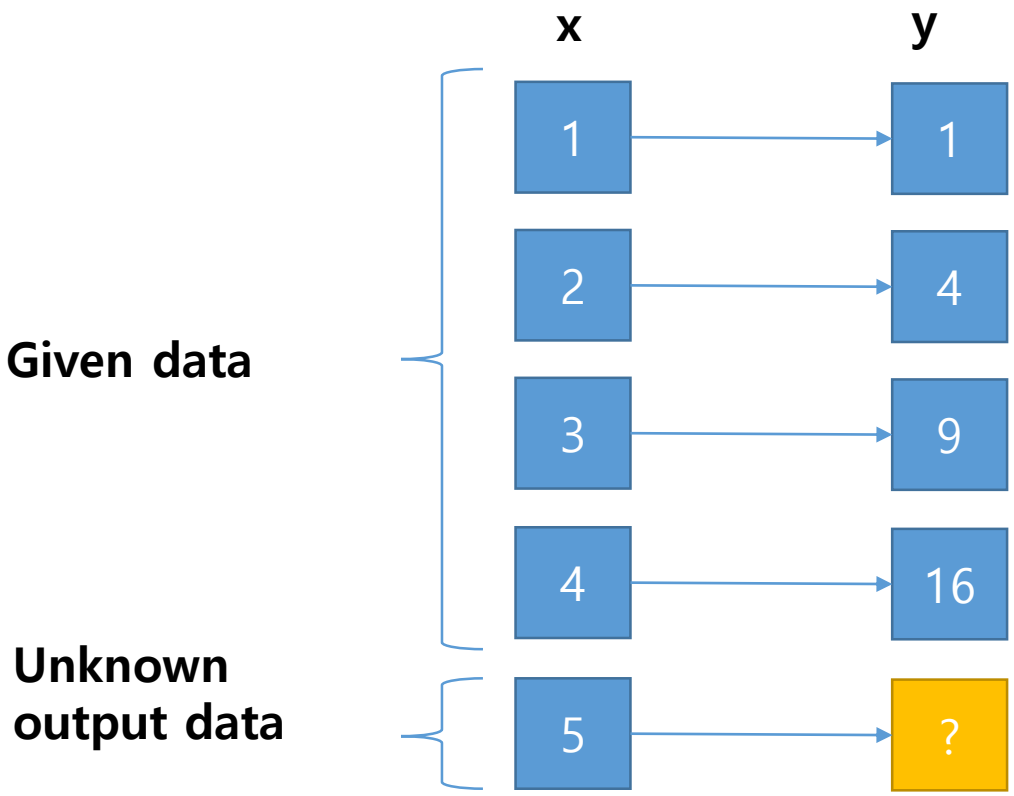
Introduction

Super Resolution based on Deep Learning

Le Van The

Deep Learning approaches

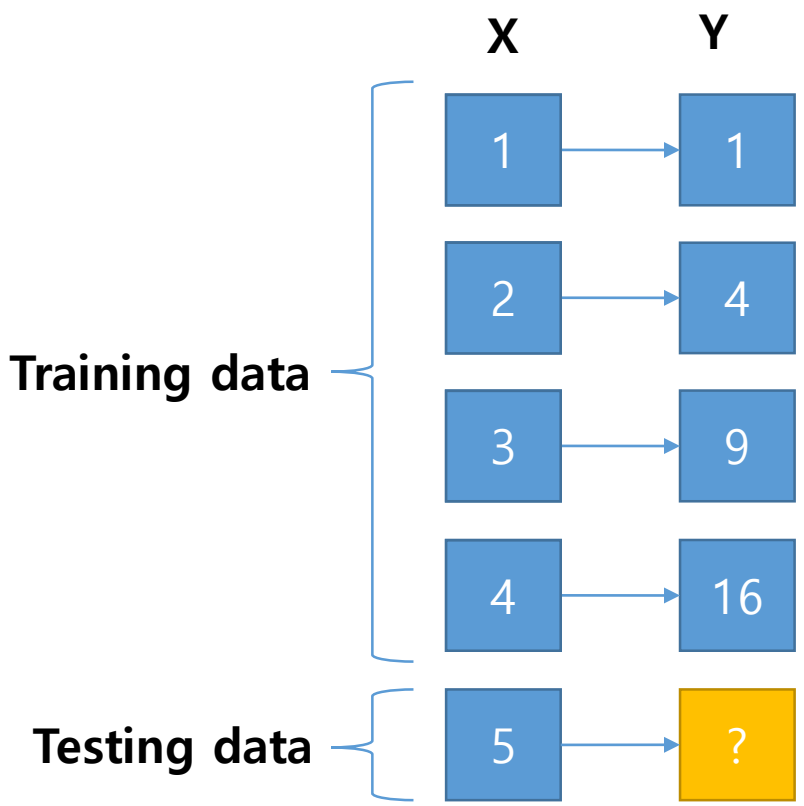
A simple problem

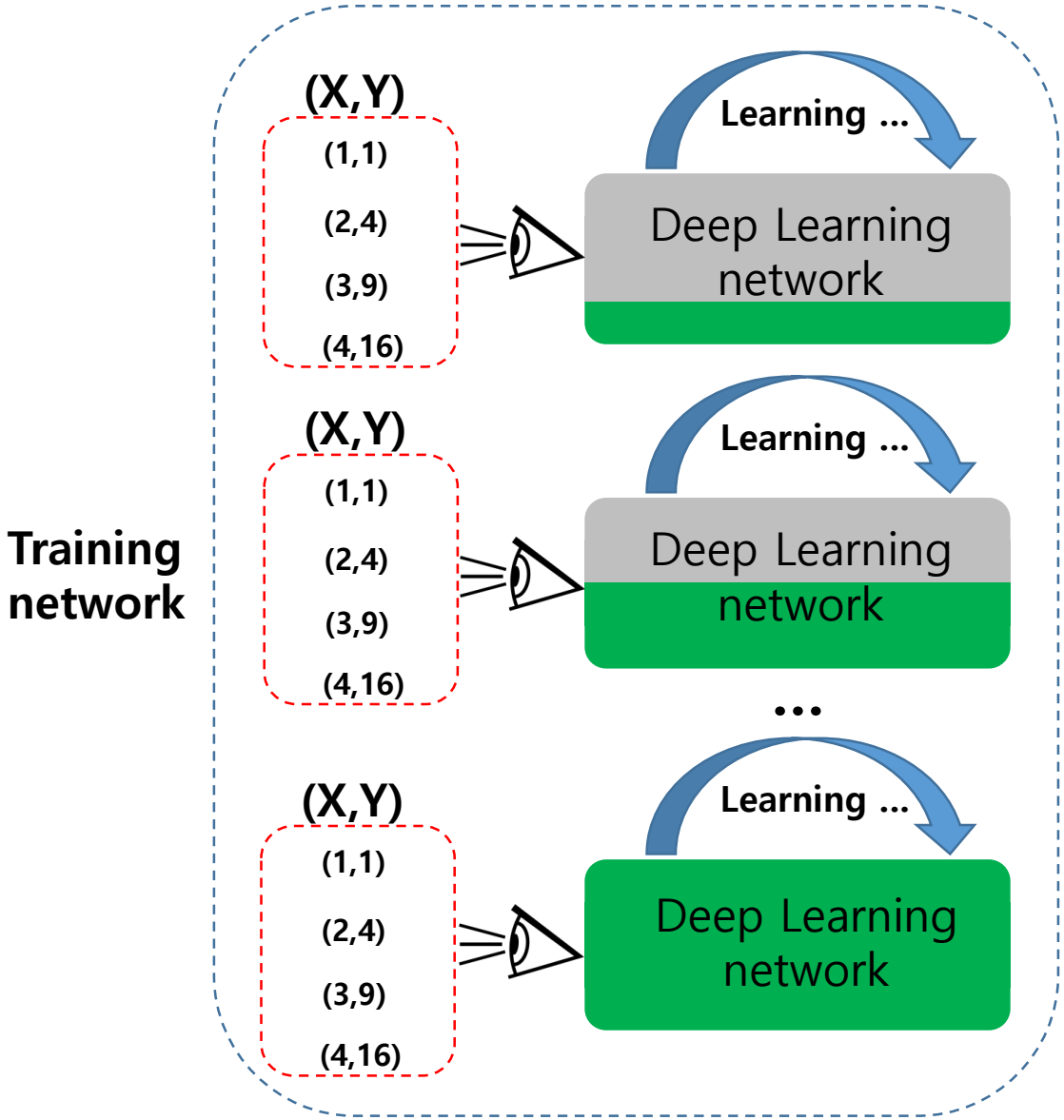


- Input (x): **a number**
- Output (y): **a number**

Deep Learning approaches

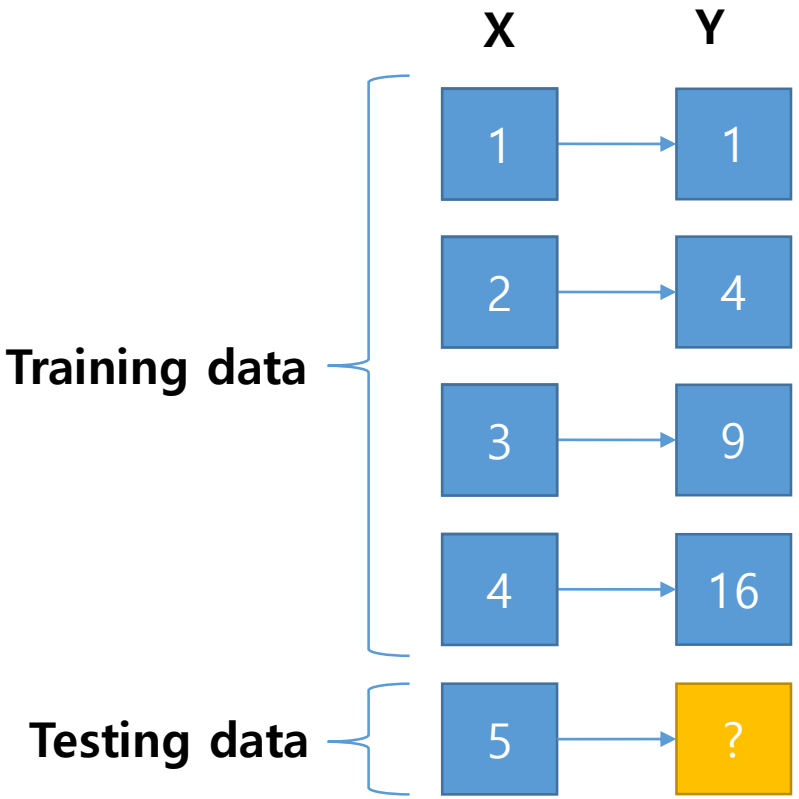
A simple problem



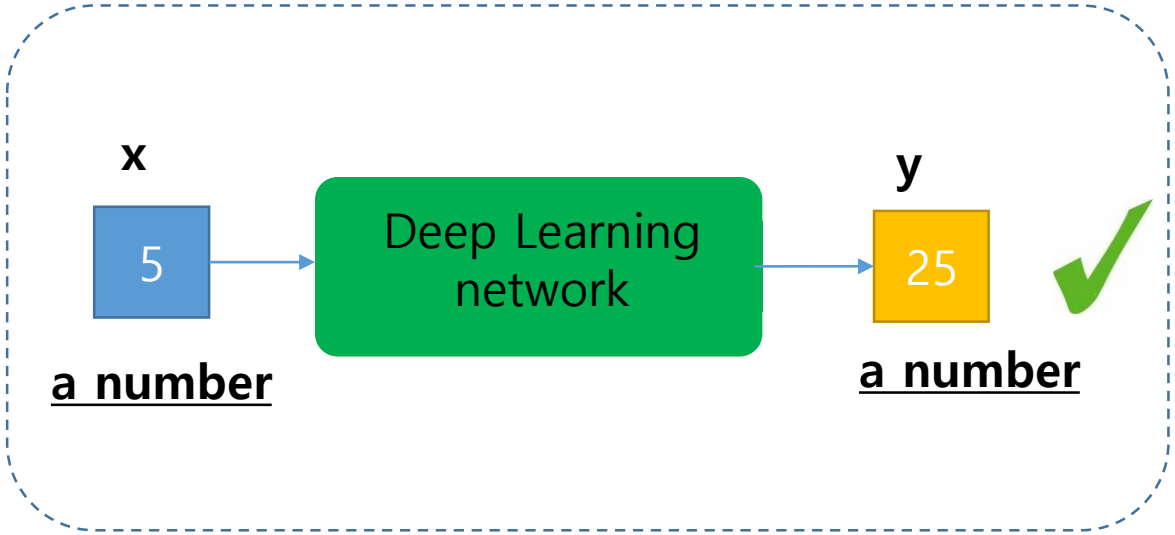


Deep Learning approaches

A simple problem

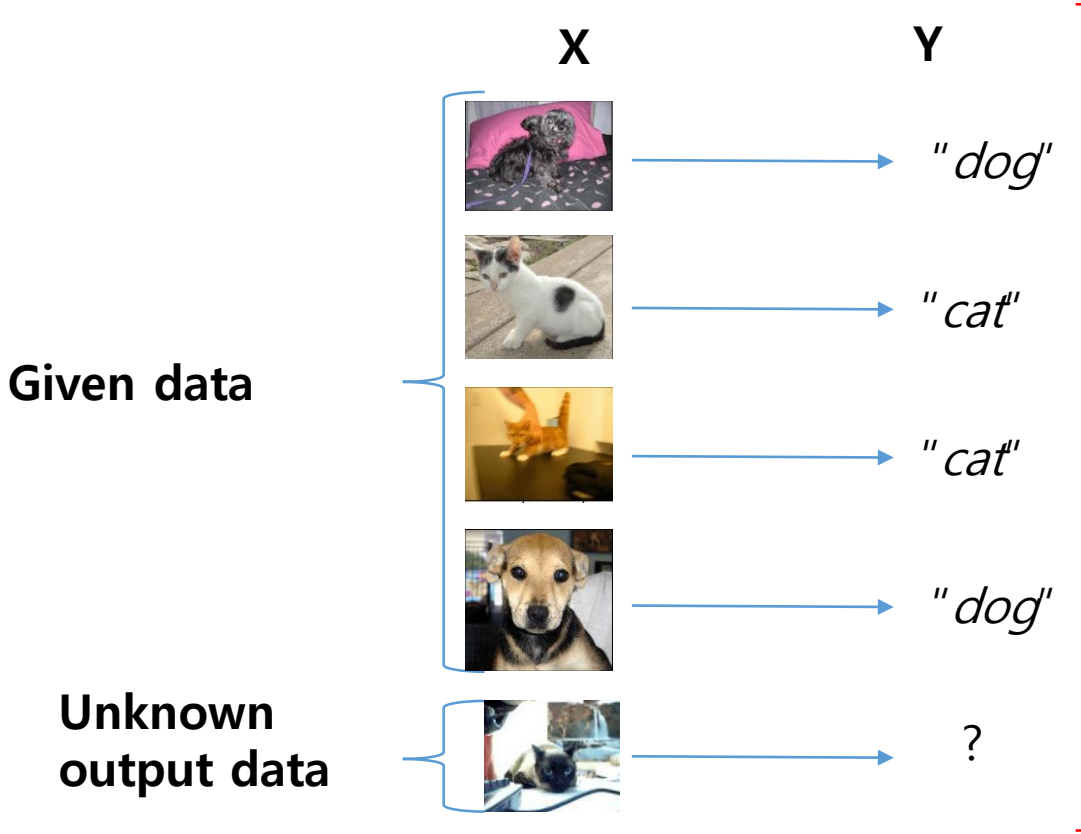


Testing



Deep Learning approaches

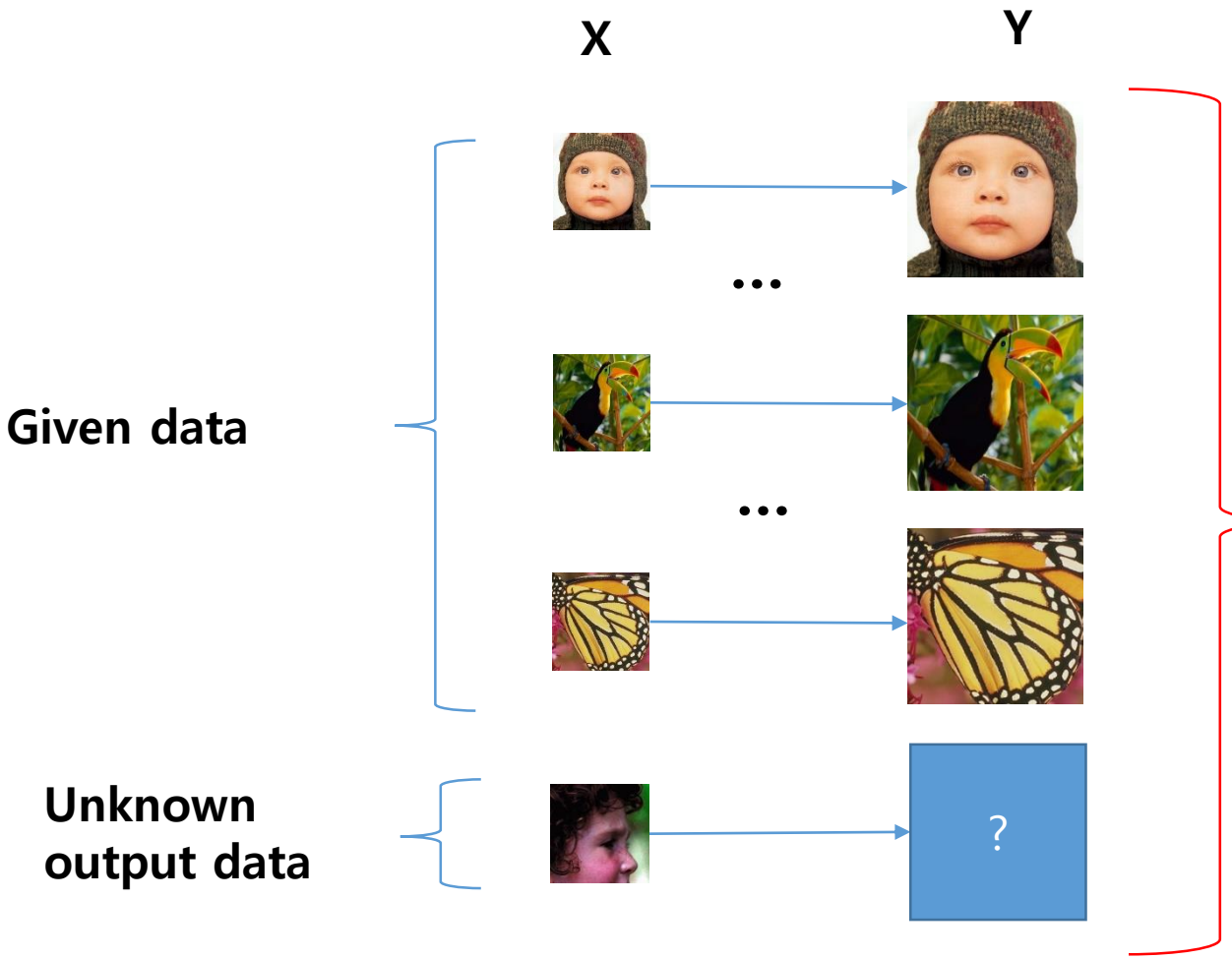
A simple problem



