Image Super-Resolution Using Deep Convolutional Networks

(3) implementation

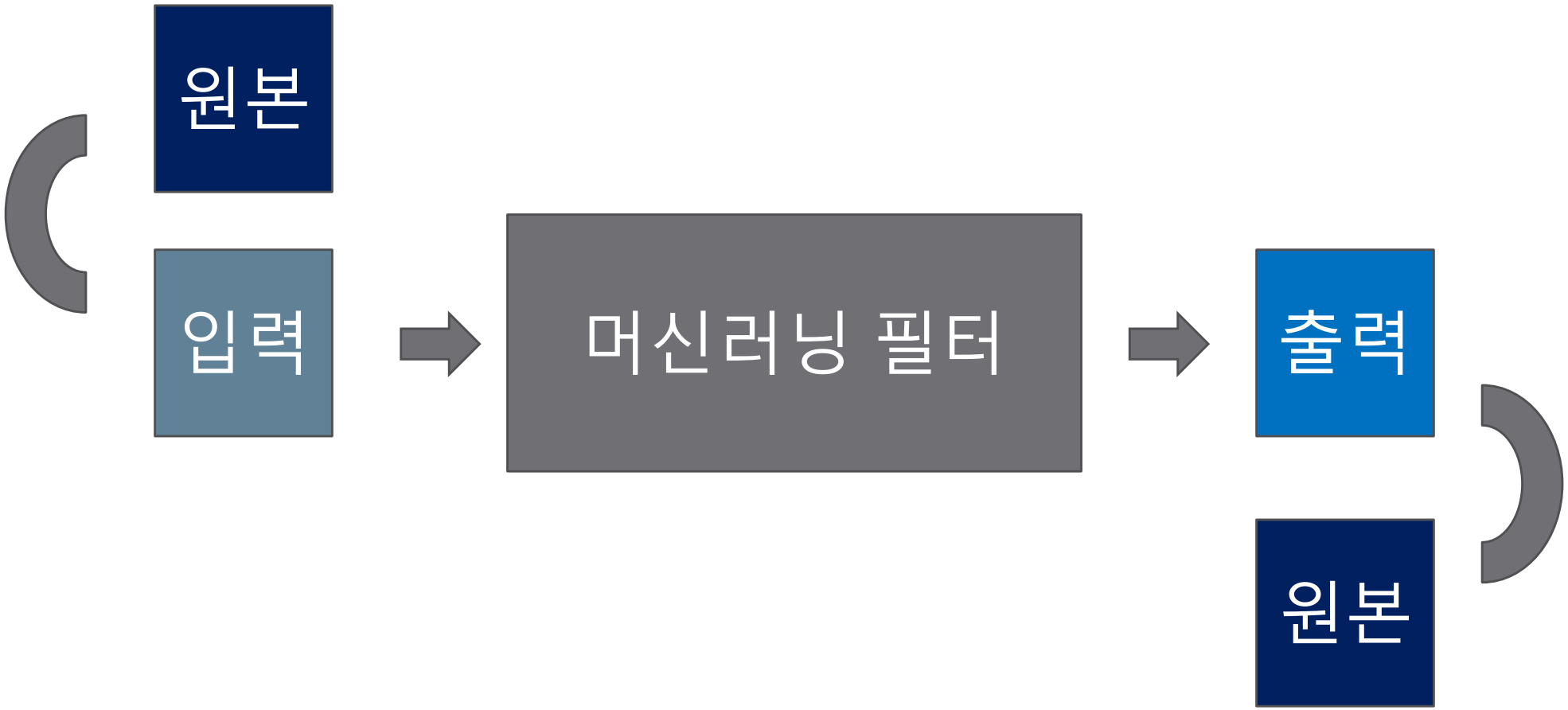
Visual Computing Lab

YoungHoon Kwon

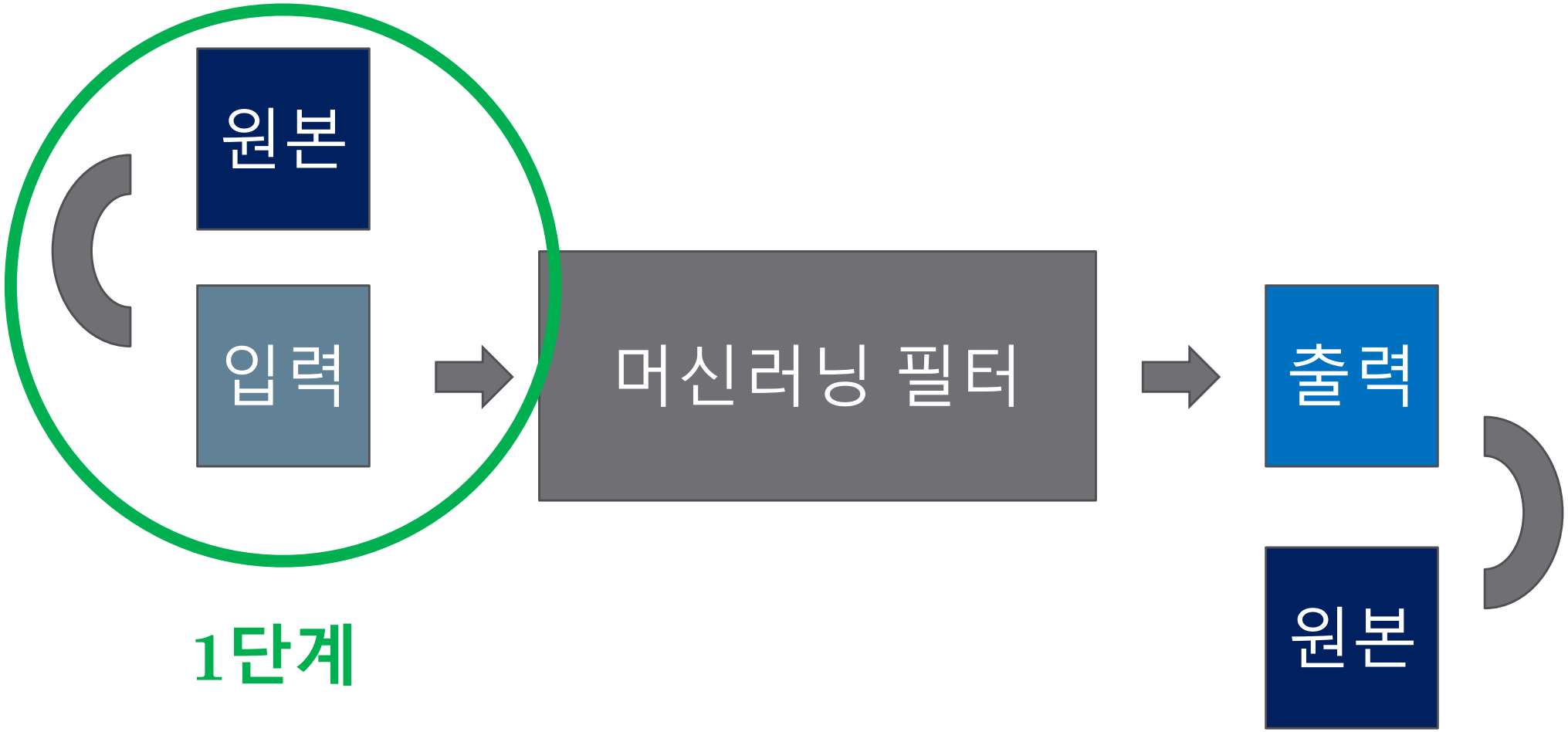
Order

- Main Idea
- Step 1
- Step 2
- Step 3
- Result
- Future works

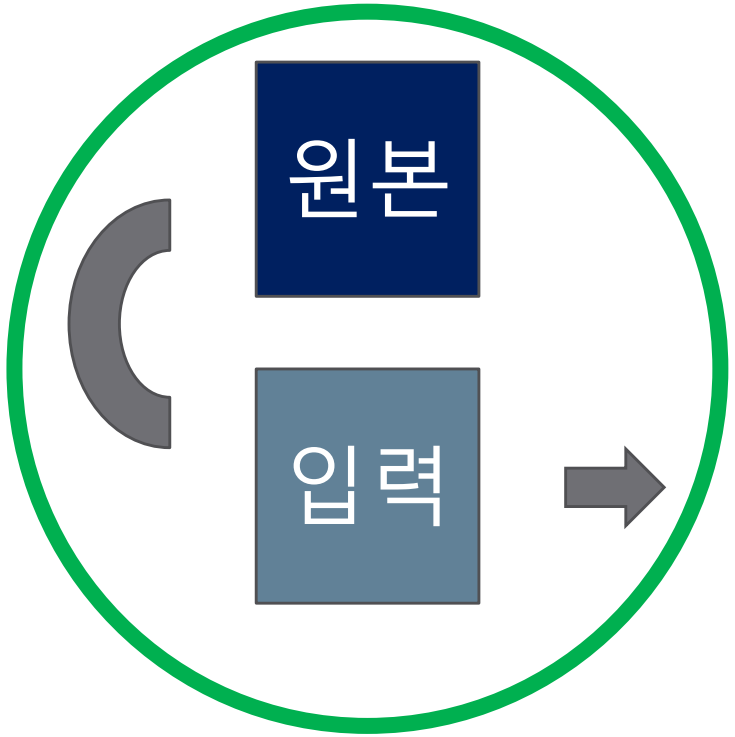
