인공지능 직무전환 과정 3
Deep Learning

Lab 05

연세대학교 컴퓨터과학과
김선주 강재연 조영현

Today

DeepDream

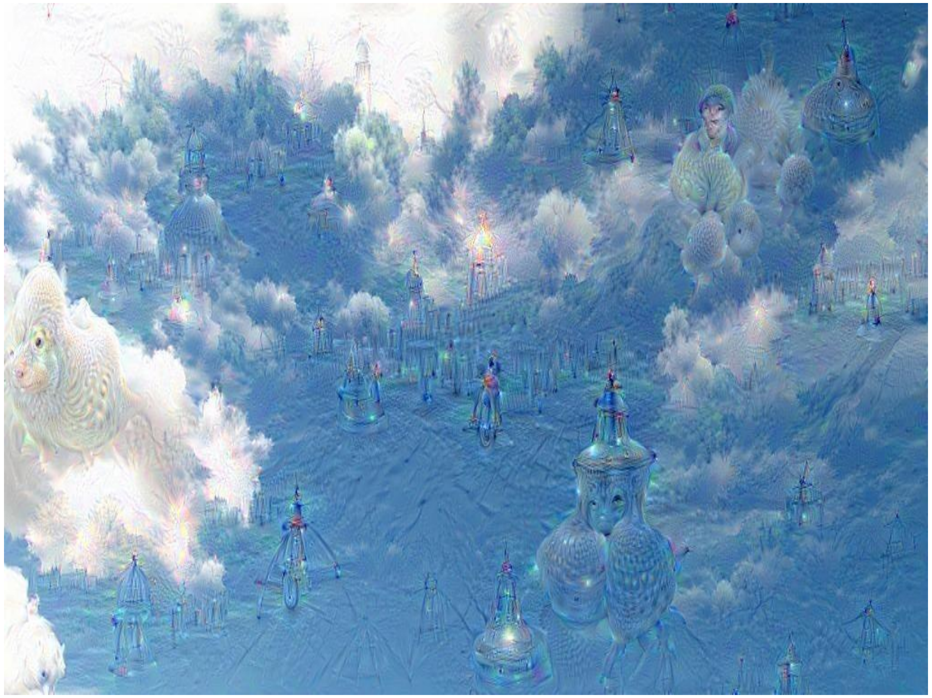
Style transfer

- Content loss
- Style loss
- Total-variation regularization

Super-Resolution

- SRCNN / VDSR
- Loss functions: MSE, MAE, Huber
- Sub-pixel convolution

DeepDream

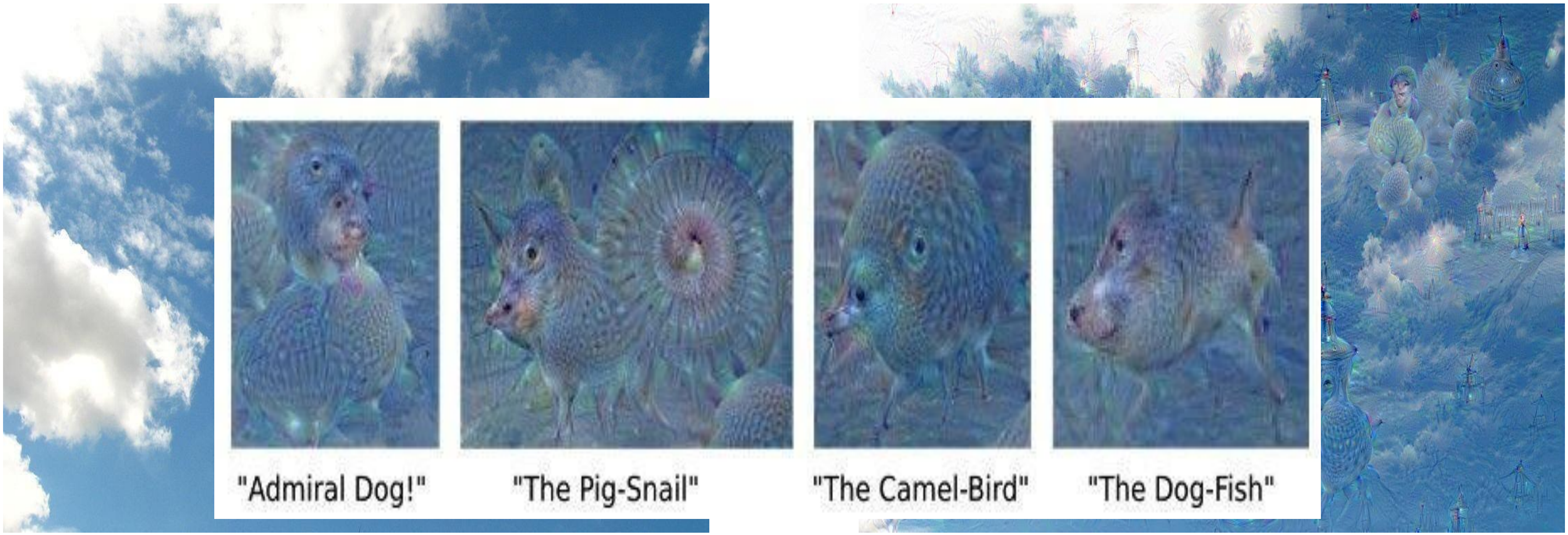