- Input (x): **image**
- Output (y): **"dog" or "cat"**

➔***Image classification problem***

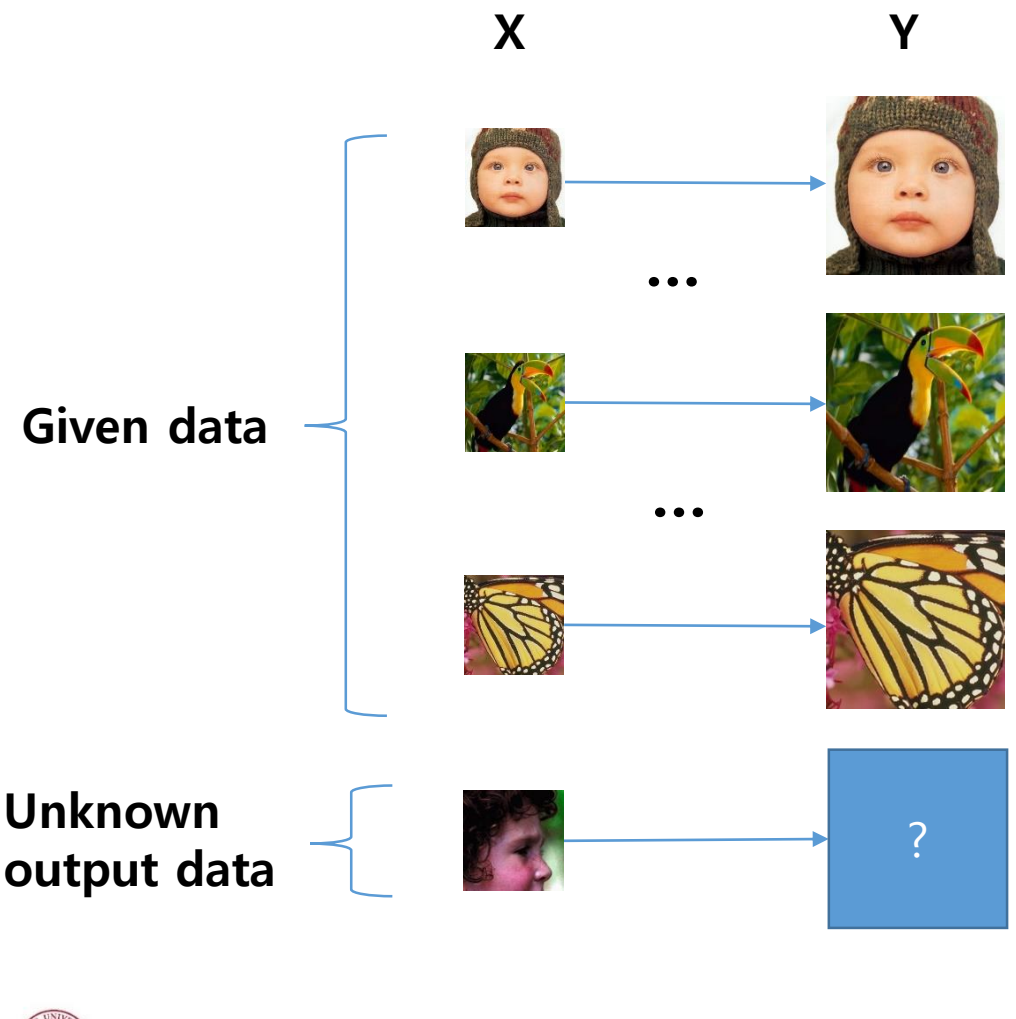
Deep Learning approaches

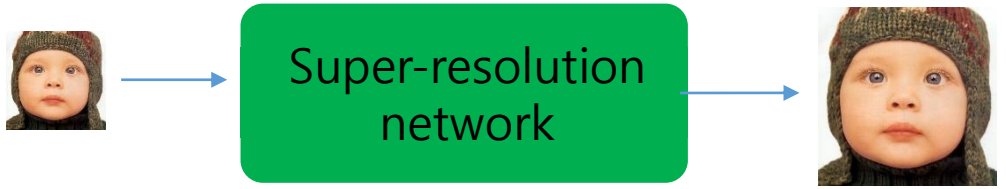


- Input (X): **image**→ low resolution
- Output (Y): **image** → high resolution

➔***Super-resolution problem***

Super-resolution based on deep learning



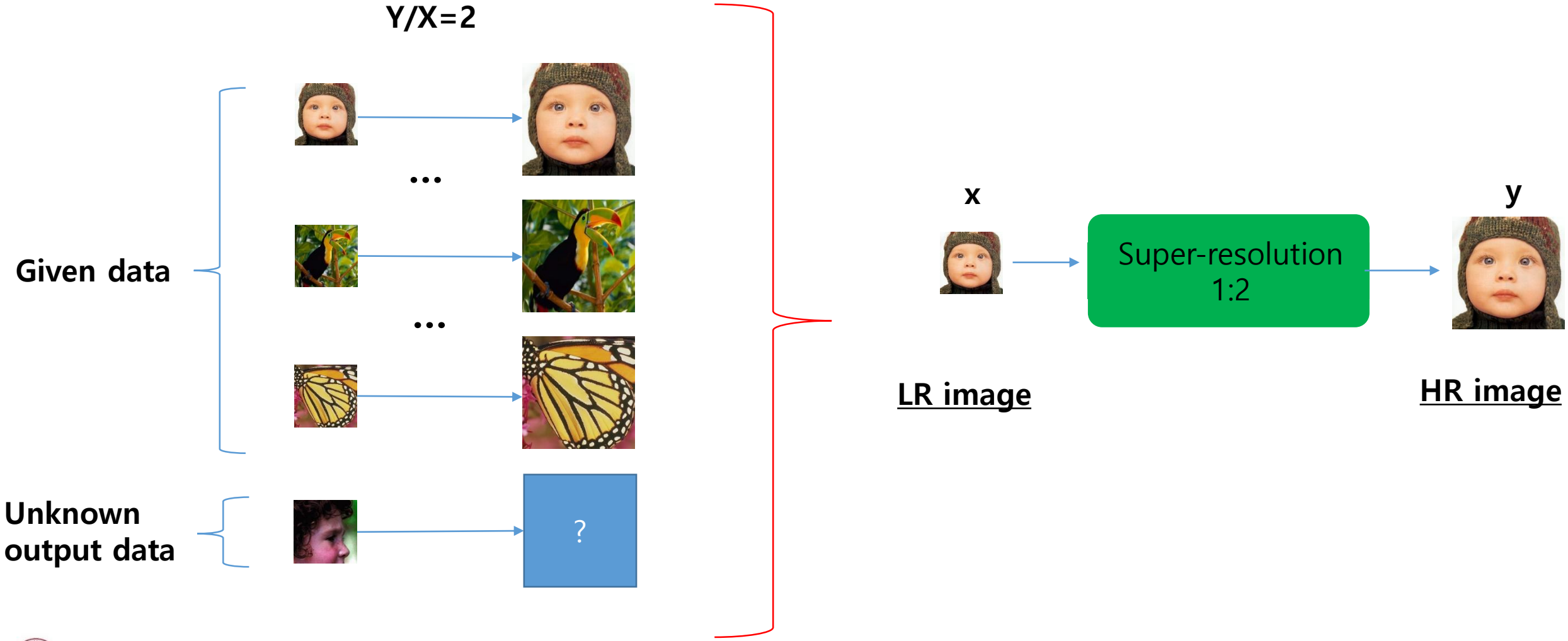


LR image **HR image**

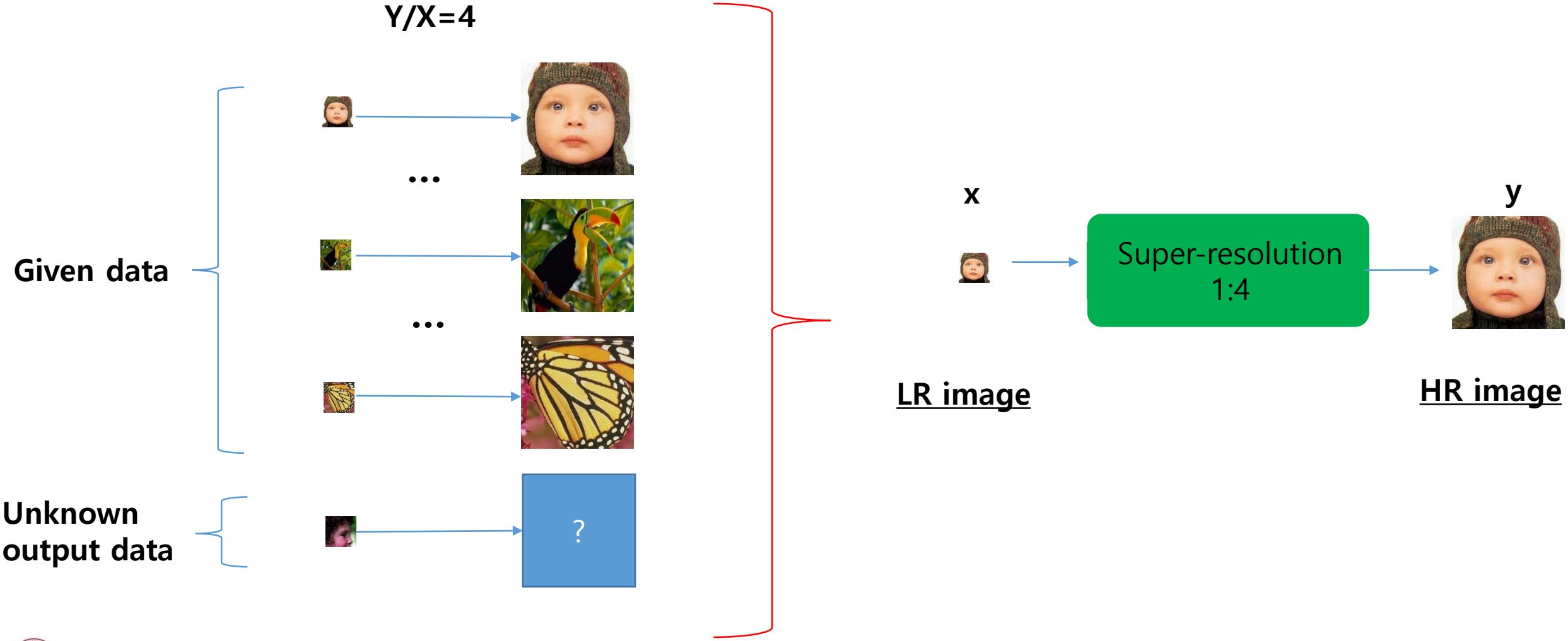
- Scale factor is the ratio scale between HR/LR (Y/X)
- Scale factor is usually 2, 3, 4, 8➔ depends on the network structures

