Image Super-Resolution Using Deep Convolutional Networks

Under the Guidance of

Dr. Shan Du

Submitted By

Vasista Avinash Behara (0689192)

Chaithanya Veera Reddy (0866559)

Ananya Marigowda (0862889)

Veeranajaneyulu Davuluri (0861594)

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.

Key Words: Image Super Resolution, deep convolutional neural network, low-resolution

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. Such a problem is typically mitigated by constraining the solution space by strong prior information.

The sparse-coding-based (SC) [9] 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 [8]. The overlapping reconstructed 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 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 a unified optimization framework.

The proposed SRCNN[1] has several appealing properties. First, its structure is intentionally designed with simplicity in mind, and yet provides superior accuracy2 compared with state-of-the-art example-based methods. Second, with moderate numbers of filters and layers, our method achieves fast speed for practical on-line usage even on a CPU. Our method is faster than several example-based methods, because it is fully feed-forward and does not need to solve any optimization problem on usage. Third, experiments show that the restoration quality of the network can be further improved when (i) larger and more diverse datasets are available, and/or (ii)

a larger and deeper model is used. On the contrary, larger datasets/ models can present challenges for existing example-based methods.

Why Convolutional Neural Network?

- We present a fully convolutional neural network for image super-resolution. The network directly learns an end-to-end mapping between low- and high-resolution images, with little pre/post-processing beyond the optimization.
- We establish a relationship between our deep-learning-based SR method and the traditional sparse-coding-based SR methods. This relationship provides a guidance for the design of the network structure.
- We demonstrate that deep learning is useful in the classical computer vision problem of super-resolution and can achieve good quality and speed.

We improve the SRCNN by introducing larger filter size in the non-linear mapping layer and explore deeper structures by adding non-linear mapping layers. Experimentally, we demonstrate that performance can be improved in comparison to the single-channel network. We also extend the original experiments from Set5 [2] and Set14 test images. In addition, we compare with several recently published methods and confirm that our model still outperforms existing approaches using different evaluation metrics.

Related Work

Super Resolution (SR)

In most digital imaging applications, high resolution images or videos are usually desired for later image processing and analysis. The desire for high image resolution stems from two principal application areas: improvement of pictorial information for human interpretation; and helping representation for automatic machine perception. Image resolution describes the details contained in an image, the higher

the resolution, the more image details. The resolution of a digital image can be classified in many ways: pixel resolution, spatial resolution, spectral resolution, temporal resolution, and radiometric resolution.

From [2], super-resolution (SR) are techniques that construct high-resolution (HR) images from several observed low-resolution (LR) images, thereby increasing the high frequency components and removing the degradations caused by the imaging process of the low-resolution camera. The basic idea behind SR is to combine the non-redundant information contained in multiple low-resolution frames to generate a high-resolution image. A closely related technique with SR is the single image interpolation approach, which can be also used to increase the image size. However, since there is no additional information provided, the quality of the single image interpolation is very much limited due to the ill-posed nature of the problem, and the lost frequency components cannot be recovered.

In the SR setting, however, multiple low-resolution observations are available for reconstruction, making the problem better constrained. The non-redundant information contained in the these LR images is typically introduced by subpixel shifts between them. These subpixel shifts may occur due to uncontrolled motions between the imaging system and scene, e.g., movements of objects, or due to controlled motions, e.g., the satellite imaging system orbits the earth with predefined speed and path.

SR arises in many areas such as:

- Surveillance video: frame freeze and zoom region of interest (ROI) in video for human perception (e.g. look at the license plate in the video), resolution enhancement for automatic target recognition (e.g. try to recognize a criminal’s face).

- Remote sensing: several images of the same area are provided, and an improved resolution image can be sought.
- Medical imaging (CT, MRI, Ultrasound etc.): several images limited in resolution quality can be acquired, and SR technique can be applied to enhance the resolution. Video standard conversion, e.g. from NTSC video signal to HDTV signal.

Convolutional Neural Network

In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptron's designed to require minimal pre-processing[3]. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

Convolutional neural networks date back decades and deep CNNs have recently shown an explosive popularity partially due to its success in image classification. They have also been successfully applied to other computer vision fields, such as object detection, face recognition, and pedestrian detection. Several factors are of central importance in this progress: (i) the efficient training implementation on modern powerful GPUs, (ii) the proposal of the rectified linear unit (ReLU) which makes convergence much faster while still presents good quality, and (iii) the easy access to an abundance of data for training larger models. Our method also benefits from these progresses.

CNNs have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars. A greyscale image, has just one channel. For the purpose of this post, we will only consider grayscale images, so we will have a single 2d matrix representing an image. The value of each pixel in the matrix will range from 0 to 255 – zero indicating black and 255 indicating white.

Formulation of Convolution Neural Networks for Super Resolution

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.

