LAB 02

# Super Resolution with Deep Learning

Sunghwan Kim

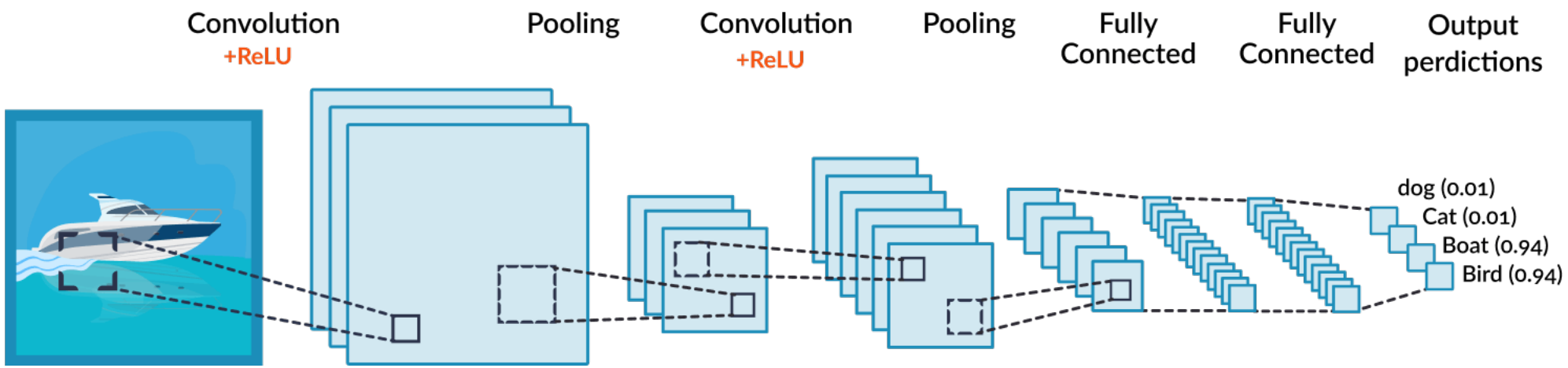
For Keynote Purposes
MS Powerpoint Online

# Index

# Convolutional Neural Network

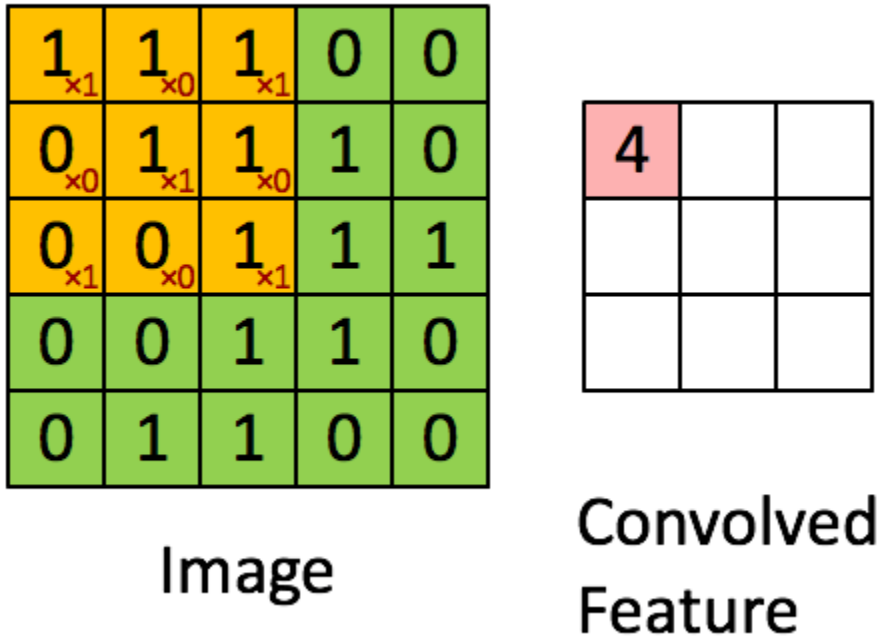
**Definition**

들어오는 2차원 데이터를 학습한 필터(filter)와 합성곱 연산을 진행하여 들어오는 데이터의 특징(feature)을 추출하여 학습하는 신경망(Neural Network).



# Convolutional Neural Network

Convolution | **합성곱** 연산



# Convolutional Neural Network

## Convolution | **합성곱** 연산

Input image

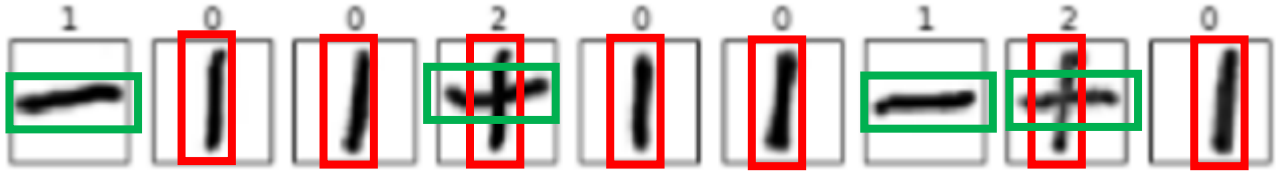
Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

