RGB image super-resolution using Deep learning using SR-CNN

Abstract

There are very limited number of methods to enhance the resolution of the RGB image. We by using the most emerging deep learning technique which is SRCNN technique tried enhancing the image's quality. SRCNN is considered to be a supervised technique, and we used around 91 images to train the model. From the trained SRCNN, a high resolution image was generated from a low resolution image. The original image was down scaled using bilinear interpolation and then was made to pass through our SRCNN model and the high resolution image was generated as an output. We used several indexes other than visual difference, to measure the quality of the image generated like SSIM (Structural Similarity Index Module), PSNR (Peak Signal to Noise Ratio), and MSE (Mean Square Error). These measures help to understand the error difference between the original image and the generated ones.

I. NOMENCLATURE

Single Image Super Resolution(SR), Peak Signal to Noise Ratio(PSNR), Mean Square Error(MSE), Structural Similarity index(SSIM), Super-Resolution Convolutional Neural Network (SRCNN)

II. INTRODUCTION

Our main aim is to increase the visual quality of an image, the need for increase in clarity of an image is in high demand. Few driving forces behind such algorithm includes CCTV surveillance, regular video information enhancement, medical diagnosis, astronomical observation, bio metric information identification, etc.[1] Deep learning techniques are used to enhance or increase the spatial resolution of the images. "Super resolution refers to a set of algorithms whose purpose is to increase the resolution of images and video"[2]. Upscaling and upsizing means the same but super-resolution (SR) methods try to do so without sacrificing or distorting the detail and visual appeal of the images or video.

Low resolution images offer low pixel density, thereby offering less information visually. Therefore, we require a high resolution image which contains more information. Few possible methods to increase resolution includes Reducing pixel size where the spatial resolution is increased by decreasing the number of pixels per unit area and thus increasing the density of the pixels, then the resolution can also be increased by increasing the chip size which too increases the spatial resolution of the image and also by super-resolution where the high resolution image is formed from combining multiple low resolution images[3].

Super resolution can be of two types, multiple image super resolution and single image super resolution. In multiple image super resolution, multiple low resolution images are taken from the same camera and the high resolution image generated by using those multiple LR images and performing subpixel shifts lead us to the SR image which is considered to contain more information than the original LR images. On the other hand in single image SR, the high resolution image is generated using that single LR image by using different algorithms[4][3]. Our model is based on Single image SR technique.

In order to increase resolution, the pixel density is increased by reducing the size of the pixels and adding new pixels to it, to make it more informative than the LR image. The new pixels value is determined by values of the surrounding pixels, and they are of many types, few of them are nearest neighbor interpolation technique, bilinear interpolation and bi-cubic interpolation, etc. The process of assigning a new value to the missing pixel on basis of the surrounding values is known as interpolation. In nearest neighbor interpolation technique, the missing pixel value depends on the value of the pixels which are placed nearest to the missing pixels. Bi-cubic interpolation where the output value takes data from all eight pixels surrounding the pixel which is worked on. We in our model use, bilinear interpolation uses this technique where the missing pixel value is the weighted average value of the four nearest points of the missing pixel. Bi-cubic is considered over the other two because of its high precision value as it takes more number of pixels and thus is more accurate[5].

CNNs (Convolution Neural Networks) and Deep CNNs have shown a huge growth recently due to its success in image classification. They have also contributed towards many other objectives like pedestrian detection, object detection, face recognition, etc. Few factors have a huge role to play in its success like (a.) The efficient training implementation on modern powerful GPUs. (b.)The efficient ReLu function which lets convergence to happen very faster without degrading the quality.[6] (c.)A very easy access to data sets like of ImageNet for training our model[7]–[9]. Our method too is benefited from the above mentioned. CNN (Convolutional Neural Network) is considered under deep neural networks. It is generally used for image classification but SRCNN is used for single image Super Resolution (SR)

III. SRCNN

SRCNN model is used to generate the high resolution images. It is a classical super-resolution technique. Our deep SRCNN is a lightweight structure, and carries forward state-of-the-art restoration quality, and achieves fast speed for practical on-line usage[7], [10]. We explore different network structures and parameter settings to gain trade-offs between performance and speed. It has 3 parts, patch extraction and representation, non-linear mapping, and reconstruction.