DeepDream modifies the image in a way that “boosts” all activations, at any layer

this creates a feedback loop: e.g. any slightly detected dog face will be made more and more dog like over time

https://github.com/google/deepdream

DeepDream



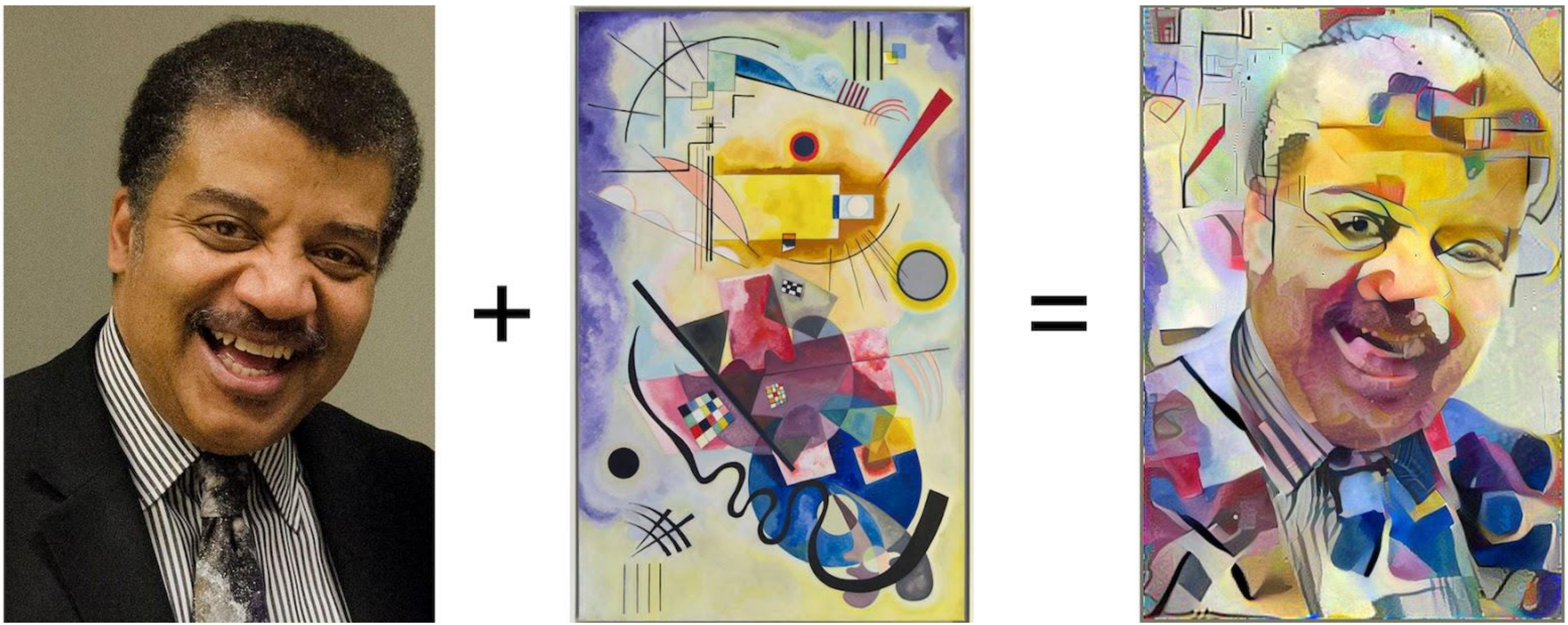
DeepDream modifies the image in a way that "boosts" all activations, at any layer