➔ If we change the scale factor, we need to change training dataset (X, Y) corresponding to scale factor.

Super-resolution based on deep learning



Super-resolution based on deep learning

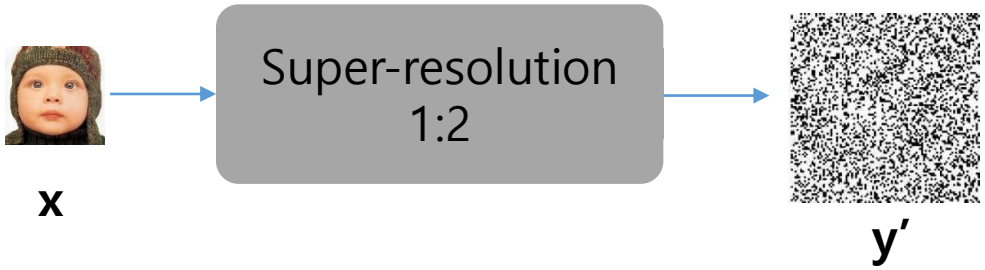


Super-resolution based on deep learning

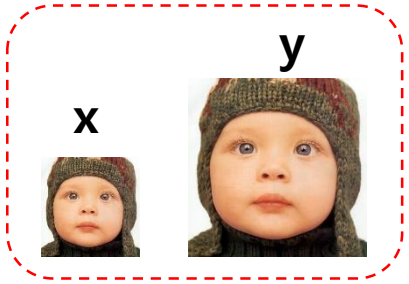
❖ **Training process for SR network**

Epoch 1:

- **Step1:** SR with initial parameters will generate SR image (**y'**)



Training data



Scale factor of x2

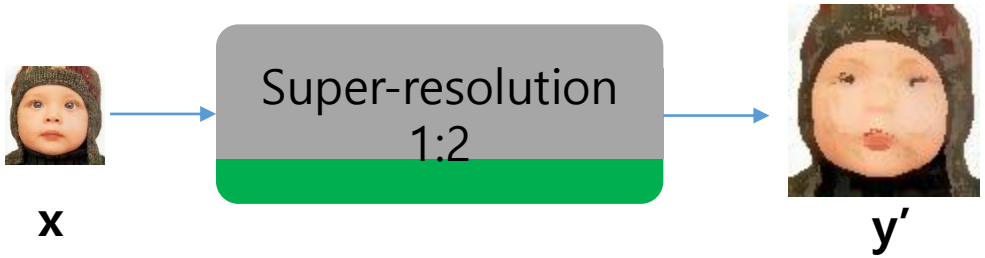
- **Step2:** Compare SR image and HR image to consider that the SR image is good enough?

