CAP 6412

# Advance Computer Vision

Project # 2

Image Super-Resolution Using Deep Convolutional Networks

**Syed Ahmed**
SY3722427

## *Introduction*

The goal of the project is to learn and understand the image super-resolution and CNN work by Dong, Chao, Chen Change Loy, Kaiming He, and Xiaoou Tang.in their publication named "Learning a deep convolutional network for image super-resolution." In ECCV 2014.proposes. The authors have proposed a restoration method making their system learn to map between low and high resolution images from end-to-end with the help of convolutional neural networks and processing beyond optimization. The paper also stresses on sparse-coding-based image super resolution by optimizing all layers with the help of a light weight structured deep CNN, providing a relationship that helps in designing network structures. I have used a training dataset of 6 images, two sets of 3 images each for performance evaluation of upscaling factor calculation. The pipeline starts with a patch extraction process that results in high-dimensional vectors representing patches form input image and being non-linearly mapped to other such vectors followed by aggregation to give the high resolution image with a higher speed and better quality. State-of-the-art SR paradigms have been used in the experiments to provide visually appealing results and highest average PSNR in much lesser time.

## *Approach*

As per the methods proposed by authors, I first extracted the frames by a video sequence shared by Dr.Gong and each frame was considered a low-resolution image, that was first up-scaled to a factor either 2 or 3, then the luminance channel 'y' alone was extracted and was input to the CNN. The goal here is to obtained F(Y) from Y that is as similar as possible to the ground truth high-resolution image X.

The patches from the low resolution image Y were extracted in order to form high-dimensional vector which was then mapped to another high-dimensional vector nonlinearly which we finally aggregate the final high-resolution image.

### ***Convolutional Layers:***

First layer for patch extraction is represented as follows:

$F_1(Y) = \max(0, W_1 * Y + B_1);$

where W1, B1 → the filters and biases respectively, * → denotes the convolution operation. Here, $W_1$ support c*f1*f1, where c → number of channels in the input image, $f_1$ is the spatial size of a filter.

Second layer for non-liner mapping:

$F_2(Y) = \max(0, W_2 * F_1(Y) + B_2);$

Increasing the layers will eventually require more training time and would also make our CNN complex.
The third layer is used for reconstruction and is represented as:

$F_3(Y) = \max(0, W_3 * F_2(Y) + B_3)$;

The operation of these layers are combined to form a CNN.

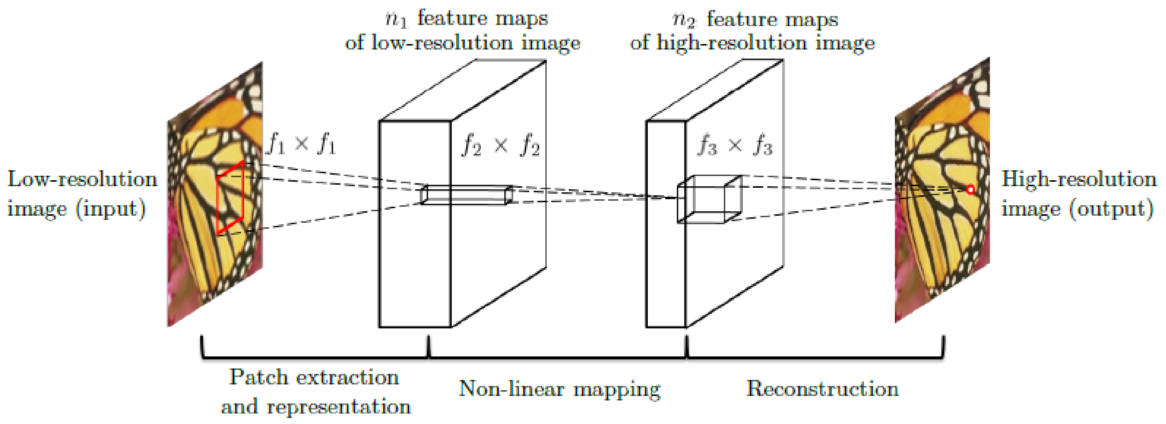


*Figure 1: Representation of Convolutional layers in SRCNN*

## *Experimental Setup*

Operating system: Ubuntu 14.04
Framework: Caffe by BVLC
Matlab: 2012b
No GPUs were used all the experiments were done on CPU mode. Intel core-i5 processor with 8GB of RAM was used.

