Computer Vision Project Report 1

Deep Network Cascade for Image Super Resolution

Team
Pooja Srinivas 20171403
Srivyshnavi T 20161235
Nitin 2018802004

Abstract
In visual information processing, high-resolution (HR) images are still desired for more useful information. However, due to the limitation of physical devices, we can only obtain low-resolution (LR) images of the specific object in some scenes such as a long-distance shooting. To handle this problem, the superresolution (SR) technique is usually employed to recover the lost information in the source image. In this paper, the idea proposed is of a new model called deep network cascade (DNC) to gradually upscale low-resolution images layer by layer, each layer with a small scale factor.
DNC is a cascade of multiple stacked collaborative local auto-encoders. In each layer of the cascade, non-local self-similarity search is first performed to enhance high-frequency texture details of the partitioned patches in the input image. The enhanced image patches are then input into a collaborative local auto-encoder(CLA) and a non-local self-similarity search. Then the super-resolution result is refined and further fed into next layer until the required image scale is achieved.

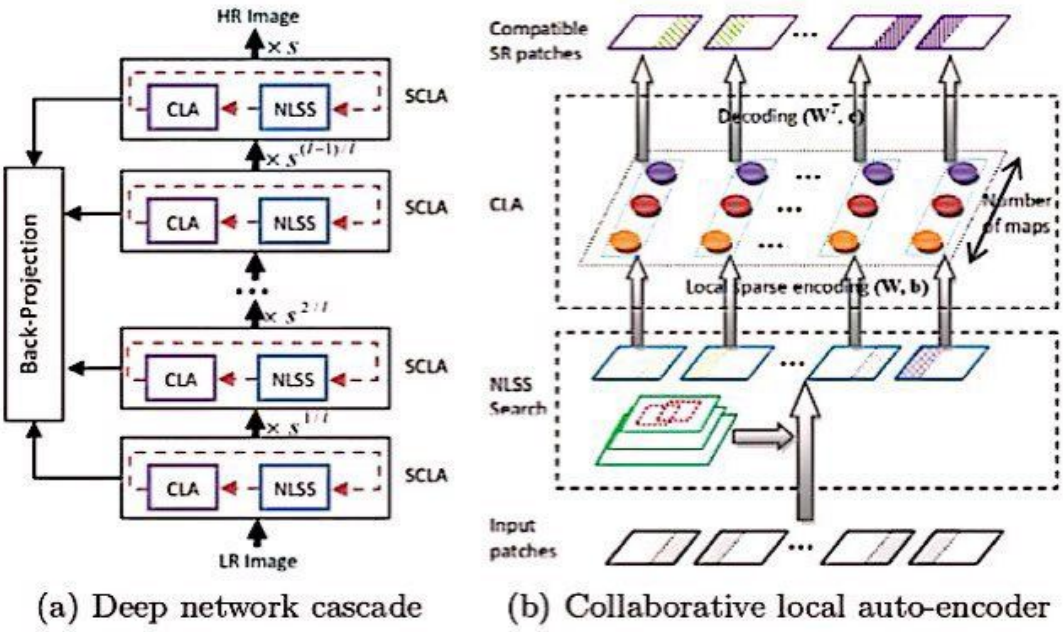
Introduction
Conventional methods
The conventional super-resolution methods attempt to recover the source image by solving the ill-posed inverse problem, $y = Hx + v$, where x is the unknown high resolution image to be estimated, y is the observed low resolution image, H is the degradation matrix, and v is the additional noise vector. Under the scarcity of observed low resolution images, the inverse process is a underdetermined problem, thus the solution is not unique. To find a reasonable solution, some sophisticated statistical priors of natural images are usually incorporated into the reconstruction process. However, these reconstruction based methods have a limit of magnification factor.

In this paper, a deep learning scheme called deep network cascade(DNC) is proposed, to gradually upscale low-resolution images layer by layer.