this creates a feedback loop: e.g. any slightly detected dog face will be made more and more dog like over time

Simple Implementation: `./DeepDream/`

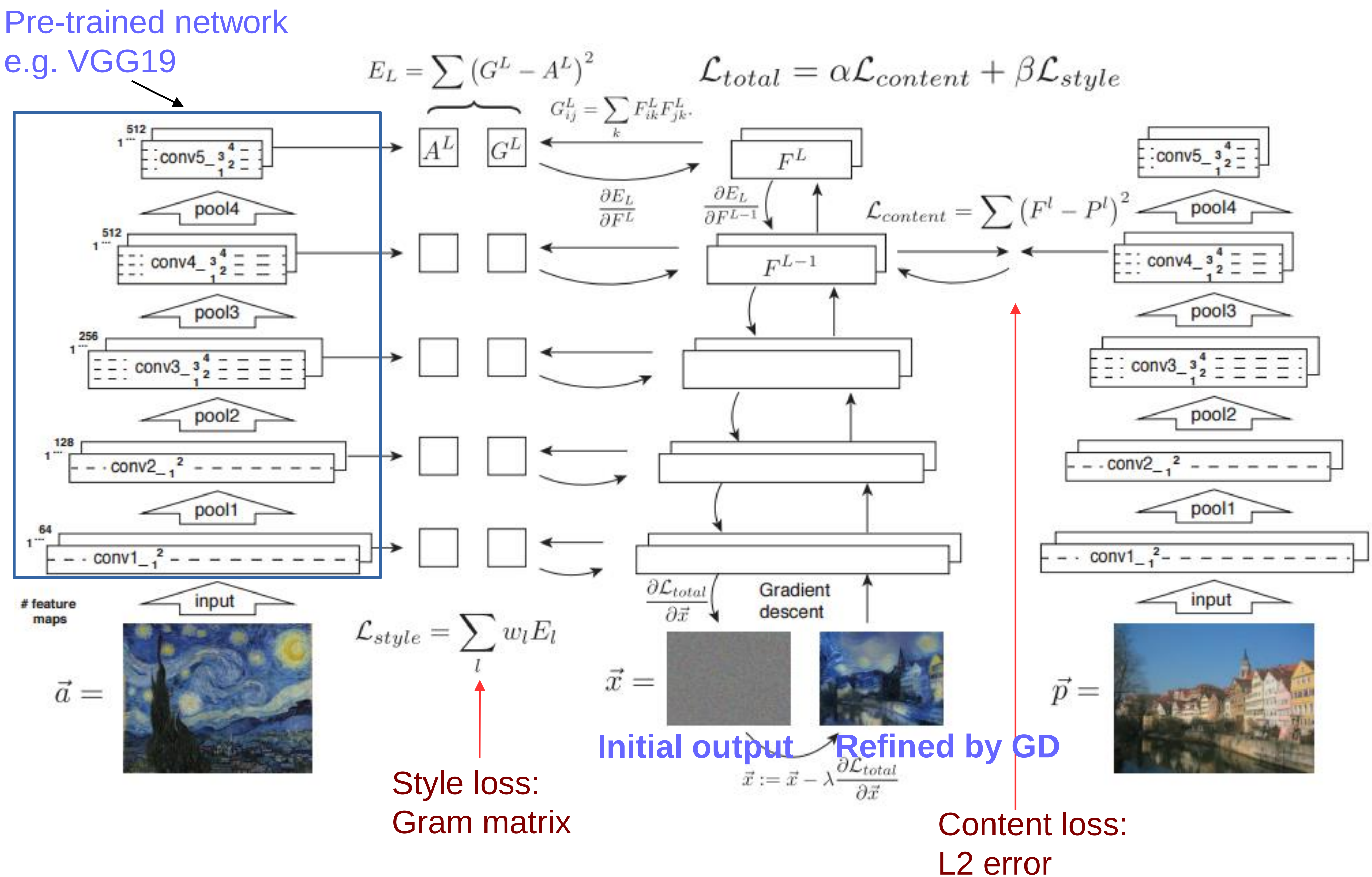
Style transfer

Take two images, and produce a new image that reflects the content of one but the artistic style of the other.



Style transfer

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.



Style transfer

Implementation: `./StyleTransfer/`

Define loss functions in `stylize.py`

- Content loss
- Style loss
- TV-reg
- Total loss

Content loss

Content loss measures how much the feature map of the generated i
mage differs from the feature map of the source image.

the feature map for the current image

$$L_c = w_c \times \sum_{i,j}(F_{ij}^{\ell} - P_{ij}^{\ell})^2$$

the feature map for the content source image

the weight of the content loss

Style loss

First, compute the Gram matrix G which represents the correlations be tween the responses of each filter, where F is as above.