Main Idea

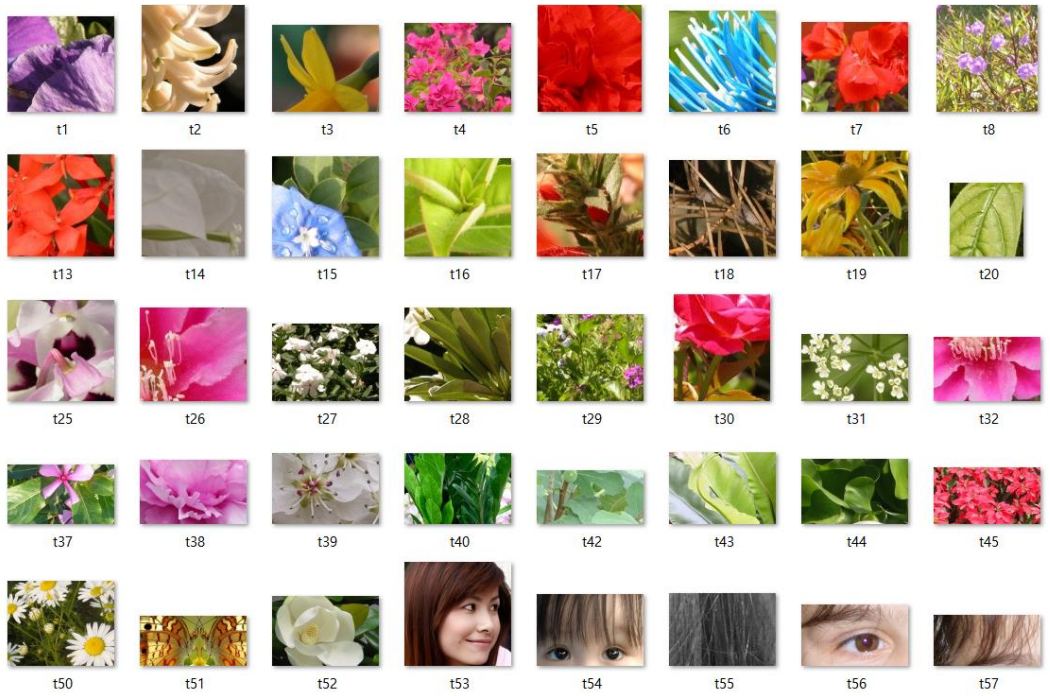


Main Idea

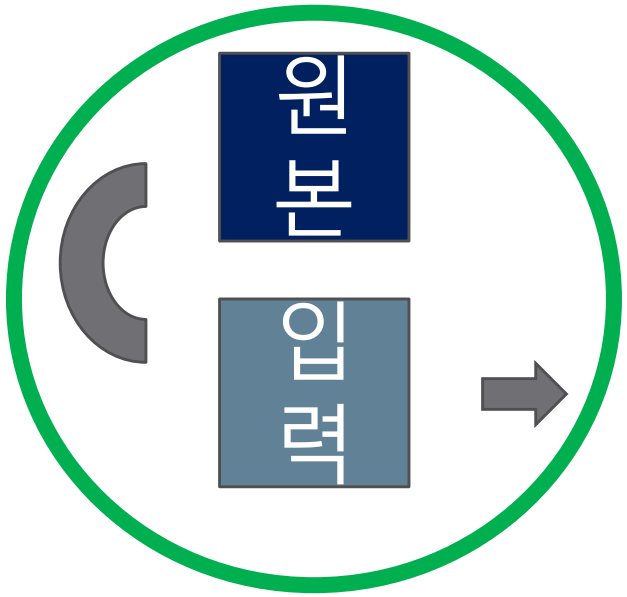


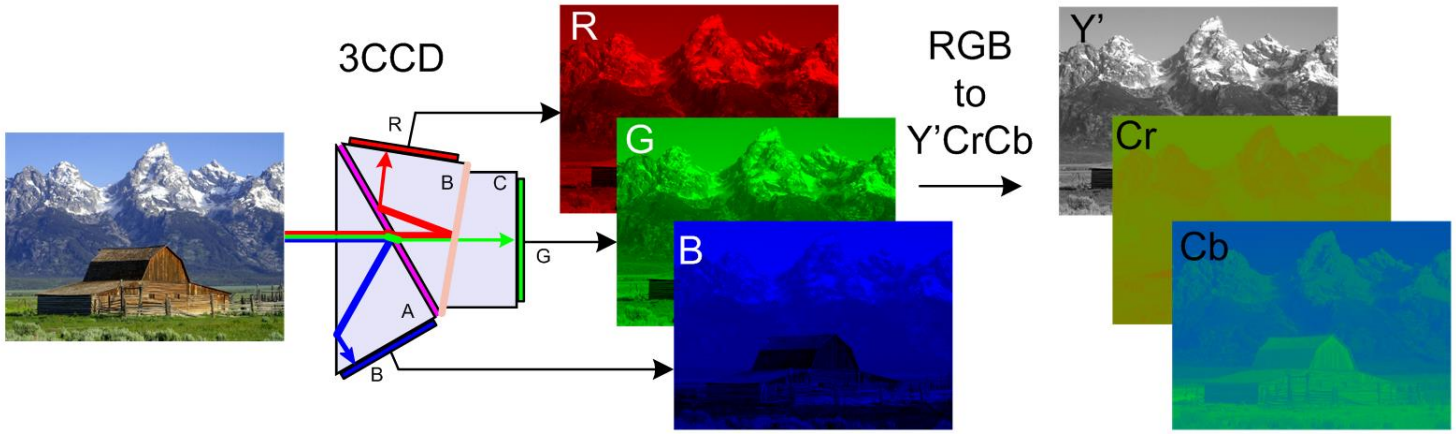
Step 1-1 Training Set



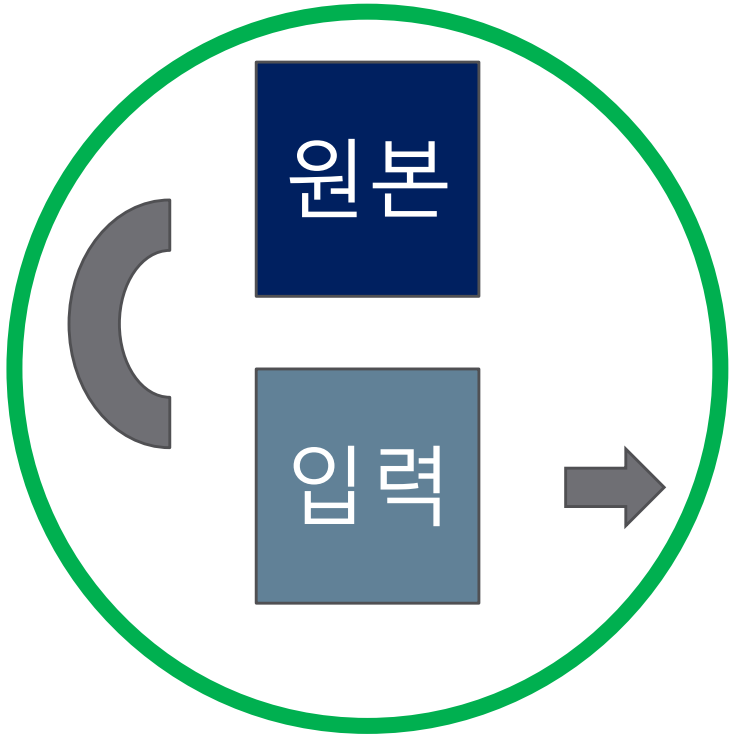


Step 1-2 RGB to YCrCb



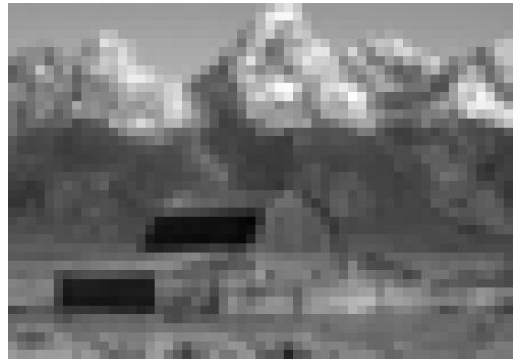


