Image Super-Resolution using Deep Learning

Final Year B.Tech Project Presentation by:

Kunal Suthar(201401131)
Maulik Limbadiya(201401189)

On-campus Project Mentor: Prof. Manjunath V. Joshi

Introduction

- **Resolution:** The number of distinct pixels in each spatial dimension, that can be displayed.
- **Super-Resolution** is the method of obtaining **High-Resolution(HR)** images from their respective **Low Resolution(LR)** images.
- The super-resolved image not only has finer details, but also has a bigger image dimension.

Problem Statement

- As this problem of Super-Resolution(SR) is of **ill-posed** nature, there are multiple ways to approach the problem.
- Multiple methods include **Motion-based SR[1]**, **Zoom-based SR[2]**, and **Learning-based SR[3]**.
- We attempt to solve the problem using **Deep Learning-based SR.**

[1] H.Demirel, S. Izadpanahi,and G.Anbarjafari, "Improved Motion-Based Localized Super Resolution Technique using discrete wavelet transform for low resolution video enhancement", 17th European Signal Processing Conference (EUSIPCO 2009), 1097-1101

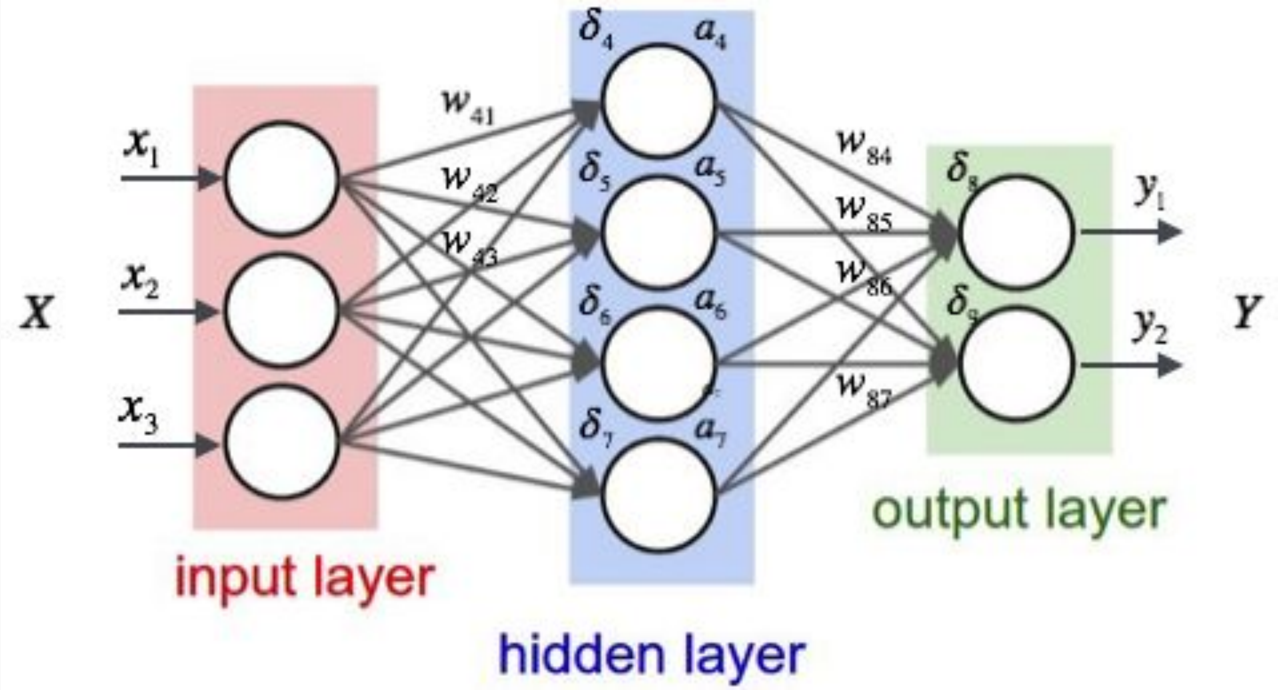
[2] M.V. Joshi, S. Chaudhuri, Super-resolution imaging: use of zoom as a cue, Indian Conf. Comput. Vis. Graph. Image Process.,Ahmedabad (2002)

[3] Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang, "Learning a Deep Convolutional Network for Image Super-Resolution", in Proceedings of European Conference on Computer Vision (ECCV), 2014

Problem Statement

Our goal in this project is to achieve **magnification factors of 2x, 4x and 8x** on an image with minimum depletion in image quality / details, by training a **deep-learning model(a Convolutional Neural Network)** which learns how to **reconstruct** details in the super-resolved(SR) image.

Neural Network



Tools and Technologies Used

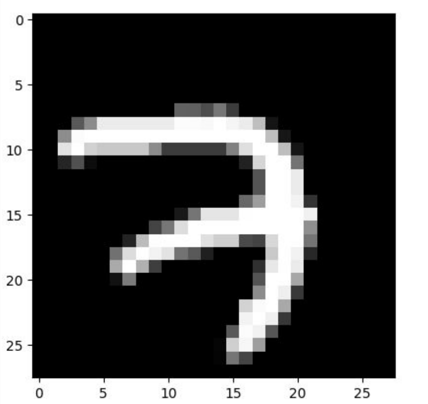
- **Keras :** A neural network API in python.
- **Tensorflow :** Neural-Network API used by Keras in the backend.
- **OpenCV:** A Computer Vision library.
- **NumPy:** Library for performing matrix computations.
- **Python 2.7:** Language used