Super-resolution based on deep learning

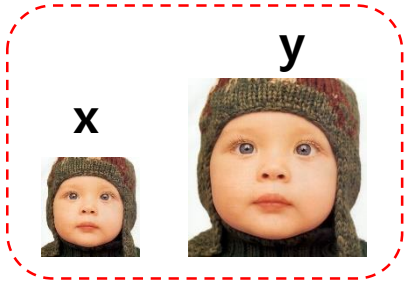
❖ **Training process for SR network**

Epoch d:

- **Step1:** SR with updated parameters will generate SR image (**y'**)



Training data



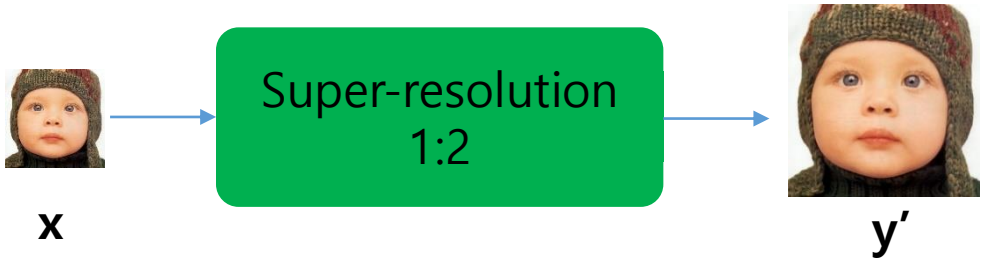
Scale factor of x2

- **Step2:** Compare SR image and HR image to consider that the SR image is good enough?

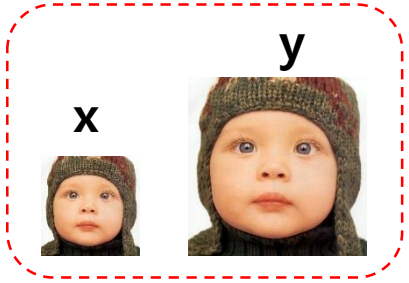
Super-resolution based on deep learning

❖ **Training process for SR network**

Epoch N: ▪ **Step1:** SR with updated parameters will generate SR image (**y'**)



Training data



Scale factor of x2

- **Step2:** Compare SR image and HR image to consider that the SR image is good enough?

Good

Super-resolution
1:2

Keep model &
STOP training

Super-resolution based on deep learning

❖ **Loss function → To check SR image is good enough or not**

Loss 1

y′ y

Loss 2

y′ y

Loss 3

y′ y

- Mean square error (MSE)

$$L_2(y, y') = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - y_i')^2$$

$$L_2\left[\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline\end{array}\ ,\ \begin{array}{|c|c|}\hline 5 & 2 \\ \hline 4 & 1 \\ \hline\end{array}\right] = \frac{1}{4}[(1-5)^2 + (2-2)^2 + (3-4)^2 + (4-1)^2] = 6.5$$

$$L_2\left[\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline\end{array}\ ,\ \begin{array}{|c|c|}\hline 1 & 3 \\ \hline 3 & 4 \\ \hline\end{array}\right] = \frac{1}{4}[(1-1)^2 + (2-2)^2 + (3-3)^2 + (4-4)^2] = 0$$

➔**Loss 1>Loss 2> Loss 3**

Super-resolution based on deep learning

❖ **Loss function → To check that SR image is good enough or not**

Loss 1

y′ y

Loss 2

y′ y

Loss 3

y′ y

- Mean absolute error (MAE)

$$L_1(y, y') = \frac{1}{N}\sum_{i=0}^{N-1}|y - y'|$$

$$L_2\left[\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}, \begin{bmatrix}5 & 2\\ 4 & 1\end{bmatrix}\right] = \frac{1}{4}[|1-5| + |2-2| + |3-4| + |4-1|] = 2$$

$$L_2\left[\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}, \begin{bmatrix}1 & 3\\ 3 & 4\end{bmatrix}\right] = 0$$

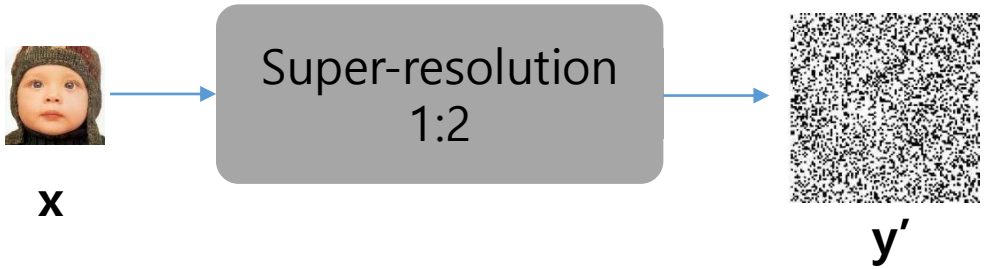
➔Loss 1>Loss 2> Loss 3

Super-resolution based on deep learning

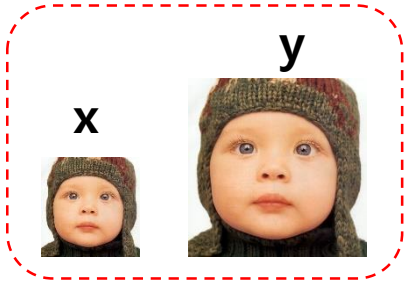
❖ **Training process for SR network**

Epoch 1:

- **Step1:** SR with initial parameters will generate SR image (**y'**)

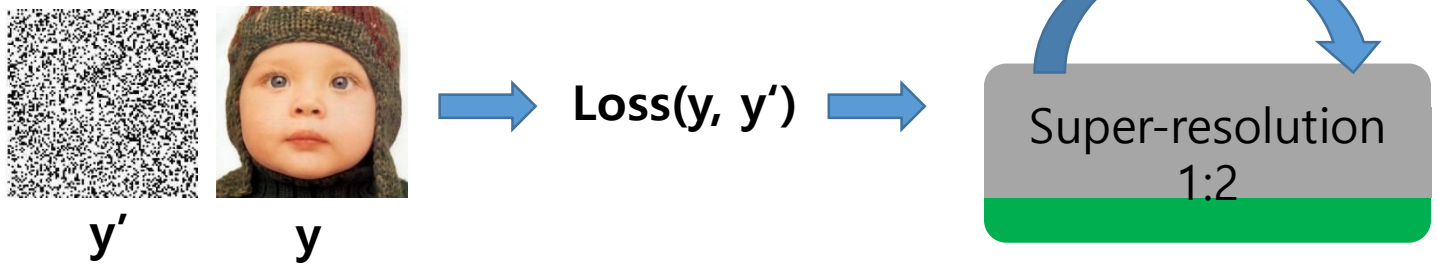


Training data



Scale factor of x2

- **Step2:** Calculate Loss value between SR and original image. Use loss value to update parameters→ objective is to reduce loss value→ make output more similar with original image.



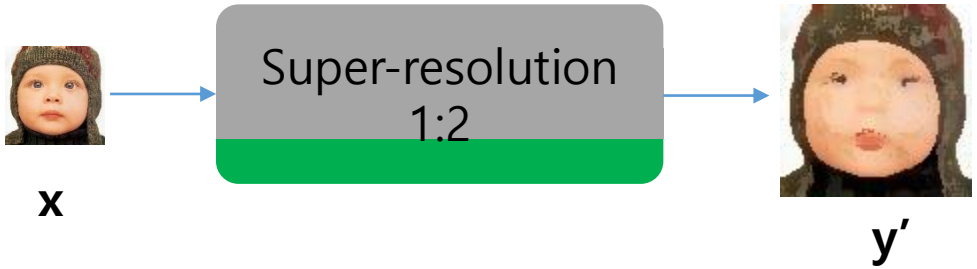
Use loss value to update parameters

Super-resolution based on deep learning

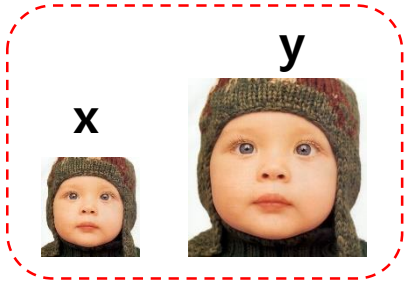
❖ **Training process for SR network**

Epoch 1:

- **Step1:** SR with initial parameters will generate SR image (**y'**)

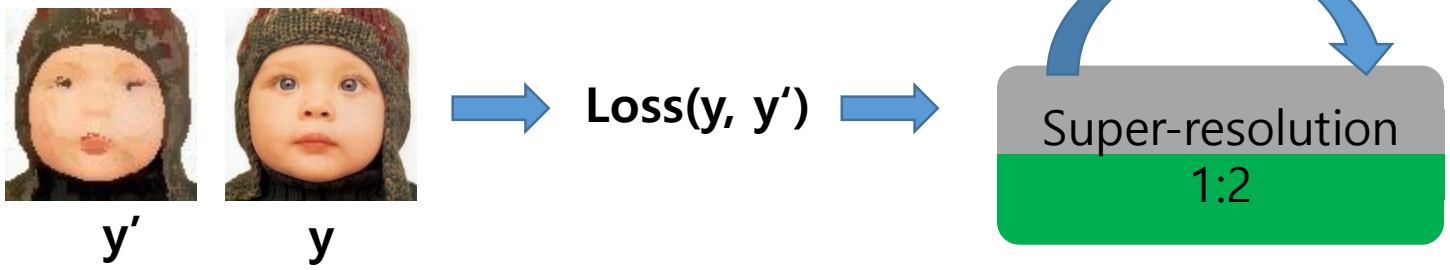


Training data



Scale factor of x2

- **Step2:** Calculate Loss value between SR and original image. Use loss value to update parameters.