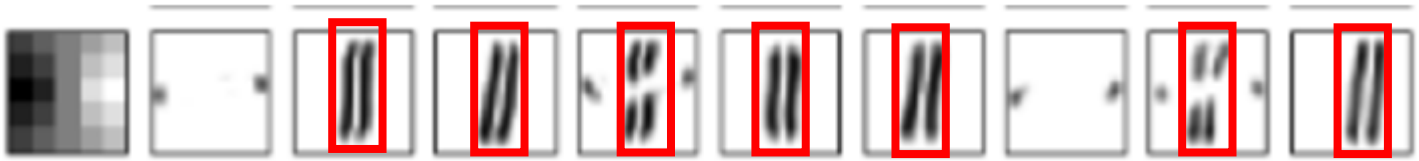
Feature map

# Convolutional Neural Network

## Why we use Convolution? => Feature Extraction!



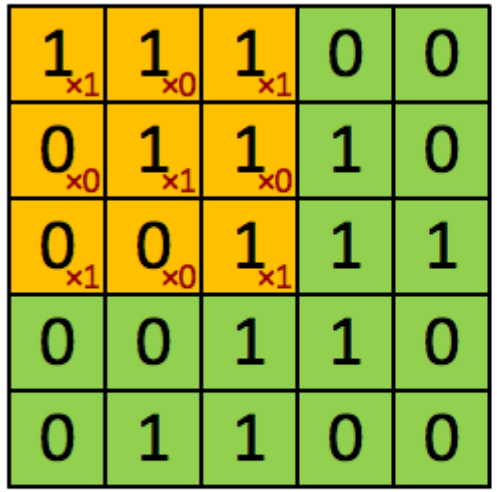

Filter that can extract vertical line



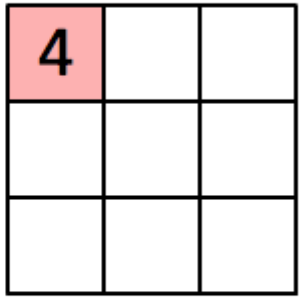
Filter that can extract horizontal line

# Convolutional Neural Network

## Convolution | **합성곱** 연산




Image

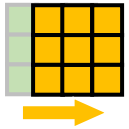



Convolved Feature

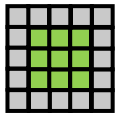
### Terminology

**Channel** : RGB input –〉 3 Channels

**Filter (Kernel)** : Kernel size 3x3



**Stride** : 1 step

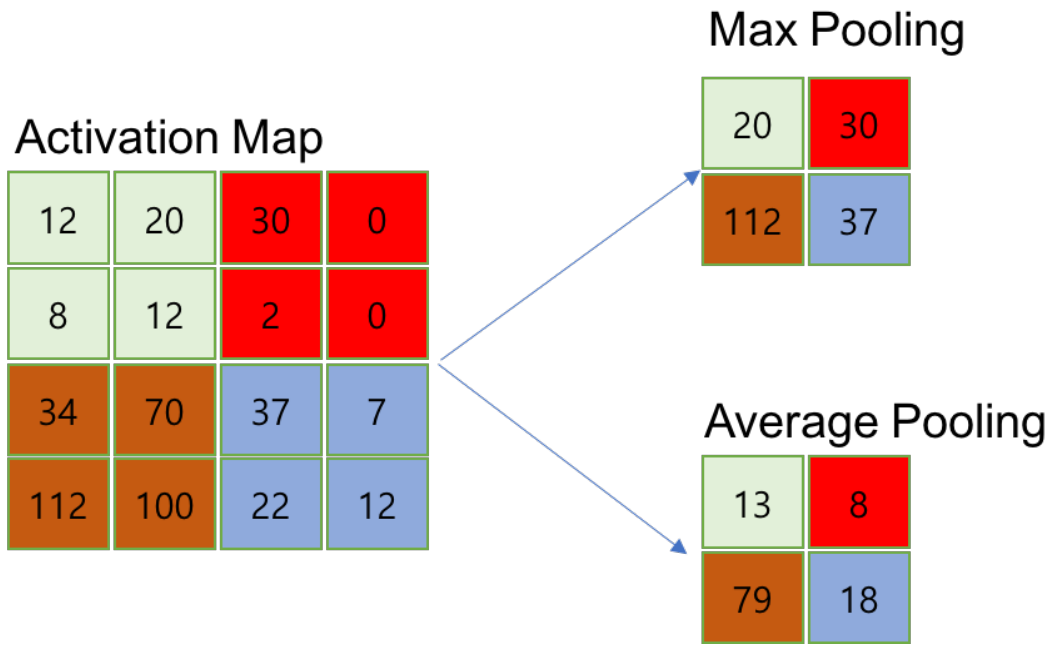


**Padding (Zero-padding)** : 1 pixel

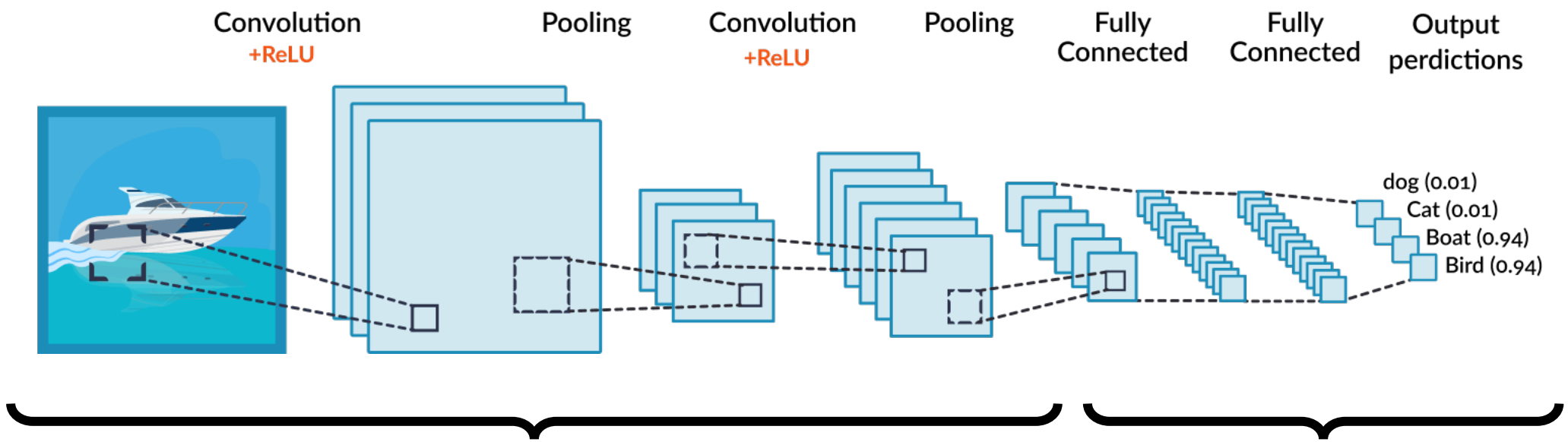
# Convolutional Neural Network

## Pooling Layer

Feature의 크기를 줄여 연산량을 줄이고, Overfitting을 방지!



# Convolutional Neural Network



# Super Resolution

Super
Resolution

이미지의 해상도를 더 큰 해상도로 키우는 것(Upscaling)

