Convolutional Neural Network

May 7^{th}, 2019

Applied Deep Learning

Yun-Nung (Vivian) Chen

HTTP://ADL.MIULAB.TW

國立臺灣大學
National Taiwan University

Slides credited from Mark Chang & Hung-Yi Lee

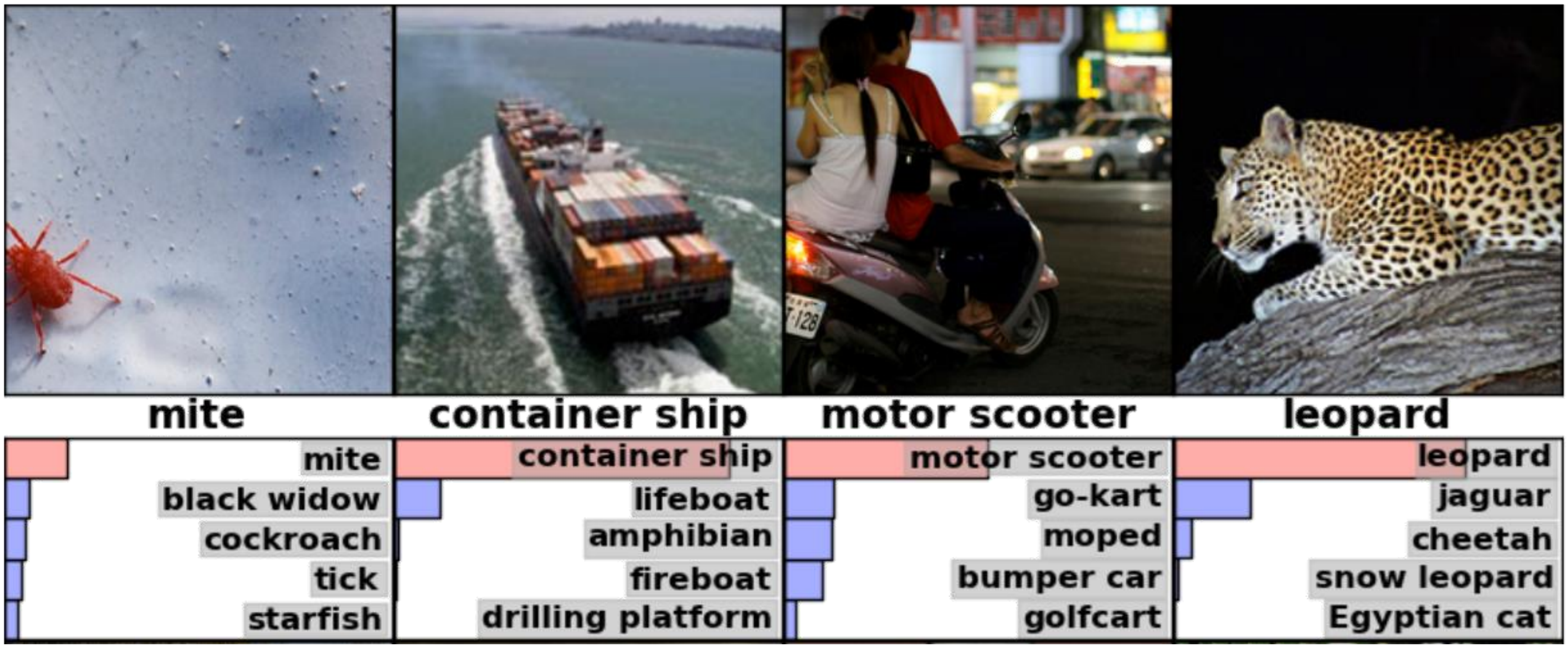
Outline

- CNN (Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- More Applications

Outline