C = the number of filters in layer l
Shape: [CxC]

$$G_{ij}^{\ell} = \sum_k F_{ik}^{\ell} F_{jk}^{\ell}$$

$$L_s^{\ell} = w_{\ell} \sum_{i,j} \left(G_{ij}^{\ell} - A_{ij}^{\ell} \right)^2$$

a scalar weight term

the Gram Matrix from the feature map of the source style image

Total-variation regularization

Helpful to also encourage smoothness in the image.
You can compute the "total variation" as the sum of the squares of differences in the pixel values for all pairs of pixels that are next to each other (horizontally or vertically). Here we sum the total-variation regularization for each of the 3 input channels (RGB).

For RGB channels

$$L_{tv} = w_t \times \sum_{c=1}^{3} \sum_{i=1}^{H-1} \sum_{j=1}^{W-1} \left((x_{i,j+1,c} - x_{i,j,c})^2 + (x_{i+1,j,c} - x_{i,j,c})^2 \right)$$

Horizontal / Vertical difference

the total variation weight

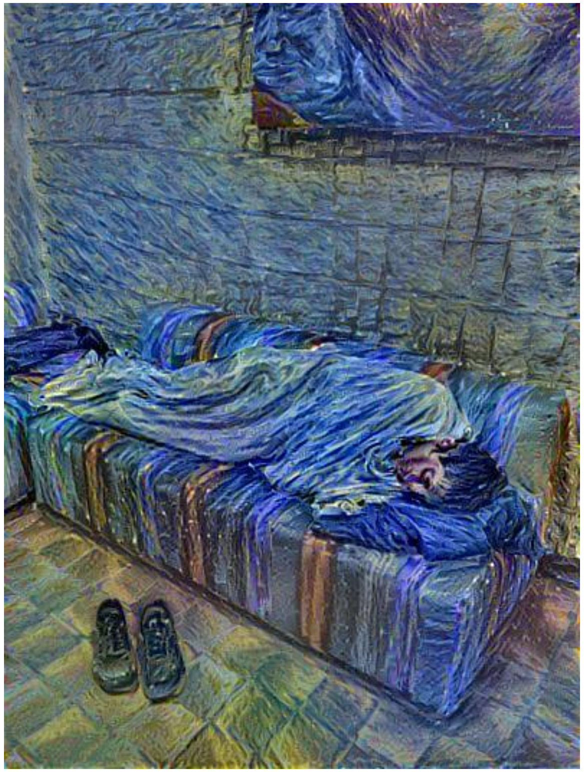
Style transfer

Result

python neural_style.py --content <content file> --styles <style file> --output <output file>

