter(conv2_data(:,:,i),
conv3_subfilter, 'same', 'replicate');
end

%% SRCNN reconstruction
im_h = conv3_data(:,:) + biases_conv3;
```

shave.m: This is used for removing borders.

```
function I = shave(I, border)
I = I(1+border(1):end-border(1), ...
      1+border(2):end-border(2), :, :);
```

compute_psnr.m: This is to compute the value of Peak signal to noise ratio for different cases.

```
function psnr=compute_psnr(im1,im2)
if size(im1, 3) == 3,
    im1 = rgb2ycbcr(im1);
    im1 = im1(:, :, 1);
end

if size(im2, 3) == 3,
    im2 = rgb2ycbcr(im2);
    im2 = im2(:, :, 1);
end

imdff = double(im1) - double(im2);
imdff = imdff(:);

rmse = sqrt(mean(imdff.^2));
psnr = 20*log10(255/rmse);
```

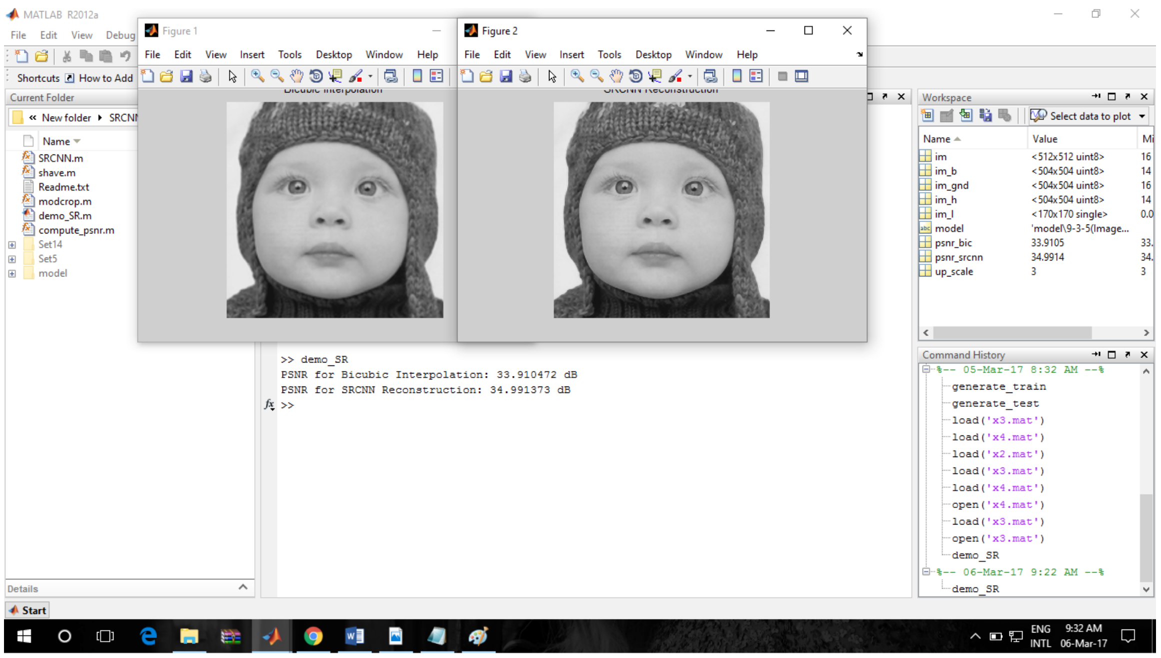
modcrop.m: This is used for upscaling.