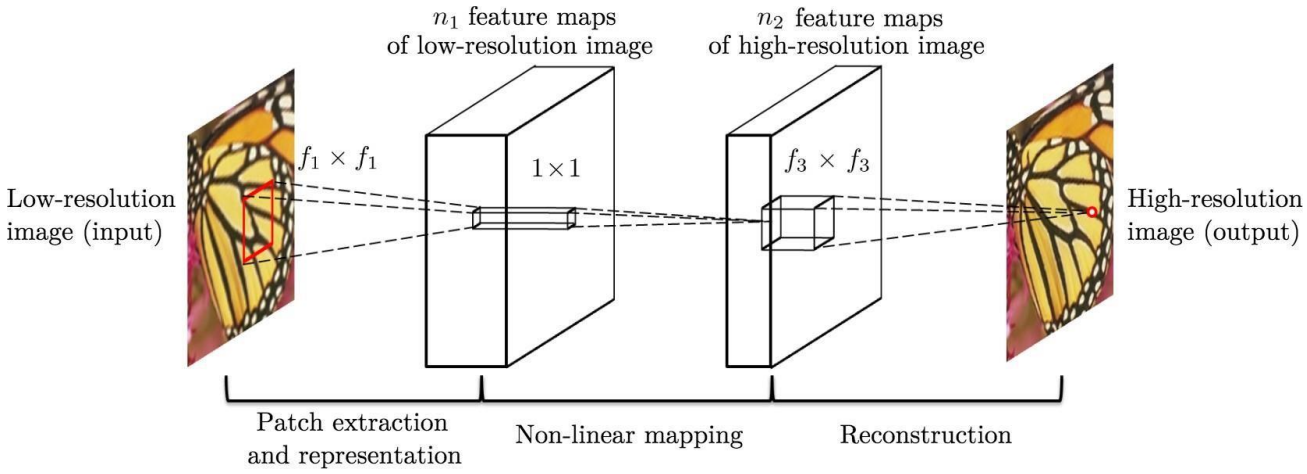


Fig. 1. Architecture of SRCNN

IV. ARCHITECTURE

A. Pre-processing

The image is first down sampled using bilinear interpolation before putting into the network. The image is converted to the YCbCr color space, although the network uses only the luminance channel (Y). The output of the network then merges the interpolated CbCr channels to output the final color image. We chose this step because we are not interested in color changes (information stored in the CbCr channel) but only in its brightness (Y channel) since our method like many Super resolution methods [7], [11]–[14] first convert image to a different color space and then process the channel. Since Y channel contains most information out of all CrCb and the channels of RGB which contains almost equal information in them we choose this approach.

B. Patch Extraction Convolution

layer 1:

It is to extract patches and represent them by pre-trained bases which is like convolving an image by a set of filters, each of them is a basis. Optimizing of each base leads to optimization of the network. The model extracts low-dimensional patches each of them is represented as a high-dimensional vector, of which these vectors consists of few feature maps.

X: High resolution original image.

Y: Image performing after bilinear interpolation.

- Size of W1: n1 filters of size c × f1 × f1, c is the number of channels.
- Activation function: ReLU (rectified linear unit)
- Output: n1 feature maps
- Parameters to optimize: c × f1 × f1 × n1 weights and n1 dimensional bias vector • Size of B_1: n_1

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

C. Non-linear Mapping

Convolution layer 2:

Non-linear mapping is done from n1 dimensional vector to n2 dimensional vector i.e from low resolution vector to high resolution vector. Each mapped vector is a representation of a high resolution patch. The structure is made deeper by adding more non-linear mapping layers.

- Size of W2: n2 filters of size n1 × 1 × 1
- Activation function: ReLU
- Output: n2 feature maps
- Parameters to optimize: n1 × 1 × 1 × n2 weights and n2 dimensional bias vector, (n2 >n1) and 1 × 1, brings more non-linearity to the model increasing the accuracy. • Size of B_2: n_2

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

D. Reconstruction Convolution

layer 3:

Reconstruction is done after mapping by performing convolution. All the above formulated high-resolution patch-wise representations are aggregated to form the final expected image.