Content

Style

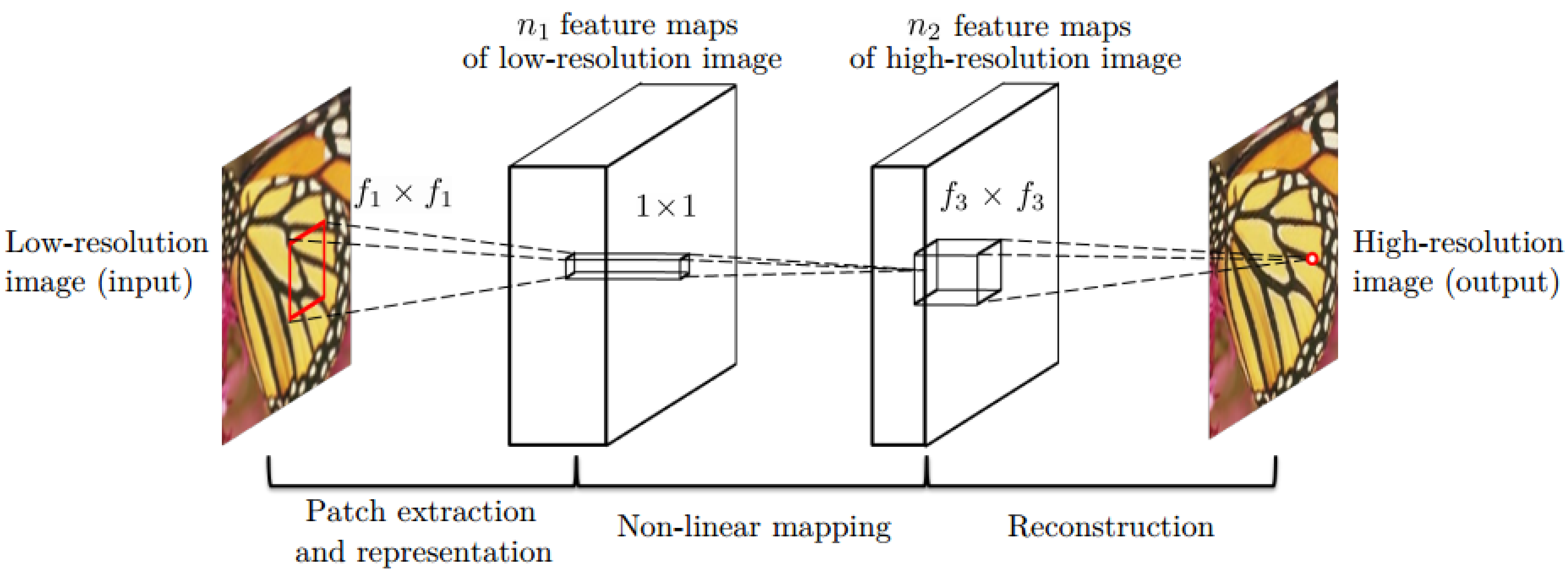


Result

https://github.com/anishathalye/neural-style

SRCNN http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html

Early work using CNN to image super-resolution



[32x32x1]	9x9 Conv, 64 filters	1x1 Conv, 32 filters	5x5 Conv, 1 filters	[20x20x1]

SRCNN

Implementation: `./SR/srcnn.py`

32x32, 24,800 training samples
    From 91 images
Using Y channel only (rather than RGB)
    Human eye is sensitive to brightness than color
No padding
ReLU

Initialized by random Gaussian mean=0, stddev=0.001, bias=0
Learning rate
    For the first two layers 0.0001
    For the last layer 0.00001

MSE loss function

J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image superresolution via sparse representation. TIP, 2010.

Peak signal-to-noise ratio

PSNR

신호가 가질 수 있는 최대 전력에 대한 잡음의 전력

화질 손실 정보를 평가할때 사용

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

MSE

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

```
tf.reduce_mean(tf.square(x - y))
```