Step 1-3 Making low-resolution

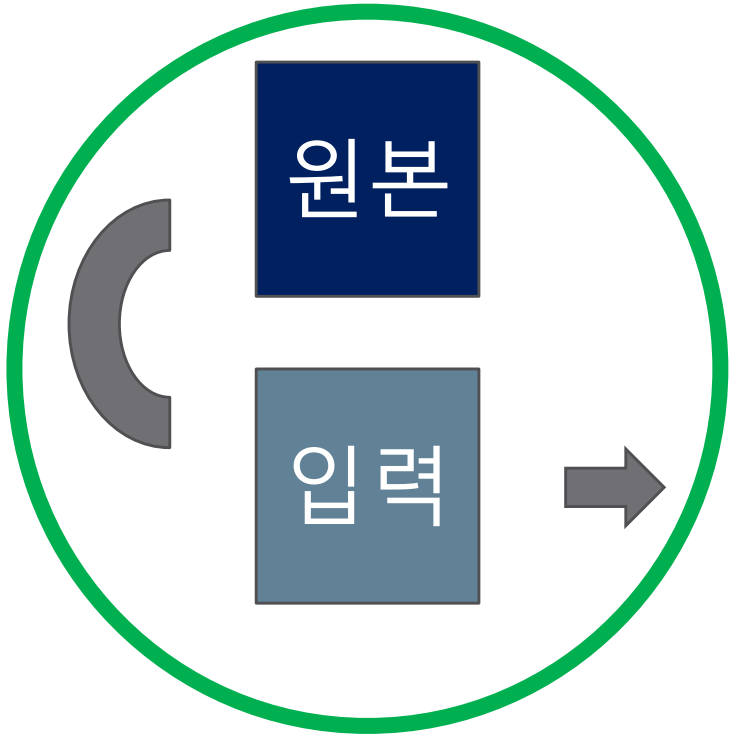


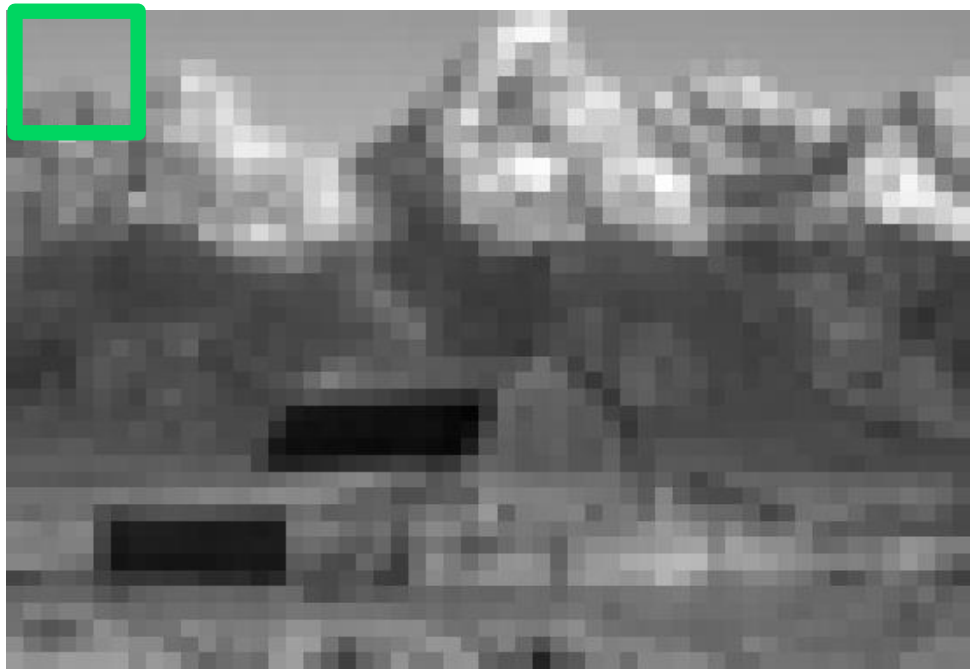
÷3

x3



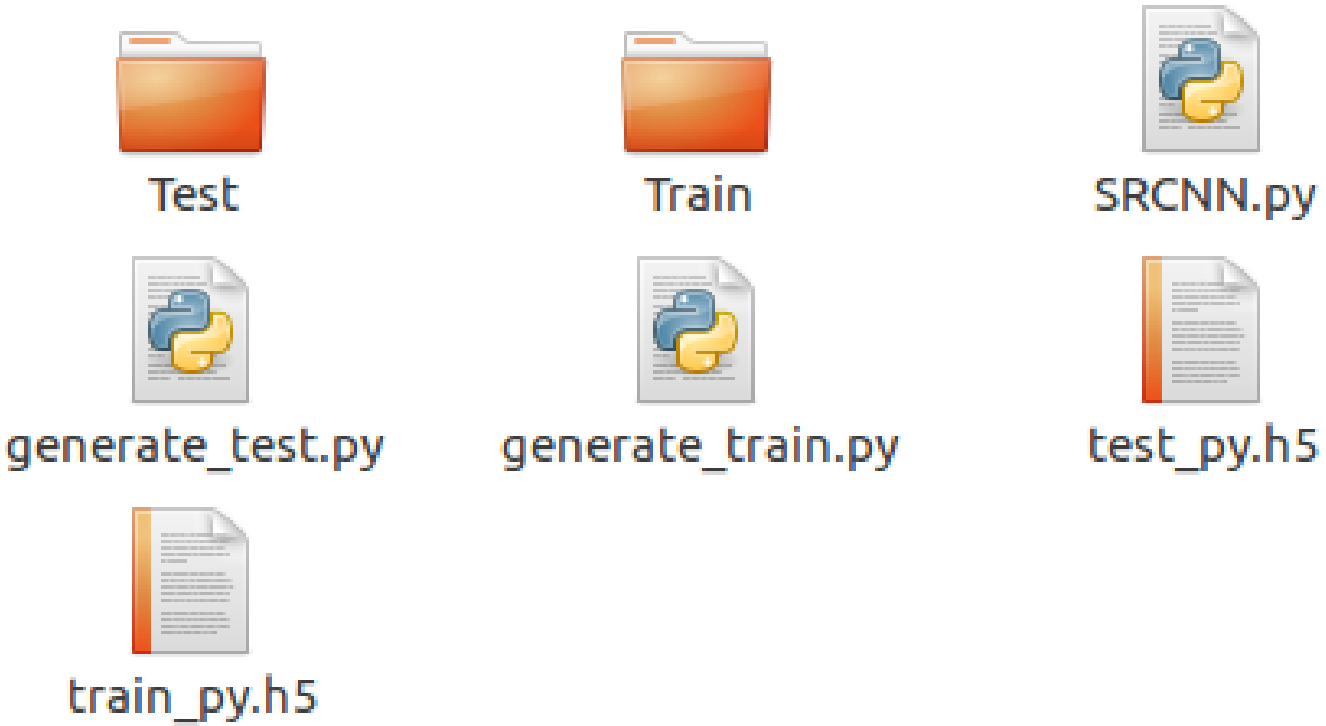
Step 1-4 Divide to 33x33





33 by 33

File Structure



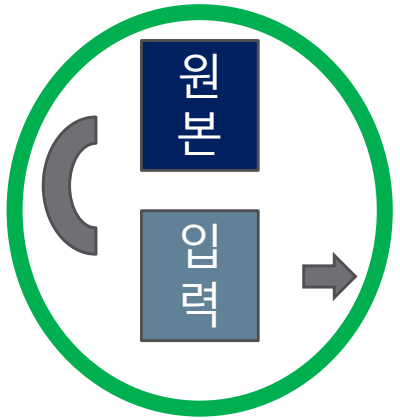
Step 1 Generate Train Import

