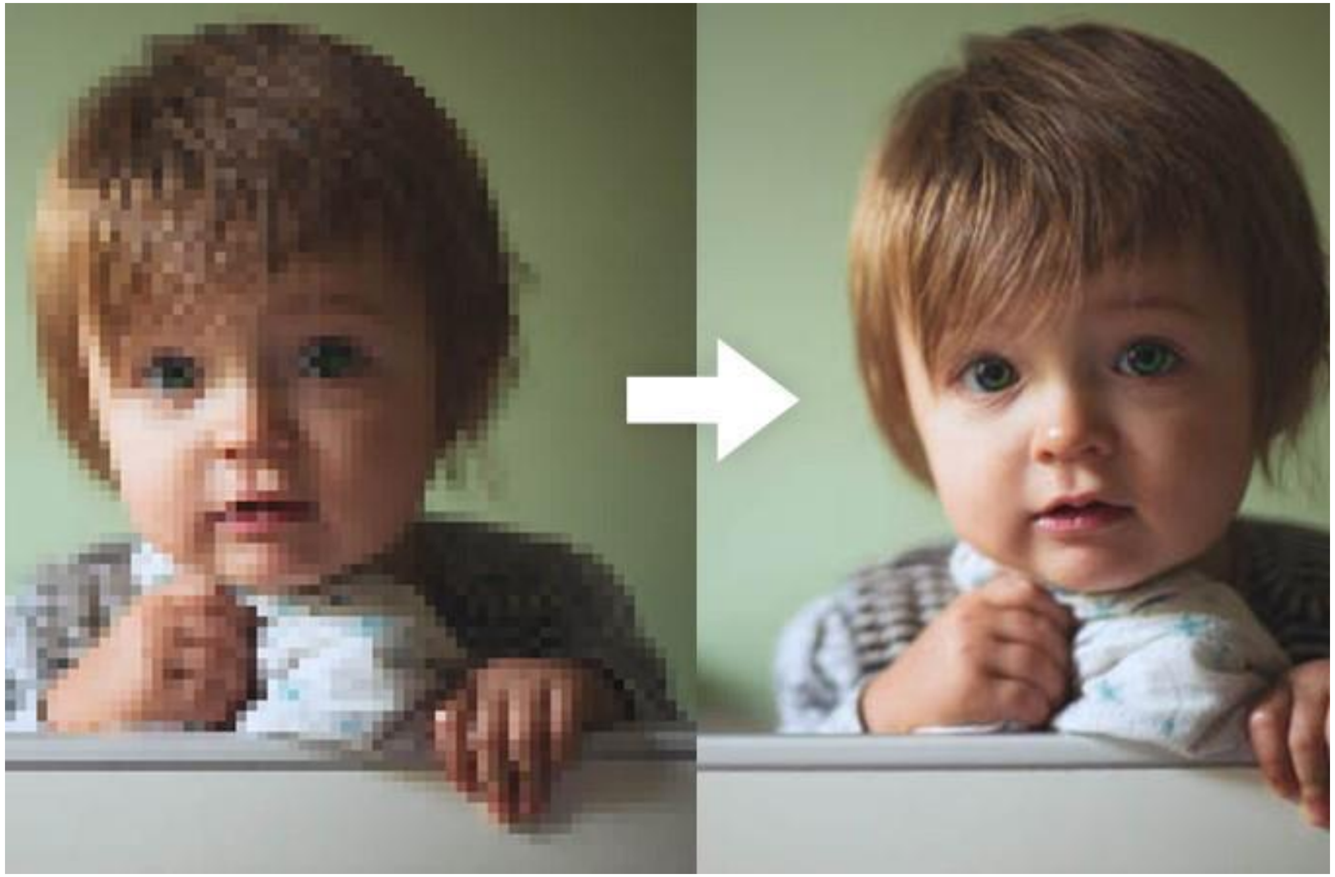
PA1 : Super Resolution

09.15.2022

What is Super Resolution?