# Super Resolution

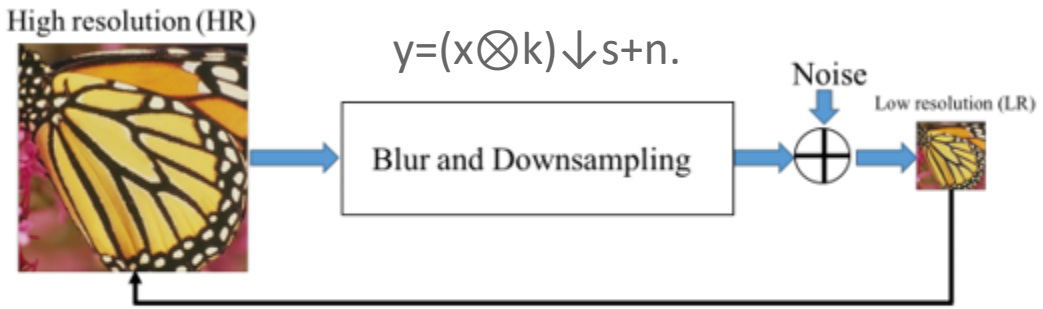


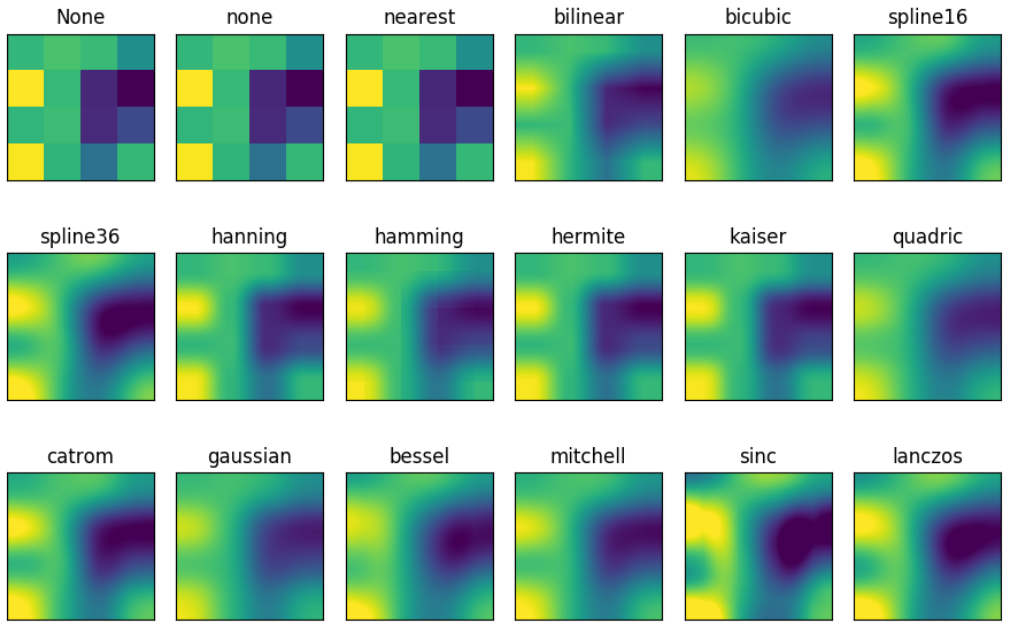

Figure 1: Sketch of the overall framework of SISR.

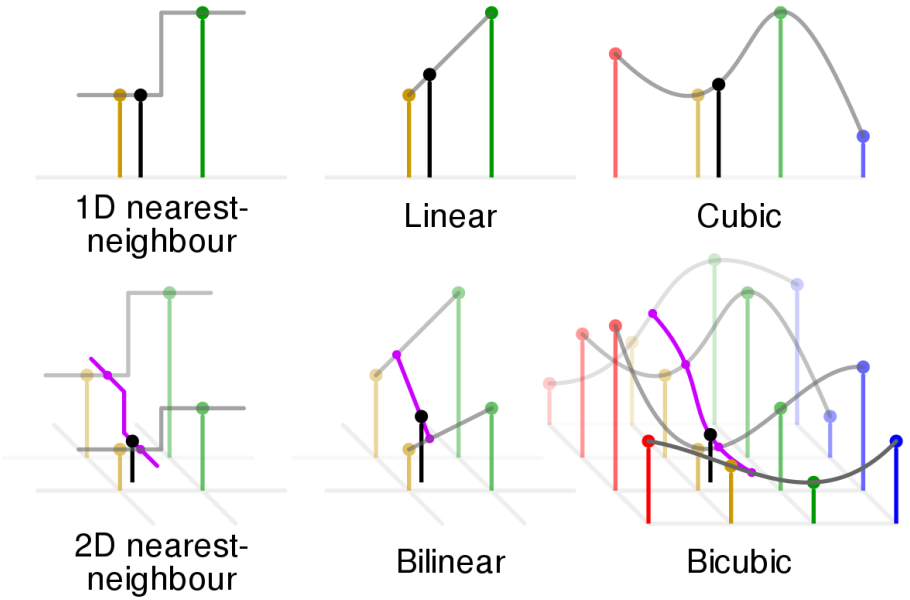
# Super Resolution

## Interpolation based Method

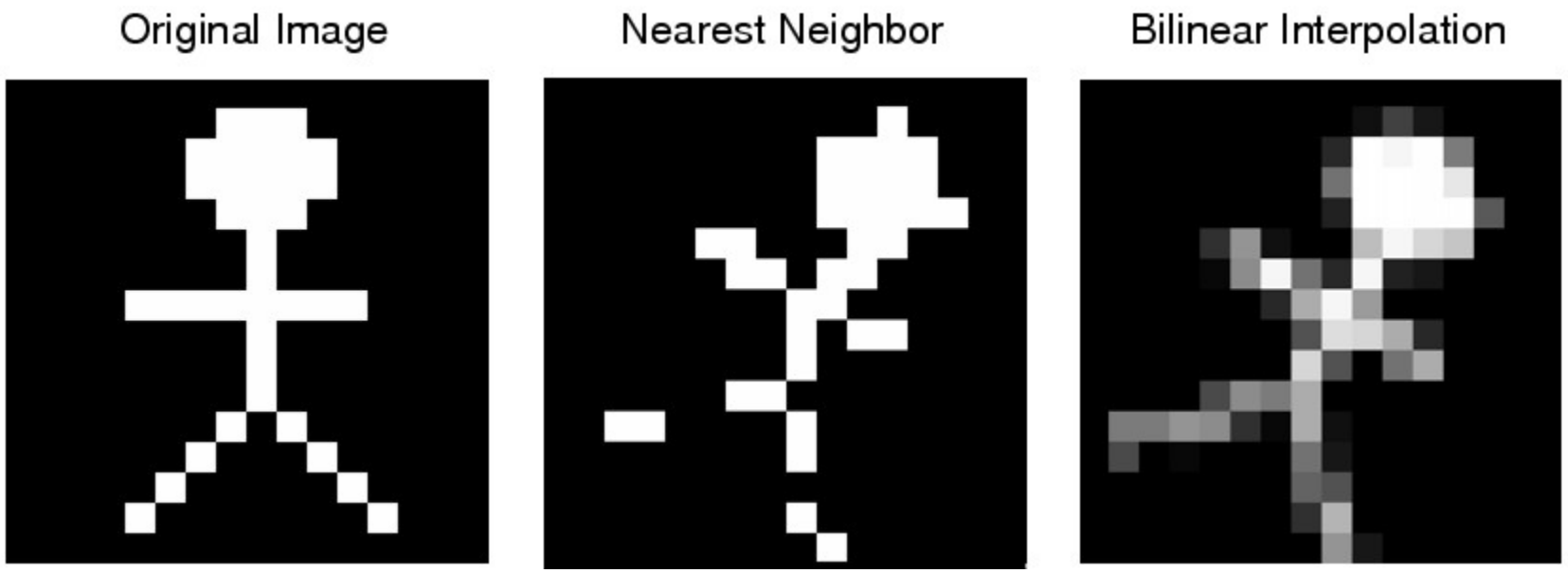
**Definition**

보간법(補間法, interpolation)은 끝점의 값이 주어졌을 때 그 사이에 위치한 값을 추정하기 위하여 모종의 방법을 통해 수많은 점들을 평균내는 방법을 말한다.