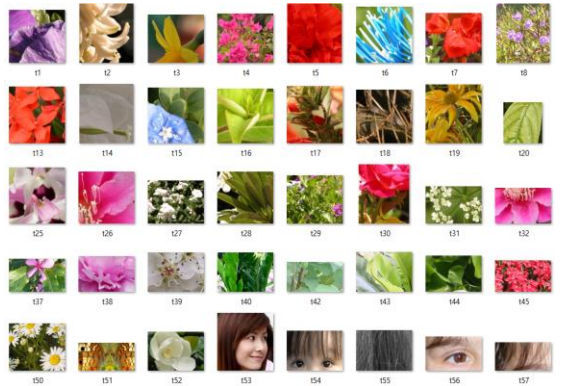
```
import numpy as np
import cv2
import os
import matplotlib.pyplot as plt
import h5py
import math
```

Step 1 Generate Train

```
folder = 'Train'
savepath = 'train_py.h5'
size_input = 33
size_label = 21
scale = 3
stride = 14

data = []
label = []

input_images = []
label_images = []
```

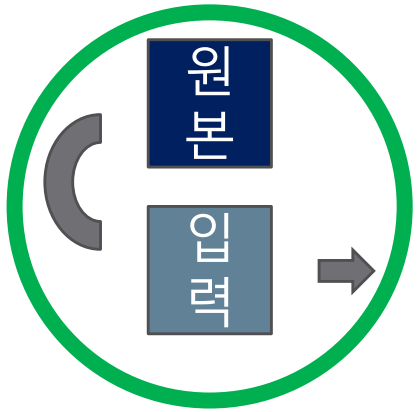
Step 1-1 Training Set

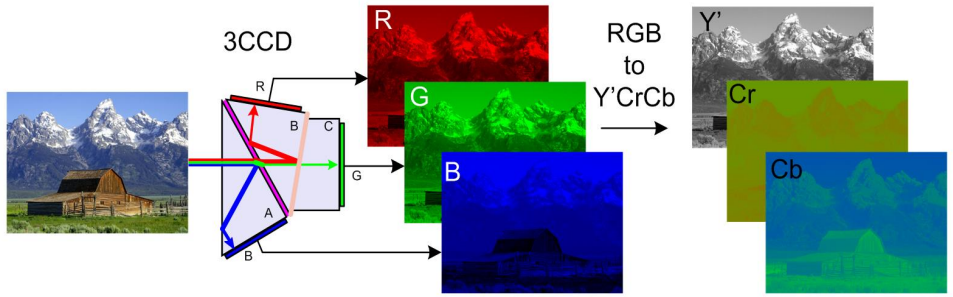




```
for (root, dir, files) in os.walk(folder):
    for file in files:
        filepath = root+'/'+file

        image = cv2.imread(filepath)
```