High-resolution image reconstruction from Low-resolution image



Reference: https://medium.com/@abhishekmalhan007/enhancement-of-images-using-super-resolution-fc36bfd58d9a

What is Super Resolution?

$$\uparrow I^l = (I^h * k)$$

I^l : Low Resolution Image
I^h: High Resolution Image
k : Kernel
↑: Upsampling
*: Convolution

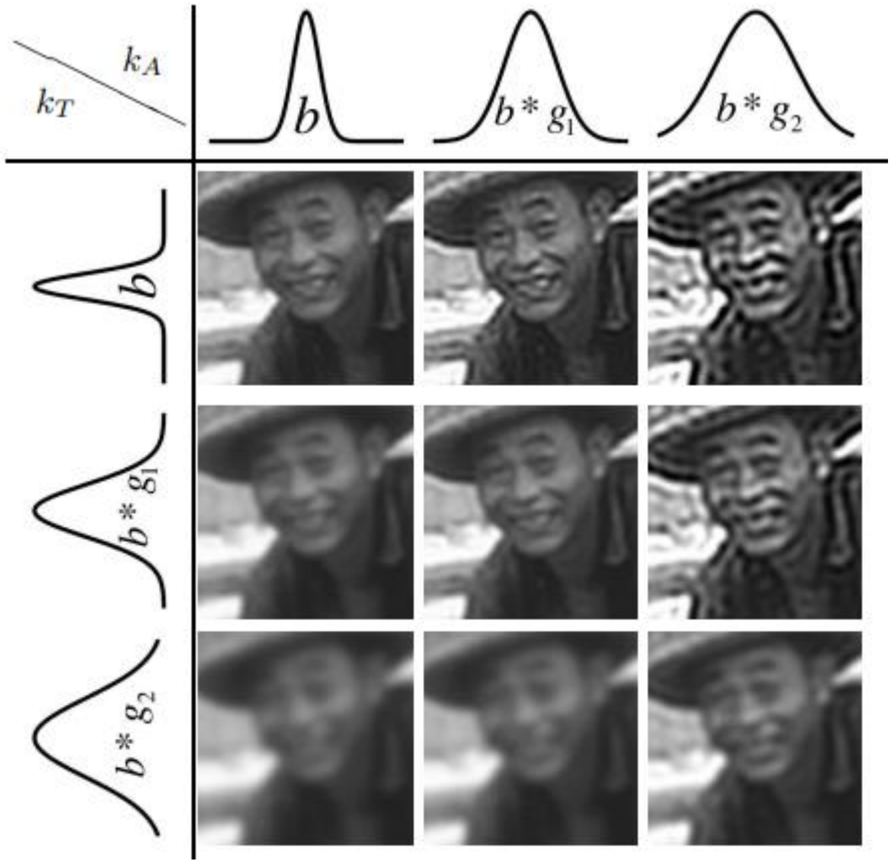
We only have I^l. (we do not know M and I^h)

There can be many pairs of I^h and K that make same I^l.

This problem is called the “Ill-posed Problem”

What is Super Resolution?

Inverse problem



$I^h = (\uparrow I^l *^{-1} \ k)$ (from $\uparrow I^l = (I^h * k)$)
where $*^{-1}$ is deconvolution

We can reconstruct I^h by applying deconvolution on $\uparrow I^l$.
k_T : kenrel which synthesizes the low-resolution image
k_A: kenrel used in deconvolution
b : bicubic interpolation
g_s: gaussian kernel whose standard deviation is s

If k_T and k_A are not equal, then result of deconvolution will be degraded.

So, if we know kernel, then we can get sharp image.

Reference: Efrat, Netalee, et al. "Accurate blur models vs. image priors in single image super-resolution." *Proceedings of the IEEE International Conference on Computer Vision*. 2013.

Preliminary : Gradient Descent

Minimum of convex function (e.g. f = $x^2 + 4x + 4$) is when x satisfies $\frac{\partial f}{\partial x} = 0$ (e.g. $2x + 4 = 0$).