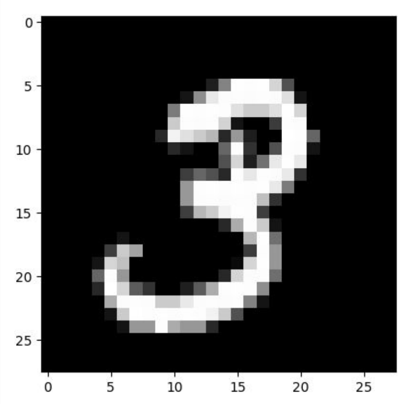
Work Done: Stage 1

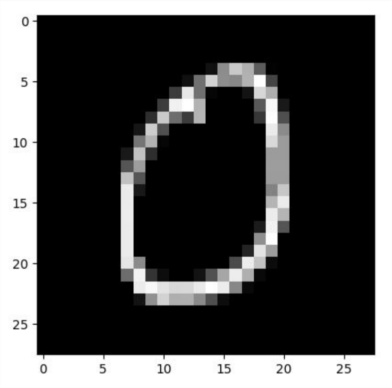
In order to get comfortable with **programming** the **neural network** and working with the **Tensorflow** and **Keras APIs**, we first took up the project of developing a classifier for **MNIST(0-9 digit)** database.

Training & Testing Dataset used:

We used Prof. Yann LeCun's **MNIST** database of handwritten digits. All of them have a dimension of **28 x 28**. It is a good database to try out **learning** and **pattern recognition** techniques on real world data while spending minimal efforts on **pre-processing** and **formatting**.







Implementation Details

We used Tensorflow's Neural-Network API to create the **Convolution Neural Network(CNN)** classifier. The **MNIST** dataset has **60000** images for **training** and **10000** images for **testing**. The **training** was done in batches of **100** images. The **training** includes approximately **20000** such steps or about **34 epochs**.

Our CNN architecture:

1. **Convolutional Layer #1:** Applies 32 5x5 filters (extracting 5x5-pixel subregions), with ReLU activation function.
2. **Pooling Layer #1:** Performs max pooling with a 2x2 filter and stride of 2 (which specifies that pooled regions do not overlap)
3. **Convolutional Layer #2:** Applies 64 5x5 filters, with ReLU activation function
4. **Pooling Layer #2:** Again, performs max pooling with a 2x2 filter and stride of 2
5. **Dense Layer #1:** 1,024 neurons, with dropout regularization rate of 0.4 (probability of 0.4 that any given element will be dropped during training to decrease overfitting).
6. **Dense Layer #2 (Logits Layer):** 10 neurons, one for each digit target class (0–9).

Results

We achieved an accuracy of **96.97%** on the test data by training the model for **20000 steps** or about **34 epochs**, where 1 step indicates training the network on 1 batch(1 batch= 100 image vectors together). For improving the accuracy of the model at every step, we used **gradient descent optimizer** for **backpropagation** at a learning rate of **0.001**, to optimize the filter values. We have also used a **dropout rate** of **0.4**, to address the problem of **overfitting**.

```
INFO:tensorflow:loss = 0.243901, step = 20704
INFO:tensorflow:Loss for final step: 0.243901.
INFO:tensorflow:Starting evaluation at 2018-02-28-23:27:39
INFO:tensorflow:Restoring parameters from mnist_convnet_model/model.ckpt-20704
INFO:tensorflow:Finished evaluation at 2018-02-28-23:27:50
INFO:tensorflow:Saving dict for global step 20704: accuracy = 0.9697, global_step = 20704, loss = 0.101608
{'loss': 0.1016076, 'global_step': 20704, 'accuracy': 0.96969998}
kunal@kunal-Lenovo-Z50-70:~/Desktop/MNIST_dataclassification$
```