- Size of W3: n2×1 ×1×c, c is channels here
- Size of B3: c

- Activation function: Identity
- Output: HR image $F(Y) = W_3 * F_2(Y) + B_3$

V. EXPERIMENT

A. Dataset

Our training dataset included around 91 colored images from ImageNet (diverse dataset) ranging from 75x75 to 375x375 pixels. All the pictures were vibrant, and they include pictures of cars, flowers, nature and other household objects. The size of training sub-images is fsub = 33. Thus the 91-image dataset can be decomposed into 24,800 sub-images (at strides=14). It is stated that SRCNN performance may get better using a larger training set, but the effect of big data is not that impressive. This is mainly because that the 91 images have already captured sufficient variability of natural images. Few pictures are mentioned below in Fig.2

Fig. 2. Some of the images from ImageNet

B. Network Structure

1) **First layer CNN:** Feature extraction of the input picture. (9 x 9 x 124 convolution kernel)
2) **Second layer CNN:** Nonlinear mapping of features extracted from the first layer (1 x 1 x 64 convolution kernel)
3) **Layer 3 CNN:** Reconstructs the mapped features to produce high resolution images (5 x 5 x 1 convolution kernels) (in Fig.3).

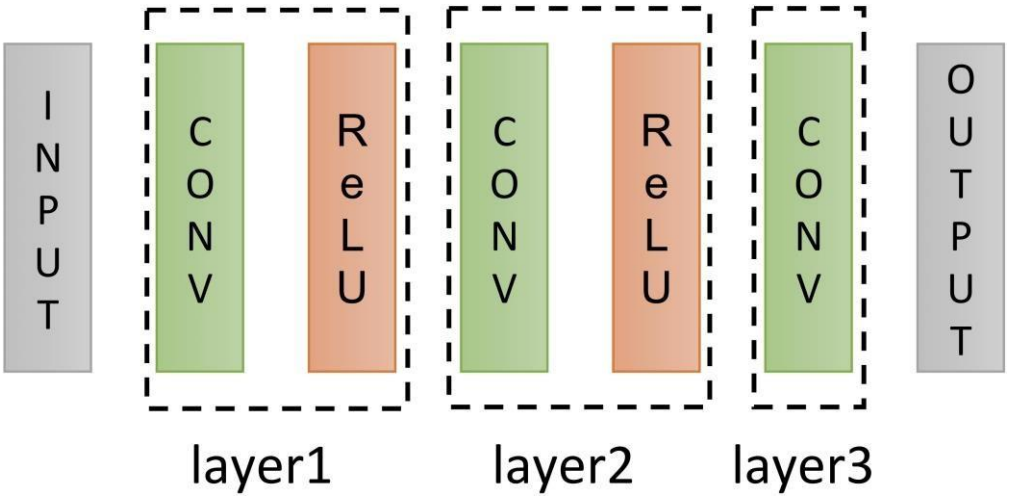


Fig. 3. A simple Network structure of SRCNN model

1) *Initializer:* Xavier/Glorot-normal Initialization initializes the weights in the network by drawing them from a distribution with zero mean and a specific variance[15]

$$Var(w_i) = \frac{1}{fan_{in}}$$

Where, fan_{in} is the number of incoming neurons

2) *Optimizer:* ADAM, an adaptive learning rate method, which means, it measures individual learning rates for different parameters. Its name comes from adaptive moment estimation, and the reason behind why it's called that is because Adam uses

estimations of first and second moments of gradient to adopt the learning rate for each and every weight of the neural network.[16], [17] Moment of a random variable is defined as the expected value of that variable to the power of z. More formally:

$$m_z = [X]^z$$ Where, *m*-

moment *X*-random variable.

3) Activation function: ReLu (Rectified Linear Unit), the most used activation function in deep learning. It helps neural network by speeding up the training process. The gradient computation is very easy, 0 or 1 depending on the sign of x. The computational step is quite easy where we put 0.0 for negative values and x for all positive value x. Gradients of hyperbolic and logistic tangent networks are smaller as compared to the positive part of the ReLU. This reflects that the positive part is updated more rapidly as the training progresses. However, this comes at a cost. On the left-hand side, the zero gradient is called as "dead neurons," in which a gradient update gives the incoming values to a ReLU such that the output is always generated as zero.

C. Accuracy assessment

Measures like SSIM and PSNR tells a lot about distortion of the image. The respective value calculated for images tells of the accuracy rate possessed by the model. Few important measures have been discussed below:

1) MSE (Mean Square error): It is the average of the square of the resultant image and the original images. The lesser the value of MSE, the more near we are to the expected value because it tends to minimize the variance.