# Super Resolution



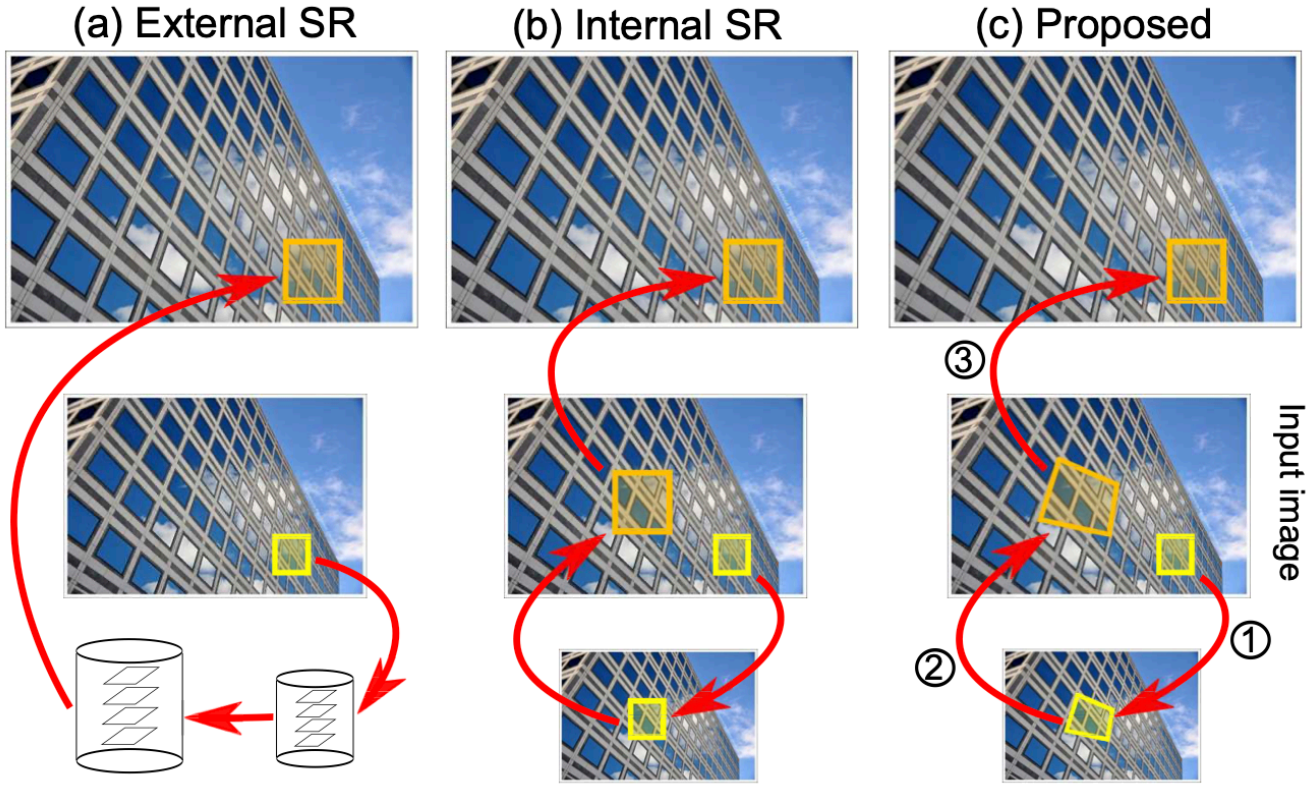

**화질의 저하가 심하다.**

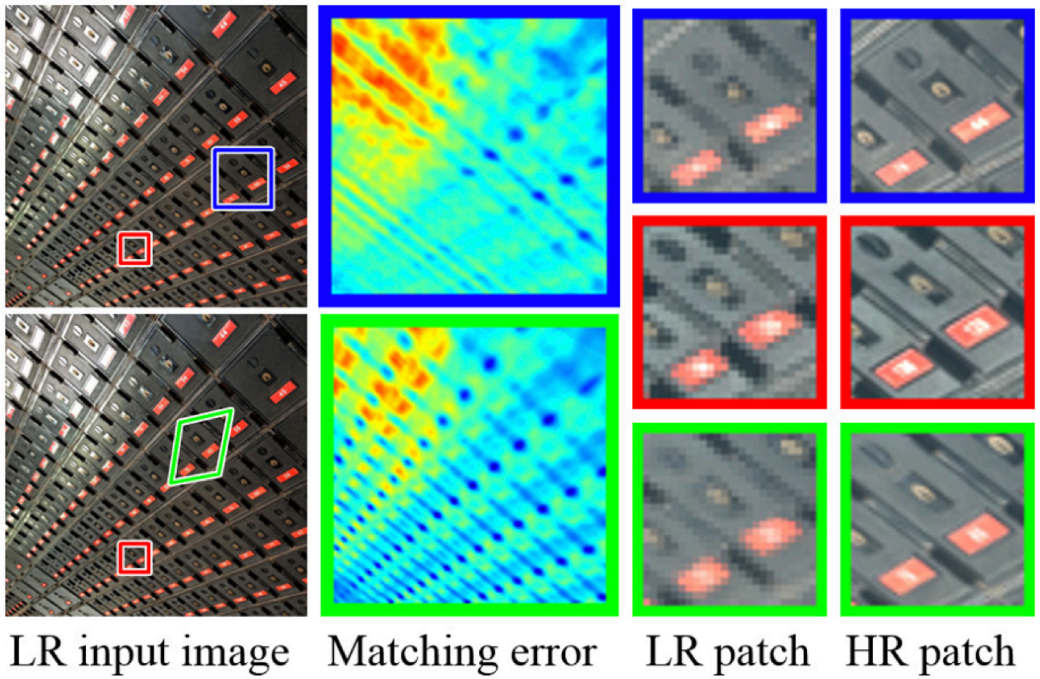
# Super Resolution

# Super Resolution

## Resconstruction based Method

## SelfExSR: Single Image Super-resolution from Transformed Self-Exemplars



# Super Resolution

## Learning based Method

## SRCNN : Image Super-Resolution Using Deep Convolutional Networks (2014 ECCV)

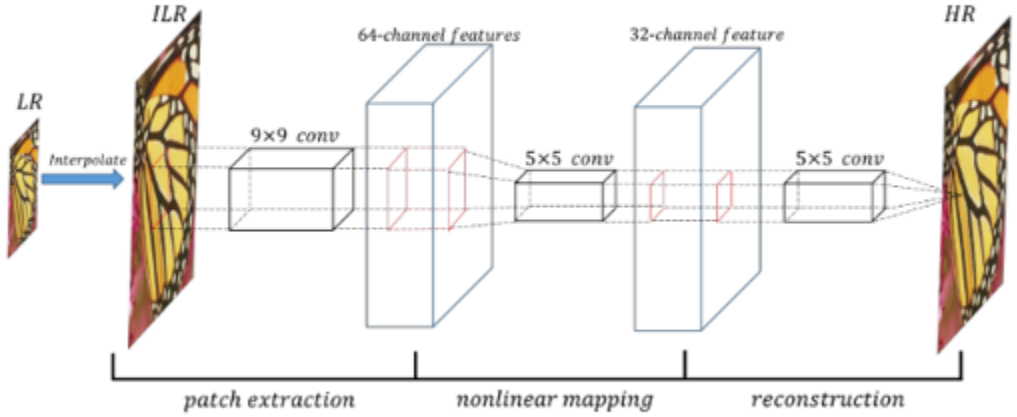
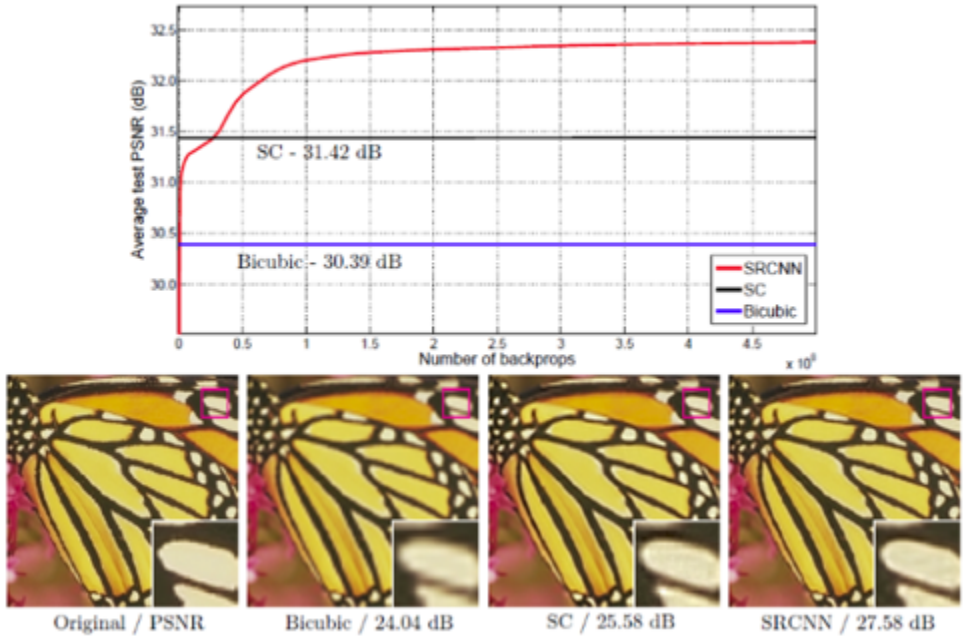



Figure 2: Sketch of the SRCNN architecture.



# Super Resolution

## SRCNN : Image Super-Resolution Using Deep Convolutional Networks (2014 ECCV)

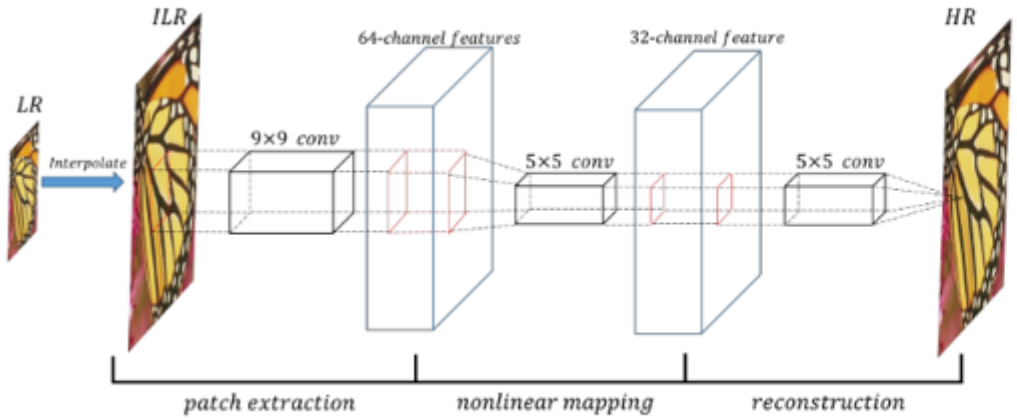



Figure 2: Sketch of the SRCNN architecture.

**Patch Extraction & Representation**

저해상도(LR)이미지에서 patch를 뽑아 feature를 추출한다.