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. Let us denote the interpolated image as Y. Our goal is to recover from Y an image 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.

The mapping of this image consists of three operations:

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

An overview of the network is depicted in the following Fig.

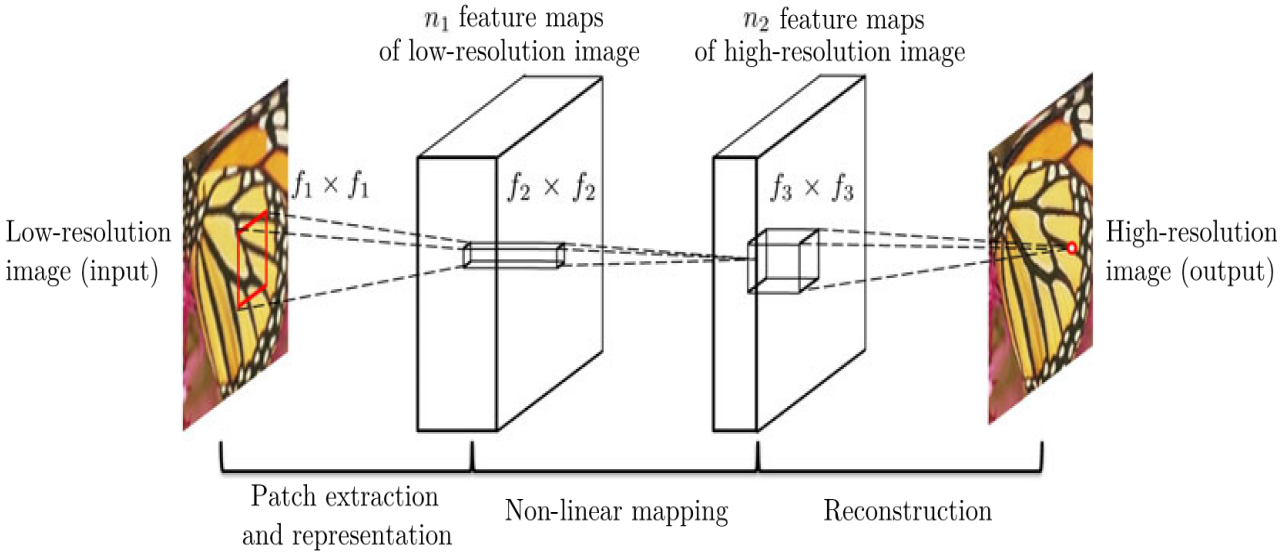


Fig 1. Stages of Convolution [1]

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:

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.

F1 (Y) = max (0, W1 * Y + B1)

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, who's each element is associated with a filter. We apply the ReLU (max (0; x)) on the filter responses.

Non-linear mapping:

The first layer extracts an n1-dimensional feature for each patch. In the second operation, we map each of these n1-dimensional vectors into an n2-dimensional one. This is equivalent to applying n2 filters which have a trivial spatial support 1×1. This interpretation is only valid for 1×1 filters. But it is easy to generalize to larger filters

like 3 × 3 or 5 × 5. 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 map.

F2 (Y) = max (0, W2 * F1 (Y) + B2)

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.

Reconstruction:

In the traditional methods, the predicted overlapping high-resolution patches are often averaged to produce the final full image. The averaging can be considered as a pre-defined filter on a set of feature maps (where each position is the "flattened" vector form of a high-resolution patch). This operation aggregates the above high-resolution patch wise representations to generate the final high-resolution image. This image is expected to be like the ground truth X.

F (Y) = W3 * F2 (Y) + B3

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

If the representations of the high-resolution patches are in the image domain, we expect that the filters act like an averaging filter; if the representations of the high-resolution patches are in some other domains, we expect that W3 behaves like first

projecting the coefficients onto the image domain and then averaging. In either way, W3 is a set of linear filters.

Relationship to Sparse-Coding-Based Methods and CNN

We show that the sparse-coding-based SR methods can be viewed as a convolutional neural network. Fig.2 shows an illustration.