- CNN (Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- More Applications

Image Recognition



http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

Why CNN for Image

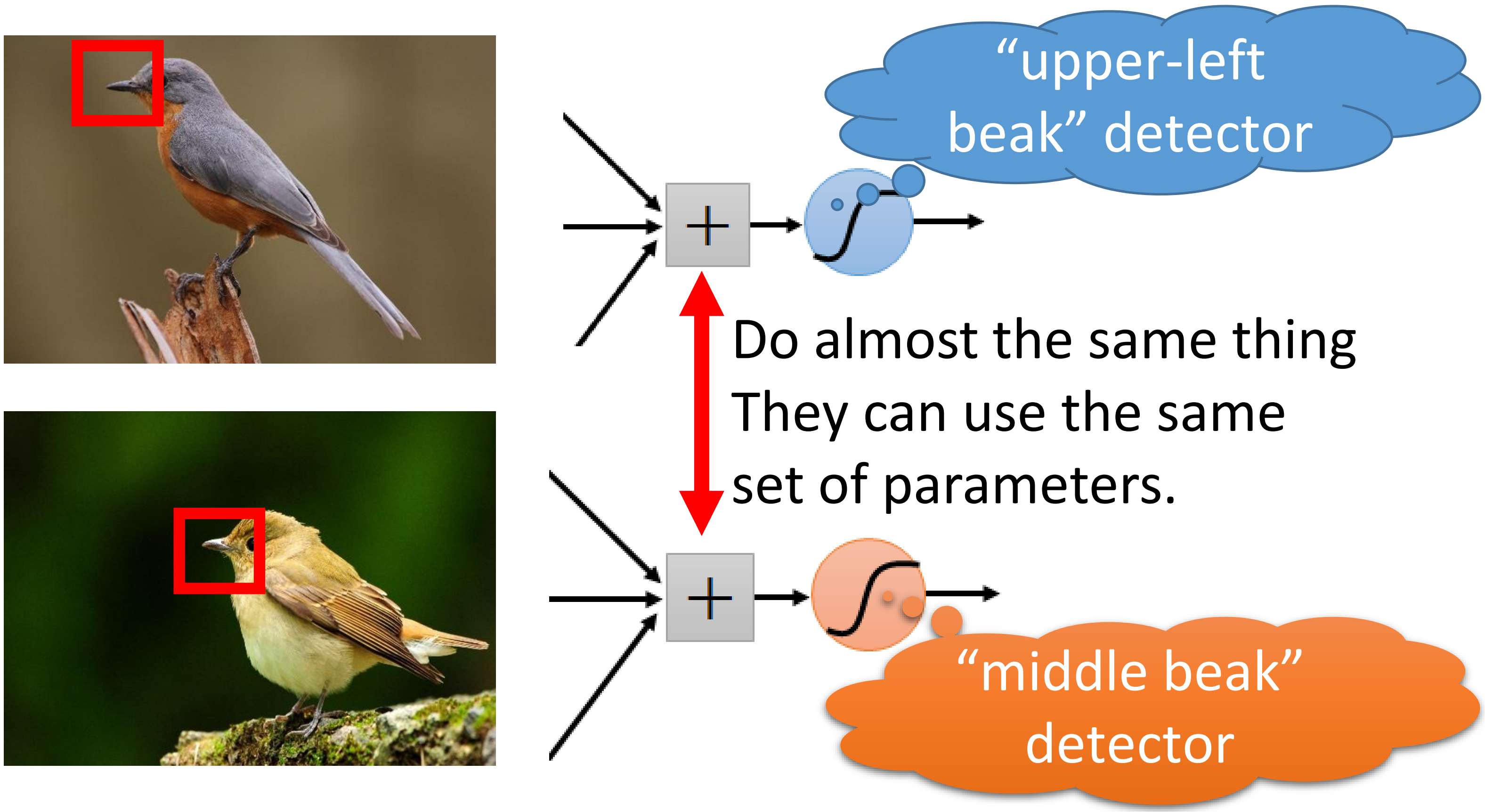
- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

Why CNN for Image

- The same patterns appear in different regions.