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||(Y_i; \Theta) - X_i||^2$$

Where *n* is the number of training samples,$F(Y_i ; \Theta)$is the minimum loss between reconstructed images and the corresponding ground truth high resolution images *X*

```
def mse(target,ref): #MSE is the sum of squared diff between the two images err =
      np.sum((target.astype('float') -ref.astype('float')) ** 2) err /=
        float(target.shape[0] * target.shape[1]) return err
```

Code Snippet 1. Function for MSE

2) PSNR (Peak Signal to Noise Ratio: The image quality depends on the pixel difference between two images. It is a comparison made between the original image and the reconstructed ones. The PSNR is basically the SNR when all pixel values are equal to the maximum possible value.

$$PSNR = 10.\, log_{10}\frac{MAX_{I2}}{MS},$$

where $MAX_I = 255$ when pixels =8

```
def psnr(target, ref): #RGB image target_data = target.astype(float) ref_data =
   ref.astype(float) diff = ref_data - target_data    diff = diff. flatten('C')
      rmse = math.sqrt(np.mean(diff ** 2.))     return 20 * math.log10(255. /
   rmse)
```

1

Code Snippet 2. Function for PSNR

3) SSIM (Structural Similarity Index): The Structural Similarity Index is a metric that quantifies image quality degradation caused by image processing like data compression or loss incurred in data transmission, in our case its, after image processing. Unlike PSNR (Peak sound to Noise Ratio), SSIM is based on visible structures in the image.

D. Test Data

Test data is collected from a camera of following specifications and the testing is done on it: The 12 MP and 13 MP sensors with ZEISS optics and 2x optical zoom deliver vivid colors. (Specifications)Primary camera 12 MP-1.4µm-f/1.75-2PD with ZEISS optics, 13 MP-1.0µm-f/2.6 with ZEISS optics, dual-tone flash. The test data is named as "our data" here.

E. Evaluation

As discussed earlier, the output images were compared with the original ones to see the efficiency of our SRCNN model using some measuring metrics like SSIM (Structural Similarity Index Model), MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ratio).The evaluated values are shown below with the image on which the testing was performed and the image quality measures values are shown in fig5 And the table (Table 1.) corresponds to the values of the metrics.

Fig.4

The (Fig.4) is code generated output image where the 1st image represents the ground truth image which undergoes down-sampling using bilinear method and the 2nd image is generated and then SRCNN model is applied on it to generate the corresponding to 0.87 is 0.92, which indicates less distortion in the image. PSNR and MSE has an inverse relationship, higher PSNR value for the output image (PSNR=29.2) shows a lower value of MSE which means less error in the final image and hence better image quality.

	Degraded	SRCNN
PSNR	27.09	29.23
MSE	381.02	232.88

Table 1

The Fig.5 shows the rise in SSIM value after performing SRCNN on the degraded images of our test data.