**Nonlinear Mapping**

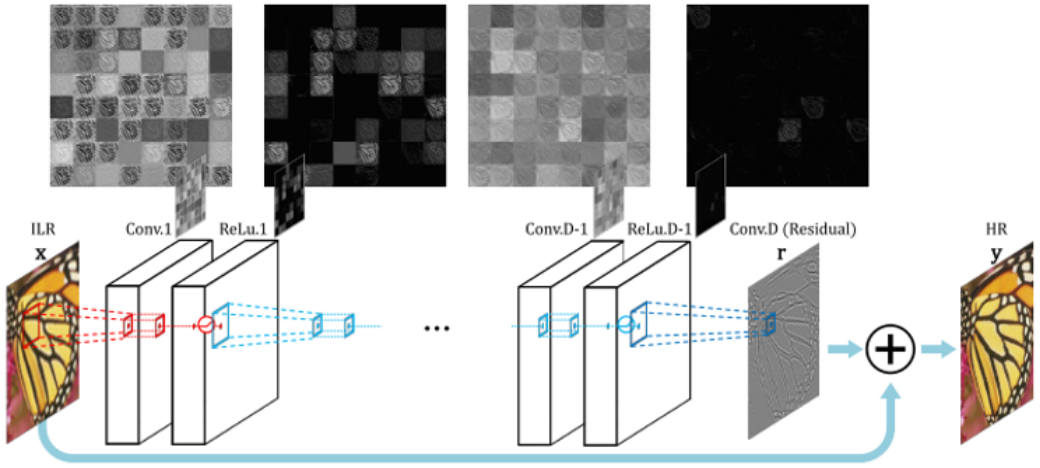
Feature가 담긴 patch들을 nonlinear하게 다른 다차원의 patch들로 mapping시킨다.

**Reconstruction**

Mapping된 patch들을 바탕으로 새로운 고해상도(HR)이미지를 생성해낸다.

# Super Resolution

## VDSR : Accurate Image Super-Resolution Using Very Deep Convolutional Networks (2016 CVPR)




Reference: "Accurate Image Super-Resolution Using Very Deep Convolutional Networks", 2016 CVPR

| Epoch | 10 | 20 | 40 | 80 |
|---|---|---|---|---|
| Residual | 36.90 | 36.64 | 37.12 | 37.05 |
| Non-Residual | 27.42 | 19.59 | 31.38 | 35.66 |
| Difference | 9.48 | 17.05 | 5.74 | 1.39 |

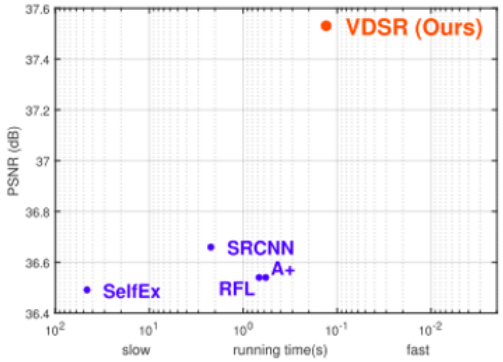
(a) Initial learning rate 0.1

| Epoch | 10 | 20 | 40 | 80 |
|---|---|---|---|---|
| Residual | 36.74 | 36.87 | 36.91 | 36.93 |
| Non-Residual | 30.33 | 33.59 | 36.26 | 36.42 |
| Difference | 6.41 | 3.28 | 0.65 | 0.52 |