Why CNN for Image

- Subsampling the pixels will not change the object

bird

subsampling

bird

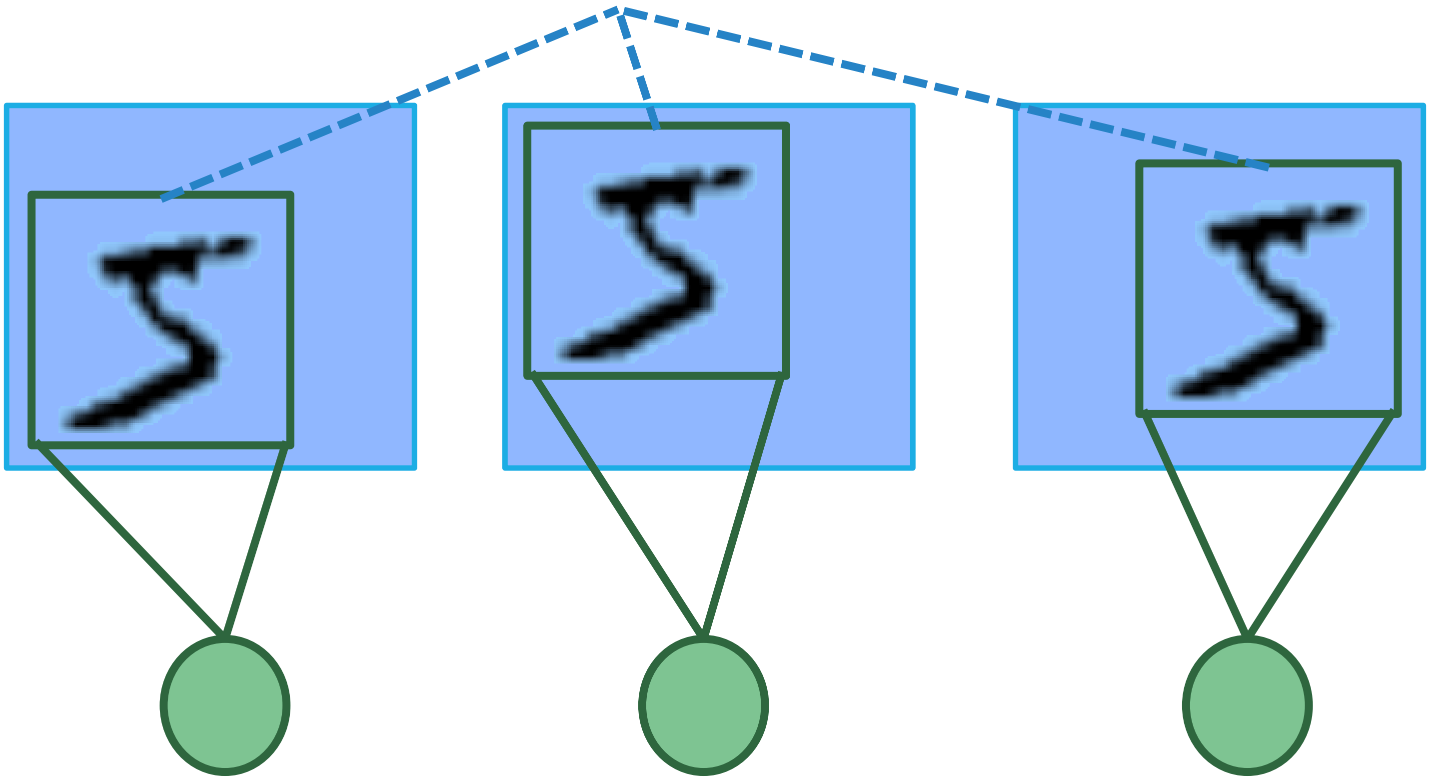
We can subsample the pixels to make image smaller