## *Training*

Given a set of high-resolution images X and its corresponding low-resolution images Y, we use Mean Squared Error (MSE) as the loss function:

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

n → #training samples
loss function favours high PSNR.
Convolutional neural networks do not preclude the usage of other kinds of loss functions, if only the loss functions are derivable. The filter weights of each layer are initialized by drawing randomly from a Gaussian distribution with zero mean and standard deviation 0.001 (and 0 for biases). The learning rate is 10□4 for the first two layers, and 10□5 for the last layer.

## *Experiments*

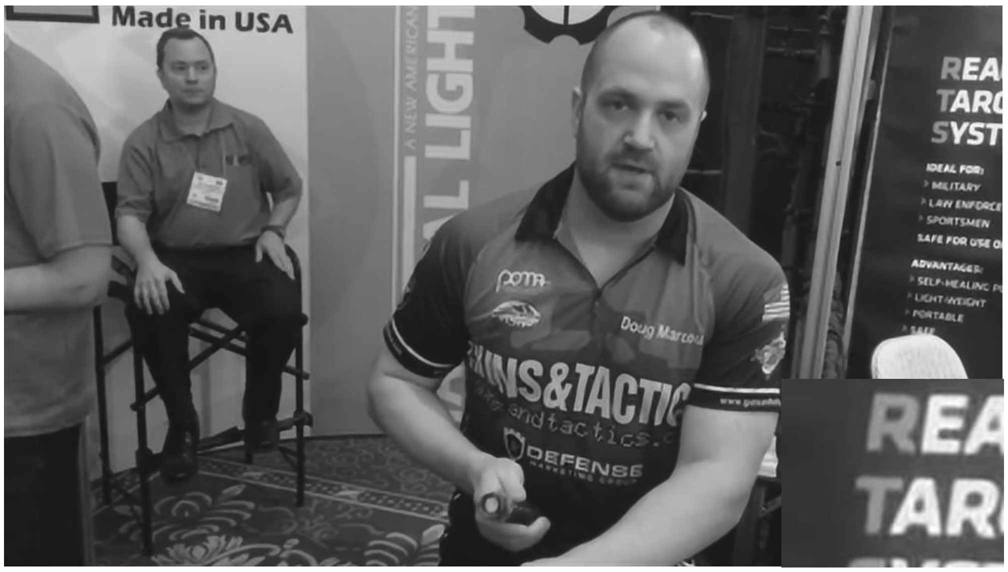
***Dataset:***
The 4K data with K=3 that was obtained from the given video and shuffled was then split into three parts. 2K of the frames were used as the training data, another K frames of them as the validation data, and the remaining K frames as the test data. K=3 was chosen due to insufficient computation resources.

***Training:***
The following parameters for training were used:
Size of sub images = 33
Stride = 14
Momentum = 0.9
Base learning rate = 0.0001
Weight decay = 0
CPU mode
Network Setting:
f1 = 9, f2 = 1, f3 = 5, n1 = 8, and n2 = 2. Tried for n1 = 6, and n2 = 2 and n1 = 8, and n2 = 2.

The learning rate was reduced to 0.00001 at the ending 10% of the training phase. About 5 million iterations were done.

***Results:***

I have reported the results on 3 test images that were randomly extracted from the given video and an upscale factor of 3.