- The first step in this method involves, reasonably enhancing the high-frequency texture details through a non-local self-similarity (NLSS) search on the input image in multi-scale. The image after NLSS should be able to bypass the assumption on the image degradation process.
- The next step is to take the NLSS results as the input to the collaborative local auto-encoder (CLA) to suppress the noises and collaborate the overlapping reconstructed patches.
    - In CLA, another extra step of weight-tying on all patches and L 1 sparse constraint on hidden neurons is performed. This is done to reduce the learnable parameters and make auto-encoder easily controllable.
- Each loop is comprised of the two steps, which forms a cascade layer, named stacked collaborative local auto-encoder (stacked CLA or SCLA). The stacked collaborative local auto-encoder refines the super-resolution images to output the final image.
- Multiple SCLA models can be successively concatenated into the deep network cascade, where the higher layer takes the output SR image of the lower layer as input.

With the increase of network layers, the magnification factor of the learned SR image can be enlarged gradually.

- A back- projection technique is also included in the model to constrain the super resolved image in each layer. This is necessary because, in the deep network synthetic error textures (e.g., noises, artifacts, etc.) will propagate and spread in the next network layers, which leads to a large deviation from the source high resolution image.
- A greedy layer-wise optimization strategy is adopted to train the DNC model.



(a) Deep network cascade (b) Collaborative local auto-encoder

Method

NLSS

The NLSS is applied on the input image itself to enhance textural high-frequency information. Since natural image patches recur many times within an image and even across different scales, we can always find some similar patches for a given patch. Concretely, in a network unit, we denote the input image by x, which comes from the SR image of the former layer or the source LR image in the first layer. Before super-resolution,the bicubic interpolation is imposed on the input image to generate the initialized SR image (marked as x again for simplification). For a patch x_i extracted from x, we perform the non-local self similarity search in multi-scale images($s^{1/l}$, scaling factor), which may come from blur and successive downscaling versions of x. Given x_i , suppose the top K nearest neighbors, x^1_i , $\cdots$, x^k_i , are chosen from these multi-scale images, we can roughly estimate a new enhanced patch. The estimated patch $\hat{x}_i$ usually contains more abundant texture information than the input patch x_i .

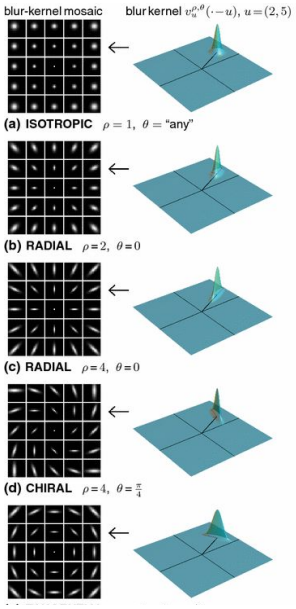
Given x_i,

K nearest neighbours : $x_i^1, x_i^2, \ldots x_i^k$

enhanced new patch $\hat{x}_i$:

$$\hat{x}_i = \sum_{j=1}^{k} \bar{w}_i^j \cdot x_i^j$$

$\bar{w}_i^j$ = set to Guassian kernel with normalization.



CLA

CLA is used to suppress the noises as well as collaborate the over-lapping patches. CLA has 2 constraints. First, a weight-tying scheme like convolutional network is used to reduce the parameter space as well as preserve a certain flexibility to other variances. Second, the compatibility constraint on overlapping patches is added into auto-encoder to induce more smooth and natural textures for the integrated SR image. Actually, the output patch z_i of CLA can be combined into a SR image x by averaging the overlapping part among patches, which is computed from the following equation,

$$\widetilde{\mathbf{x}} = (\sum_{i=1}^{n} \mathcal{F}_i^{\mathsf{T}} \mathcal{F}_i)^{-1} \sum_{i=1}^{n} (\mathcal{F}_i^{\mathsf{T}} \mathbf{z}_i).$$