Step 1-2 RGB to GRAY

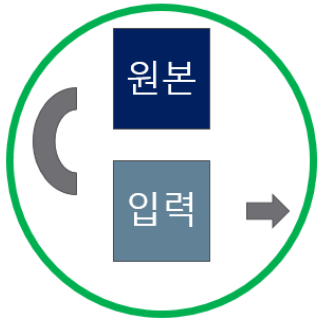


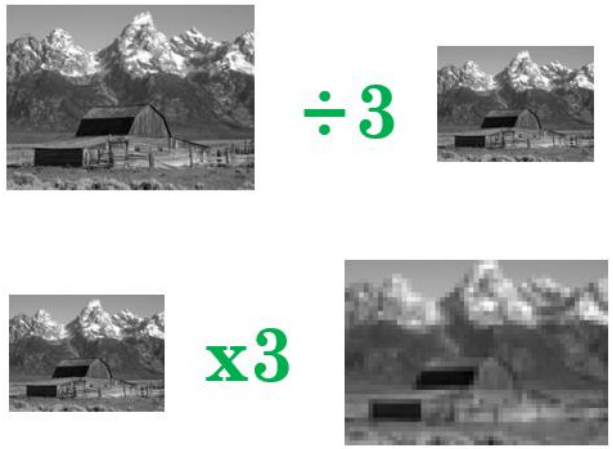



```
for (root, dir, files) in os.walk(folder):
    for file in files:
        filepath = root+'/'+file

        image = cv2.imread(filepath)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        #image = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
        image = image[:,:]
        image = image.astype('float') / 255.0          # int to float
```

Step 1-3 Making low-resolution

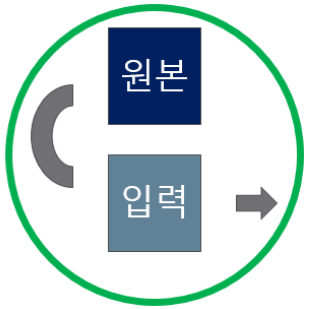
```
for (root, dir, files) in os.walk(folder):
    for file in files:
```

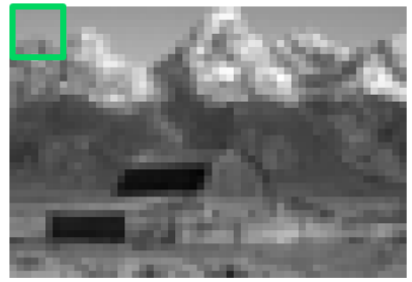





```
im_input = cv2.resize(im_label, (0,0), fx=1.0/scale, fy=1.0/scale, interpolation = cv2.INTER_CUBIC)
im_input = cv2.resize(im_input, (0,0), fx=scale, fy=scale, interpolation = cv2.INTER_CUBIC) # return original size

input_images.append(im_input)
label_images.append(im_label)
```

Step 1-4 Divide to 33x33





33 by 33

```
for (root, dir, files) in os.walk(folder):
    for file in files:
```

⋮

```
        (hei, wid) = im_label.shape

        for x in range(0, hei - size_input + 1, stride) :
            for y in range(0, wid - size_input + 1, stride) :

                subim_input = im_input[x:x+size_input, y: y+size_input]
                subim_label = im_label[x:x+size_input, y: y+size_input]

                subim_input = subim_input.reshape([size_input,size_input,1])
                subim_label = subim_label.reshape([size_input,size_input,1])

                data.append(subim_input)
                label.append(subim_label)
```

Step 1 Generate_Train

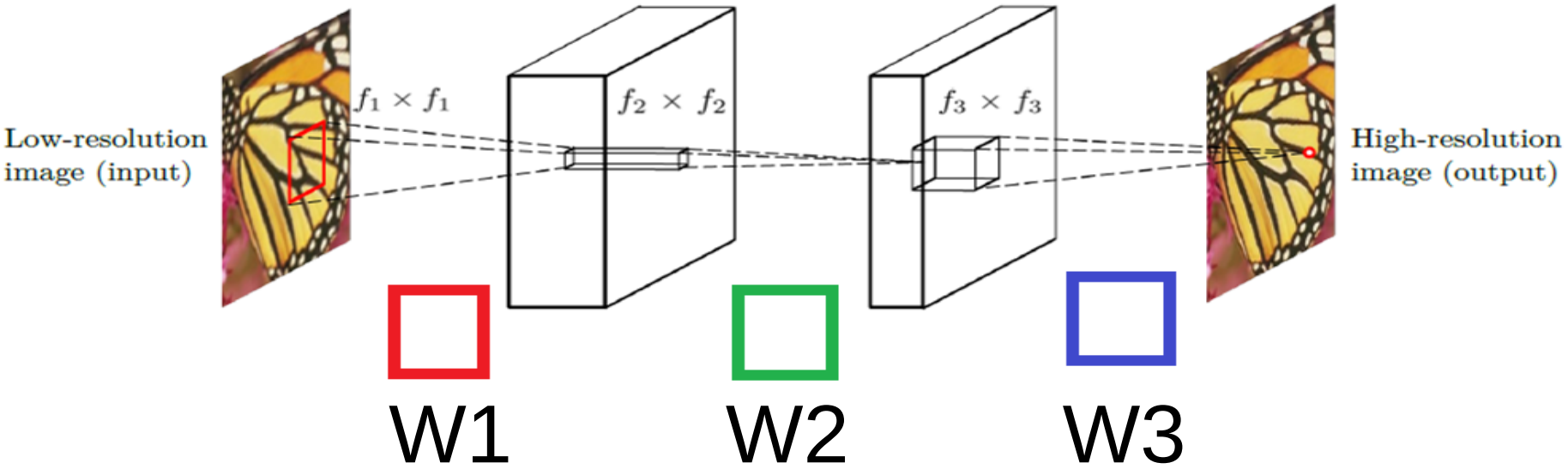
```
with h5py.File(savepath,'w') as hf:
    hf.create_dataset('dataset_1',data=data)
    hf.create_dataset('dataset_2',data=label)
```

```
def modcrop(imgs, modulo):
    if np.size(imgs.shape)==3:
        (sheight, swidth,_) = image.shape
        sheight = sheight - np.mod(sheight,modulo)
        swidth = swidth - np.mod(swidth,modulo)
        imgs = imgs[0:sheight,0:swidth,:]
    else:
        (sheight, swidth) = image.shape
        sheight = sheight - np.mod(sheight,modulo)
        swidth = swidth - np.mod(swidth,modulo)
        imgs = imgs[0:sheight,0:swidth]

    return imgs
```