But computers are difficult to solve the equation ($\frac{\partial f}{\partial x} = 0$) direclty. Instead, computer can solve this problem through **Gradient Descent.**

Gradient Descent Algorithm

$x_{t+1} = x_t - \boxed{\mathrm{lr}} * \frac{\partial f}{\partial x}(x_t)$ where x_t: value of x at iteration t

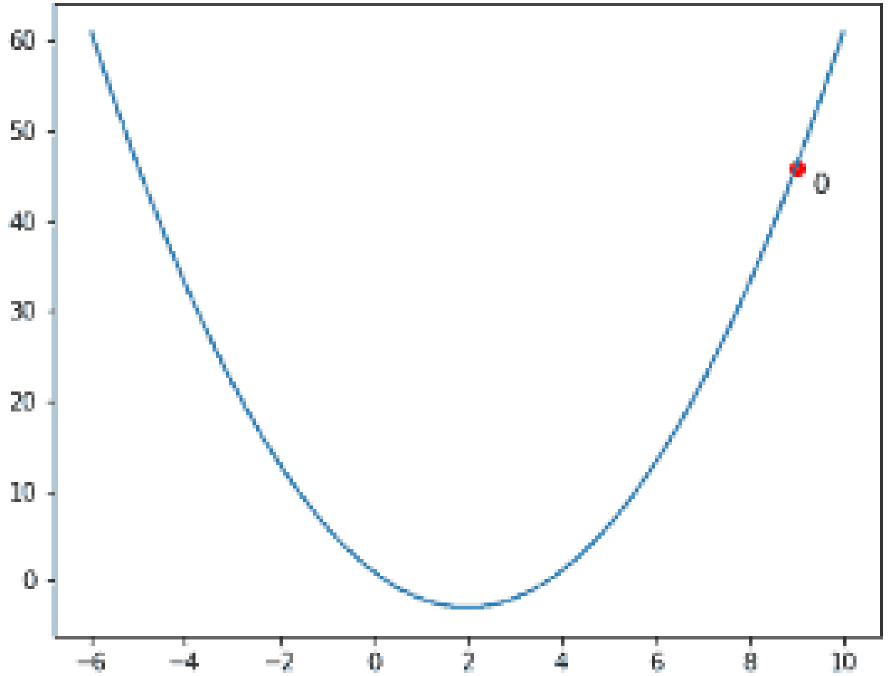
lr: step size(hyper-parameter)

Step1. Choose initial point (initial value of x)

Step2. Calculate gradient at the point x_t ($\frac{\partial f}{\partial x}(x_t)$, e.g. 2 x_t + 4)

Step3. Update position (x_{t+1}) by adjusting it in the opposite direction to gradient(lr $* \frac{\partial f}{\partial x}(x_t)$). Step size(lr) is a scale factor of gradient.

Step4. Repeat Step2 and 3 until gradient become tiny.

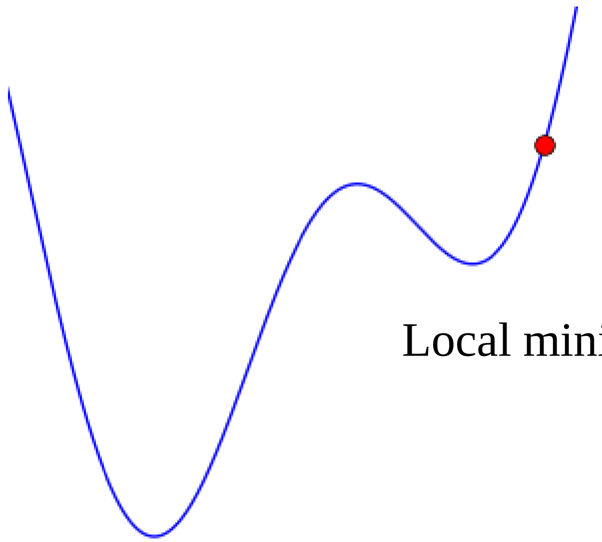


Blue line represents energy function we want to optimize. The red line indicates the progression of gradient descent.