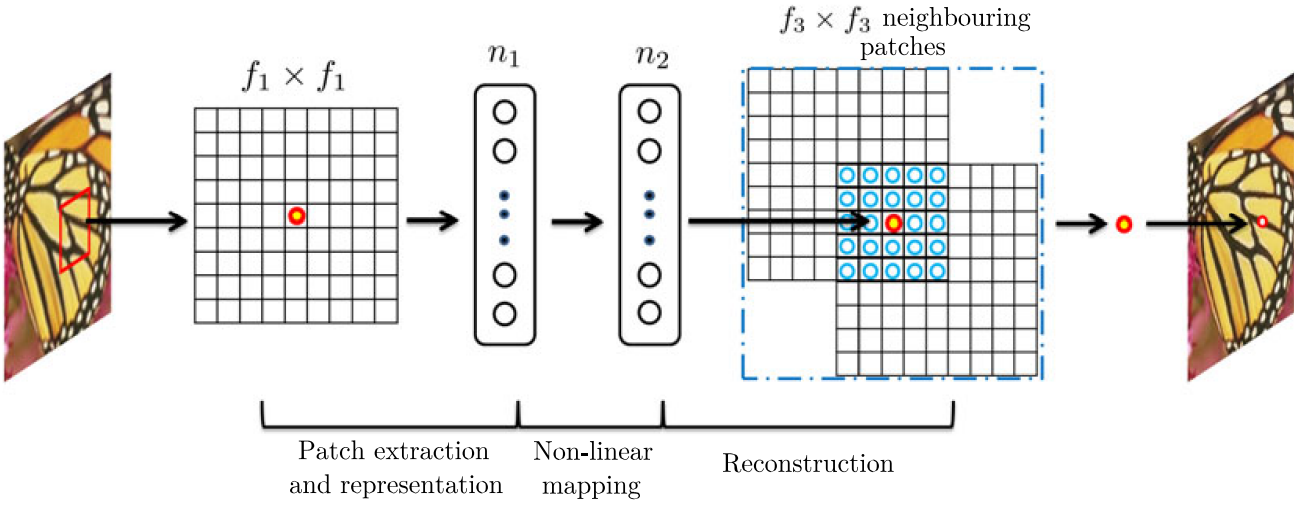


Fig 2. An illustration of sparse-coding-based methods in the view of a convolutional neural network. [1]

In the sparse-coding-based methods, let us consider that an 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 n1, this is equivalent to applying n1 linear filters (f1x f1) on the input image (the mean subtraction is also a linear operation so can be absorbed). This is illustrated as the left part of Fig. 2.

The sparse coding solver will then iteratively process the n1 coefficients. The outputs of this solver are n2 coefficients, and usually n2 ¼ n1 in the case of sparse coding. These n2 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. See the middle part of Fig. 2.

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 ¼ 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 n2 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 n2 feature maps. If the high-resolution patches used for reconstruction are of size f3xf3, then the linear filters have an equivalent spatial support of size f3 xf3. See the right part of Fig. 2.

The above discussion shows that the sparse-coding based SR method can be viewed as a kind of convolutional neural network (with a different non-linear mapping). But not all operations have been considered in the optimization in the sparse-coding-based SR methods. On the contrary, in our convolutional neural network, the low-resolution dictionary, high-resolution dictionary, non-linear mapping, together with mean subtraction and averaging, are all involved in the filters to be optimized. So our method optimizes an end-to-end mapping that consists of all operations.

Training

Learning the end-to-end mapping function F requires the estimation of network parameters $\Theta = \{W1, W2, W3, B1, B2, B3\}$. 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

In the training phase, the ground truth images {Xi} are prepared as fsub ×fsub ×c-pixel sub-images randomly cropped from the training images. By “sub-images” we mean these samples are treated as small “images” rather than “patches”, in the sense that “patches” are overlapping and require some averaging as post-processing but “sub-images” need not. To synthesize the low-resolution samples {Yi}, we blur a sub-image by a Gaussian kernel, sub-sample it by the upscaling factor, and upscale it by the same factor via bicubic interpolation.

To avoid border effects during training, all the convolutional layers have no padding, and the network produces a smaller output ((fsub−f1−f2−f3 +3)2×c). The MSE loss function is evaluated only by the difference between the central pixels of Xi and the network output. Although we use a fixed image size in training, the convolutional neural network can be applied on images of arbitrary sizes during testing.

Experiment

We first investigate the impact of using different datasets on the model performance. Next, we examine the filters learned by our approach. We then explore different architecture designs of the network and study the relations between super-resolution performance and factors like depth, number of filters, and filter sizes. Subsequently, we compare our method with recent state-of the-arts both quantitatively and qualitatively. Super-resolution is only applied on the luminance channel (Y channel in YCbCr color space) in the first/last layer, and performance (e.g., PSNR and SSIM) is evaluated on the Y channel. At last, we extend the network to cope with grey scale images and evaluate the performance.

Data Set

As shown in the literature, deep learning generally benefits from big data training. For comparison, we use 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. The size of training sub-images is fsub = 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. We observe a similar trend even if we use the larger Set14 set. The upscaling factor is 3.

The training time on ImageNet is about the same as on the 91-image dataset since the number of backpropagations is the same. As can be observed, with the same number of backpropagations (i.e.,8×108), the SRCNN +ImageNet achieves 32.52 dB, higher than 32.39 dB yielded by that trained on 91 images. The results positively indicate that SRCNN performance may be further boosted using a larger training set, but the effect of big data is not as impressive as that shown in high-level vision problems. This is mainly because that the 91 images have already captured sufficient variability of natural images. On the other hand, our SRCNN is a relatively small network (8,032parameters), which could not over fit the 91images (24,800 samples). Nevertheless, we adopt the ImageNet, which contains more diverse data, as the default training set in the following experiments.