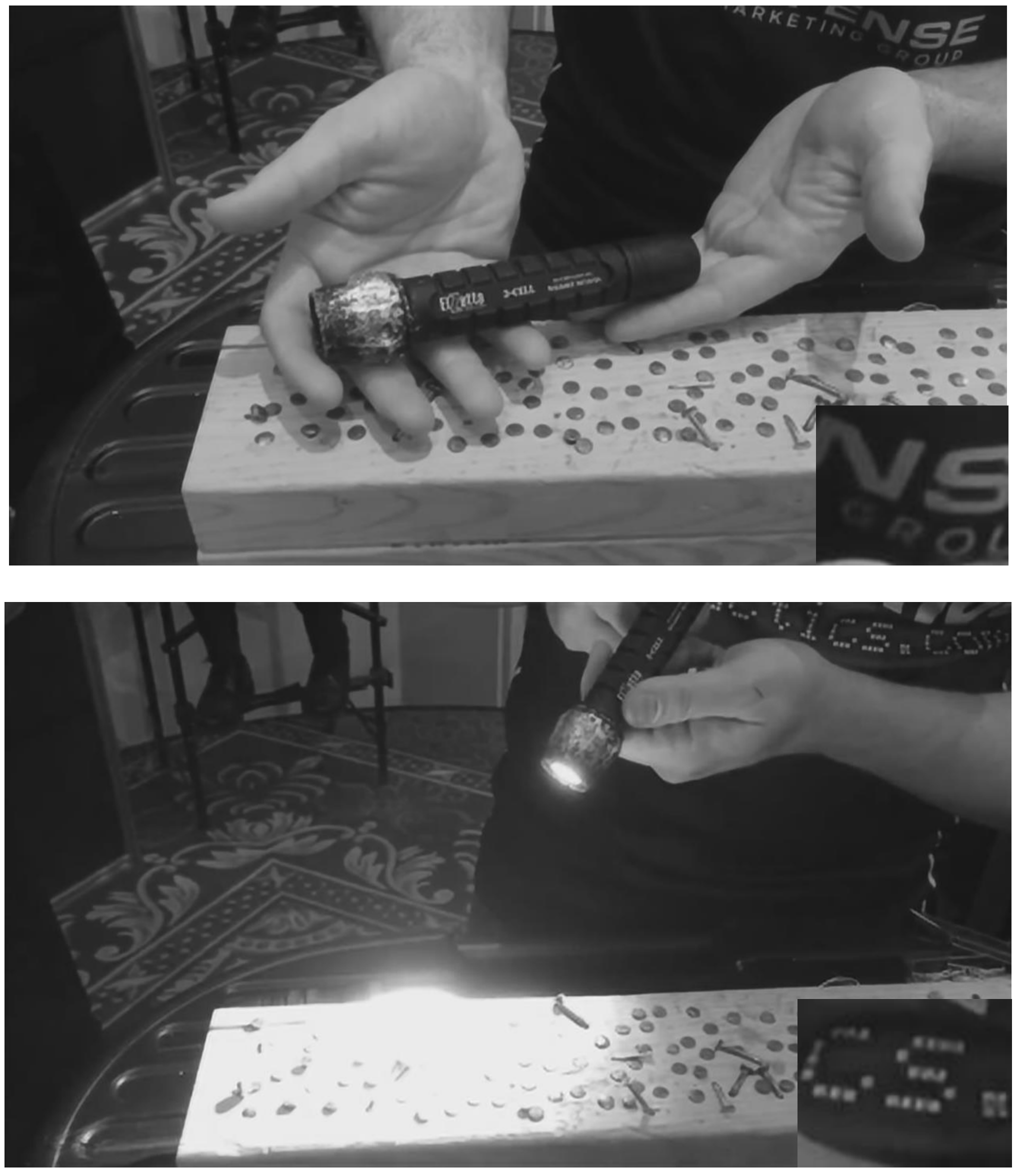


*Figure 2: Resulting images with scale 3*

The same images were also tested for up-scaling factor and the output images have been displayed below.