```
function imgs = modcrop(imgs, modulo)
if size(imgs,3)==1
    sz = size(imgs);
    sz = sz - mod(sz, modulo);
    imgs = imgs(1:sz(1), 1:sz(2));
else
    tmpsz = size(imgs);
    sz = tmpsz(1:2);
    sz = sz - mod(sz, modulo);
    imgs = imgs(1:sz(1), 1:sz(2),:);
end
```

4. RESULTS AND SNAPSHOTS

For different images given as inputs, the following are the results.

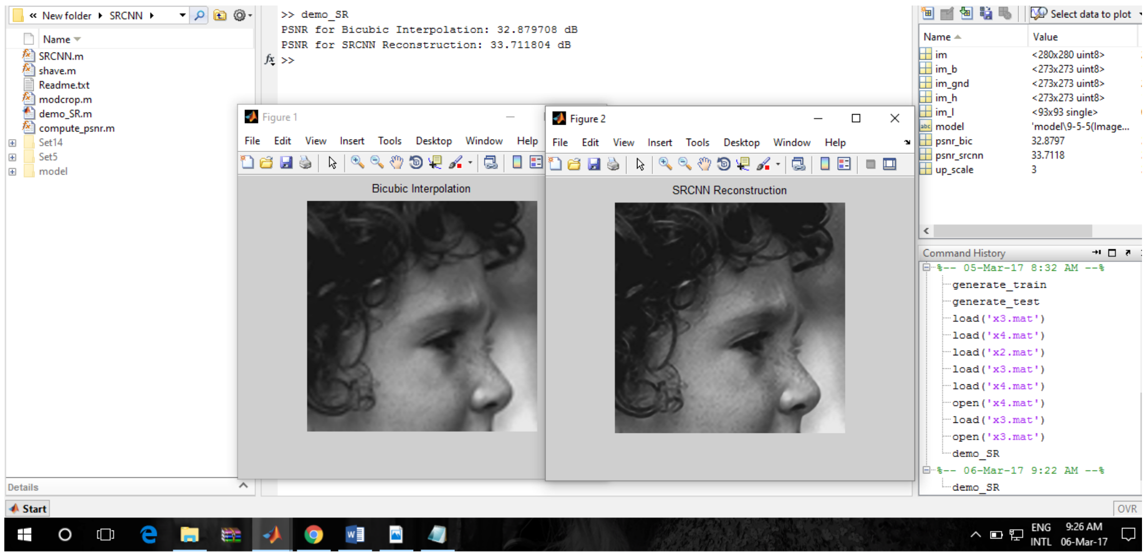
```
>> load('x3.mat')
>> open('x3.mat')

ans =

    weights_conv1: [81x64 double]
     biases_conv1: [64x1 double]
     biases_conv2: [32x1 double]
    weights_conv2: [64x9x32 double]
    weights_conv3: [32x25 double]
     biases_conv3: 0.0013
```



```
>> demo_SR
PSNR for Bicubic Interpolation: 33.910472 dB
PSNR for SRCNN Reconstruction: 34.991373 dB
>>
```

Name	Value	Mi
im	<512x512 uint8>	16
im_b	<504x504 uint8>	14
im_gnd	<504x504 uint8>	16
im_h	<504x504 uint8>	14
im_l	<170x170 single>	0.0
model	'model\9-3-5(Image...	
psnr_bic	33.9105	33.
psnr_srcnn	34.9914	34.
up_scale	3	3

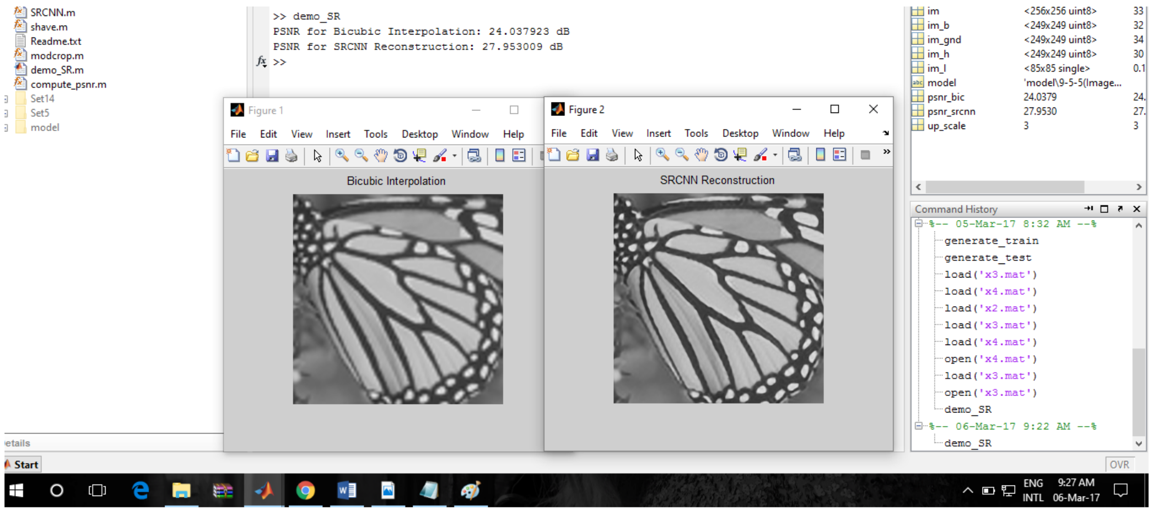
```
%-- 05-Mar-17 8:32 AM --%
generate_train
generate_test
load('x3.mat')
load('x4.mat')
load('x2.mat')
load('x3.mat')
load('x4.mat')
open('x4.mat')
load('x3.mat')
open('x3.mat')
demo_SR
%-- 06-Mar-17 9:22 AM --%
demo_SR
```