Less parameters for the network to process the image

Image Recognition

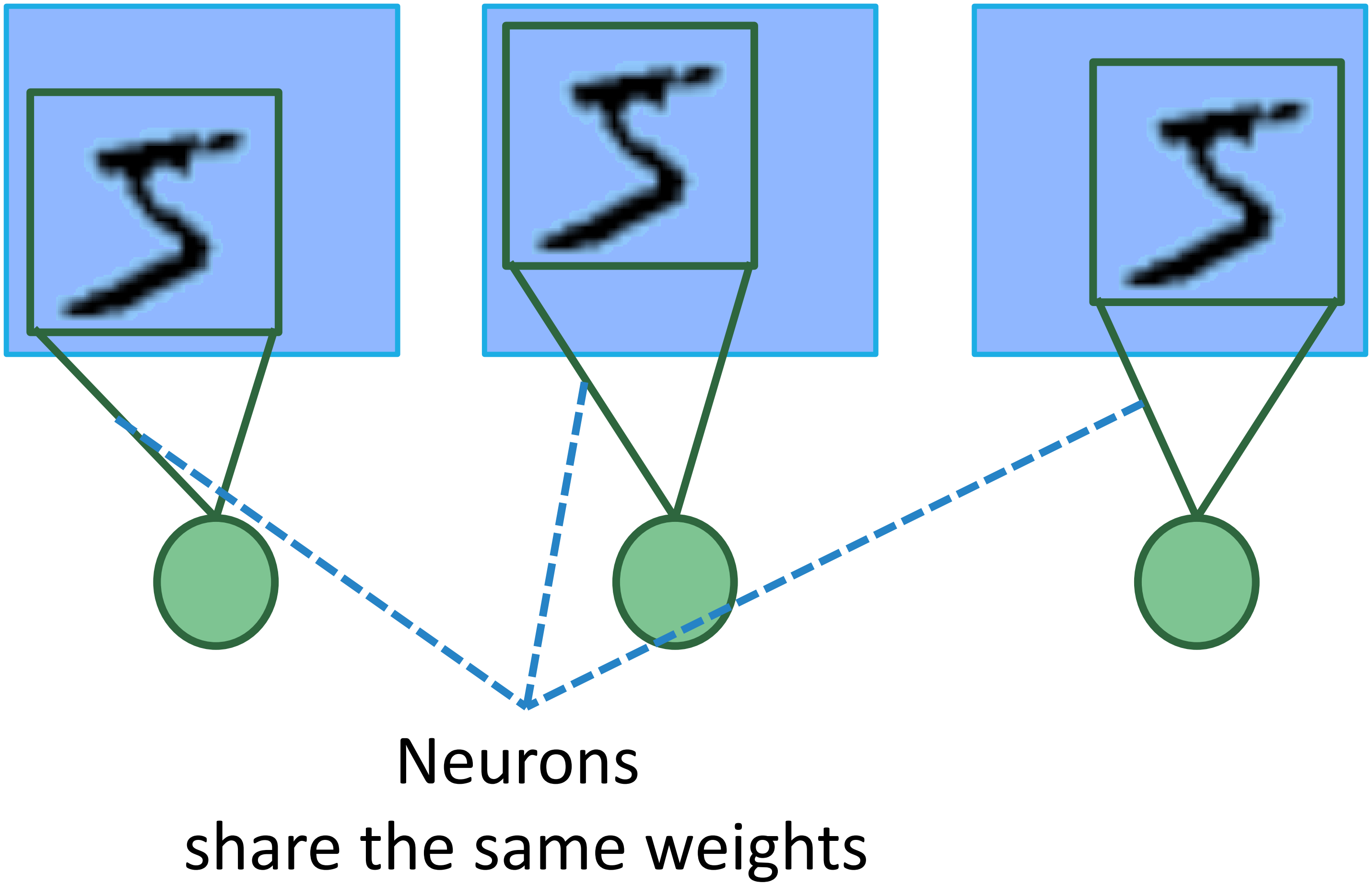
Local Connectivity

Neurons connect to a small region