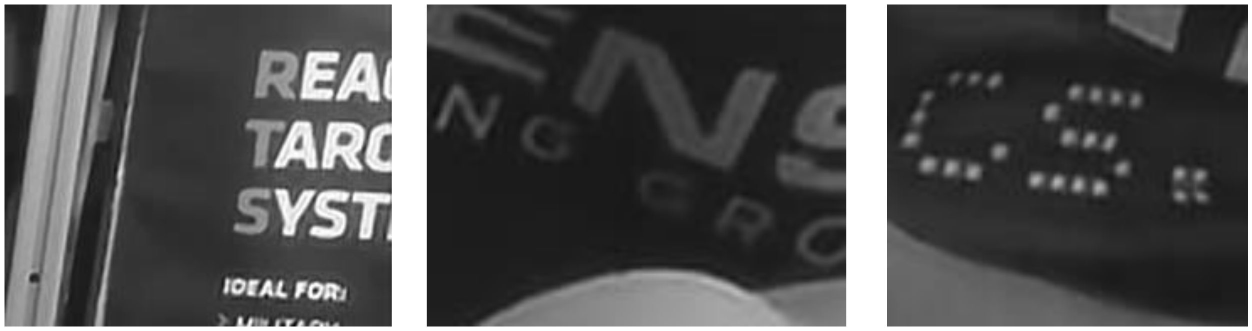


*Figure 3: Parts of results images with scale 2*

From the comparison of the above images, it can be said that the images with scale 2 have better clarity than the former.

The output images on the intermediate backpropagations for both the scaling factors and both the paradigms (SRCNN and bi-cubic interpolation) have been shared here for reference.

The below section shows comparison of the performance of the models on both the upscaling factors 2 and 3:

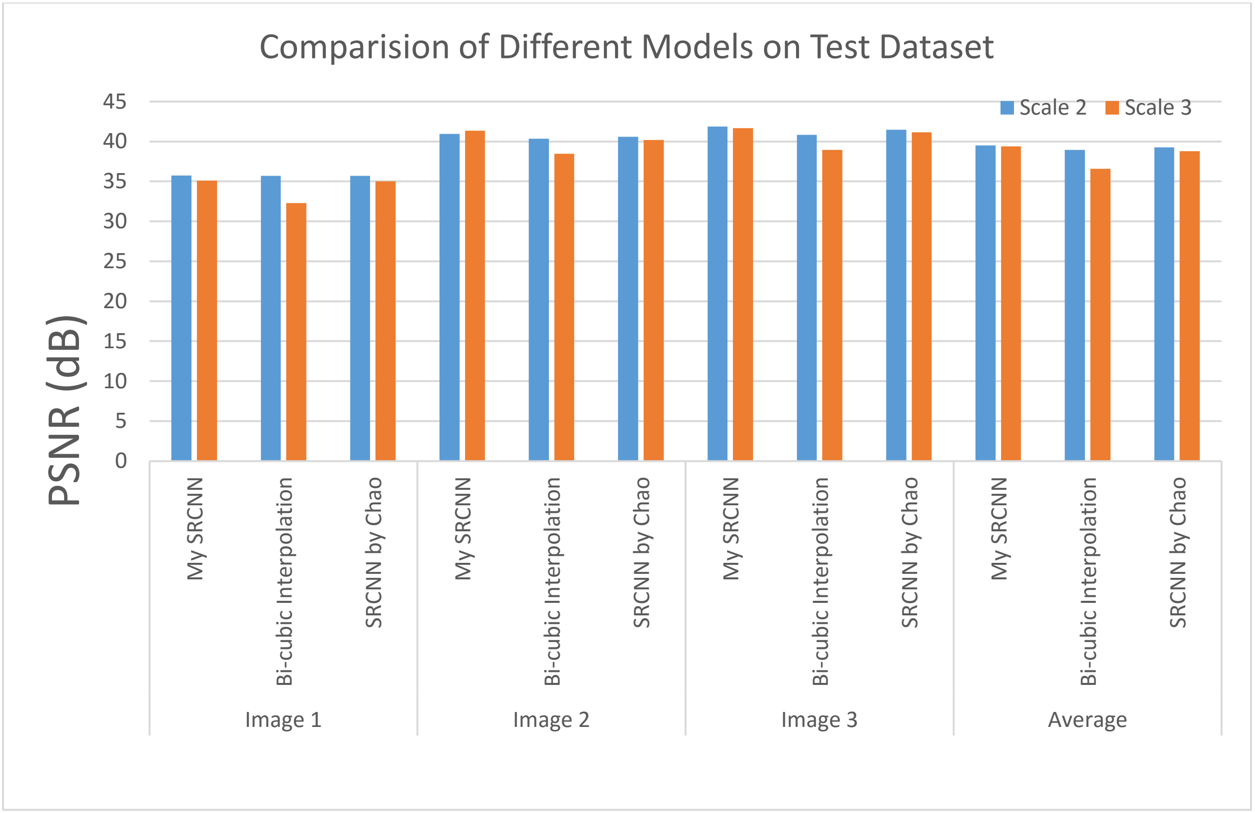


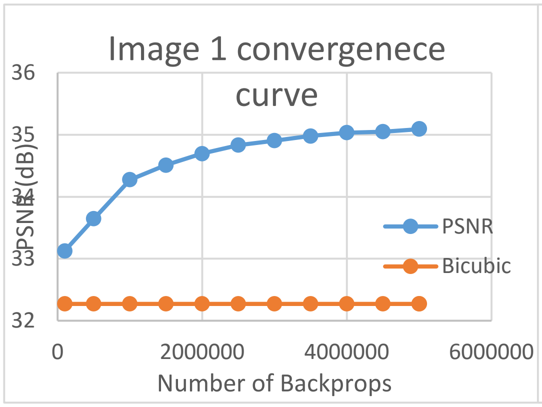
*Figure 4: Comparison of different model performances on both scales.*

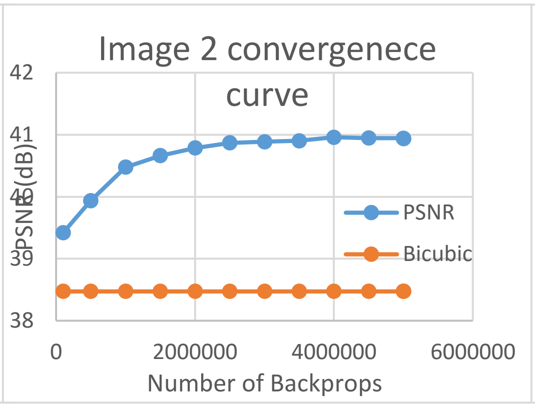
| | | PSNR | | | |
|---|---|---|---|---|---|
| Model | Scale | Image 1 | Image 2 | Image 3 | Average |
| My SRCNN | 2 | **35.717118** | **40.93817** | **41.850726** | ***39.502005*** |
| | 3 | **35.097445** | **41.329571** | **41.678445** | ***39.368487*** |
| Bi-cubic Interpolation | 2 | 35.678135 | 40.360482 | 40.844449 | 38.961022 |
| | 3 | 32.272841 | 38.473112 | 38.943847 | 36.563267 |
| SRCNN by Chao | 2 | 35.70215 | 40.587621 | 41.463722 | 39.251164 |
| | 3 | 35.032932 | 40.165723 | 41.153286 | 38.78398 |

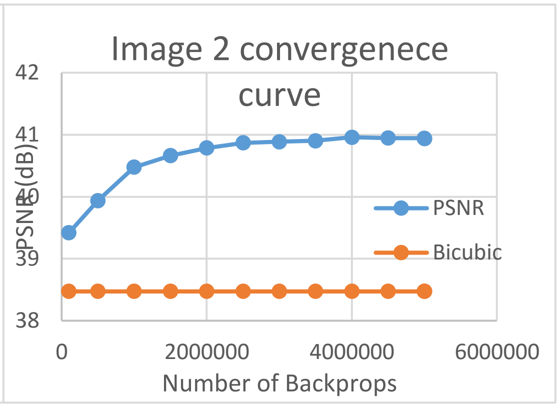
*Table 1: Comparison of different model performances on both scales.*