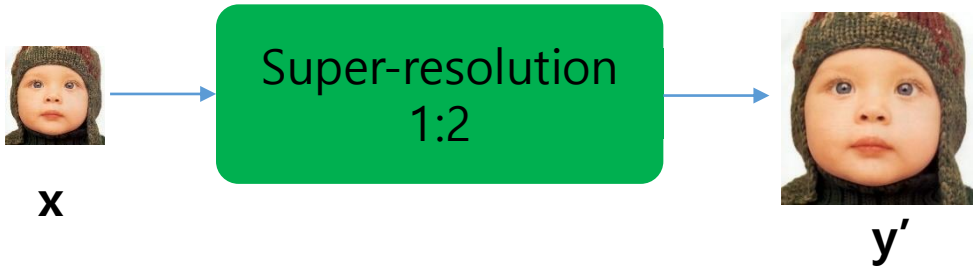


Use loss value to update parameters

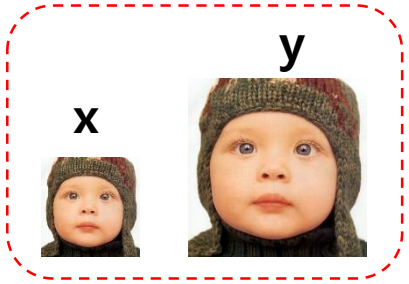
Super-resolution based on deep learning

❖ **Training process for SR network**

Epoch N: ▪ **Step1:** SR with initial parameters will generate SR image (**y'**)

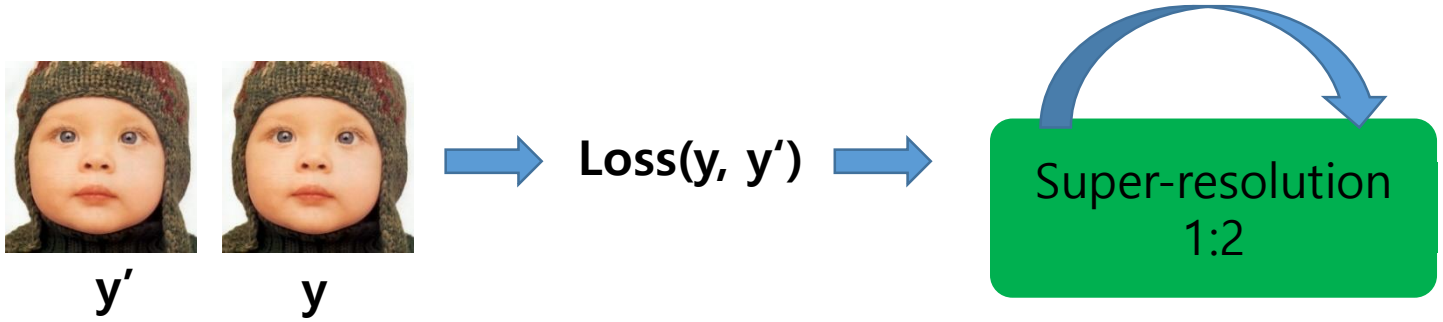


Training data



Scale factor of x2

- **Step2:** Calculate Loss value between SR and original image. Use loss value to update parameters.

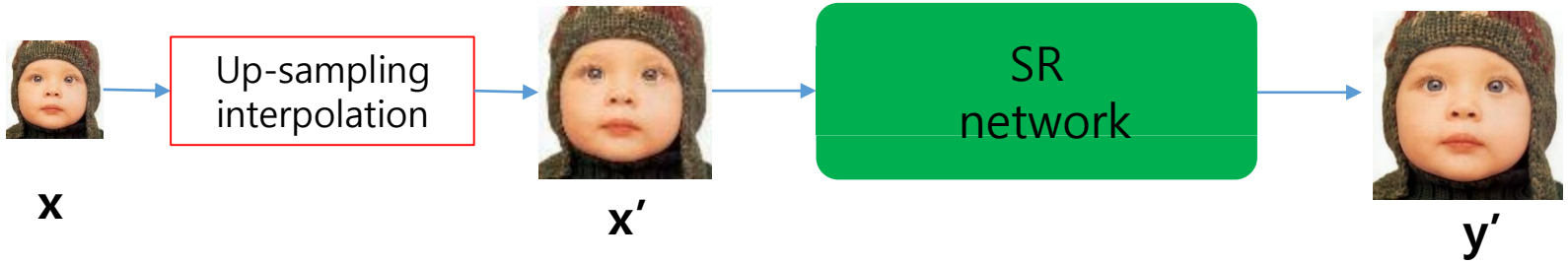


Keep model &
STOP training

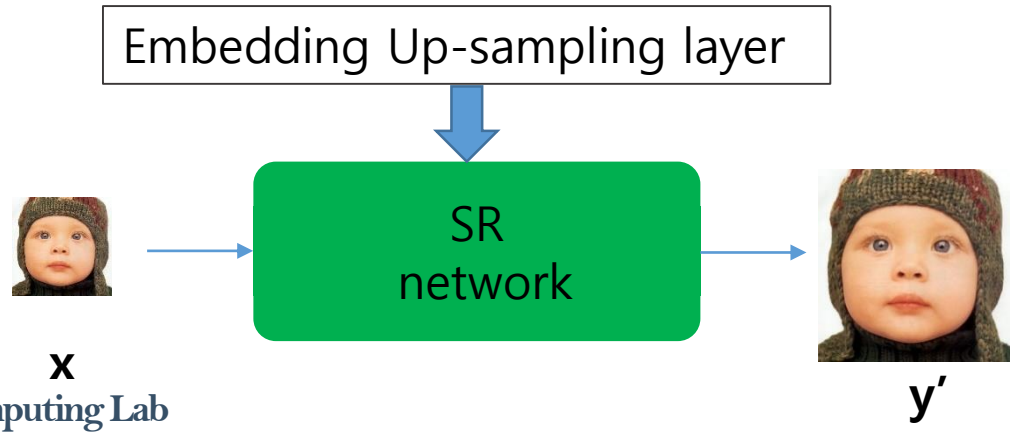
Super-resolution based on deep learning

❖ **Up-sampling → embedded in structure to up-scale image**

- **Method#1:** Up-sampling ***outside*** of SR network



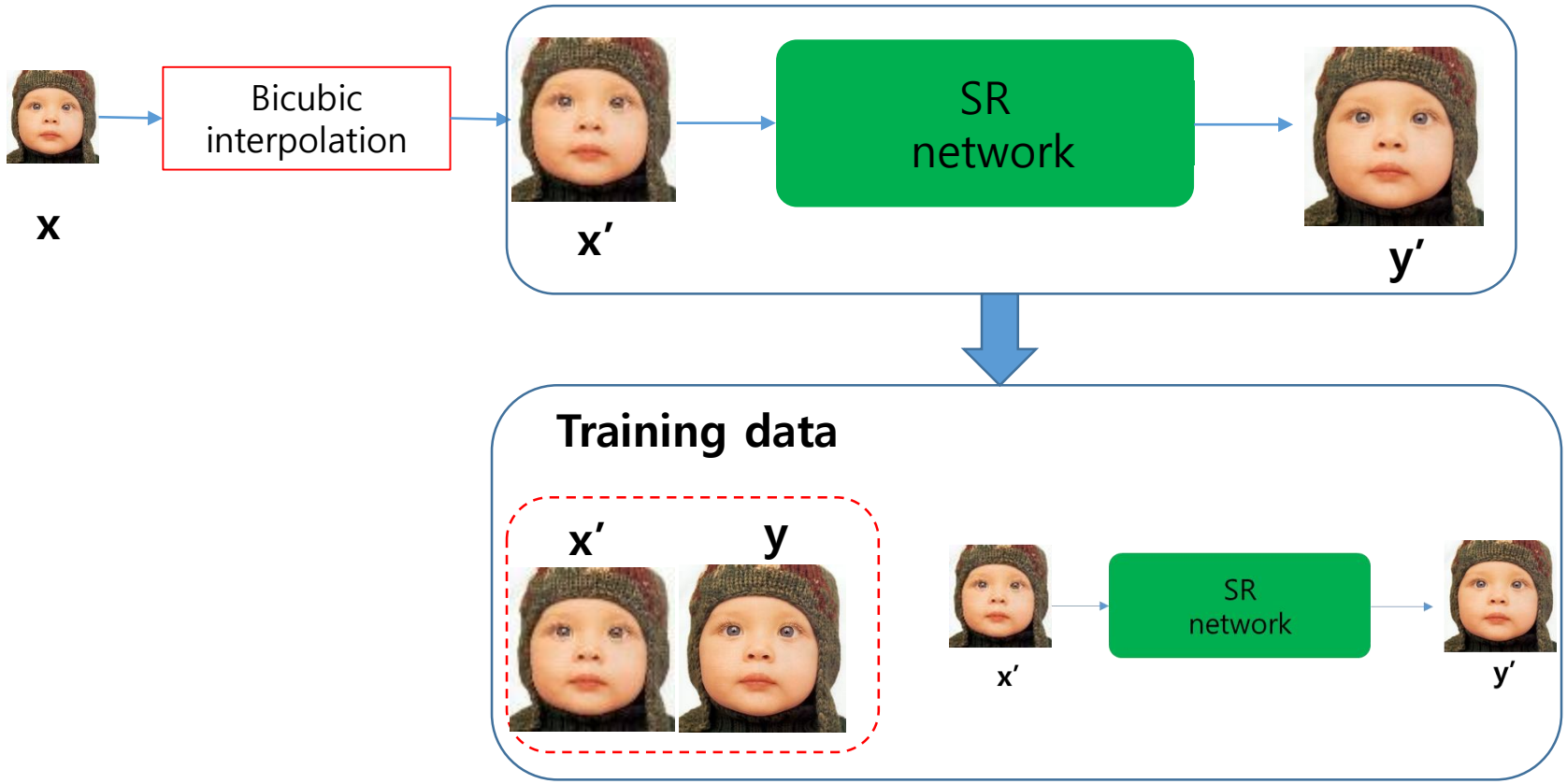
- **Method#2:** Up-sampling ***inside*** of SR network



Super-resolution based on deep learning

❖ **Up-sampling**

- **Method#1:** Up-sampling ***outside*** of SR network

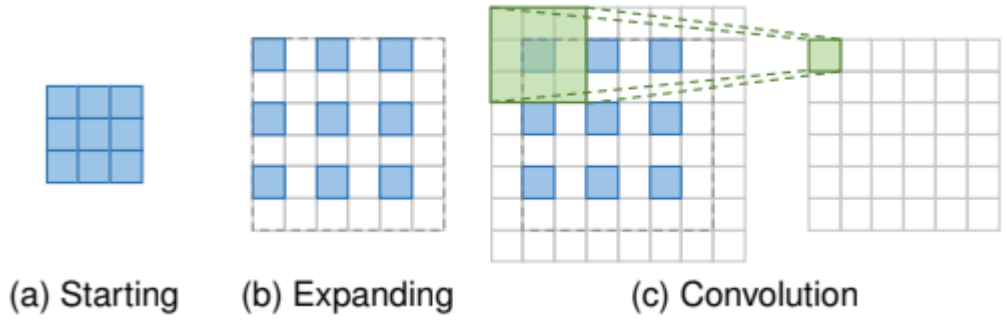


Super-resolution based on deep learning

- **Method#2:** Up-sampling ***inside*** of SR network

  Transposed convolutional layer

Input size: 3x3 **Output size: 9x9**



Transposed convolutional layer with scale factor of 2

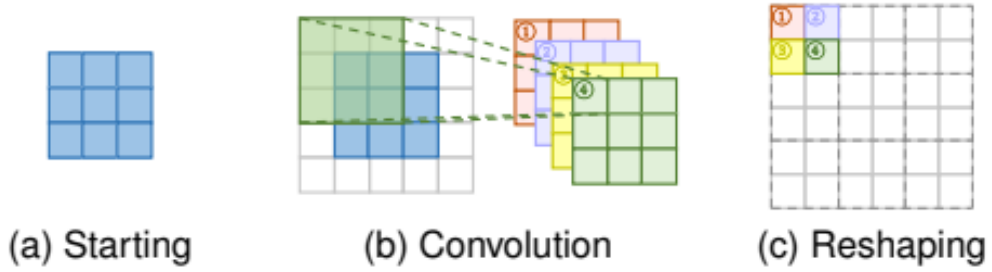
- **Step1:** Expanding **xA** the input with zeros. **A** is scale factor and it usually is 2,3.
- **Step2:** Use convolution with kernel 3x3, stride 3 and padding 1.
- **Note:** - If scale factor is **4**, we will use 2 times of up-sampling **x2**.
  - If scale factor is **8**, we will use 3 times of up-sampling **x2**.