Formulation

The input to CLA is obtained from NLSS search. CLA contains two constraints: the sparse constraint and the compatibility constraint. Concretely, CLA can be formulated into the following optimization problem,

$$\min_{W,b,c} \; l(x, W, b, c) + \gamma g(U) + \eta h(z)$$

$\tilde{x} = \left(\sum_{i=1}^{n} F_i^T F_i\right)^{-1} \sum_{i=1}^{n} (F_i^T z_i)$

z_i – output patch of CLA

$\tilde{x}$ – SR image

$z_i \cong F_i \tilde{x}$ (ideally)

⇓

Compatibility constraint

Sparsity using hyperbolic tangent function.

$y_i = \sigma(W \hat{x}_i + b) \Rightarrow L_1$ norm

$\sigma(x) = \dfrac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}}$ (gain is a)

encoder level :

$u_i = \dfrac{y_i}{\sqrt{y_i^T y_i}} \Rightarrow L_2$ norm

decoder level :

$z_i = W^T u_i + c$

L_1 norm to regulaise the sparsity :

$g(U) = \frac{1}{n} \sum_{i=1}^{n} \|u_i\|_1$

$U = [u_1, u_2 \cdots u_n]$

Regulaisation :

$h(z) = \frac{1}{2n} \sum_{i=1}^{n} \| z_i - F_i \tilde{x} \|^2$

$Z = [z_1, z_2 \cdots z_n]$

$\tilde{x}$ = SR image from compatibility constriant

Algorithm

Input: LR image x, patch size p, upscale factor s, stacked layer number l, neighbor number K, balance parameters γ, η.

Output: SR image x, W, b, c.

1: Set t = 1. Initialize W^t , b^t , c^t with (denoising) auto-encoder;

2: repeat

3: Initialize $x^{(t)}$ by interpolating $x^{(t-1)}$ ($x^{(0)} = x$) with a scale factor $s^{1/l}$.

4: repeat

5: Sample overlapping patches $X^{(t)} = [F_1 x^{(t)}, \cdots, F_n x^{(t)}]$ from $x^{(t)}$.

6: Compute $X^{(t)}$ by using the NLSS search.

7: L-BFGS optimization for new $W^{(t)}$, $b^{(t)}$, $c^{(t)}$.

8: Predict new SR image $x^{(t)}$.

9: Perform the back-projection operation for $x^{(t)}$.

10: $W^{(t+1)} = W^{(t)}$; $b^{(t+1)} = b^{(t)}$; $c^{(t+1)} = c^{(t)}$; $x^{(t+1)} = x^{(t)}$

11: $t = t + 1$.

12: **until** reach a satisfied solution.

13: **until** reach l layers.

14: **return** SR image x (t) .

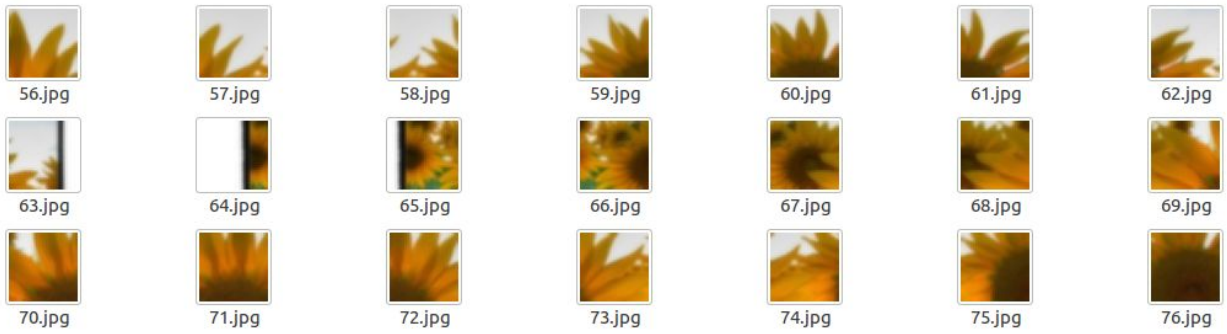
NLSS Result:

gray_image

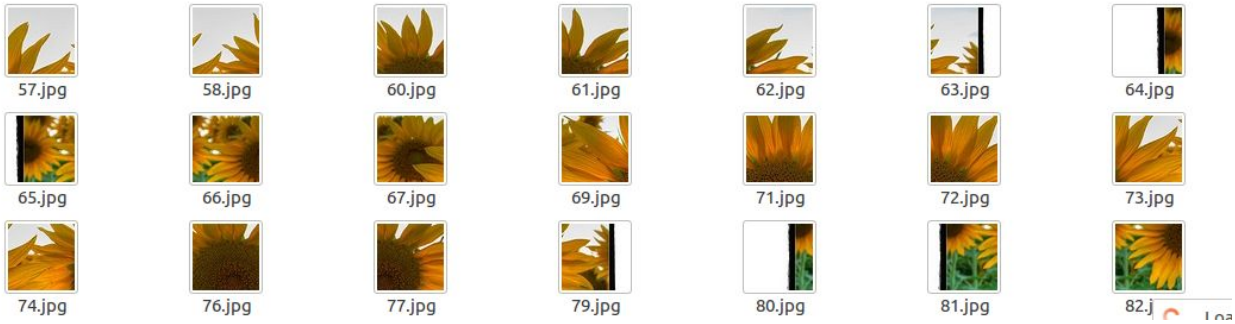
NLSS_image

Training Data:

We used around 1000 high resolution images and generated 1000 low resolution images to further train the autoencoder we collected 110000 image patches to train the auto encoder Below are some of the example of data we used to train the auto encoder



Train data



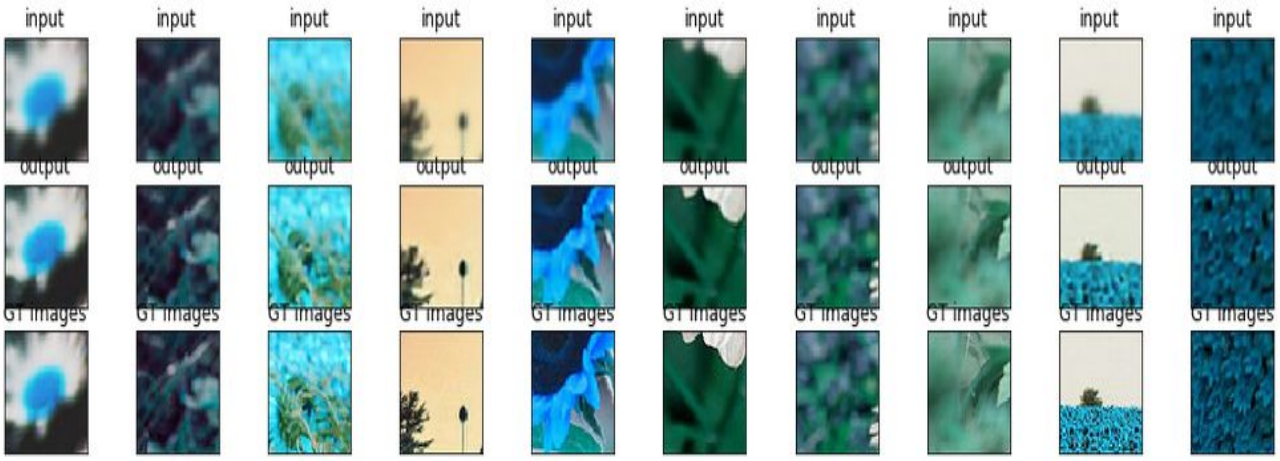
Label data

Summary of autoencoder network:

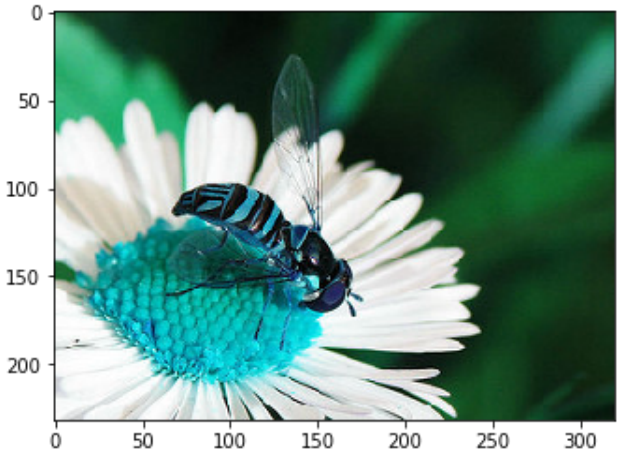
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 56, 56, 32)	896
max_pooling2d (MaxPooling2D)	(None, 28, 28, 32)	0
conv2d_1 (Conv2D)	(None, 28, 28, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 14, 14, 32)	9248
up_sampling2d (UpSampling2D)	(None, 28, 28, 32)	0
conv2d_3 (Conv2D)	(None, 28, 28, 32)	9248
up_sampling2d_1 (UpSampling2	(None, 56, 56, 32)	0
conv2d_4 (Conv2D)	(None, 56, 56, 3)	867

Total params: 29,507
Trainable params: 29,507
Non-trainable params: 0

Result of autoencoder :



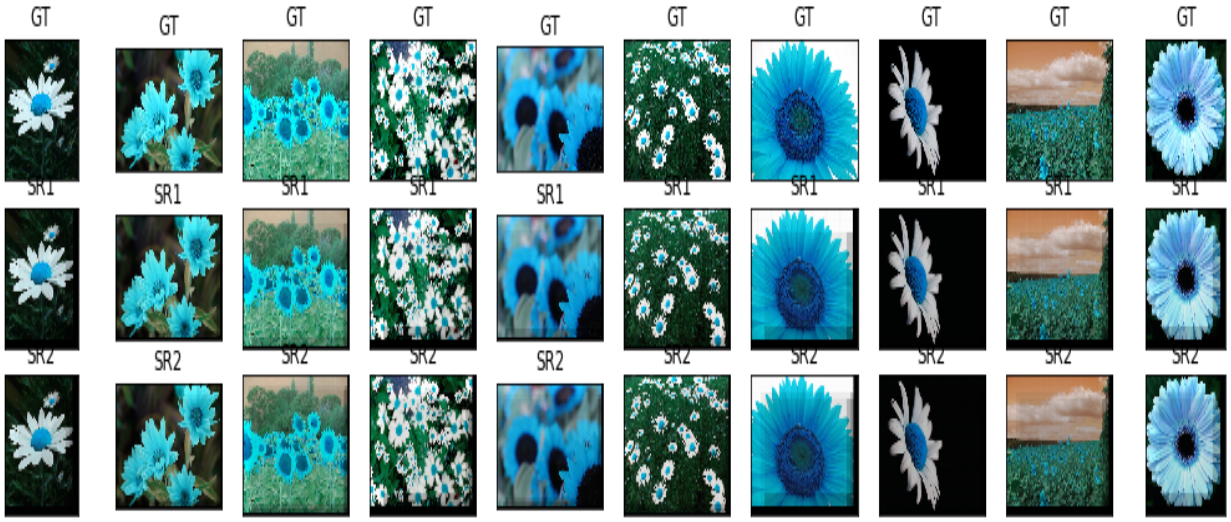
Result with one level Super resolution:



Input image

super resolution image (missed the boundary condition)

Result with stacked auto encoders:



GitHub link :

https://github.com/savera2020/super-resolution