The convergence curve for each image and then the average of all the test images have been reported below:

It can be seen in the curve that after 4.5 million back props, the PSNR value was either constant or decreased very slightly. In order to further improve the performance of the system, I reduced the learning rate by 1/10th as mentioned in the section above. As shown, the performance started increasing slightly, only 0.5 million iterations were continued from this point but with more iterations, the performance could have been increased further.







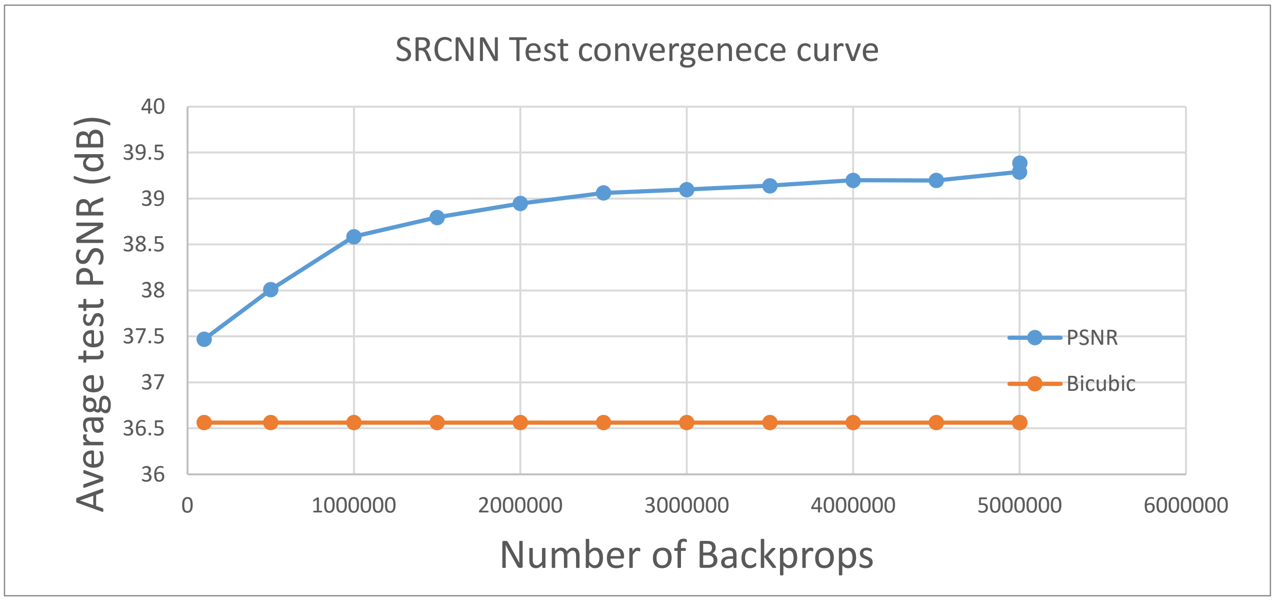


*Figure 5: The test convergence curve of SRCNN and bi-cubic interpolation paradigms on test dataset*

Different filter numbers were also tried out to check the restoration speed and the results have been shown below:

| | |
|---|---|
| n1 = 8<br>n2 = 2<br>back props = 5 million | PSNR: 39.28<br>Time: 137 secs |
| n1 = 4<br>n2 = 2<br>back props = 3 million | PSNR:39.03<br>Time: 96 secs |