(b) Initial learning rate 0.01

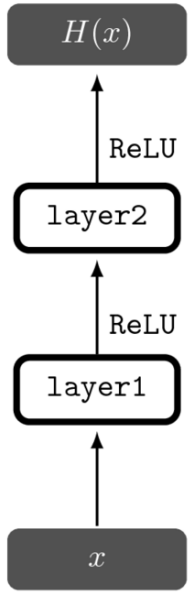
| Epoch | 10 | 20 | 40 | 80 |
|---|---|---|---|---|
| Residual | 36.31 | 36.46 | 36.52 | 36.52 |
| Non-Residual | 33.97 | 35.08 | 36.11 | 36.11 |
| Difference | 2.35 | 1.38 | 0.42 | 0.40 |

(c) Initial learning rate 0.001

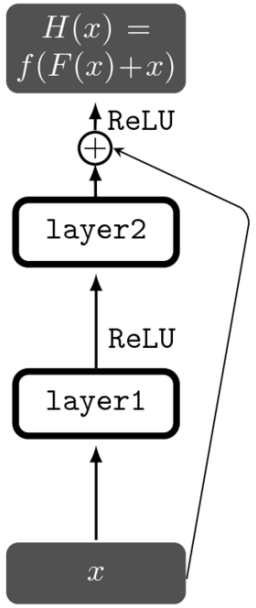


# Super Resolution

## Residual Network : Skip Connection




일반적 뉴럴넷




Skip connection

Without residual

$Network: y = H(x)$

$Object: minimize\ H(x) - y$

With residual

$Network: y = H(x) + x$

$Object: find\ H(x) - x = 0$

# Super Resolution

# Super Resolution

## Accurate Image Super-Resolution Using Very Deep Convolutional Networks

Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee
Department of ECE, ASRI, Seoul National University, Korea
{j.kim, deruci, kyoungmu}@snu.ac.kr

### Abstract

*We present a highly accurate single-image super-resolution (SR) method. Our method uses a very deep convolutional network inspired by VGG-net used for ImageNet classification [19]. We find increasing our network depth shows a significant improvement in accuracy. Our final model uses 20 weight layers. By cascading small filters many times in a deep network structure, contextual information over large image regions is exploited in an efficient way. With very deep networks, however, convergence speed becomes a critical issue during training. We propose a simple yet effective training procedure. We learn residuals only and use extremely high learning rates ($10^4$ times higher than SRCNN [6]) enabled by adjustable gradient clipping. Our proposed method performs better than existing methods in accuracy and visual improvements in our results are easily noticeable.*

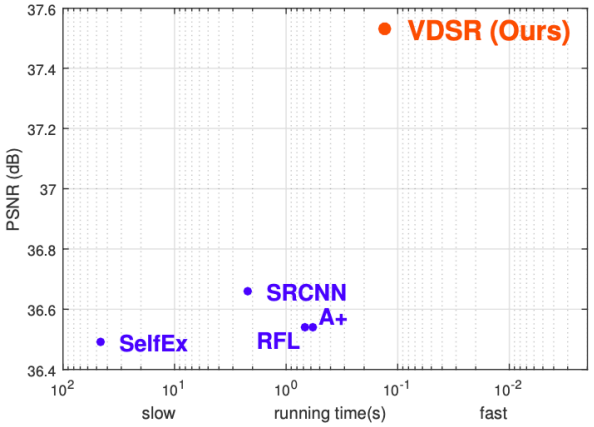



**Figure 1:** Our VDSR improves PSNR for scale factor $\times 2$ on dataset Set5 in comparison to the state-of-the-art methods (SRCNN uses the public slower implementation using CPU). VDSR outperforms SRCNN by a large margin (0.87 dB).

# Super Resolution

**Training dataset** Different learning-based methods use different training images. For example, RFL [18] has two methods, where the first one uses 91 images from Yang et al. [25] and the second one uses 291 images with the addition of 200 images from Berkeley Segmentation Dataset [16]. SRCNN [6] uses a very large ImageNet dataset.

We use 291 images as in [18] for benchmark with other methods in this section. In addition, data augmentation (rotation or flip) is used. For results in previous sections, we used 91 images to train network fast, so performances can be slightly different.

**Test dataset** For benchmark, we use four datasets. Datasets 'Set5' [15] and 'Set14' [26] are often used for benchmark in other works [22, 21, 5]. Dataset 'Urban100', a dataset of urban images recently provided by Huang et al. [11], is very interesting as it contains many challenging images failed by many of the existing methods. Finally, dataset 'B100', natural images in the Berkeley Segmentation Dataset used in Timofte et al. [22] and Yang and Yang [24] for benchmark, is also employed.

Results on BSDS100

# Super Resolution

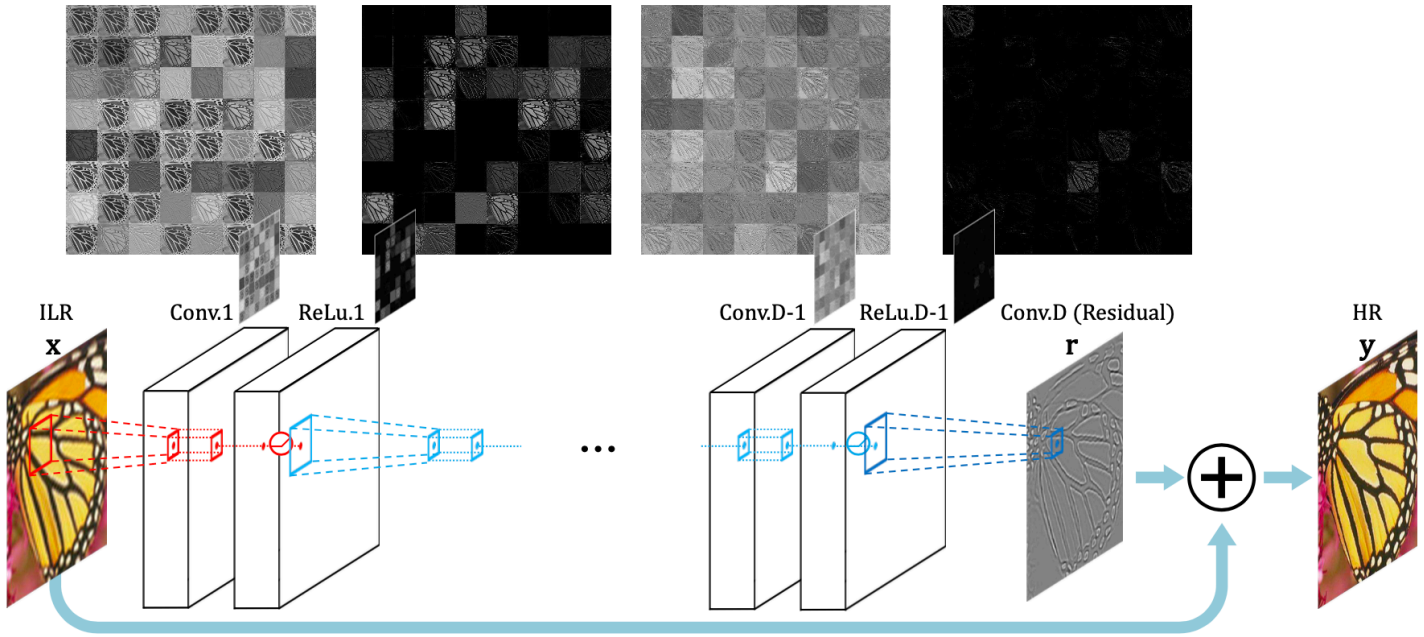



**Figure 2:** Our Network Structure. We cascade a pair of layers (convolutional and nonlinear) repeatedly. An interpolated low-resolution

**Model** SRCNN consists of three layers: patch extraction/representation, non-linear mapping and reconstruction. Filters of spatial sizes $9 \times 9$, $1 \times 1$, and $5 \times 5$ were used respectively.