Code:

For demo.m which is the main file that we execute,

```
close all;
clear all;
```

we read the ground truth image with this line

```
im = imread('set14\zebra.bmp');
```

Here we set the parameters, which include upscale factor and the ImageNet trained networks

```
up_scale = 3;
model = 'model\9-5-5(ImageNet)\x3.mat';
up_scale = 3;
model = 'model\9-3-5(ImageNet)\x3.mat';
up_scale = 2;
model = 'model\9-5-5(ImageNet)\x2.mat';
up_scale = 4;
model = 'model\9-5-5(ImageNet)\x4.mat';
```

To work on illuminance only, we only take Y luminance channel in YCbCr Space.

```
if size(im,3)>1
    im = rgb2ycbcr(im);
    im = im(:, :, 1);
end
im_gnd = modcrop(im, up_scale);
im_gnd = single(im_gnd)/255;
```

ModCrop.m file includes the following code which is a function used to crop the images

```
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


we get a Bicubic interpolated image after this
im_l = imresize(im_gnd, 1/up_scale, 'bicubic');
im_b = imresize(im_l, up_scale, 'bicubic');


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

SRCNN method is as follows

```
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;
```

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]);

to compute PSNR, we use a PSNR function
psnr_bic = compute_psnr(im_gnd,im_b);
psnr_srcnn = compute_psnr(im_gnd,im_h);

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);
```