Step 2 Make filters

2단계

Step 2-1 Make filters



필터 W	9*9*1	1*1*64	5*5*32
출력	64	32	1
상수항	B1(64)	B2(32)	B3(1)

Step 2-2 Define function

$$F_1(\mathbf{Y}) = \max(0, W_1 * \mathbf{Y} + B_1), \quad (1)$$

$$F_2(\mathbf{Y}) = \max(0, W_2 * F_1(\mathbf{Y}) + B_2). \quad (2)$$

$$F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3. \quad (3)$$

Step 2-3 Loss function

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i;\Theta) - \mathbf{X}_i||^2$$

$$\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$$

Step 2-1 Make filters

	W1	W2	W3
필터 W	9*9*1	1*1*64	5*5*32
출력	64	32	1
상수항	B1(64)	B2(32)	B3(1)

```
X = tf.placeholder("float", [None, 33, 33, 1])
Y = tf.placeholder("float", [None, 33, 33, 1])

W1 = init_weights([9, 9, 1, 64])        # 9x9x1 conv, 64 outputs
W2 = init_weights([1, 1, 64, 32])      # 1x1x64 conv, 32 outputs
W3 = init_weights([5, 5, 32, 1])       # 5x5x32 conv, 1 output

B1 = tf.Variable(tf.zeros([64]), name="Bias1")
B2 = tf.Variable(tf.zeros([32]), name="Bias2")
B3 = tf.Variable(tf.zeros([1]), name="Bias3")
```

Step 2-2 Define function

$$F_1(\mathbf{Y}) = \max(0, W_1 * \mathbf{Y} + B_1), \quad (1)$$

$$F_2(\mathbf{Y}) = \max(0, W_2 * F_1(\mathbf{Y}) + B_2). \quad (2)$$

$$F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3. \quad (3)$$

```
L1 = tf.nn.relu(tf.nn.conv2d(X, W1,                          # l1 shape=(?, 33, 33, 64)
                strides=[1, 1, 1, 1], padding='SAME') + B1)
L2 = tf.nn.relu(tf.nn.conv2d(L1, W2,                         # l2 shape=(?, 33, 33, 32)
                strides=[1, 1, 1, 1], padding='SAME') + B2)

hypothesis = tf.nn.conv2d(L2, W3,                          # l3 shape=(?, 33, 33, 1)
                strides=[1, 1, 1, 1], padding='SAME') + B3
```