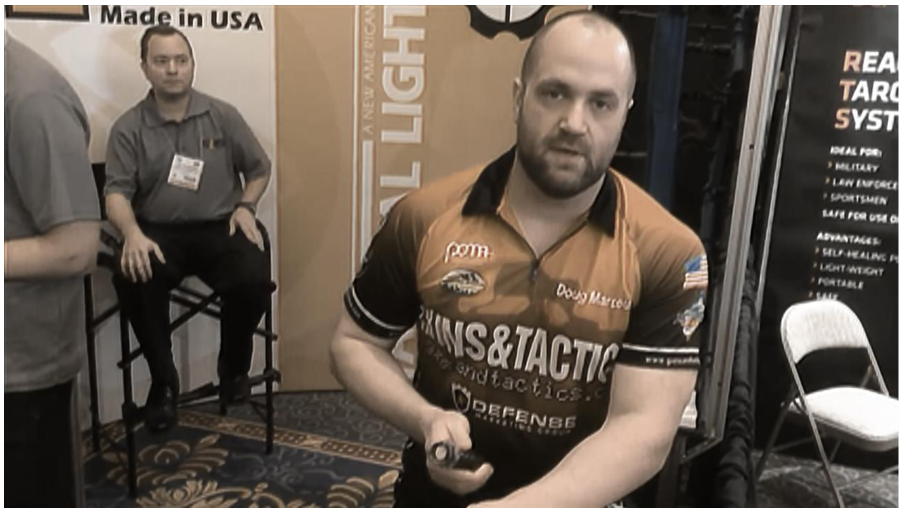
*Table 2: Qualitative results of using different filter numbers in SRCNN.*

## *Experiments on RGB channels:*

The system was then trained on the same dataset but with the following network architecture:

c = 3, f1 = 9, f2 = 1, f3 = 5, n1 = 8 , 6 , and n2 = 2.