Reference : https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

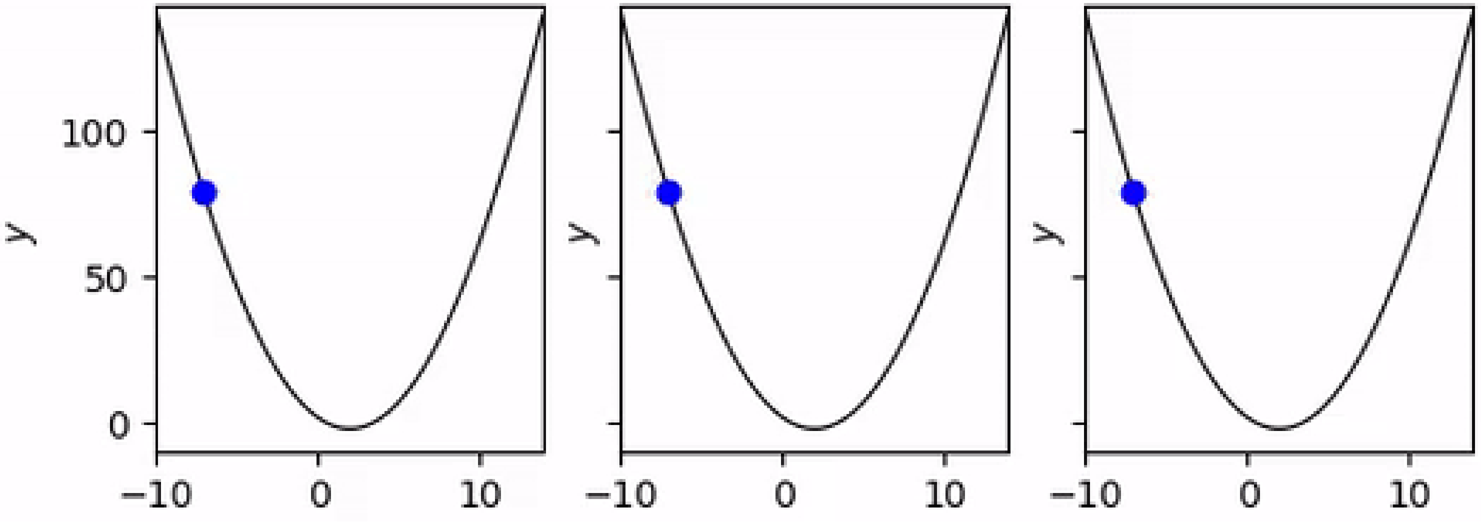
Preliminary : Gradient Descent

Impact of step size



Local minimum

Global minimum



Small step size Appropriate step size Large step size

If step size is too large, it can diverge.
If step size is too small, convergence takes long time.

If step size is small, then
It may converge to the local minimum instead of the global minimum.

Reference
https://medium.com/x8-the-ai-community/gradient-descent-intuition-how-machines-learn-d29ad7464453
https://www.kaggle.com/code/ohseokkim/bird-species-standing-on-the-shoulders-of-giant

Overview of PA1

Minimize this loss function(E) using Gradient Descent that you have to implement

$$E = \left(I^{l} - D(\,I^{h})\right)^{2} + Prior$$

$$I_{t+1}^{h} = I_{t}^{h} - \alpha \frac{\partial E}{\partial I_{t}^{h}}$$

D : Downsampling

t : iteration of Gradient descent

α : step size

I_{t}^{h} : High-resolution image at iteration t

Prior will be discussed in method 2

Files/Libraries

Language : Python
Libraies : numpy, openCV
In openCV, only the following functions are available:
cv2.imread, cv2.imwrite, cv2.cvtColor, cv2.Laplacian, cv2.Sobel, cv2.resize
Files:
Upsampled.png : Initial of gradient descent (I_0^h) (Unsampled image from low-resolution image)
HR.png: High resolution image (Ground Truth) (I_{gt})

Upsampled.png

HR.png

Method 1 : Gradient Descent (6pts)

Optimize this loss function without prior using Gradient Descent algorithm that you have to implement

Loss : $E = ((I^l - D(I^h)))^2$

Update: $I_{t+1}^h = I_t^h - \alpha \frac{\partial E}{\partial I_t^h}$

Gradient : $\frac{\partial E}{\partial I_t^h} = \mathrm{U}(D(I^h) - I^l)$

t : iteration of Gradient descent
α : step size
I_t^h : High resolution image candidate at iteration
I^l : Low resolution image(input)

U: Upsampling (Bilinear)
D: Downsampling (Bilinear)

Please use "**Upsampled.png**" as initial high resolution(I_0^h)
Please get low resolution image (I^l) by applying D on **"HR.png"** (I^{gt})

Method 1 : Gradient Descent (6pts)

Pseudo Code

I^h= load(Upsampled.png)

I^h= Grayscale(I^h)

height, width = I_0^h.shape

I^{gt} = load(HR.png) #Ground Truth

I^{gt}= Grayscale(I^{gt})

I^l = bilinear(I^{gt}, (h//4,w//4)) # Input of the algorithm (low resolution image) *#1pt*

Max_iteration = 1000

#Gradient Descent

Iterate Max_iteration times:

I_t^d = bilinear(I^h , (height//4,width//4))

Calculate gradient : $\frac{\partial E}{\partial I_t^h}$ *# 2 pts*

Update I_t^d *# 1 pt*

Write I^h *#2pts*

You can tune step size and Max_iteration.