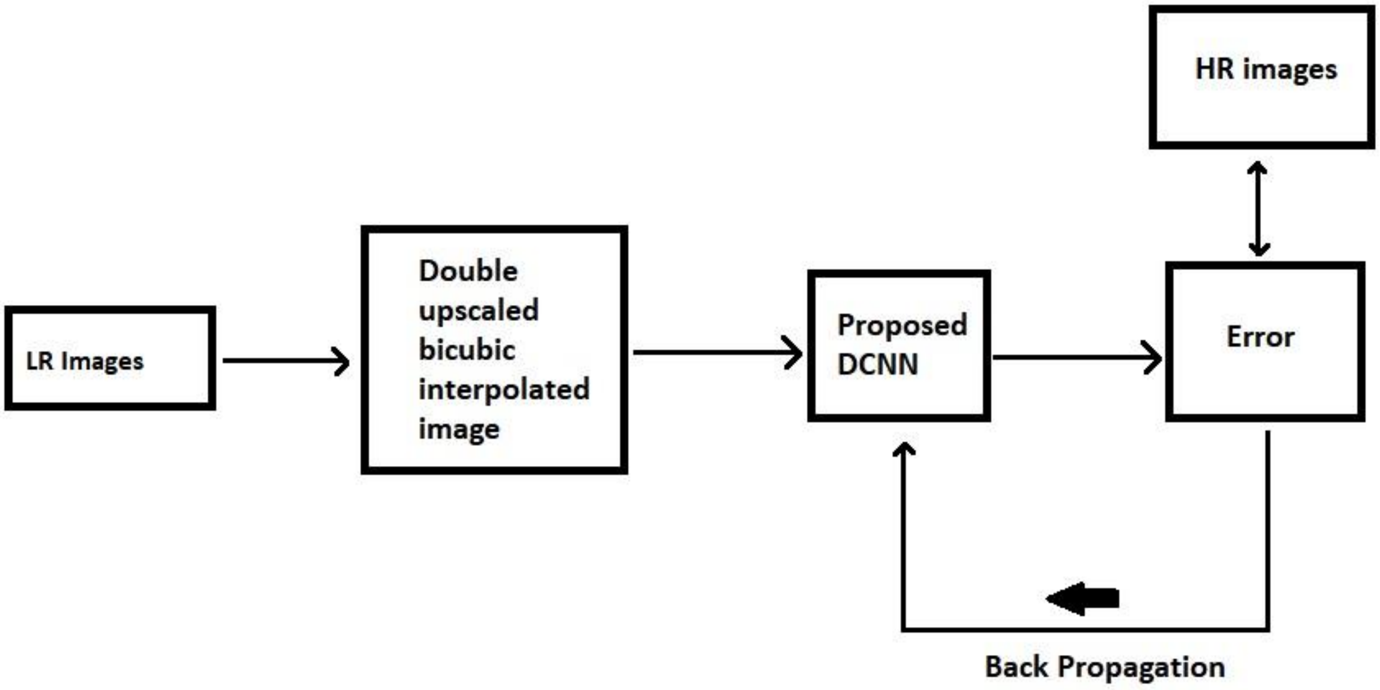
Work Done: Stage 2

Coming back to Image Super-Resolution, we attempt to utilize the **learning property** of the **CNN** here to **reconstruct** details, rather than using it as a **classifier**.

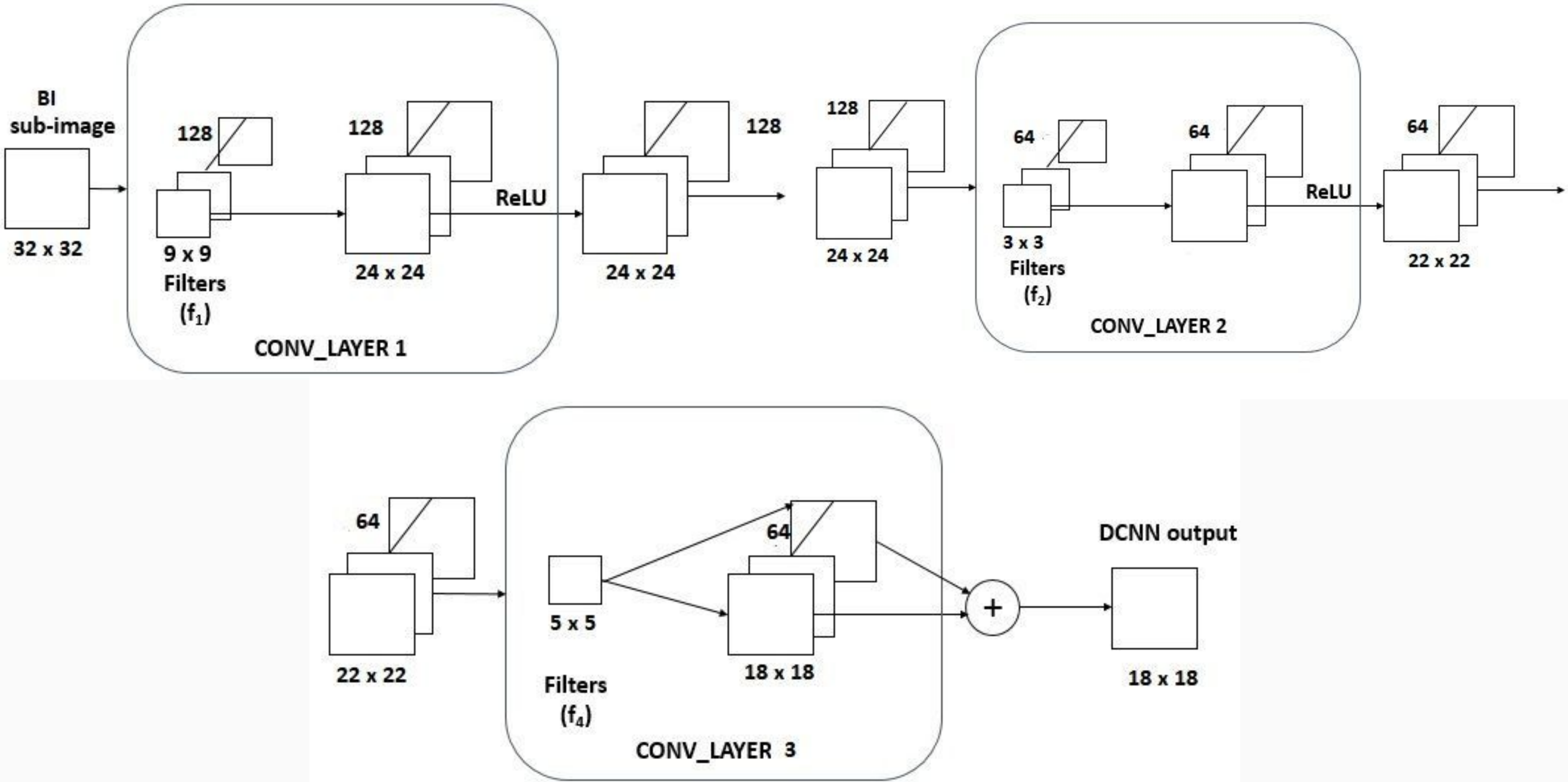
Implementation: Pre-processing images

For creating the **LR** images, we first down-scale the size of the **ground truth** image by 2. We then **double** the size by using **bicubic interpolation,** thus creating the **LR** image. We diminish the size by 2 and then super-resolve it by 2 in order to compare the SR image with the ground truth. For feeding the images in the **DCNN**, we create overlapping patches of the LR input image of size **32 x 32** and the respective HR ground truth. They are stored in the form of vectors in **h5py** files. We retrieve them from the h5py files for the purpose of training and testing.