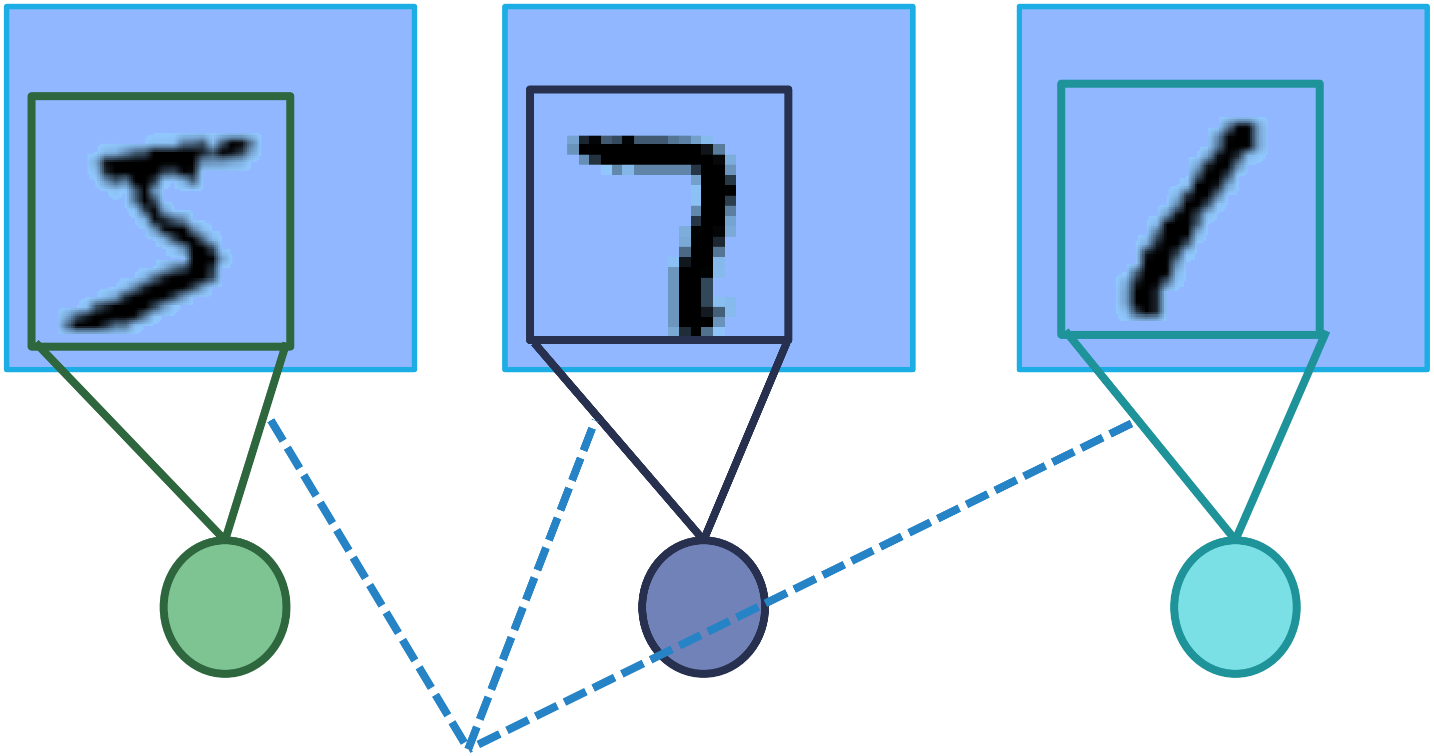
Parameter Sharing

- The same feature in different positions



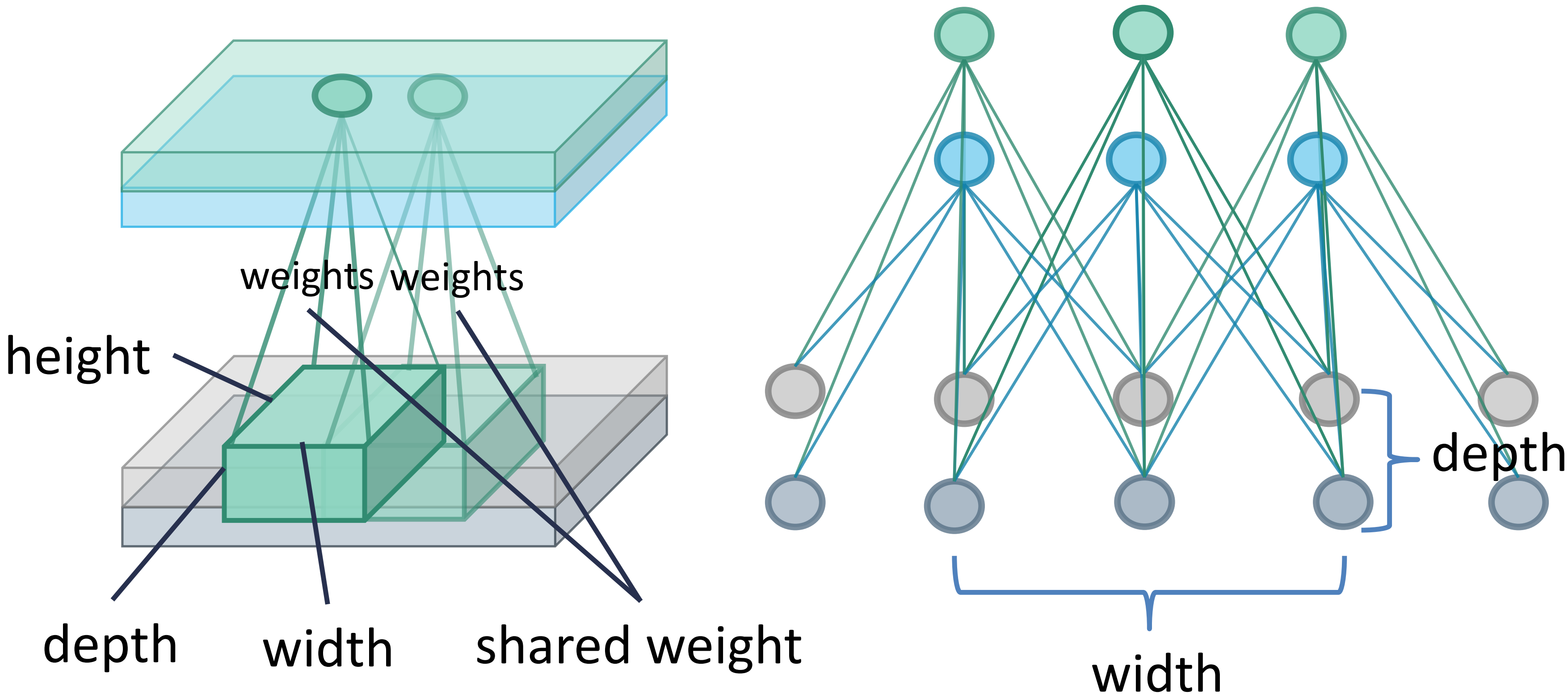
Parameter Sharing