MAE

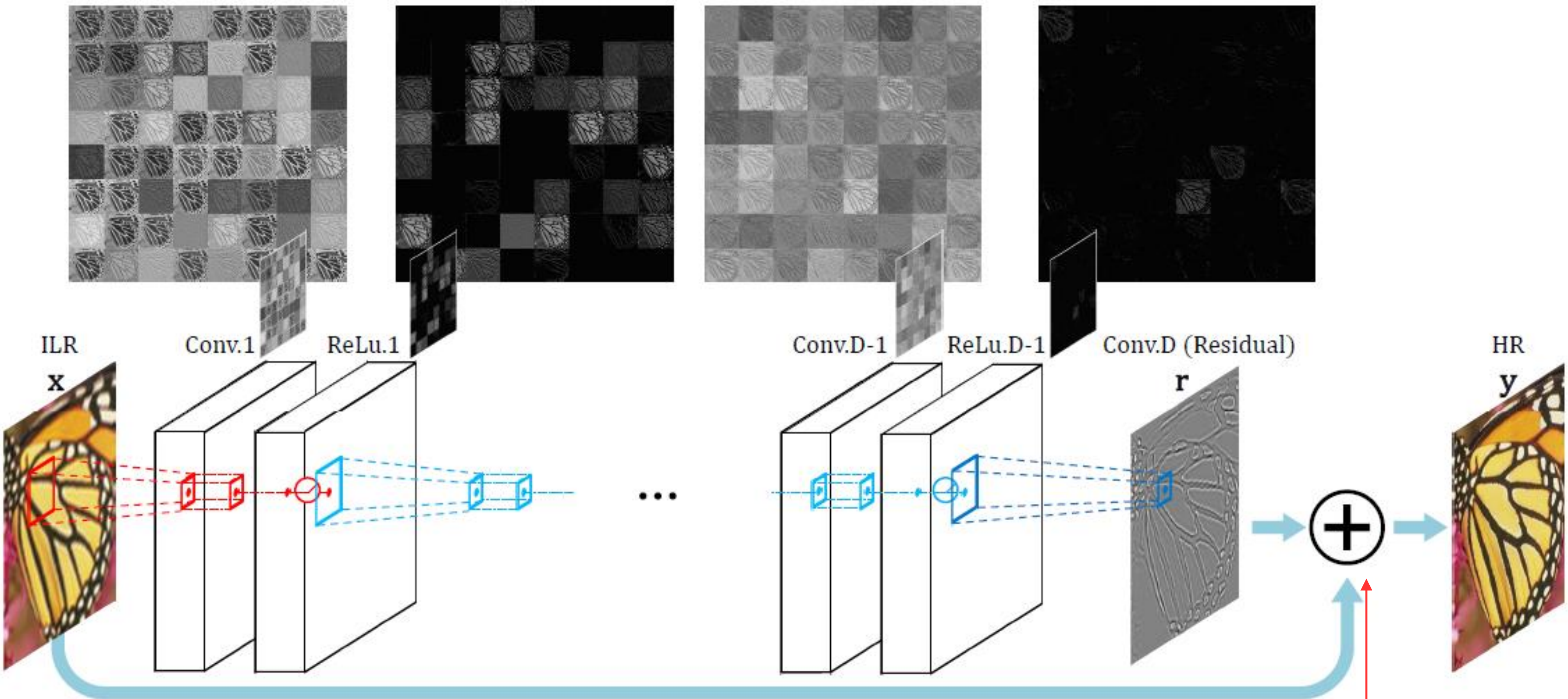
More robust to outliers

```
tf.reduce_mean(tf.abs(x - y))
```

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n} = \frac{\sum_{i=1}^{n}|e_i|}{n}.$$

VDSR http://cv.snu.ac.kr/research/VDSR/

Very deep CNN for SR



[41x41x1]

3x3 Conv,
64 filters

......

[41x41x1]

Residual learning

VDSR

Implementation: `./SR/vdsr.py`

41x41, 7,968 training samples
  From 291 images 91 images + BSDS200
Using Y channel only (rather than RGB)
  Human eye is sensitive to brightness than color
ReLU
Momentum 0.9
Weight decay 0.0001
High learning rate 0.1
  Decreased by 10 every 20 epochs