Method 2 : Super Resolution with prior (9pts)

Equation of Prior

$$\text{Loss} : E = \boxed{\left(I^{l} - D\left(I_t^h\right)\right)^2} + \boxed{\beta\left(\nabla I_t^h - \nabla I^{\mathrm{T}}\right)^2}$$

Loss from Method 1 Prior

$$\text{Update:}\quad I_{t+1}^h = I_t^h - \alpha \frac{\partial E}{\partial I_t^h}$$

$$\text{Gradient} : \frac{\partial E}{\partial I_t^h} = \boxed{\mathrm{U}\left(D\left(I_t^h\right) - I^{l}\right)} + \boxed{\beta\left(\nabla^2 I_t^h - \nabla^2 I^{T}\right)}$$

Gradient from Method 1 Gradient of Prior

α, β: hyper parameters

$\nabla \mathrm{I}^{\mathrm{T}}$: Gradient we are aiming for

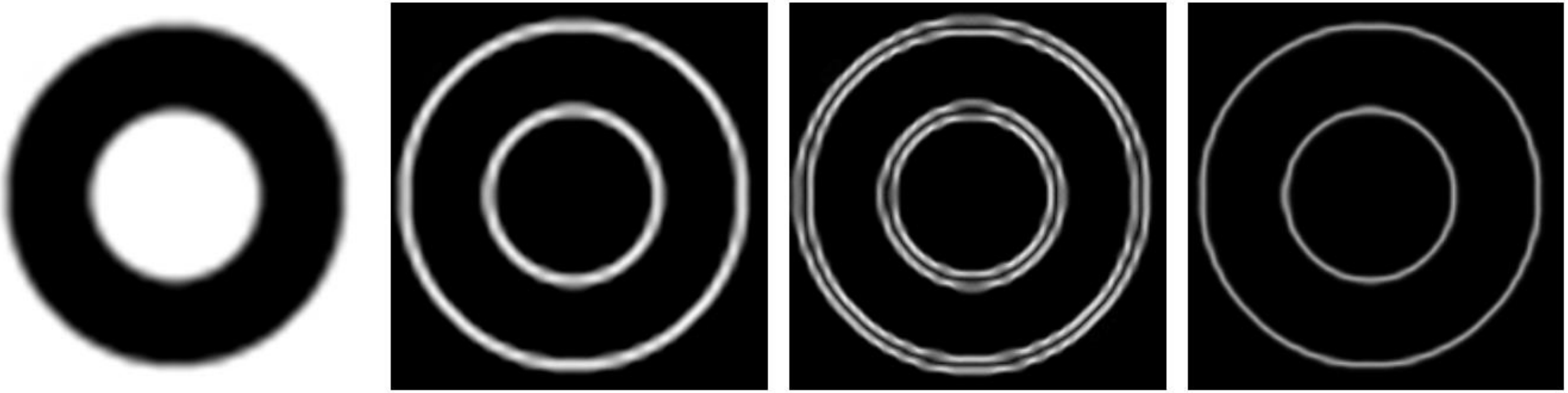
∇ : Sobel operator(Use cv2.Sobel)

∇^2: Laplacian operator(Use cv2.Laplacian)

I_t^h : High-resolution image at iteration t

Please see next slide

Method 2 : Super Resolution with prior (9pts)



(a) Low Resolution (b) Gradient of (a) (c) Second Derivatives of (a) (d) Results of (b)-(c)

The low-resolution image has thick edges due to lack of high-frequency components (Fig. (b)).

To recover high-frequency component, we use an additional loss term as a prior which make edges sharper.