However, we argue that increasing depth significantly boosts performance. We successfully use 20 weight layers ($3 \times 3$ for each layer). Our network is very deep (20 vs. 3 [6]) and information used for reconstruction (receptive field) is much larger ($41 \times 41$ vs. $13 \times 13$).

## 3. Proposed Method

### 3.1. Proposed Network

For SR image reconstruction, we use a very deep convolutional network inspired by Simonyan and Zisserman [19]. The configuration is outlined in Figure 2. We use $d$ layers where layers except the first and the last are of the same type: 64 filter of the size $3 \times 3 \times 64$ where a filter operates on $3 \times 3$ spatial region across 64 channels (feature maps). The first layer operates on the input image. The last layer, used for image reconstruction, consists of a single filter of size $3 \times 3 \times 64$.

Kernel Size: 3 x 3, Channel: 64,
Convolution Layer: 20

# Super Resolution

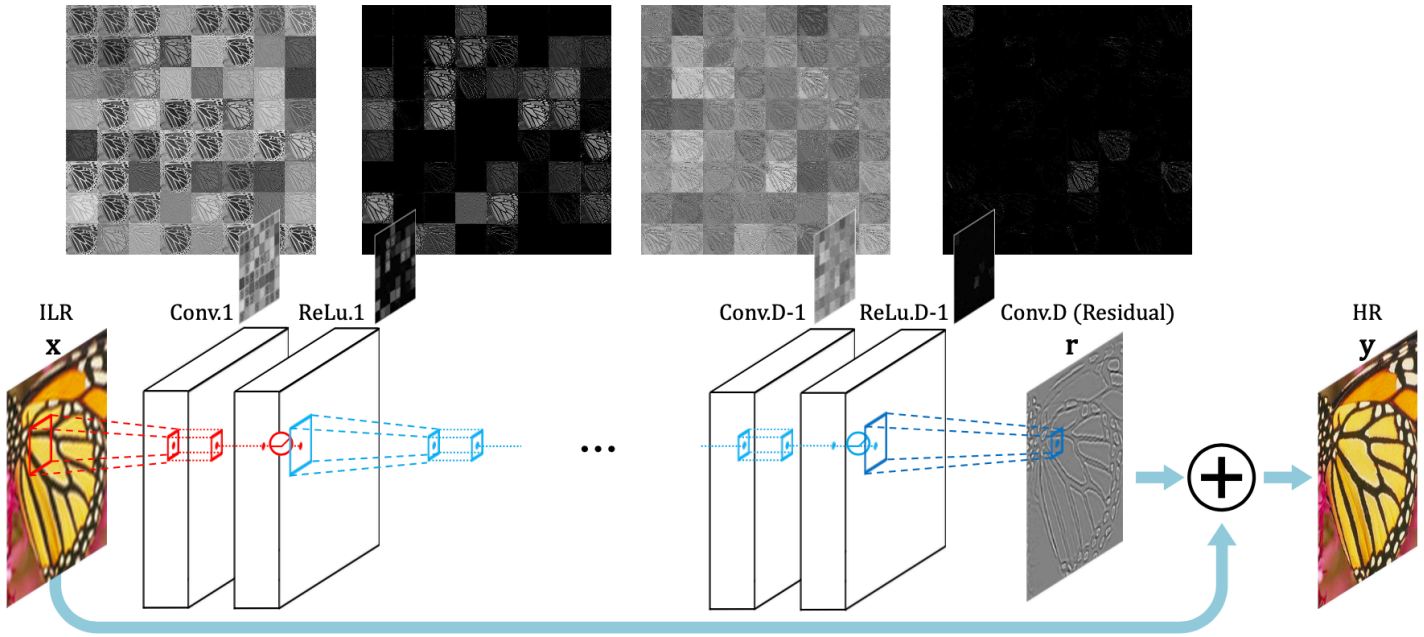



**Figure 2:** Our Network Structure. We cascade a pair of layers (convolutional and nonlinear) repeatedly. An interpolated low-resolution

One problem with using a very deep network to predict dense outputs is that the size of the feature map gets reduced every time convolution operations are applied. For example, when an input of size $(n+1)\times(n+1)$ is applied to a network with receptive field size $n \times n$, the output image is $1 \times 1$.

Stride: 1

To resolve this issue, we pad zeros before convolutions to keep the sizes of all feature maps (including the output image) the same. It turns out that zero-padding works surprisingly well. For this reason, our method differs from most other methods in the sense that pixels near the image boundary are also correctly predicted.