He initialization

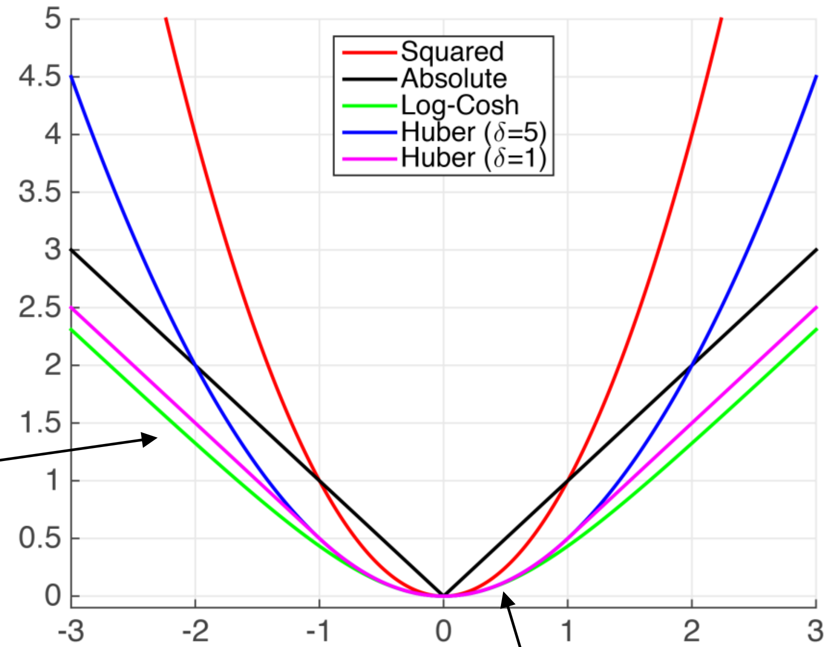
MSE loss function

Gradient clipping
Multi-scale

Huber loss

Quadratic for small values of a, and linear for large values
MSE와 MAE의 장점을 모두 가짐

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

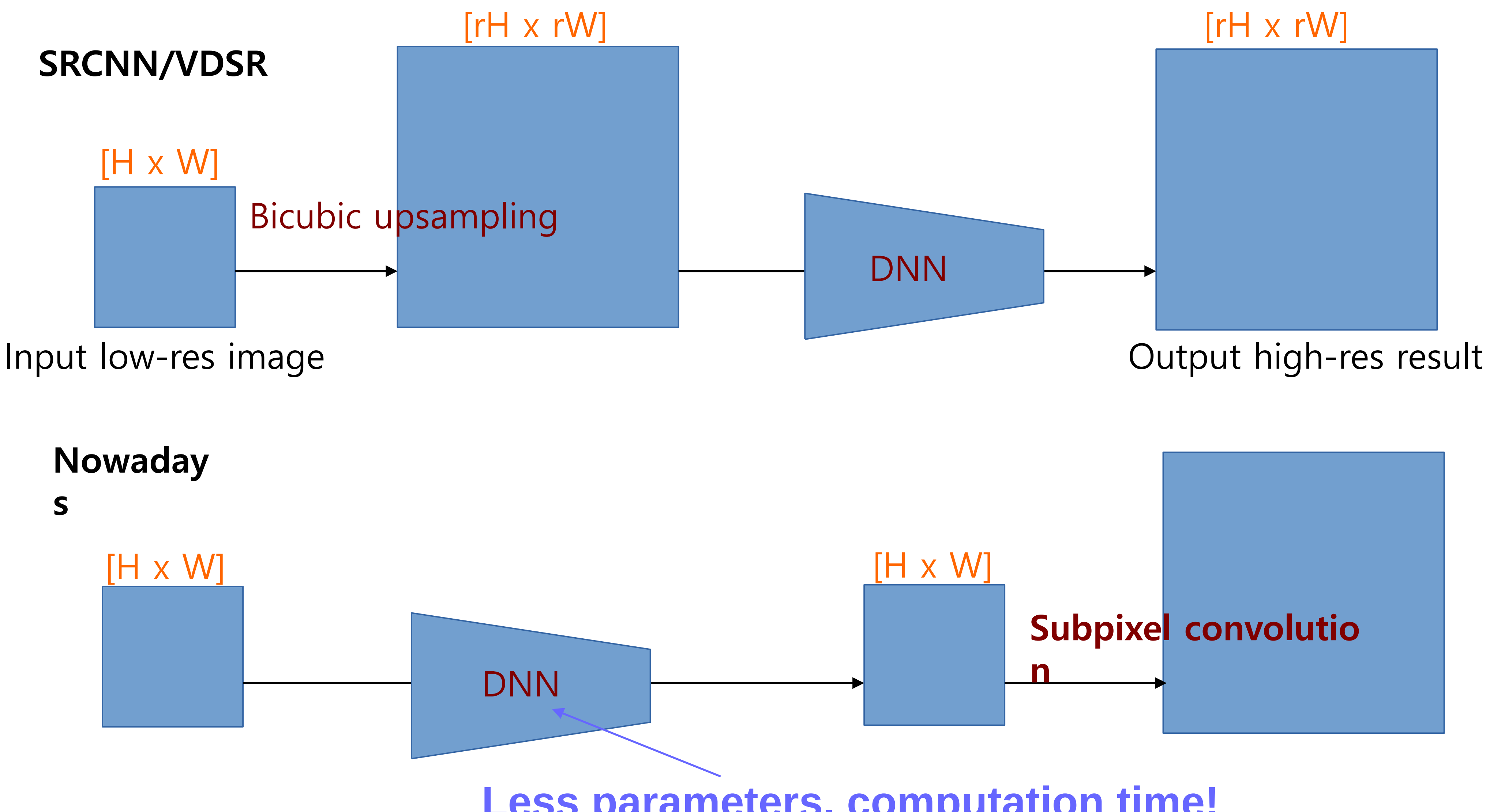


Less sensitive to outliers
(L1 like)

Stable convergence (L2 like)