***Results:***



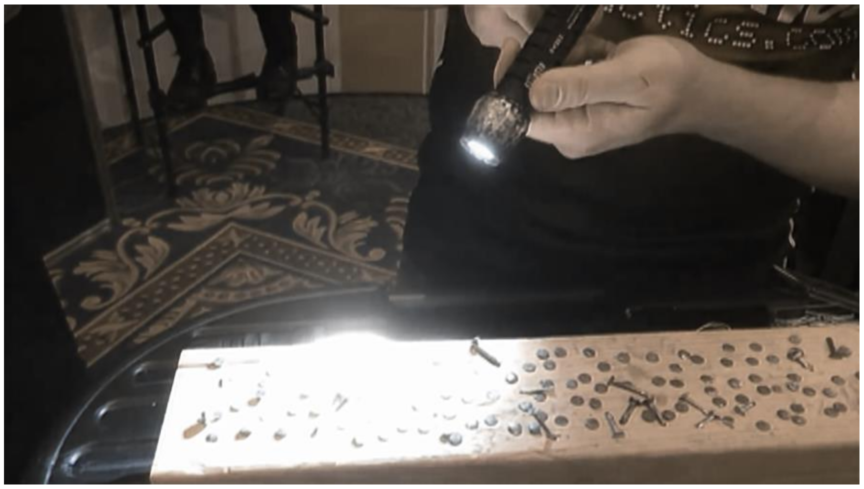


*Figure 2: Resulting images with scale 3 and RGB channels*

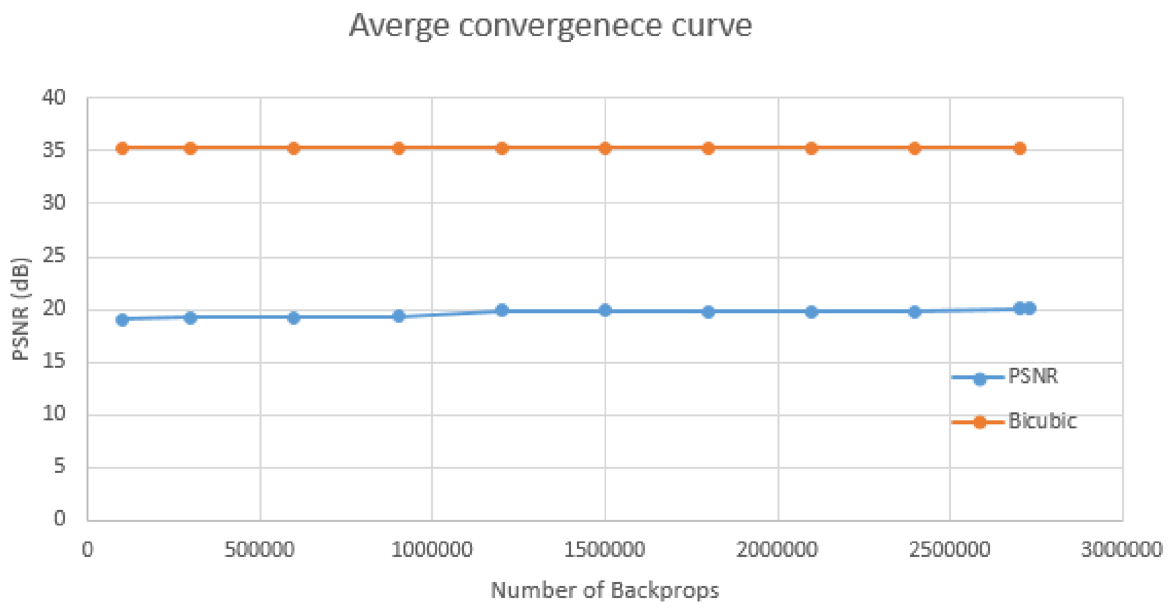


*Figure 6: The test convergence curve of SRCNN and bi-cubic interpolation paradigms on test dataset on RGB Channels*

Only 2.7 million iterations were done on RGB and it was noticed that the system was learning very slowly, I tried to optimize the system but any change in the learning rate, decay, momentum was resulting in the fall of the network into bad local minimum.

***Attached files***

- Log files for both the training phases (Y and RGB channel)
- Evaluation code and SRCNN implementation code.
- Resulting images have been attached here.
- Caffe solverprototxt for RGB channel and the modified code for RBG SCRNN implementation.

## *Conclusion*

- The model is very simple, robust and easy to implement. With a lightweight structure, the SRCNN has achieved superior performance than the state-of-the-art methods. Additional performance can be further gained by exploring more filters and different training strategies.
- Reducing the learning rate at the end of the training phase yields better performance.
- The scaling plays an important role in reconstruction the image but requires a lot of time.
- As the filter size in the network increases the complexity and the time required to be trained also increases and moreover, no significant improvement can be seen in comparison to a system trained with relatively lower filter scale. Hence this parameter is very crucial and is to be selected as a trade-off between required performance and the available time.
- When the weight decay of the training phase was changed, no significant improvement was noticed.
- A small set of data is sufficient for the system to learn and produce significant results.

This approach of reconstruction can be extended to videos where super resolution could be used for frame reconstruction in videos. Another significant work that could be seen here is the proof of deep learning being able to solve the SR problems, since their approach yields better results with boast in performance, we could use similar approaches to reconstruct a set images a once instead of using a single image at a time. This paper also suggests that CNNs don't avoid the usage of loss functions and adapts the metric used during training time. This fact could also be one of the motivations for future work of this flexibility. Another motivation is to utilize better training approaches and using more filters in order to obtain even better results for the SR problems in computer vision like noise removal etc.,

## *References:*

- *C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In Proc. Eur. Conf. Comp. Vis., 2014.*
- *Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: IEEE Conference on Computer Vision and Pattern Recognition (2004)*
- *Mamalet, F., Garcia, C.: Simplifying convnets for fast learning. In: International Conference on Artificial Neural Networks, pp.58–65. Springer (2012)*
- *Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: ACM Multimedia. pp. 675–678 (2014)*
- *Yang, C.Y., Ma, C., Yang, M.H.: Single-image super-resolution: A benchmark. In: European Conference on Computer Vision, pp. 372–386 (2014)*