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');
```

RESULTS & DISCUSSION:

By varying the upscale factor and the training set for that particular upscale factor, we try to get the results of different images.

For the image of zebra from set14 using the upscale factor of 5 and ImageNet data set of 9-5-5 as filters, we get the following images as results.

The following result gets the values as follows:

PSNR for Bicubic Interpolation: 26.633759 dB

PSNR for SRCNN Reconstruction: 29.290147 dB

By changing the upscale factor to 3 and using the same filter Data set of 9-5-5, we get the following result.

PSNR for Bicubic Interpolation: 26.633759 dB

PSNR for SRCNN Reconstruction: 28.864286 dB

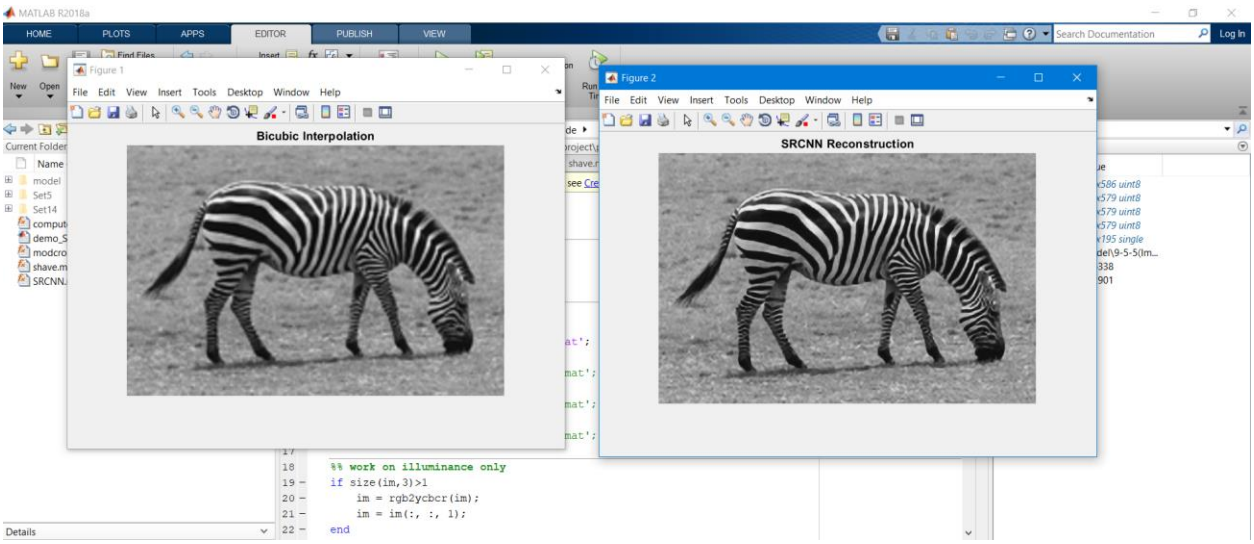


Fig 3: output of zebra image from set14

Let us change the image and check the results for butterfly image in 3 cases with upscale factor of 1,2 and 3

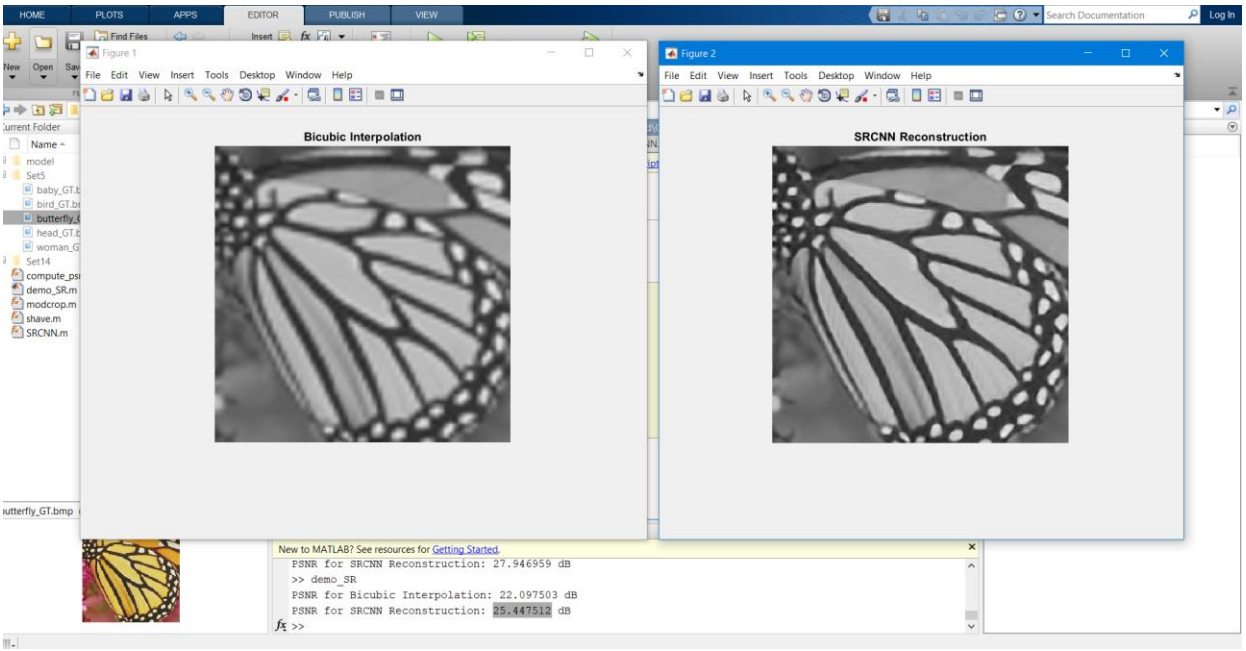
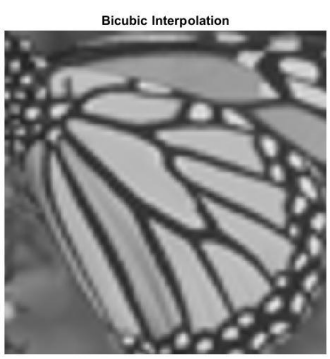


Fig 4: output of image butterfly from set5

This image is also shown in the paper. So, we use this image to compare it with the results in the paper.

By using the upscale factor of 2 and 9-5-5 set(using more filters get better results)

We get the results as



PSNR for Bicubic Interpolation: 27.433190 dB

PSNR for SRCNN Reconstruction: 32.746595 dB

This result is promising as it is a better result.

Upscale factor	Bicubic	SRCNN
2	27.433190	32.746595
3	24.036424	27.946959
4	22.097503	25.447512

Table 1 : comparison of image in different upscale values

For the zebra image, the above results obtained show us a significant improvement of SRCNN over the bicubic image. We also prove the paper by getting the results as close to them as possible. We can even derive from the table that using an upscale factor of 2 results better results in both than upscale factor of 4.

Here are some other images that we used to get more results.

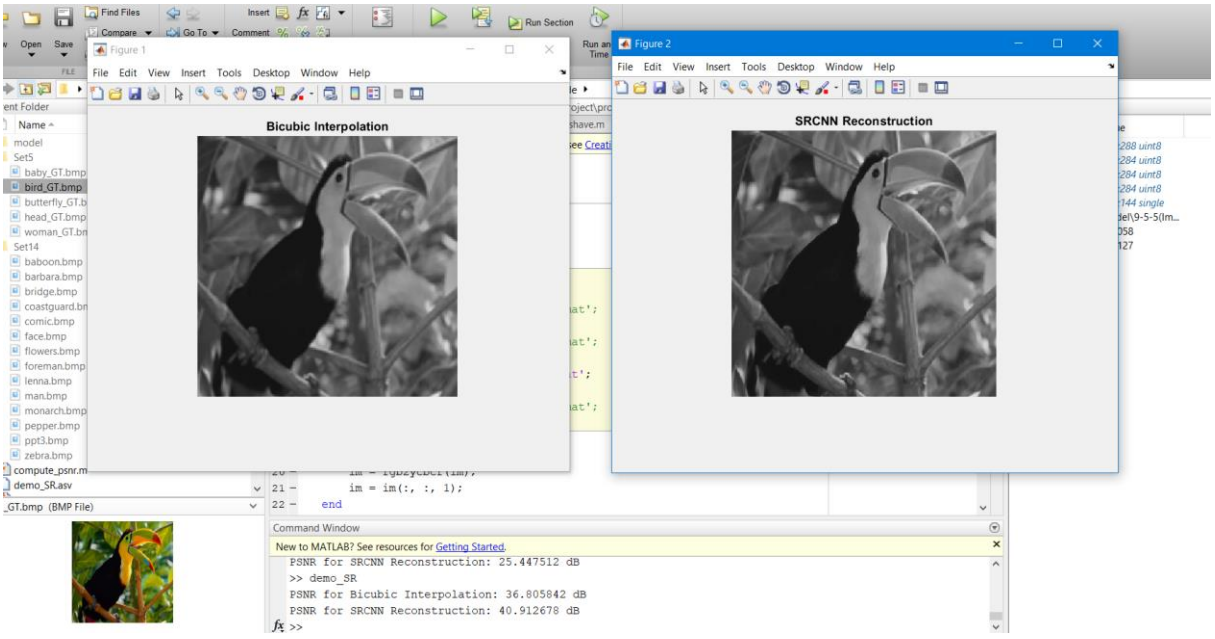


Figure 1
File Edit View Insert Tools Desktop Window Help
Bicubic Interpolation
Figure 2
File Edit View Insert Tools Desktop Window Help
SRCNN Reconstruction
Command Window
New to MATLAB? See resources for Getting Started.