```
>> demo_SR
PSNR for Bicubic Interpolation: 32.879708 dB
PSNR for SRCNN Reconstruction: 33.711804 dB
>>
```

Name	Value
im	<280x280 uint8>
im_b	<273x273 uint8>
im_gnd	<273x273 uint8>
im_h	<273x273 uint8>
im_l	<93x93 single>
model	'model\9-5-5(Image...
psnr_bic	32.8797
psnr_srcnn	33.7118
up_scale	3

```
%-- 05-Mar-17 8:32 AM --%
generate_train
generate_test
load('x3.mat')
load('x4.mat')
load('x2.mat')
load('x3.mat')
load('x4.mat')
open('x4.mat')
load('x3.mat')
open('x3.mat')
demo_SR
%-- 06-Mar-17 9:22 AM --%
demo_SR
```



```
>> demo_SR
PSNR for Bicubic Interpolation: 24.037923 dB
PSNR for SRCNN Reconstruction: 27.953009 dB
>>
```

im	<256x256 uint8>	33
im_b	<249x249 uint8>	32
im_gnd	<249x249 uint8>	34
im_h	<249x249 uint8>	30
im_l	<85x85 single>	0.1
model	'model\9-5-5(Image...	
psnr_bic	24.0379	24.
psnr_srcnn	27.9530	27.
up_scale	3	3

```
%-- 05-Mar-17 8:32 AM --%
generate_train
generate_test
load('x3.mat')
load('x4.mat')
load('x2.mat')
load('x3.mat')
load('x4.mat')
open('x4.mat')
load('x3.mat')
open('x3.mat')
demo_SR
%-- 06-Mar-17 9:22 AM --%
demo_SR
```

5. CONCLUSION

This report explains a novel deep learning approach for single image super resolution. Conventional sparse-coding based SR methods can be reformulated into a deep convolutional neural network. The proposed approach, SRCNN, learns an end-to-end mapping between low- and high-resolution images, with little extra pre/post-processing beyond the optimization. With a lightweight structure, the SRCNN has achieved superior performance than the state-of-the-art methods. It is a conjecture that additional performance can be further gained by exploring more filters and different training strategies. Besides, the proposed structure, with its advantages of simplicity and robustness, could be applied to other low-level vision problems, such as image deblurring or simultaneous SR+ denoising. One could also investigate a network to cope with different upscaling factors.

6. REFERENCES

[1] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, “Deep network cascade for image super-resolution,” in Proc. Eur. Conf. Comput. Vis., 2014.

[2] “Image Super-Resolution Using Deep Convolutional Networks” Chao Dong, Chen Change Loy, Member, IEEE, Kaiming He, Member, IEEE, and Xiaoou Tang, Fellow, IEEE.

[3] http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html

[4] http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[5] http://deeplearning.net/tutorial/lenet.html