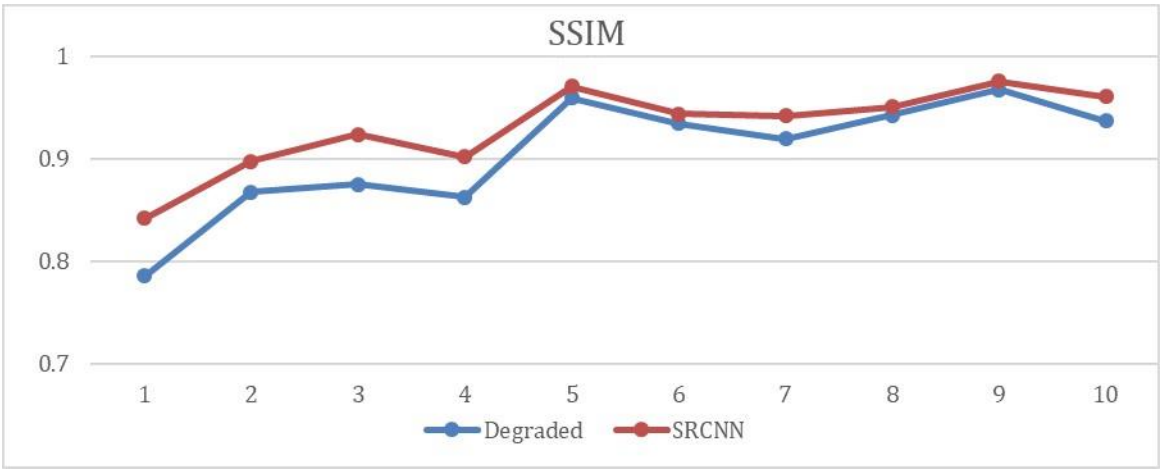


Fig. 5. Rise in SSIM after performing SRCNN

1) Comparison between different sets:: 10 images were taken for testing from test set and the performance of the model was judged, the test data is not a part of training data and the SSIM value is pretty good which proves efficiency and accuracy of our model. The average SSIM value after performing SRCNN on training data [set 5 and set 14] is 0.93 and 0.87 respectively and almost equal to the average SSIM value after performing SRCNN on test data(our data) which is 0.9, shown in (Fig.5). The high SSIM and PSNR value on the test data indicates that SRCNN model can be used on real life images. (The average value was evaluated from the value given in Table 5, 6, and 7 is shown in Table 2)

2) Compared with other paper and the model proposed by them: Different columns represent the model proposed by different papers and their output. Here, comparison has been done between the evaluated PSNR values that was predicted by their paper. The ground truth image was first resampled at a scale of 2 using Bi-cubic interpolation (and the image is named as A) and then it is used as an input to different model which is proposed by different papers mentioned in our Table 3 and thus the high-resolution image is generated as an output. The PSNR and SSIM value is used as a metric to calculate the efficiency of their model. The PSNR(eval) and SSIM(eval) are the evaluated values which is calculated by subtracting the average PSNR value of the resampled images of set14 (29.78, from Table2) from the average PSNR value of the high resolution images generated through different model and the same goes for SSIM(eval) which is calculated by subtracting the average SSIM value of the resampled images of set14 (0.87, from Table 2) from the average SSIM value of the high resolution images generated through different model.

Average value									
SET 14				SET 5				Our data	
	SRCNN				SRCNN				SRCNN	
PSNR	MSE	SSIM		PSNR	MSE	SSIM		PSNR	MSE	SSIM
29.78	281.61	0.87		33.07	96.68	0.93		31.74	189.14	0.9

Table 2

set14							
Eval. Mat.	SC[12]	NE+LLE[11]	KK[14]	ANR[13]	A+[13]	SRCNN[7]	SRCNN + BL
PSNR(eval)	-1.92	1.53	1.88	1.57	2.05	**2.22**	1.898083
SSIM(eval)	-0.0733	0.0306	0.0339	0.0317	0.0369	0.038	**0.039167**

Table 3

set5							
Eval. Mat.	SC[12]	NE+LLE	KK[14]	ANR[13]	A+[13]	SRCNN[7]	SRCNN + BL
PSNR(eval.)	-2.24	2.11	2.54	2.17	2.88	3	**3.266**
SSIM(eval.)	-0.0478	0.0191	0.0212	0.02	0.0245	0.0243	**0.034**

Table 4

Our model (SRCNN + Bi-linear) is presented in the last column, the 8th column where the images are resampled using bilinear interpolation instead of bi-cubic interpolation. Despite of the fact the bi-cubic interpolation is considered to a better technique than bilinear, the SSIM(eval) value for the model is the highest showing a higher gap between the average SSIM value of the resampled images(Bi-linear in this case) and average SSIM value of the high resolution images, thus proving that our model provides with more accuracy than other models.

Same explanation as above goes for [set 5] images which is shown in Table 4 and here too the PSNR(eval) and SSIM(eval) is maximum for the model SRCNN + bilinear, showing that this model provides images of better quality than as compared to other models.

VI. Conclusion

This paper is an implementation of SRCNN model which is not considered to be that deep, hence it contains only 3 layers, looking at the need of an image with high visual capacity in many fields. The SRCNN model can be enhanced more by adding more filters and adding more features to the training data. With its robust nature and simplicity, it could be applied to more low vision problems such as image deblurring or simultaneous SR+ denoising. There are many metrics proposed for image comparison like MSE, PSNR and SSIM, where we can measure the destruction difference between the original image and the high resolution image. After undergoing a lot of test, result declarations and thus comparisons, it was proved that this model presented in our paper provides outputs of high quality images. In above results, when compared with models proposed by other papers, this model gave better results which was inferred from the PSNR(eval) and SSIM(eval) values calculated above in the evaluation section.

REFERENCES

[1] L. Yue, H. Shen, J. Li, Q. Yuan, H. Zhang, and L. Zhang, "Image super-resolution: The techniques, applications, and future," *Signal Processing*, vol. 128, no. May, pp. 389–408, 2016.