Super-resolution based on deep learning

- **Method#2:** Up-sampling ***inside*** of SR network

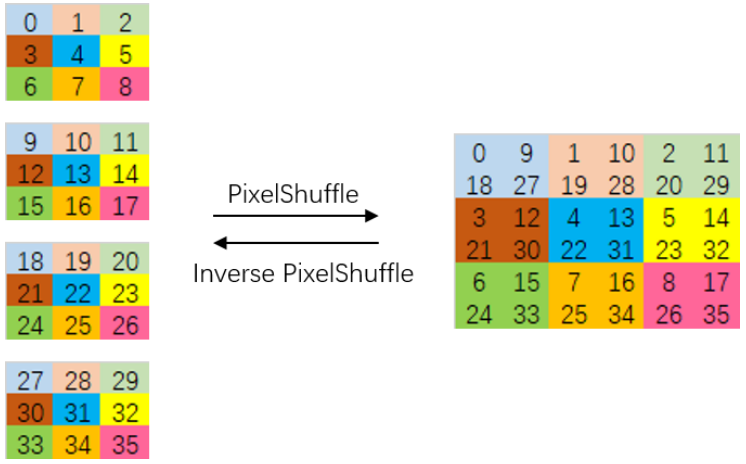
Pixel Shuffle layer

Input size: 3x3 **Output size: 9x9**



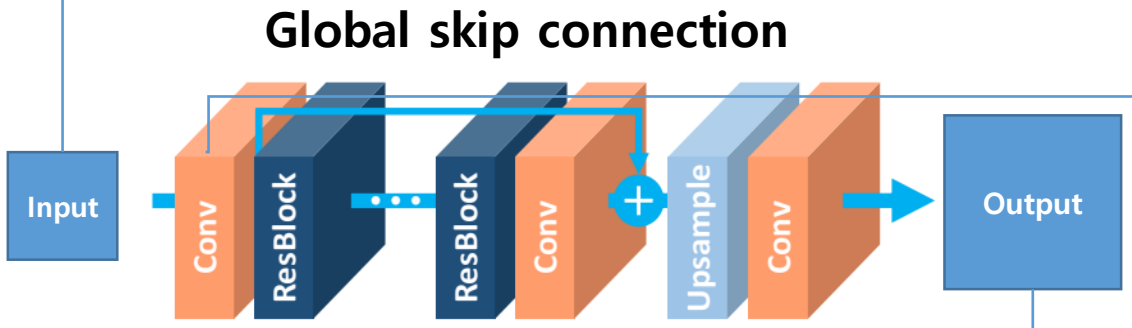
Pixel shuffle layer with scale factor of 2

- **Step1:** From 1 channel, it use convolution to generate A^2 channel. **A** is scale factor and it usually is 2,3.
- **Step2:** Merge A^2 to 1 output channel with the up-sampling size.



- **Note:** - If scale factor is **4**, we will use 2 times of up-sampling **x2**.
  - If scale factor is **8**, we will use 3 times of up-sampling **x2**.

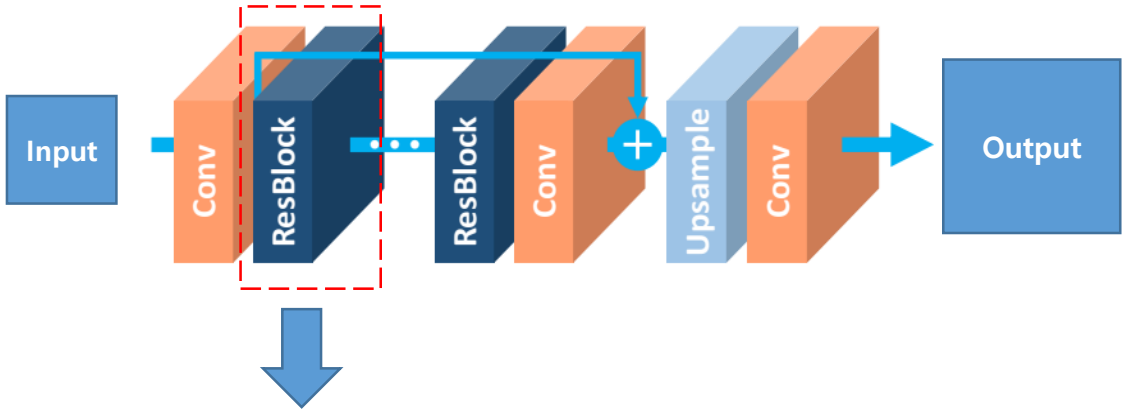
Enhanced Deep Residual Networks



The architecture of the single-scale SR network (EDSR)

```
def edsr(scale, num_filters=64, num_res_blocks=8, res_block_scaling=None):
    x_in = Input(shape=(None, None, 3))
    x = Lambda(normalize)(x_in)

    x = b = Conv2D(num_filters, 3, padding='same')(x)
    for i in range(num_res_blocks):
        b = res_block(b, num_filters, res_block_scaling)
    b = Conv2D(num_filters, 3, padding='same')(b)
    x = Add()([x, b])

    x = upsample(x, scale, num_filters)
    x = Conv2D(3, 3, padding='same')(x)

    x = Lambda(denormalize)(x)
    return Model(x_in, x, name="edsr")
```

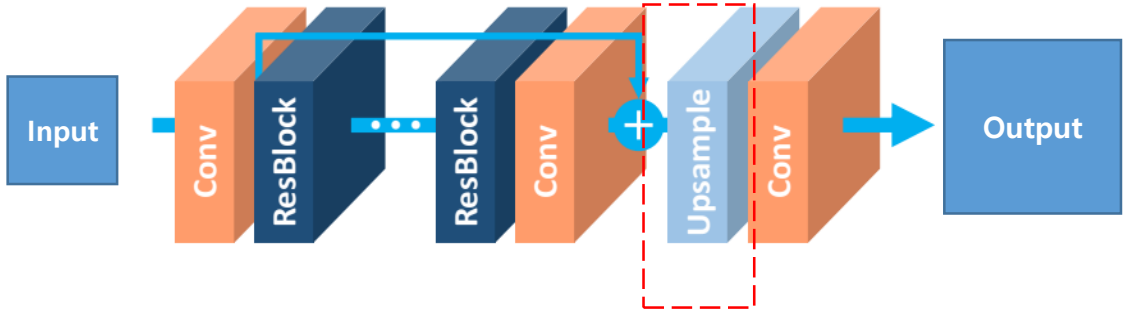
Enhanced Deep Residual Networks



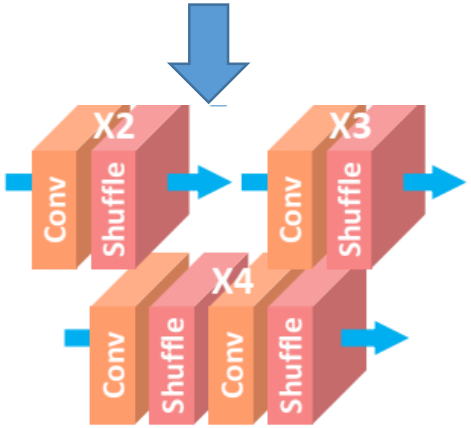
ResBlock

```
def res_block(x_in, filters, scaling):
    x = Conv2D(filters, 3, padding='same', activation='relu')(x_in)
    x = Conv2D(filters, 3, padding='same')(x)
    if scaling:
        x = Lambda(lambda t: t * scaling)(x)
    x = Add()([x_in, x])
    return x
```

Enhanced Deep Residual Networks

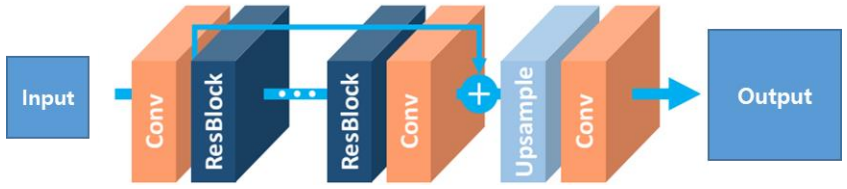



```
def upsample(x, scale, num_filters):
    def upsample_1(x, factor, **kwargs):
        x = Conv2D(num_filters * (factor ** 2), 3, padding='same', **kwargs)(x)
        return Lambda(pixel_shuffle(scale=factor))(x)

    if scale == 2:
        x = upsample_1(x, 2, name='conv2d_1_scale_2')
    elif scale == 3:
        x = upsample_1(x, 3, name='conv2d_1_scale_3')
    elif scale == 4:
        x = upsample_1(x, 2, name='conv2d_1_scale_2')
        x = upsample_1(x, 2, name='conv2d_2_scale_2')

    return x
```