Reference
Yan, Xing, and Jianbing Shen. "Fast gradient-aware upsampling for cartoon video." *2010 International Conference on Image Analysis and Signal Processing*. IEEE, 2010.
Sun, Jian, Zongben Xu, and Heung-Yeung Shum. "Image super-resolution using gradient profile prior." *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008.

Method 2 : Super Resolution with prior (9pts)

Implementation details of Gradient of Prior($\nabla^2 I^T$)

$$\nabla^2 I^T = \gamma \cdot \nabla^2 I_0^h \cdot \frac{G^T}{G_0^h}$$

$\frac{G^T}{G_0^h}$ gives weight to be sharper on $\nabla^2 I_0^h$

$G^T = G_0^h - \left|\nabla^2 I_0^h\right|$ (Sharp edge : Figure(d) in 12nd Slide)

G_0^h: Gradient of I_0^h(Use cv2.Sobel)

$\nabla^2 I_0^h$: Laplacian of I_0^h(Use cv2.Laplacian)

γ: hyper-parameter (4 or 6 is the best)

*** $\left|\nabla^2 I_0^h\right|$, G_0^h should be normalized! And, $\boldsymbol{G^T}$ should be in $[\mathbf{0}, \mathbf{1}]$**

$*$ I_0^h : initial high resolution(I_0^h) (“Upsampled.png”)

Method 2 : Gradient Descent (9pts)

Pseudo Code

I^h= load(Upsampled.png)
I^h= Grayscale(I^h)
height, width = I_0^h.shape
I^{gt} = load(HR.png) #Ground Truth
I^{gt}= Grayscale(I^{gt})
I^l = bilinear(I^{gt}, (h//4,w//4)) # Input of the algorithm
Gamma = 6
G_0^h= normalize(abs(Sobel(I_h, x_direction)) + abs(Sobel(I_h, y_direction))) +1e-10 **# 1pt**
$\nabla^2 I_0^h$ = Laplacian(I^h)
$G^T = G_0^h -$ normalize($\nabla^2 I_0^h$) **#1pt**
G^T = clip($G^T, \min = 0, \max = 1.0$)
$\nabla^2 I^T$ = Gamma * $\nabla^2 I_0^h * \frac{G^T}{G_0^h}$ **# 1pts**
$\nabla^2 I^T$ = clip($\nabla^2 I^T$, min=0.0, max=255.0)
Beta=0.0001

Max_iteration = 600
Iterate Max_iteration times:
    $\nabla^2 I_t^h$ = Laplacian(I^h)
    I_t^d = bilinear(I^h , (height//4,width//4))
    Gradient = Gradient from method 1 - Beta * $\left(\nabla^2 I_t^h - \nabla^2 I^T\right)$ **# 3pts**
    Update I_t^h

Write I^h **# 3pt**

You can tune step size, Gamma, Max_iteration and Beta yourself.
(e.g. Max_iteration about 100 iters, $\alpha(step\ size) =$ 2, Beta = 0.001, Gamma=6)
Method 2 can diverge, so you need to stop iterations well.

Metric (2pts)

Please show the evaluation of each method using both metrics. (2pts)

1. MSE (**M**ean **S**quare **E**rror) = $\sum \frac{(I_{gt}-I^h)^2}{H \cdot W}$

where H: height, W: width, I_{gt}:Ground Truth HR, I^h:Estimated HR

2. PSNR (**P**eak **S**ignal to **N**oise **R**atio) = $10 \log_{10}(\frac{R^2}{MSE})$

Where R: maximum value of pixel of images
(e.g. uint8 type image: R=255, float type image: R=1.0)

Result Example

Upsampled.png (I_0^h)
PSNR:23.05

Method 1
PSNR:26.20

Method 2
PSNR:26.45

Method 2 shows similar or better results than Method 1.

Information

Due Date: Sep.30 11:55pm

NEVER COPY!!! NO MERCY!
Please submit codes, results and reports.

Additional credits: 5pts (Report 2pts + Results/codes 3pts)
Additional credit can be earned if you submit notable reports, code or results.

If you have any questions about PA1, please contact TA

TA: Changyeon Won **(**원창연**)**
E-mail: cywon1997@gm.gist.ac.kr
Office Hour: Mon ~Thur 9:30am~11:30am
Office Location: Electrical Engineering and Computer Science Building C, Room 405
Call: 062-715-6375