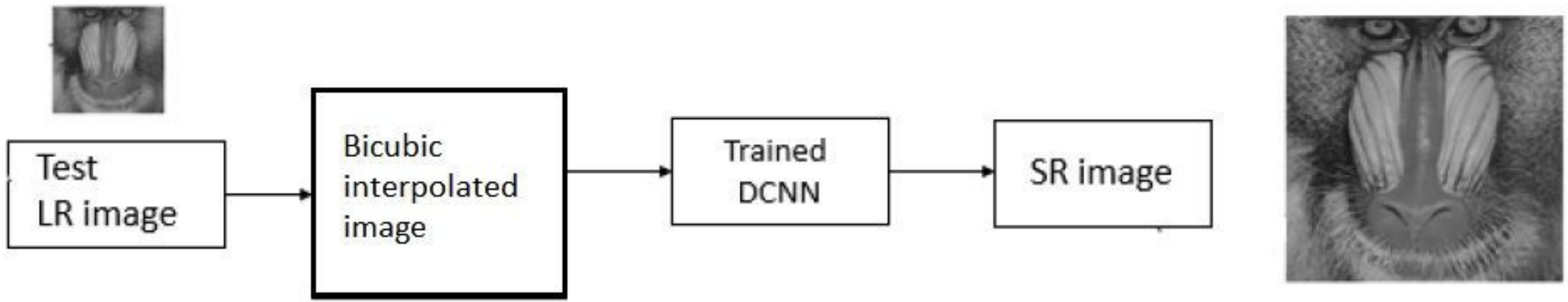
Implementation: Training



Implementation: DCNN Architecture



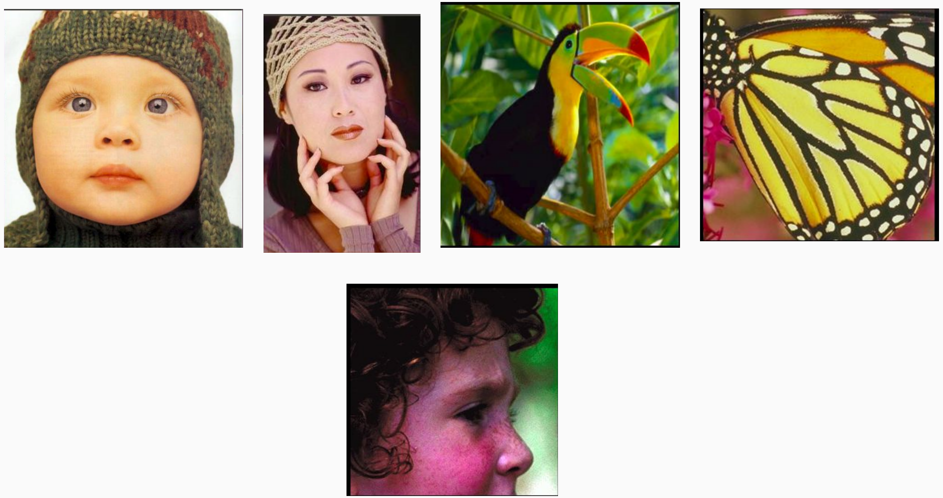
Implementation: Testing



Implementation: Dataset used

We have used the same dataset as in paper **[4]**. Training images comprises 91 color images of varying sizes. The 5 images in the Set5 section of the dataset is used to evaluate the performance.

Sample Training Images



Testing images from Set5 section

[4] Timofte R, De Smet V, Van Gool L (2013) Anchored neighborhood regression for fast example-based super-resolution. In: The IEEE International Conference on Computer Vision (ICCV), ,1920-1927

Implementation: Regularization

As the problem of super-resolution is ill-posed, we further try to improve the image quality by adding a **regularization** parameter in the **loss/error function** of the neural network, which is as follows:

$$loss\ function = MSE(true\ image - reconstructed\ image) + \lambda\left(\sum_{i=1}^{m}\sum_{j=1}^{n}(reconstructed\ image(i,j) - reconstructed\ image(i,j-1))^2 + \sum_{i=1}^{m}\sum_{j=1}^{n}(reconstructed\ image(i,j) - reconstructed\ image(i-1,j))^2\right)$$