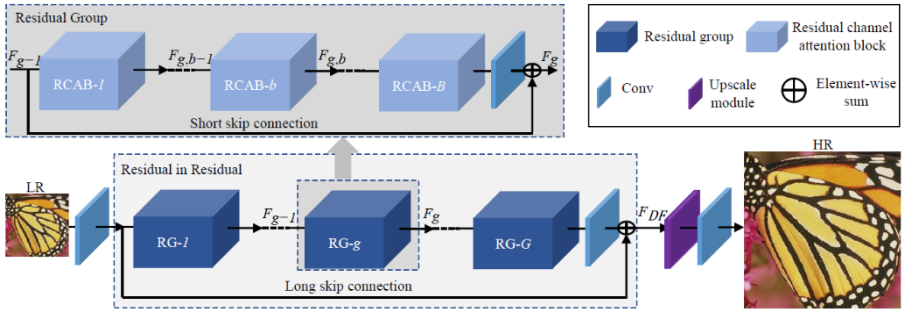


4=2x2

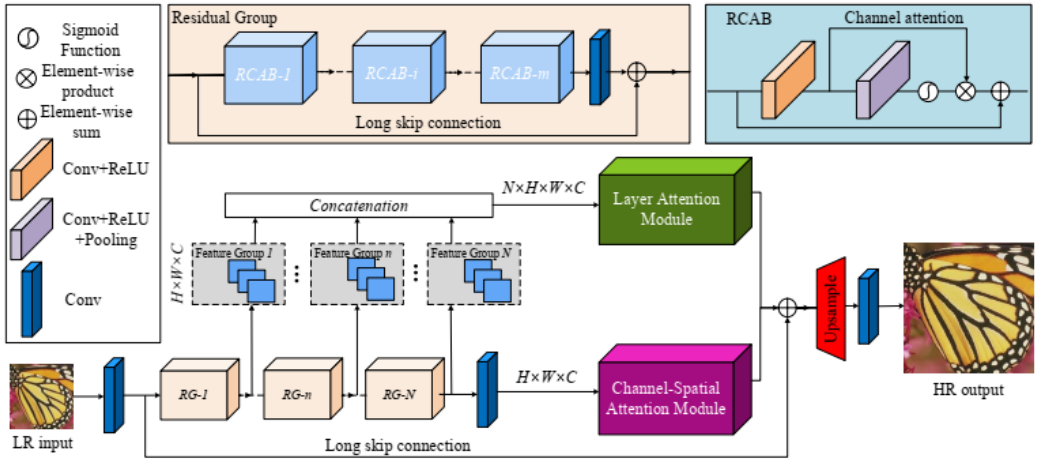
The state-of-the-art SR networks



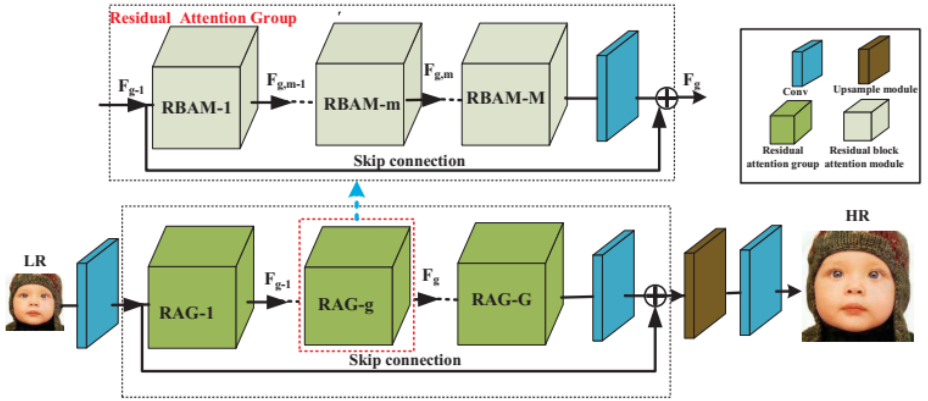
single-scale SR network (EDSR)



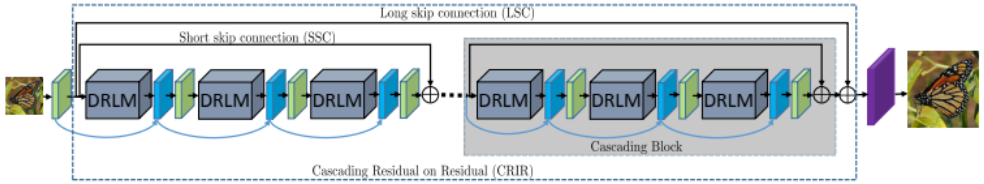
Network architecture of the residual channel attention network (RCA



Holistic attention network(HAN)



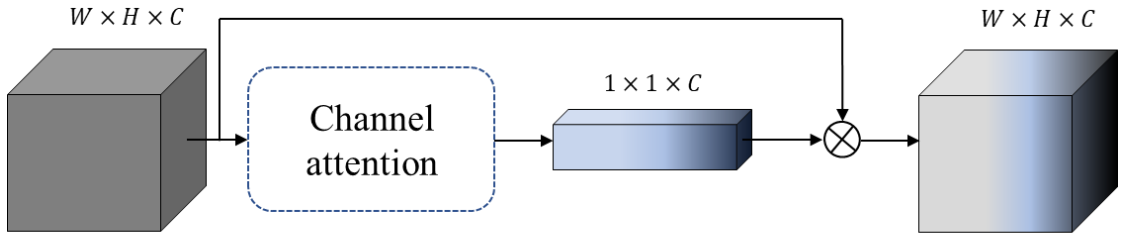
Network framework of our residual block attention network (RBAN)



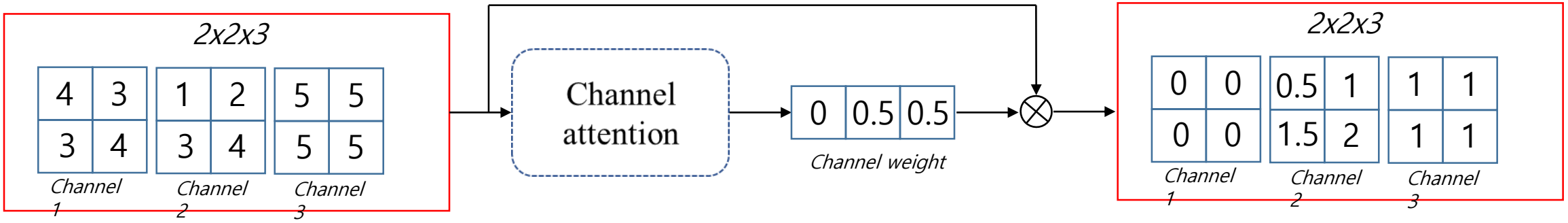
Dense Residual Laplacian Module (DRLM).

Attention mechanism

❖ **Channel attention → which channel is more important?**

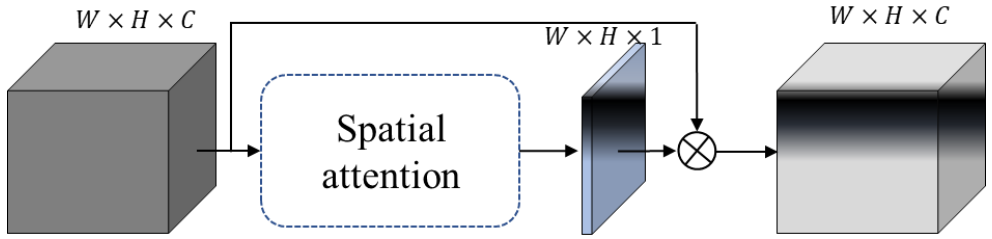


a, Channel attention architecture

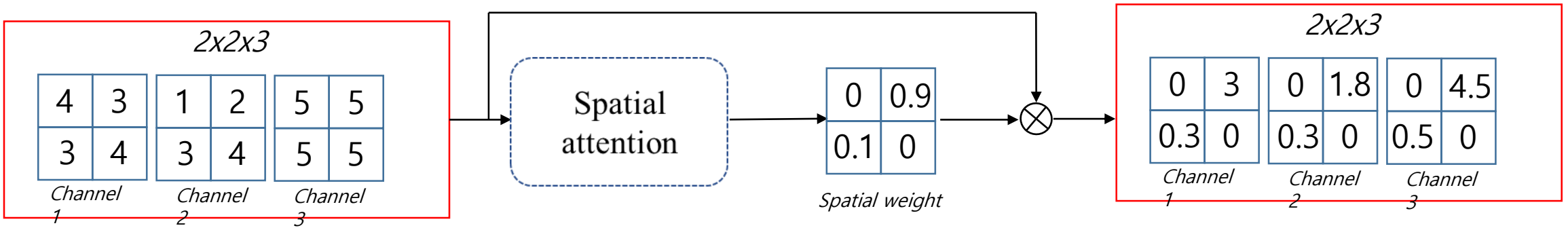


Attention mechanism

❖ **Spatial attention → which spatial region is more important?**

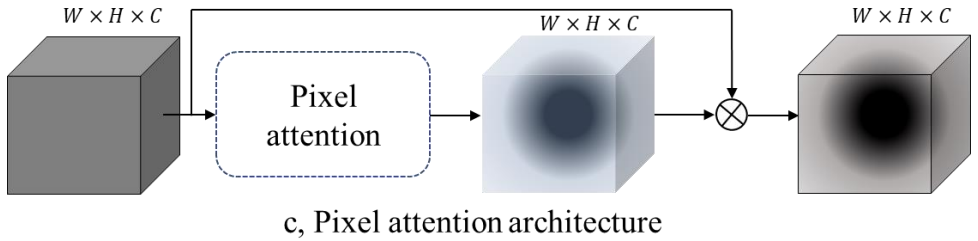


b, Spatial attention architecture

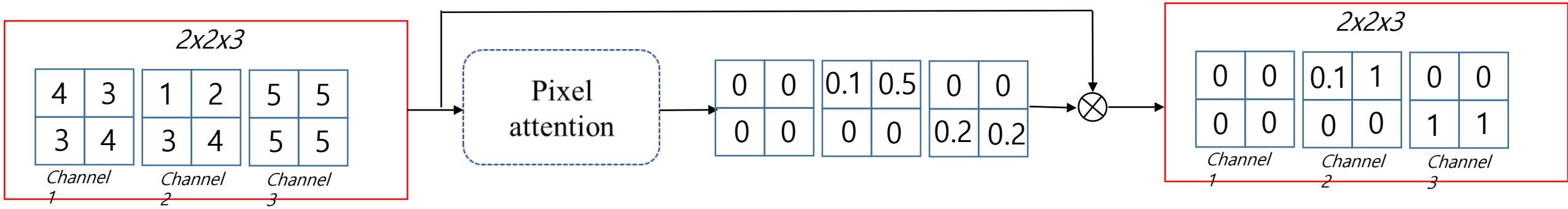


Attention mechanism

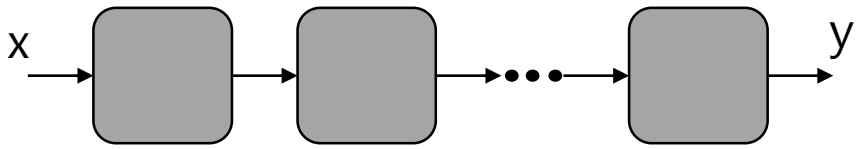
❖ Pixel attention → Which pixel is more important?