Padding: 1

# Super Resolution

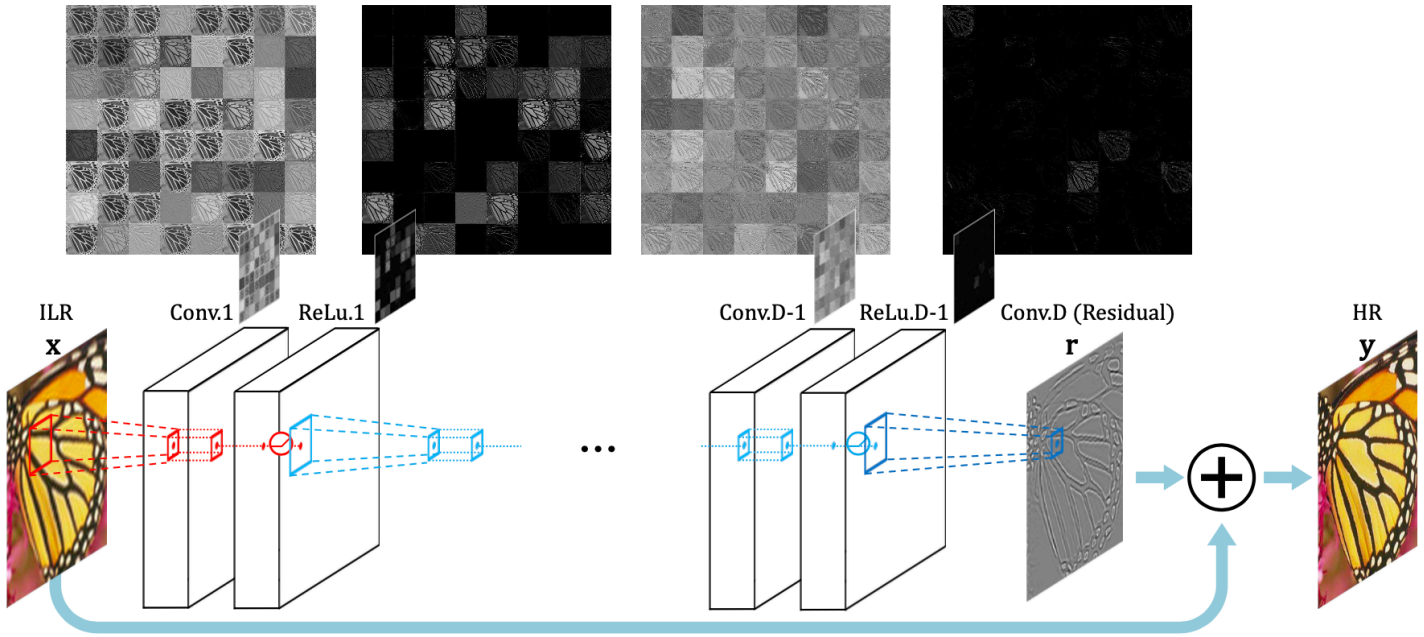



**Figure 2:** Our Network Structure. We cascade a pair of layers (convolutional and nonlinear) repeatedly. An interpolated low-resolution

**Residual-Learning** In SRCNN, the network must preserve all input detail since the image is discarded and the output is generated from the learned features alone. With many weight layers, this becomes an end-to-end relation requiring very long-term memory. For this reason, the vanishing/exploding gradients problem [2] can be critical. We can solve this problem simply with residual-learning.

As the input and output images are largely similar, we define a residual image $\mathbf{r} = \mathbf{y} - \mathbf{x}$, where most values are likely to be zero or small. We want to predict this residual image. The loss function now becomes $\frac{1}{2}||\mathbf{r} - f(\mathbf{x})||^2$, where $f(\mathbf{x})$ is the network prediction.

In networks, this is reflected in the loss layer as follows. Our loss layer takes three inputs: residual estimate, network input (ILR image) and ground truth HR image. The loss is computed as the Euclidean distance between the reconstructed image (the sum of network input and output) and ground truth.

Residual Learning Required

# Super Resolution

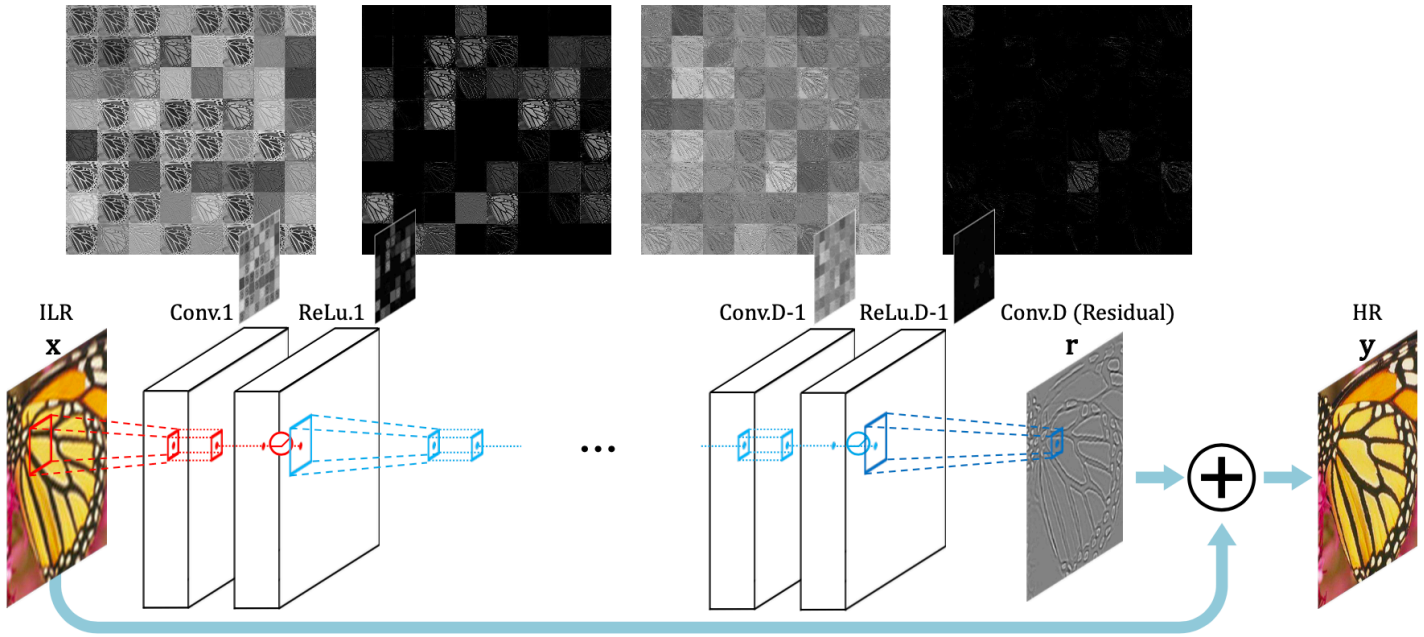
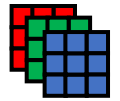
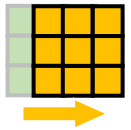



**Figure 2:** Our Network Structure. We cascade a pair of layers (convolutional and nonlinear) repeatedly. An interpolated low-resolution

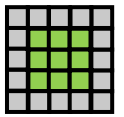


**Channel** : 64 channel

**Filter (Kernel)** : Kernel size 3x3



**Stride** : 1 step



**Padding (Zero-padding)** : 1 pixel

# Super Resolution

### 3.2. Training

We now describe the objective to minimize in order to find optimal parameters of our model. Let $\mathbf{x}$ denote an interpolated low-resolution image and $\mathbf{y}$ a high-resolution image. Given a training dataset $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, our goal is to learn a model $f$ that predicts values $\hat{\mathbf{y}} = f(\mathbf{x})$, where $\hat{\mathbf{y}}$ is an estimate of the target HR image. We minimize the mean squared error $\frac{1}{2}||\mathbf{y} - f(\mathbf{x})||^2$ averaged over the training set is minimized.

Mean Squared Error (MSE)

Training is carried out by optimizing the regression objective using mini-batch gradient descent based on back-propagation (LeCun et al. [14]). We set the momentum parameter to 0.9. The training is regularized by weight decay ($L_2$ penalty multiplied by 0.0001).

Gradient Descent (Momentum 0.9, Weight decay 0.0001)

# Super Resolution

We train all experiments over 80 epochs (9960 iterations with batch size 64). Learning rate was initially set to 0.1 and then decreased by a factor of 10 every 20 epochs. In total, the learning rate was decreased 3 times, and the learning is stopped after 80 epochs. Training takes roughly 4 hours on GPU Titan Z.

80 epoch (Each epoch: 9960 iteration)

Learning Rate: 0.1 (decrease by a factor of 10 every 20 epoch)