[2] R. Al-falluji, A. Youssif, and S. Guirguis, "Single Image Super Resolution Algorithms: A Survey and Evaluation," *Int. J. Adv. Res. Comput. Eng. Technol.*, 2017.

[3] S. C. Park, M. K. Park, and M. G. Kang, “Super-resolution image reconstruction: A technical overview,” *IEEE Signal Processing Magazine*. 2003.

[4] W. T. Freeman, T. R. Jones, and E. C. Pasztor, “Example-based super-resolution,” *IEEE Comput. Graph. Appl.*, 2002.

[5] Matlab and Simulink, “Nearest Neighbor, Bilinear, and Bicubic Interpolation Methods,” *MathWorks, Inc.*, pp. 8–9, 2018.

[6] V. Nair and G. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” in *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[7] C. Dong, C. C. Loy, K. He, and X. Tang, “Image Super-Resolution Using Deep Convolutional Networks.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, 2016.

[8] C. Szegedy *et al.*, “Going deeper with convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015.

[9] J. Wu, “Introduction to convolutional neural networks,” *Natl. Key Lab Nov. Softw. Technol.*, pp. 1–31, 2017.

[10] M. Lin, Q. Chen, and S. Yan, “Network In Network,” pp. 1–10, 2013.

[11] Hong Chang, Dit-Yan Yeung, and Yimin Xiong, “Super-resolution through neighbor embedding,” no. January 2004, pp. 275–282, 2004.

[12] 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, 2010.

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

[14] 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, 2010.

[15] X. Glorot and Y. Bengio, “Understanding the Difficulty of Training Deep Feedforward Neural Networks,” in *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*, 2010.

[16] S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” in *ICLR 2018*, 2018.

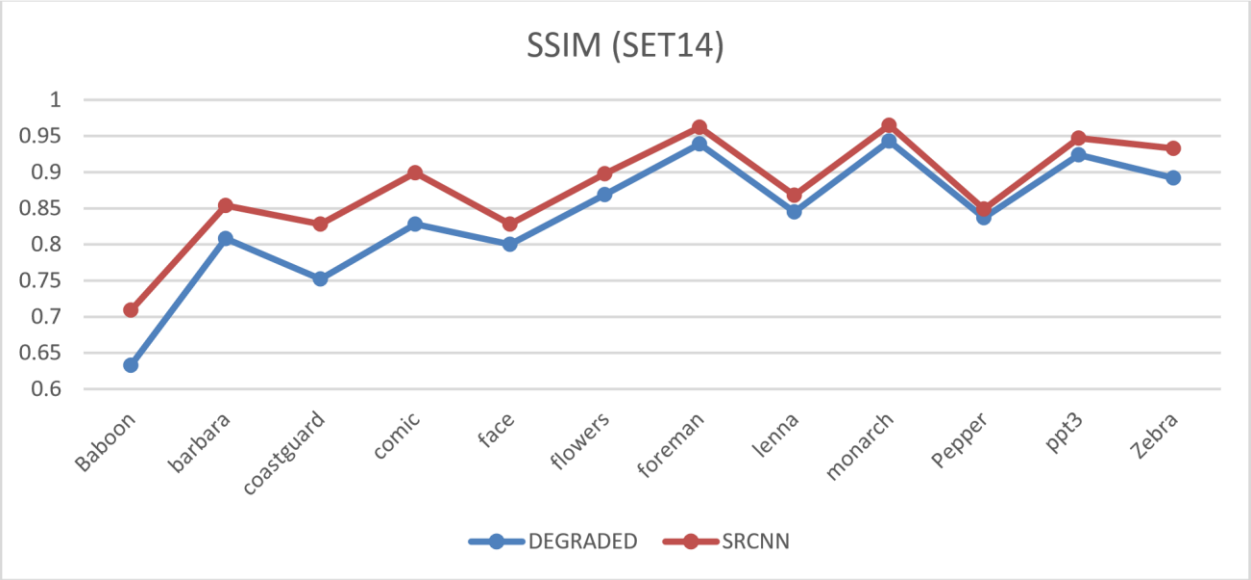
[17] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” pp. 1–15, 2014.