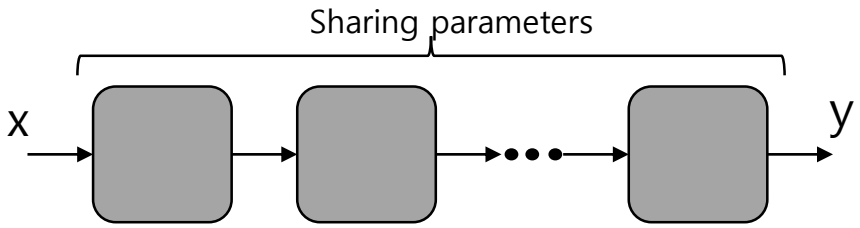
c, Pixel attention architecture



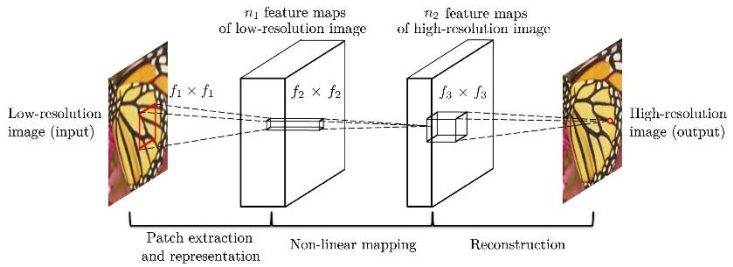
Popular structures



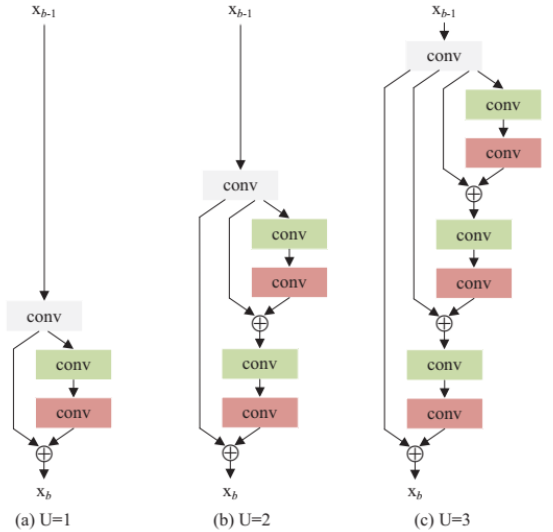
a, Linear learning



a, Recursive learning

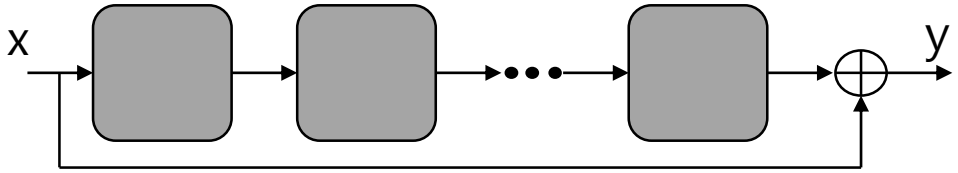


SRCNN network



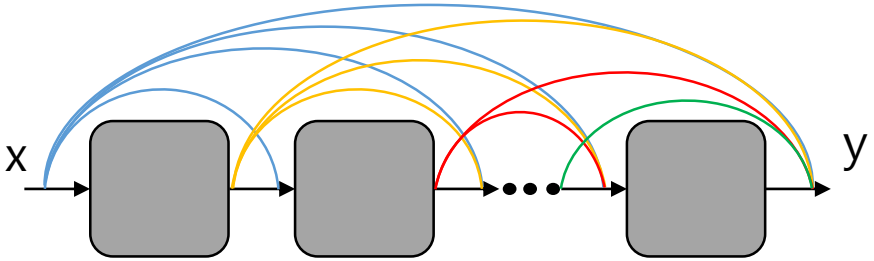
Recursive blocks in Deep Recursive Residual Network

Popular structures

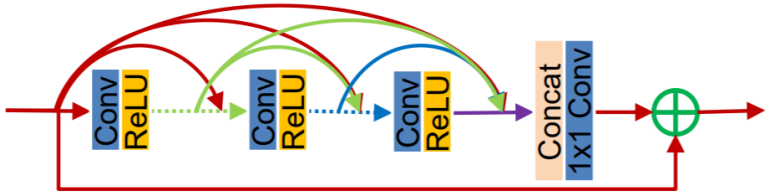


c, Residual learning

ResBlock in EDSR



d, Dense learning



Residual dense block in Residual Dense Network

SR network evaluation

- ❖ Common testing datasets (just including high-resolution images)
- ❖ Image Quality Assessment (PSNR and SSIM)

To evaluate the SR network is good or not when comparing to many other SR networks

Methods	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	×2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
SRCNN [3]	×2	36.66	0.9542	32.45	0.9067	31.36	0.8879	29.50	0.8946	35.60	0.9663
FSRCNN [4]	×2	37.05	0.9560	32.66	0.9090	31.53	0.8920	29.88	0.9020	36.67	0.9710
VDSR [15]	×2	37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140	37.22	0.9750
LapSRN [19]	×2	37.52	0.9591	33.08	0.9130	31.08	0.8950	30.41	0.9101	37.27	0.9740
MemNet [31]	×2	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195	37.72	0.9740
EDSR [22]	×2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
SRMDNF [38]	×2	37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204	38.07	0.9761
D-DBPN [6]	×2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RDN [41]	×2	38.24	0.9614	34.01	0.9212	32.34	0.9017	32.89	0.9353	39.18	0.9780
RCAN [40]	×2	38.27	0.9614	34.12	0.9216	32.41	0.9027	33.34	0.9384	39.44	0.9786
SRFBN [21]	×2	38.11	0.9609	33.82	0.9196	32.29	0.9010	32.62	0.9328	39.08	0.9779
SAN [2]	×2	38.31	**0.9620**	34.07	0.9213	32.42	0.9028	33.10	0.9370	39.32	0.9792
HAN(ours)	×2	38.27	0.9614	34.16	0.9217	32.41	0.9027	33.35	0.9385	39.46	0.9785
HAN+(ours)	×2	**38.33**	0.9617	**34.24**	**0.9224**	**32.45**	**0.9030**	**33.53**	**0.9398**	**39.62**	**0.9787**

Quantitative results in *Holistic attention network(HAN) paper*

SR network evaluation

❖ Image Quality Assessment (PSNR and SSIM)

- Peak Signal-to-Noise Ratio (PSNR)

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{L^2}{\frac{1}{N}\sum_{i=1}^{N}(I(i) - \hat{I}(i))^2}\right),$$

```
from skimage.metrics import peak_signal_noise_ratio

psnr = peak_signal_noise_ratio(image1, image2, data_range=255)
```

- Structural Similarity (SSIM)

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_+ c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

```
from skimage.metrics import structural_similarity

ssim = structural_similarity(image1, image2,  data_range=255)
```

SR network evaluation

❖ Testing process

Down-sampling

interpolation

Up-sampling

SR network

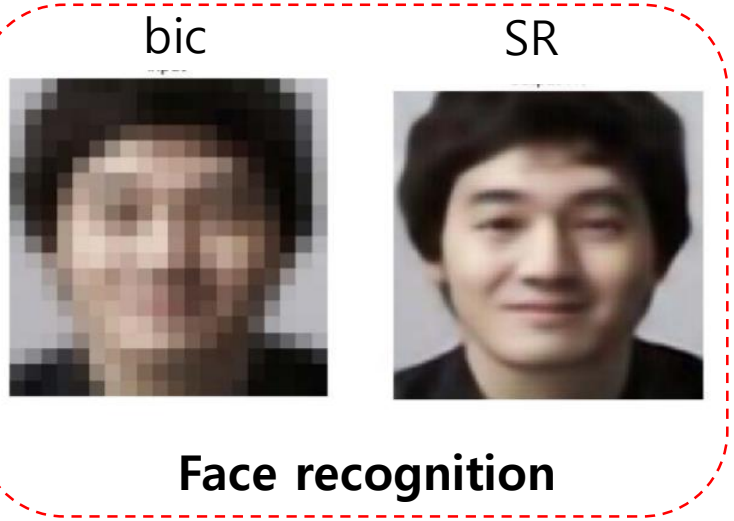
HR image in testing dataset

LR image

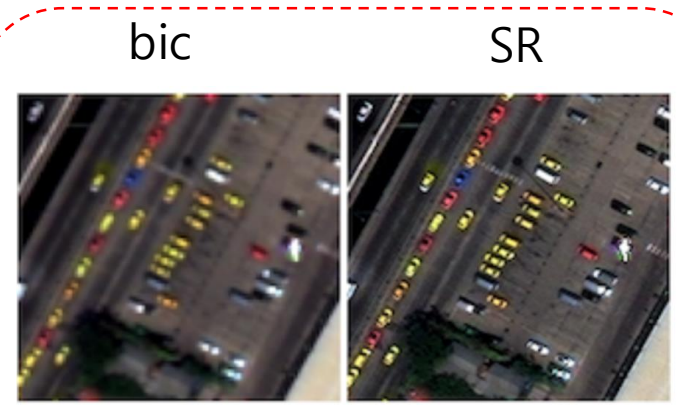
SR image

Compute:
- PSNR (HR,SR)
- SSIM (HR,SR)

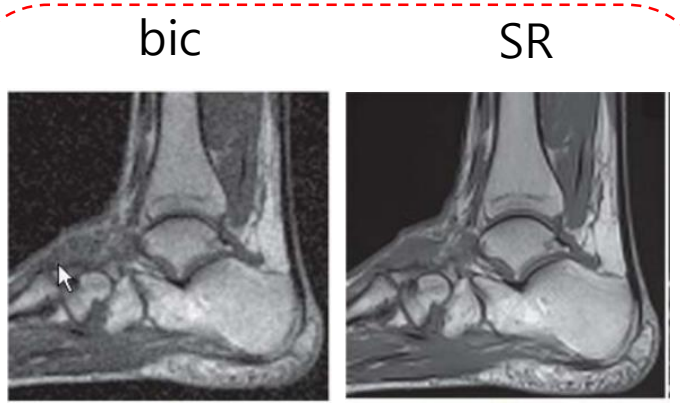
SR application



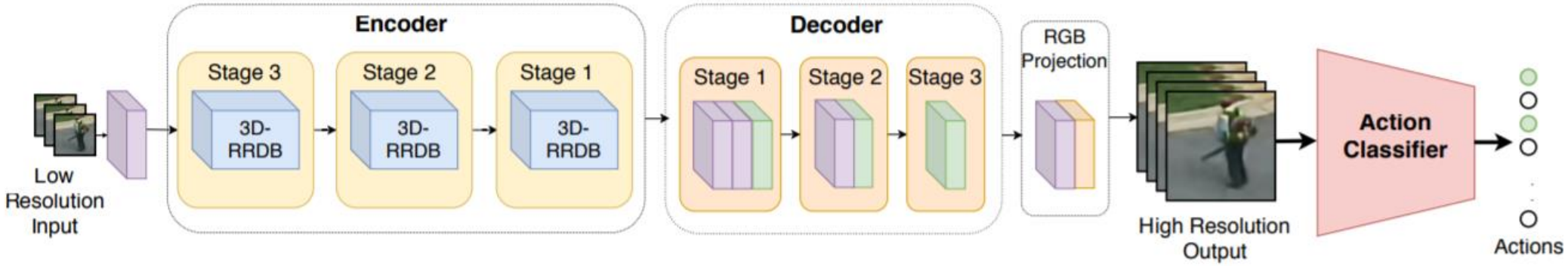
Face recognition



Tiny object detection



Medical imaging



Video action recognition