MSE= Mean Square Error function, λ= Smoothness parameter,

m,n= dimensions of the image

Results: Low Resolution(LR) Input Image

Input

Results: Bicubic Interpolation v SR

Bicubic Interpolated Image

SR

Results: Ground Truth v SR

Ground Truth

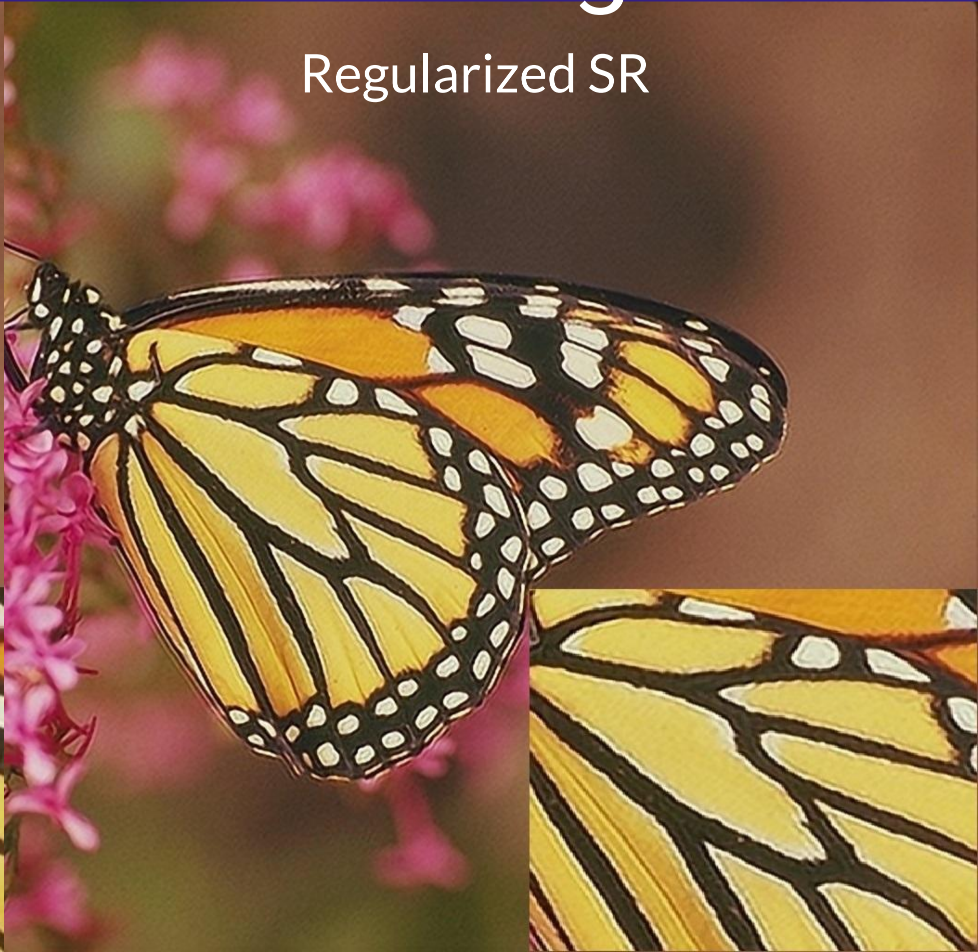
SR

Results: 2x Super-Resolved(SR) Image

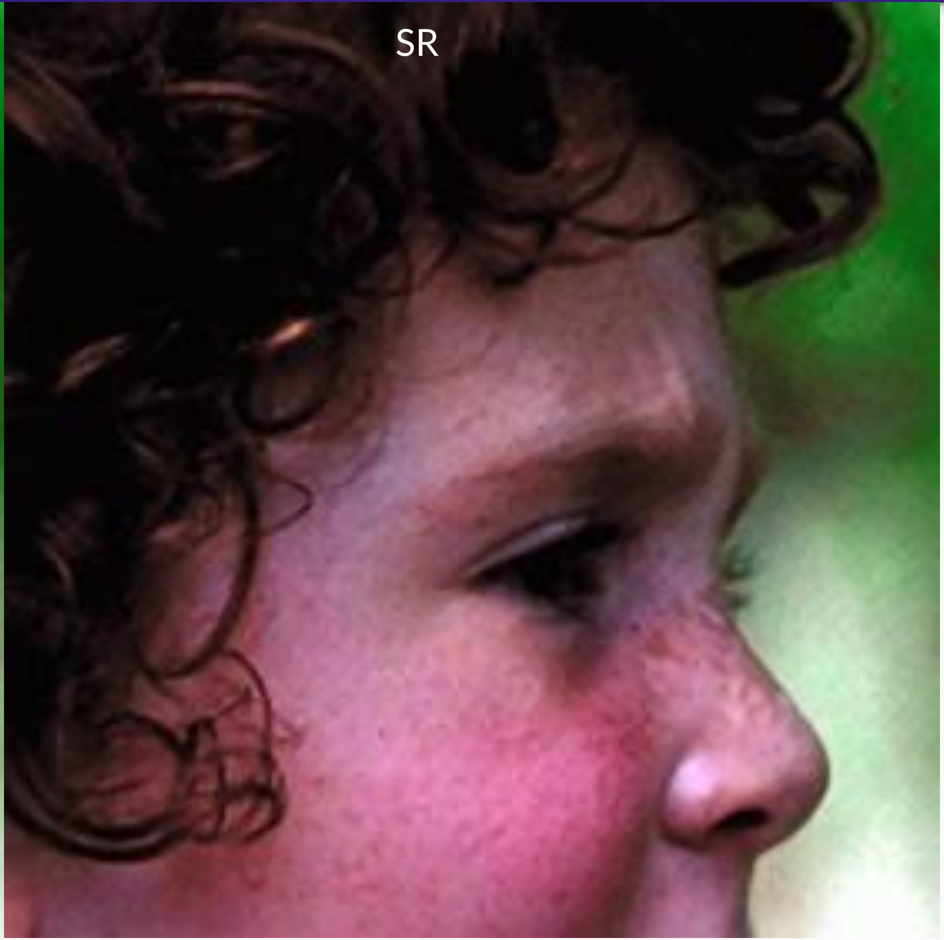
Non-Regularized SR

Regularized SR

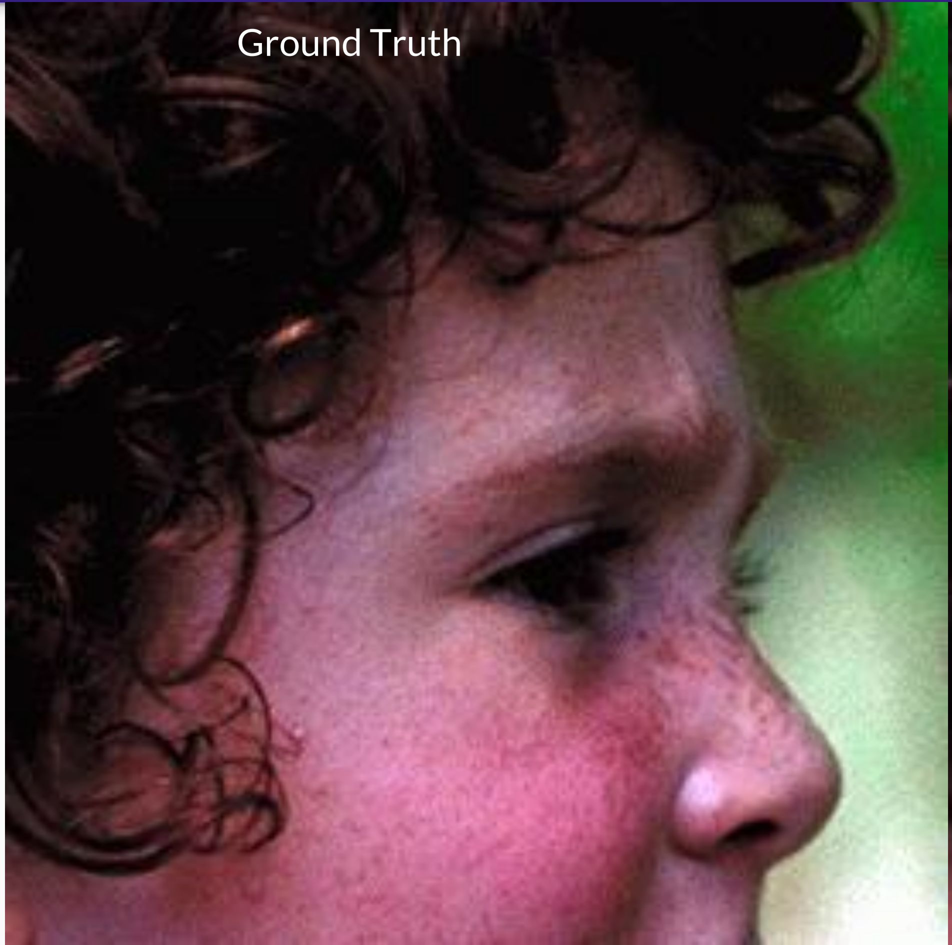
Results: 4x Super-Resolved(SR) Image

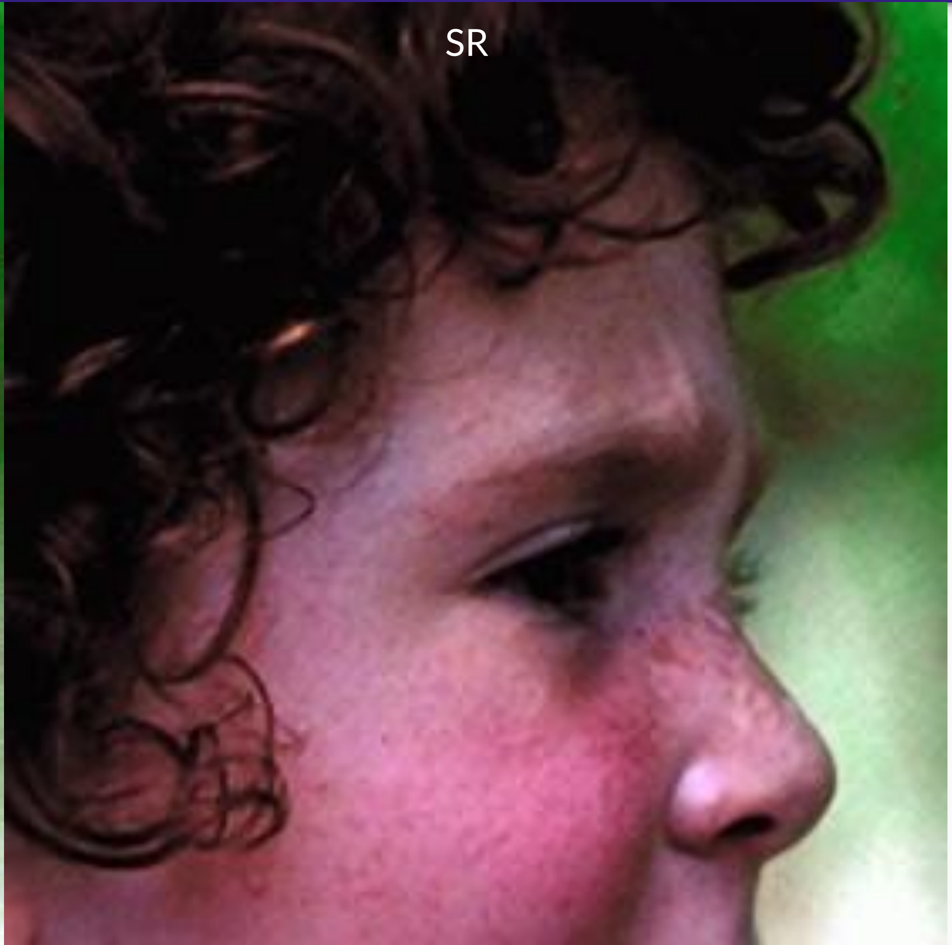


Results: Bicubic Interpolation v SR