PSNR for SRCNN Reconstruction: 25.447512 dB
>> demo_SR
PSNR for Bicubic Interpolation: 36.805842 dB
PSNR for SRCNN Reconstruction: 40.912678 dB

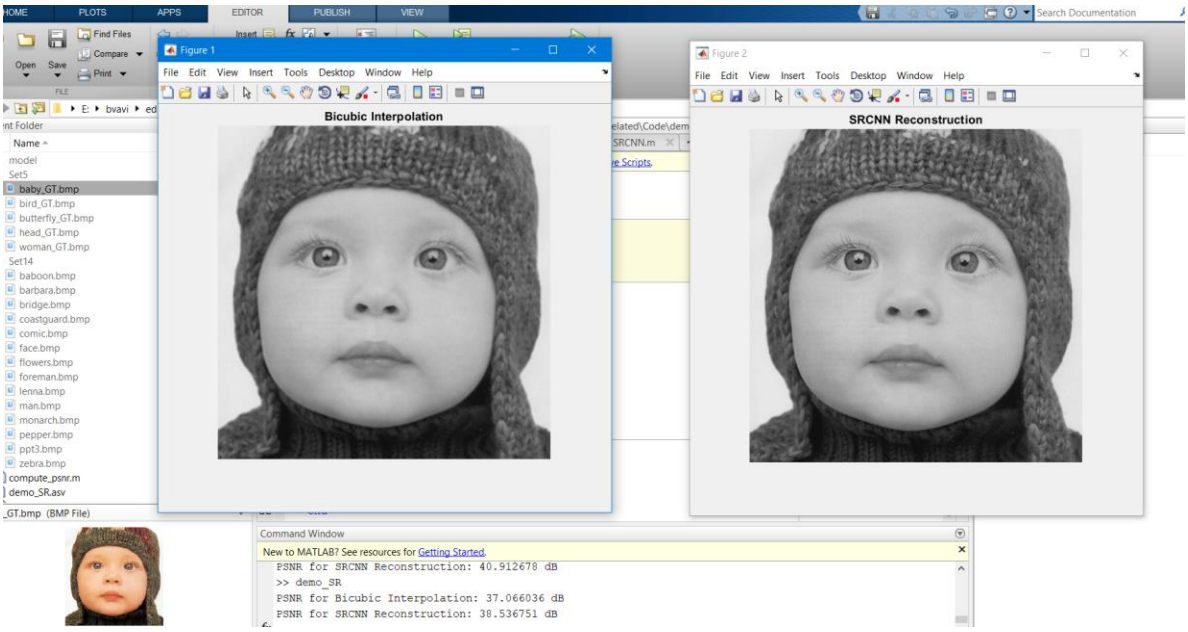


Figure 1
File Edit View Insert Tools Desktop Window Help
Bicubic Interpolation
Figure 2
File Edit View Insert Tools Desktop Window Help
SRCNN Reconstruction
Command Window
New to MATLAB? See resources for Getting Started.
PSNR for SRCNN Reconstruction: 40.912678 dB
>> demo_SR
PSNR for Bicubic Interpolation: 37.066036 dB
PSNR for SRCNN Reconstruction: 38.536751 dB

COMPARISIONS:

We try to explain the comparison of SRCNN with the state of the arts methods which they compared in the paper that we have taken[1]. I have taken the table in which they explained about set14 results between all the different methods.

The Average Results of PSNR (dB), SSIM, IFC, NQM, WPSNR (dB) and MSSIM on the Set14 Dataset

Eval. Mat	Scale	Bicubic	SC [48]	NE+LLE [4]	KK [24]	ANR [39]	A+ [39]	SRCNN
PSNR	2	30.23	-	31.76	32.11	31.80	32.28	**32.45**
	3	27.54	28.31	28.60	28.94	28.65	29.13	**29.30**
	4	26.00	-	26.81	27.14	26.85	27.32	**27.50**
SSIM	2	0.8687	-	0.8993	0.9026	0.9004	0.9056	**0.9067**
	3	0.7736	0.7954	0.8076	0.8132	0.8093	0.8188	**0.8215**
	4	0.7019	-	0.7331	0.7419	0.7352	0.7491	**0.7513**
IFC	2	6.09	-	7.59	6.83	7.81	**8.11**	7.76
	3	3.41	2.98	4.14	3.83	4.23	**4.45**	4.26
	4	2.23	-	2.71	2.57	2.78	**2.94**	2.74
NQM	2	40.98	-	41.34	38.86	41.79	**42.61**	38.95
	3	33.15	29.06	37.12	35.23	37.22	**38.24**	35.25
	4	26.15	-	31.17	29.18	31.27	**32.31**	30.46
WPSNR	2	47.64	-	54.47	53.85	54.57	**55.62**	55.39
	3	39.72	41.66	43.22	43.56	43.36	44.25	**44.32**
	4	35.71	-	37.75	38.26	37.85	38.72	**38.87**
MSSSIM	2	0.9813	-	0.9886	0.9890	0.9888	0.9896	**0.9897**
	3	0.9512	0.9595	0.9643	0.9653	0.9647	0.9669	**0.9675**
	4	0.9134	-	0.9317	0.9338	0.9326	0.9371	**0.9376**

Table 2 : comparison of different methods for set14 dataset

We even compare with the result that we have obtained. The PSNR of SRCNN we got is 32.746595 which is very close and even better to the result that they have shown in the table for upscale factor of 2. This means the results we obtain are close to the ones mentioned in this paper. So, we can use this table which is theoretically our results to compare ours with other methods.