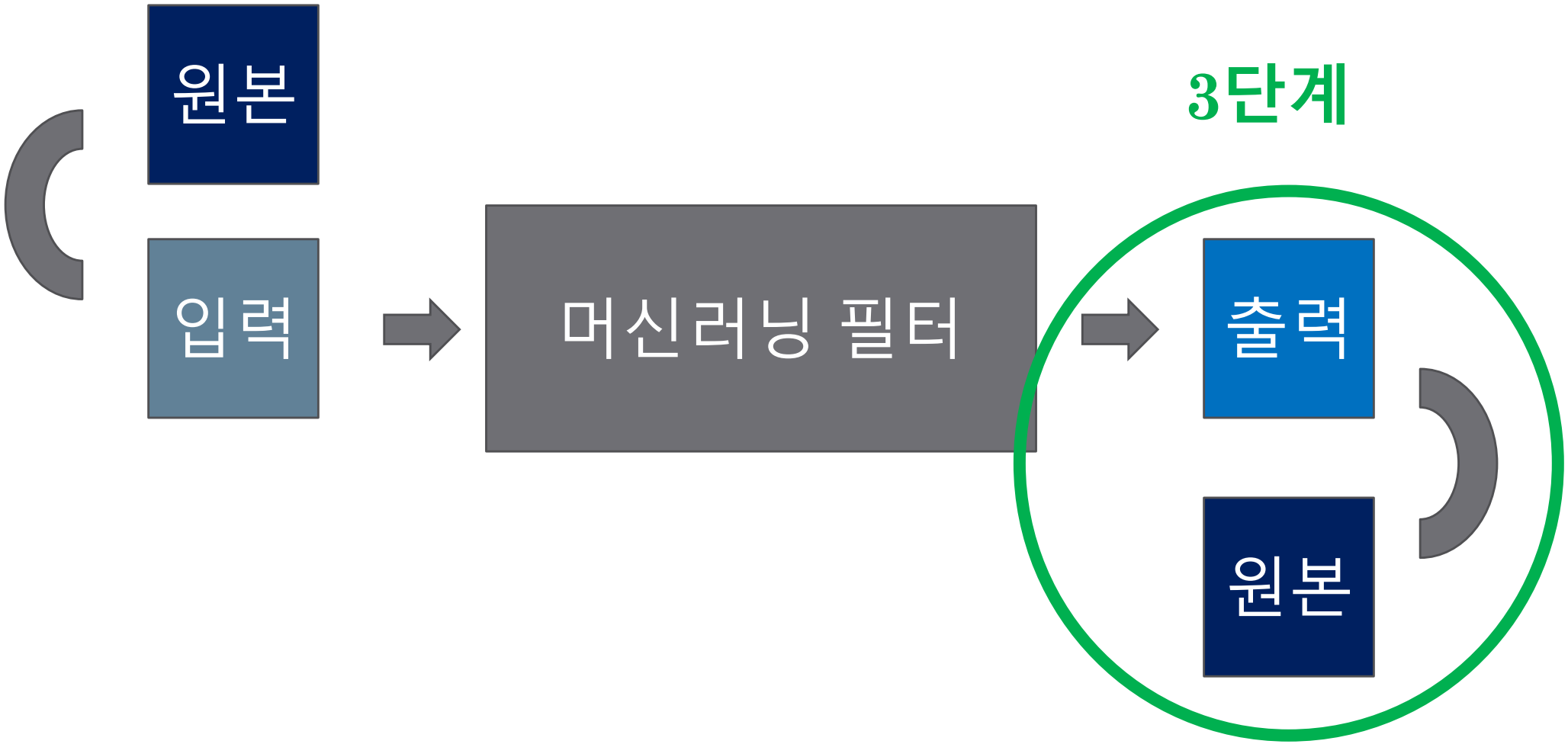
Step 2-3 Loss function

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i;\Theta) - \mathbf{X}_i||^2$$

$$\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$$

```
cost = tf.reduce_mean(tf.reduce_sum(tf.square(hypothesis - Y), reduction_indices=1))

optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

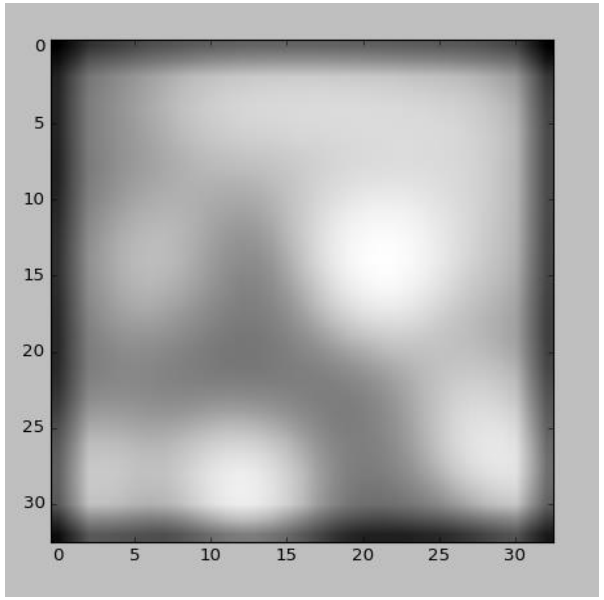
Main Idea

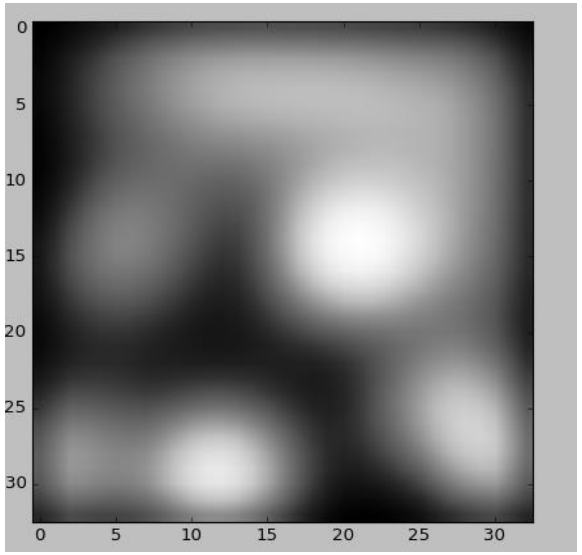


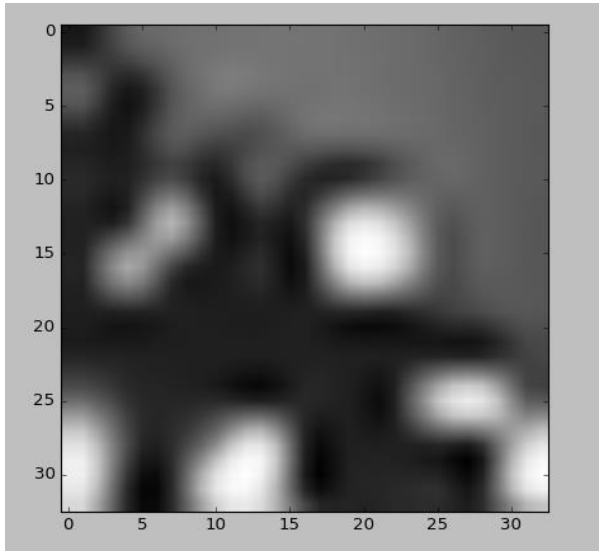
Step 3 Run Tensorflow

```
with tf.Session() as sess:

    tf.initialize_all_variables().run()

    for i in range(train_num):
        sess.run(optimizer, feed_dict={X: data, Y: label})

        step+=1
        if step%100==0 : print (step," : ",sess.run(cost, feed_dict={X:data, Y: label }))
```

Result








```
(995, ' : ', 1.0389839)
(996, ' : ', 1.0383712)
(997, ' : ', 1.0377547)
(998, ' : ', 1.0371395)
(999, ' : ', 1.0365237)
(1000, ' : ', 1.0359094)
```

1000회

```
(1600, ' : ', 0.70586002)
(1700, ' : ', 0.66383034)
(1800, ' : ', 0.62883192)
(1900, ' : ', 0.6000275)
(2000, ' : ', 0.57646954)
```

2000회

Future works

- 33by33 → 21by21
- 1 channel -> 3 channels
- Batch normalization