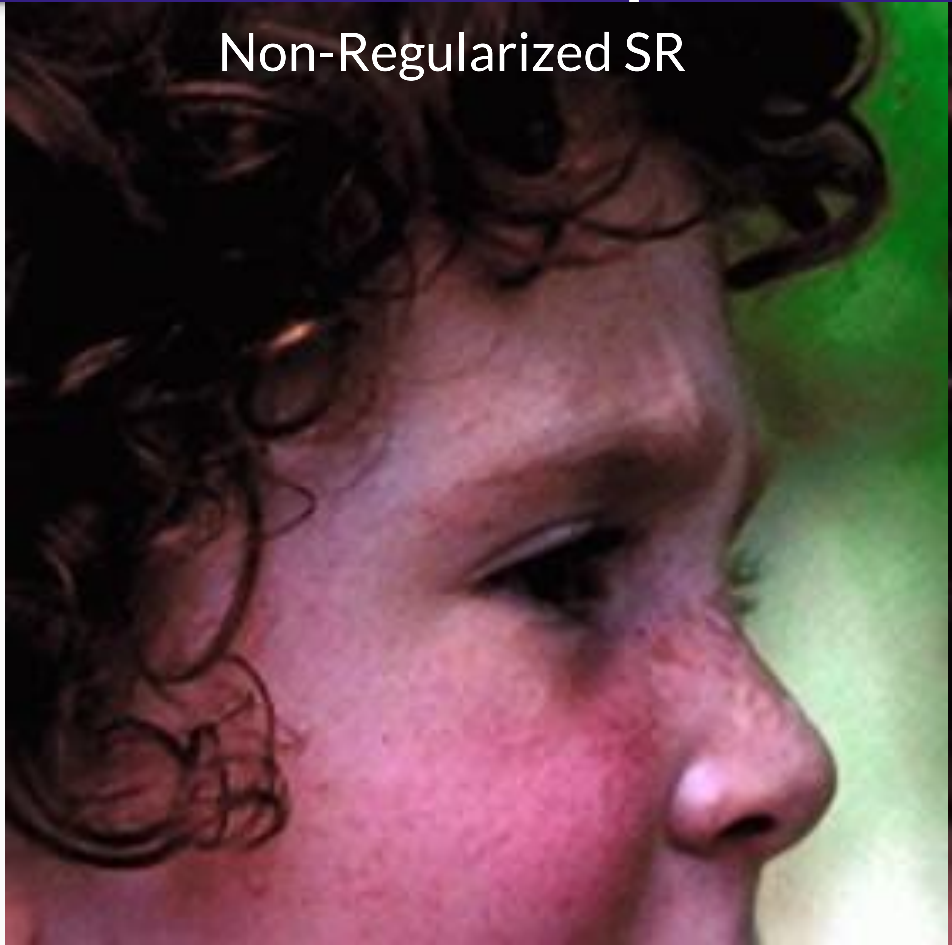


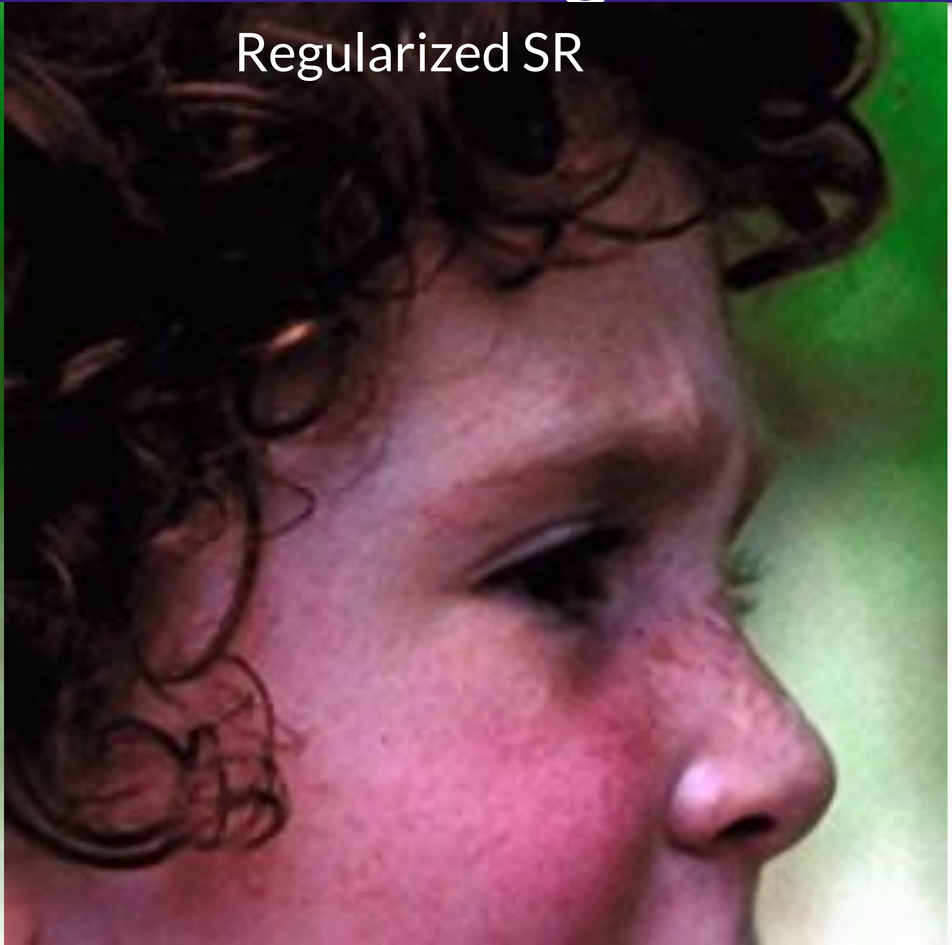
Results: Ground Truth v SR



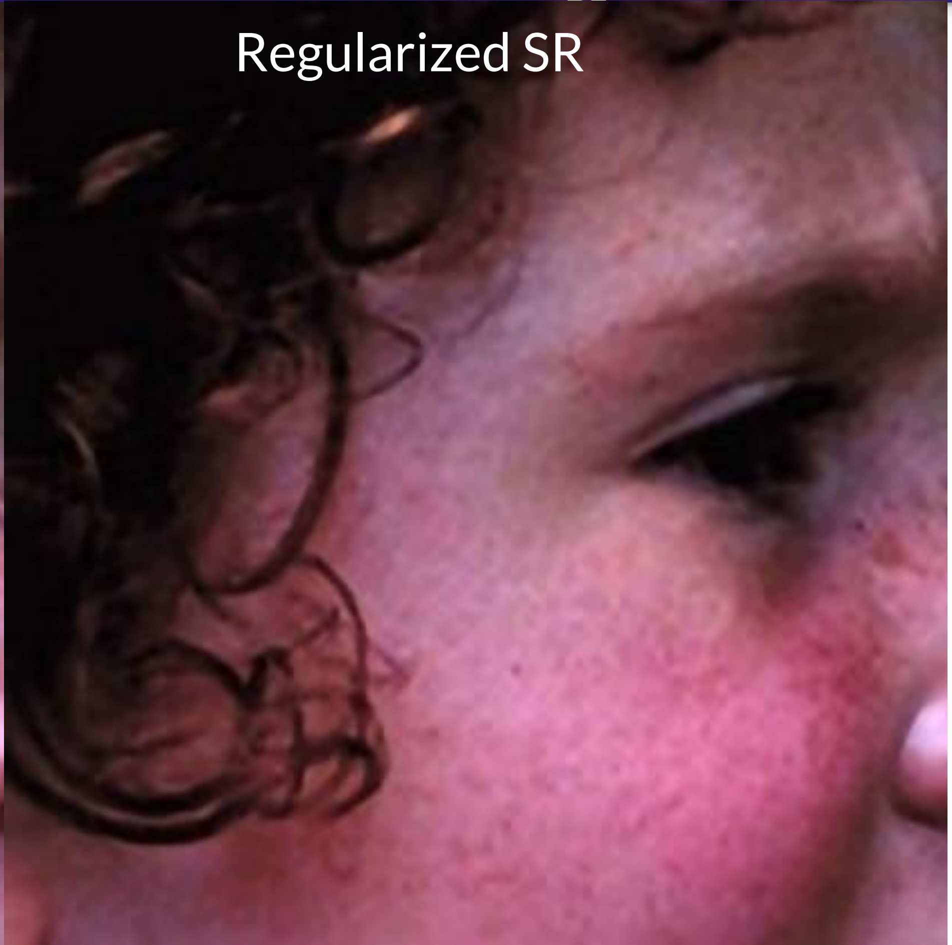


Results: 2x Super-Resolved(SR) Image





Results: 4x Super-Resolved(SR) Image



Results: Bicubic Interpolation v SR

Bicubic Interpolated Image

SR

Results: Ground Truth v SR

Ground Truth

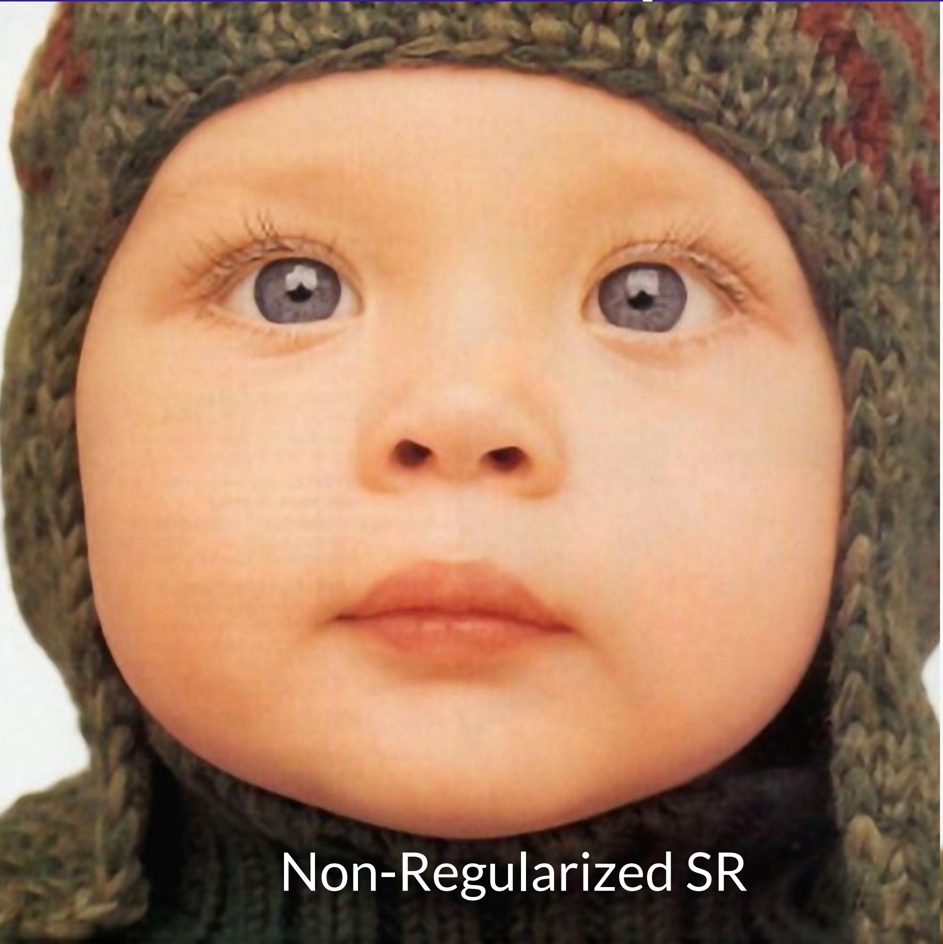
SR

Results: 2x Super-Resolved(SR) Image

Non-Regularized SR

Regularized SR

Results: 4x Super-Resolved(SR) Image



Non-Regularized SR

Regularized SR

Results

The metric used for quantitative comparison here is the **PSNR(Peak Signal to Noise Ratio)[5]**, measured in dB and the formula to compute it is as follows:

$$\mathbf{PSNR} = 10 * \log_{10}((255)^2 \ /MSE)$$

Where MSE is the **Mean Square Error** between the ground truth image vector and the Super-resolved image vector.

[5] Prathmesh R. Madhu and M.V. Joshi, “Digital Heritage Reconstruction using deep learning-based Super-resolution”, 1-22

Results: PSNR Values

Test images	Bicubic-Interpolated image(in dB)	Non-Regularized SR (in dB)	Regularized SR (in dB)
Baby	36.00	31.56	31.86
Bird	36.75	31.52	32.84
Butterfly	27.01	23.99	25.88
Head	33.55	30.60	30.69
Woman	31.76	27.23	28.69

Conclusion

- We attain higher magnification by reusing the neural network, and using the super-resolved image as input generates 4x magnification and similarly we can do it for 8x.
- We also tweak the loss function of the network with a regularization parameter to produce a better output.
- The results obtained using this approach, show observable qualitative and quantitative improvements in the image quality, as compared to a digital zoom performed using bicubic interpolation.

Thank You