ANNEXURE

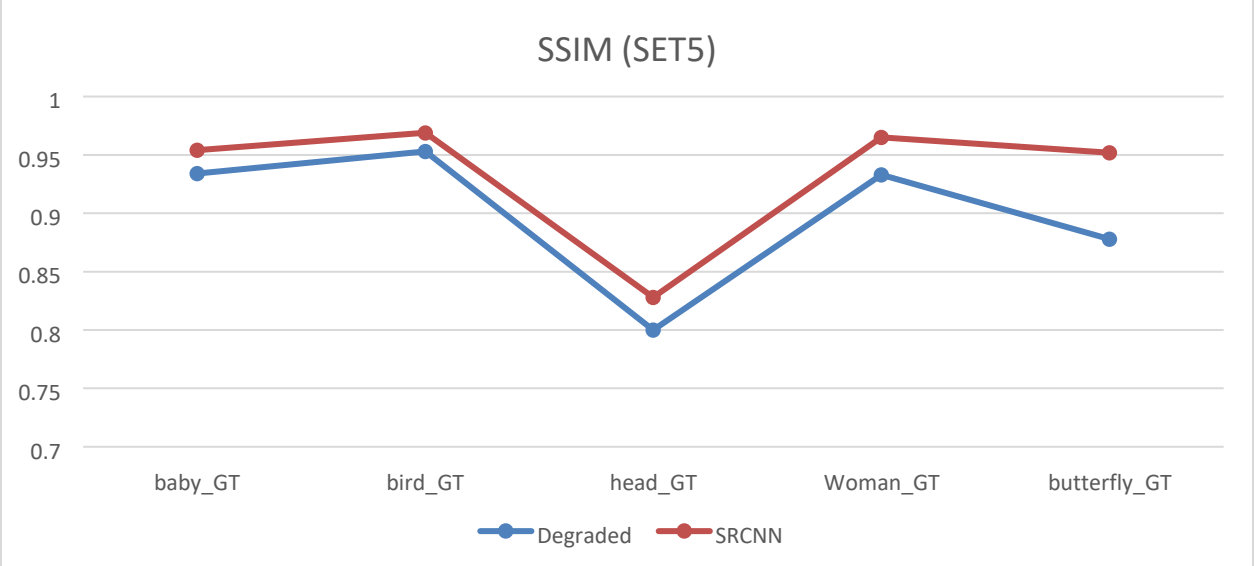
A. RESULTS ON DATA

Calculated values of PSNR, MSE and SSIM on SET 14, SET 5, OUR SET

Images	SET 14					
	DEGRADED				SRCNN	
	PSNR	MSE	SSIM		PSNR	MSE	SSIM
Baboon	22.27	33.9739459	0.633		23.031	970.58	0.709
barbara	25.871	504.67	0.808		23.558	490.915	0.854
coastguard	27.355	358.614	0.752		28.653	265.978	0.828
comic	23.619	847.623	0.828		25.856	506.488	0.899
face	30.865	159.844	0.8		31.628	134.076	0.828
flowers	27.226	369.456	0.869		29.639	211.951	0.898
foreman	32.228	116.78	0.939		36.536	43.306	0.962
lenna	31.341	143.228	0.845		33.097	35.597	0.868
monarch	30.017	194.273	0.943		35.109	60.151	0.965
Pepper	31.449	139.704	0.837		32.786	102.692	0.849
ppt3	24.623	672.753	0.924		27.113	379.223	0.947



Images	SET 5						
	DEGRADED				SRCNN		
	PSNR	MSE	Degraded		PSNR	MSE	SRCNN
baby_GT	34.241	73.454	0.934		36.246	46.3	0.954
bird_GT	32.969	98.459	0.953		36.537	43.301	0.969
head_GT	30.896	158.692	0.8		31.657	133.172	0.828
Woman_GT	29.326	227.812	0.933		33.641	84.337	0.965
butterfly_GT	24.758	652.226	0.878		30.439	176.301	0.952



Images	OUR SET						
	DEGRADED				SRCNN		
	PSNR	MSE	Degraded		PSNR	MSE	SRCNN
1	23.107	953.783	0.786		24.12	755.373	0.842
2	27.937	313.657	0.868		29.156	236.874	0.898
3	27.092	381.022	0.875		29.23	232.879	0.924
4	28.381	283.171	0.863		0.902	165.132	0.902

5	36.59	42.767	0.96		38.201	29.515	0.971
6	38.169	29.736	0.935		39.899	13.966	0.944
7	33.364	89.889	0.92		34.353	71.595	0.942
8	38.779	25.836	0.943		39.651	21.135	0.951
9	35.866	50.524	0.968		40.216	18.558	0.976
10	33.893	79.588	0.937		36.553	43.141	0.961