The state of the arts methods that we mention in this table are:

SC - sparse coding-based method of Yang et al. [9]

NE+LLE - neighbour embedding + locally linear embedding method [4]

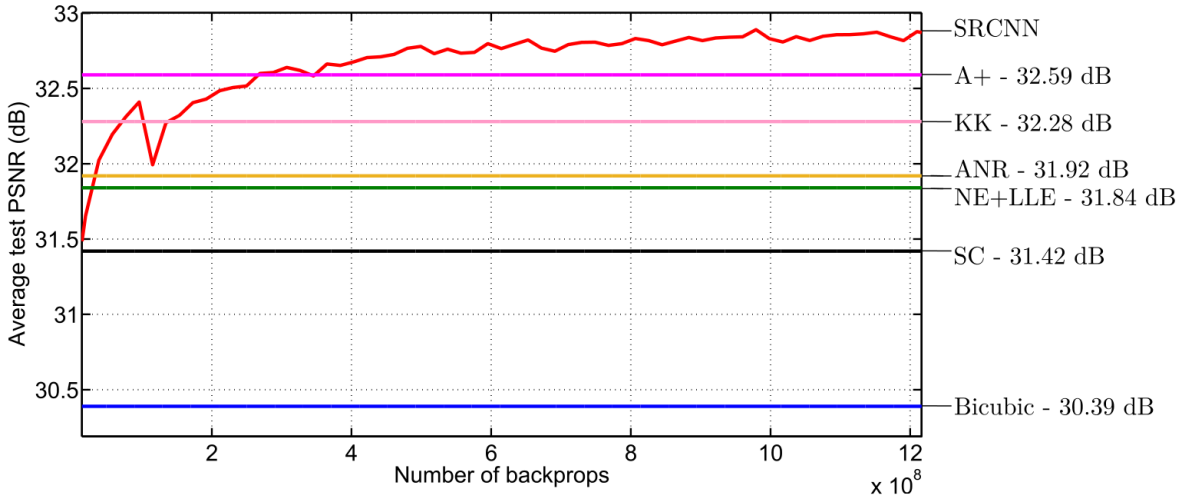
ANR - Anchored Neighbourhood Regression method [6]

A+ - Adjusted Anchored Neighbourhood Regression method [7], and

KK - the method described in [5], which achieves the best performance among external example-based methods, according to the comprehensive evaluation conducted in Yang et al.'s work [8].

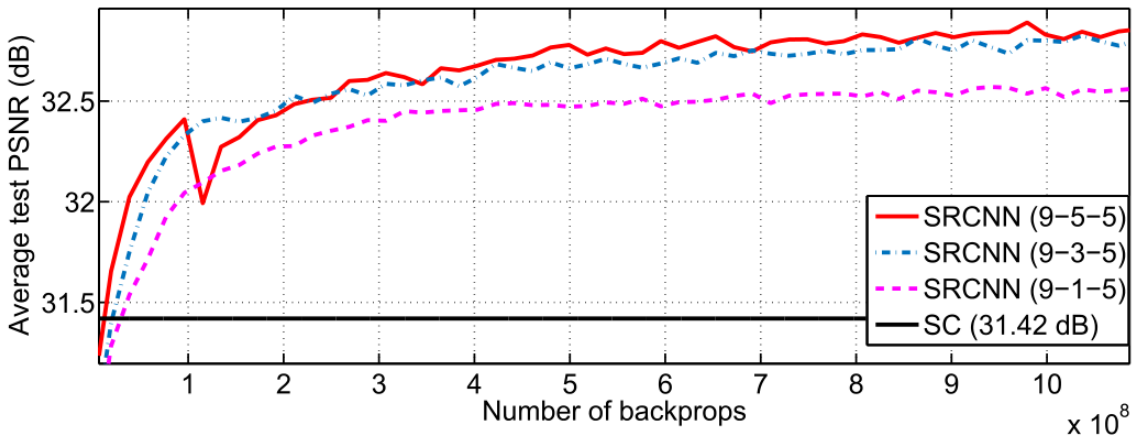
All the methods are available publicly and the authors used set5 and set14 to implement them. The same image is taken for the comparison and compared. This shows that SC which is base for the SRCNN method is close to bicubic and KK is by far the best performing when we don not consider this method. They have used many evaluations metrics apart form PSNR but we basically PSNR which shows that SRCNN is the best performing among all the methods.

We are using some of the graphs used in [1] to show how the results are compared which we could not generate.



Graph 1: Comparison of average PSNR vs Backprops for different methods

The above graph shows in detail how the methods compare in each other with average PSNRs and number of backdrops. All the methods are same in backdrops always but the SRCNN changes with each value. As we increase the number of backdrops which are increased with filter size, we get better results in SRCNN.



Graph 2 : Backprops and average PSNR values of different filter values

As we discussed, the more the filters, the better the image gets. This graph shows the comparison of using different filters i.e., trained data set used by training with different number of filters. This shows that using more filters i.e., 9-5-5 gets better results.

CONCLUSION:

This report mainly explains about the deep convolution neural networks and its approach for single image SR. Conventional Sparse coding methods can be reframed using Convolutional neural networks. SRCNN, does and end – to – end mapping between low- and high-resolution images to reconstruct the bicubic image with little pre/post processing. This easy structure of SRCNN also able to achieve higher performance than state of the arts methods. If we use more filters and with higher data sets, we can get performance results. We could also try to implement the SRCNN method of colored images and get a colored image output for future enhancements. Studying the paper, we understood that they had done a great job even with the colored images. This can be applied in image and resolution problems like image blurring and denoising etc.

REFERENCES:

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, “Image Super-Resolution Using Deep Convolutional Networks”, IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 2, Feb. 2016.

[2] "Image super-resolution: Historical overview and future challenges" By Jianchao Yang and Thomas Huang, University of Illinois at Urbana-Champaign

[3] "An Intuitive Explanation of Convolutional Neural Networks" by Ujjwalkarn, August 11, 2016 - https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[4] H. Chang, D. Y. Yeung, and Y. Xiong, “Super-resolution through neighbor embedding,” presented at the IEEE Conf. Comput. Vis. Pattern Recog., Washington, DC, USA, 2004.

[5] K. I. Kim and Y. Kwon, “Single-image super-resolution using sparse regression and natural image prior,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 6, pp. 1127–1133, Jun. 2010.

[6] R. Timofte, V. De Smet, and L. Van Gool, “Anchored neighborhood regression for fast example-based super-resolution,” in Proc. IEEE Int. Conf. Comput. Vis., 2013, pp. 1920–1927.

[7] R. Timofte, V. De Smet, and L. Van Gool, “A+: Adjusted anchored neighborhood regression for fast super-resolution,” in Proc. IEEE Asian Conf. Comput. Vis., 2014, pp. 111–126.

[8] C. Y. Yang, C. Ma, and M. H. Yang, “Single-image super-resolution: A benchmark,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 372–386.

[9] J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” IEEE Trans. Image Process., vol. 19, no. 11, pp. 2861–2873, Nov. 2010.