Sub-pixel convolution

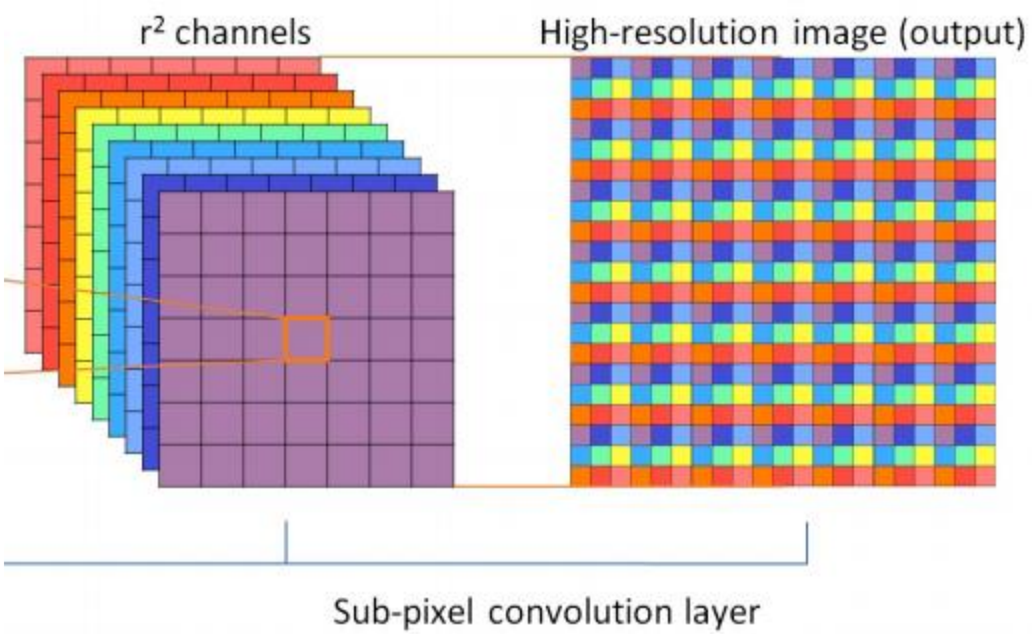
A sub-pixel convolution layer that aggregates the feature maps from L R space and builds the SR image in a single step.



VDSR with sub-pixel conv

Implementation: `./SR/vdsr_sp.py`

`Conv → tf.depth_to_space`



Shi, Wenzhe, et al. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.