- Different features in the same position

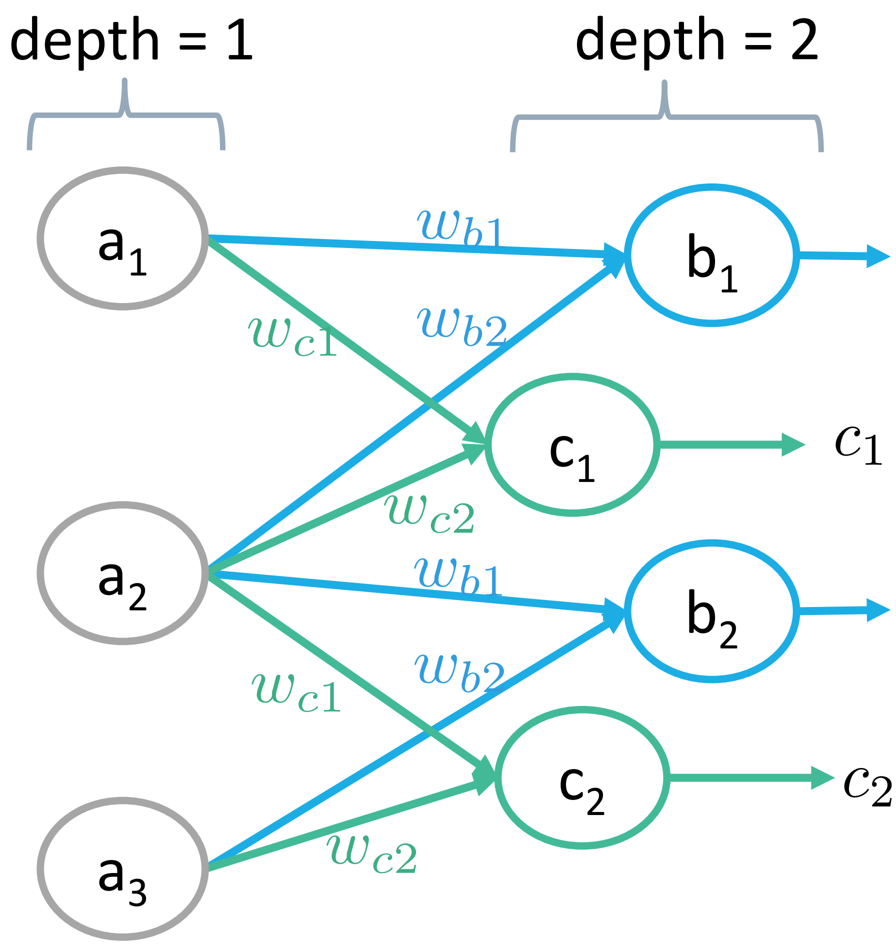


Neurons
have different weights

Convolutional Layers



Convolutional Layers



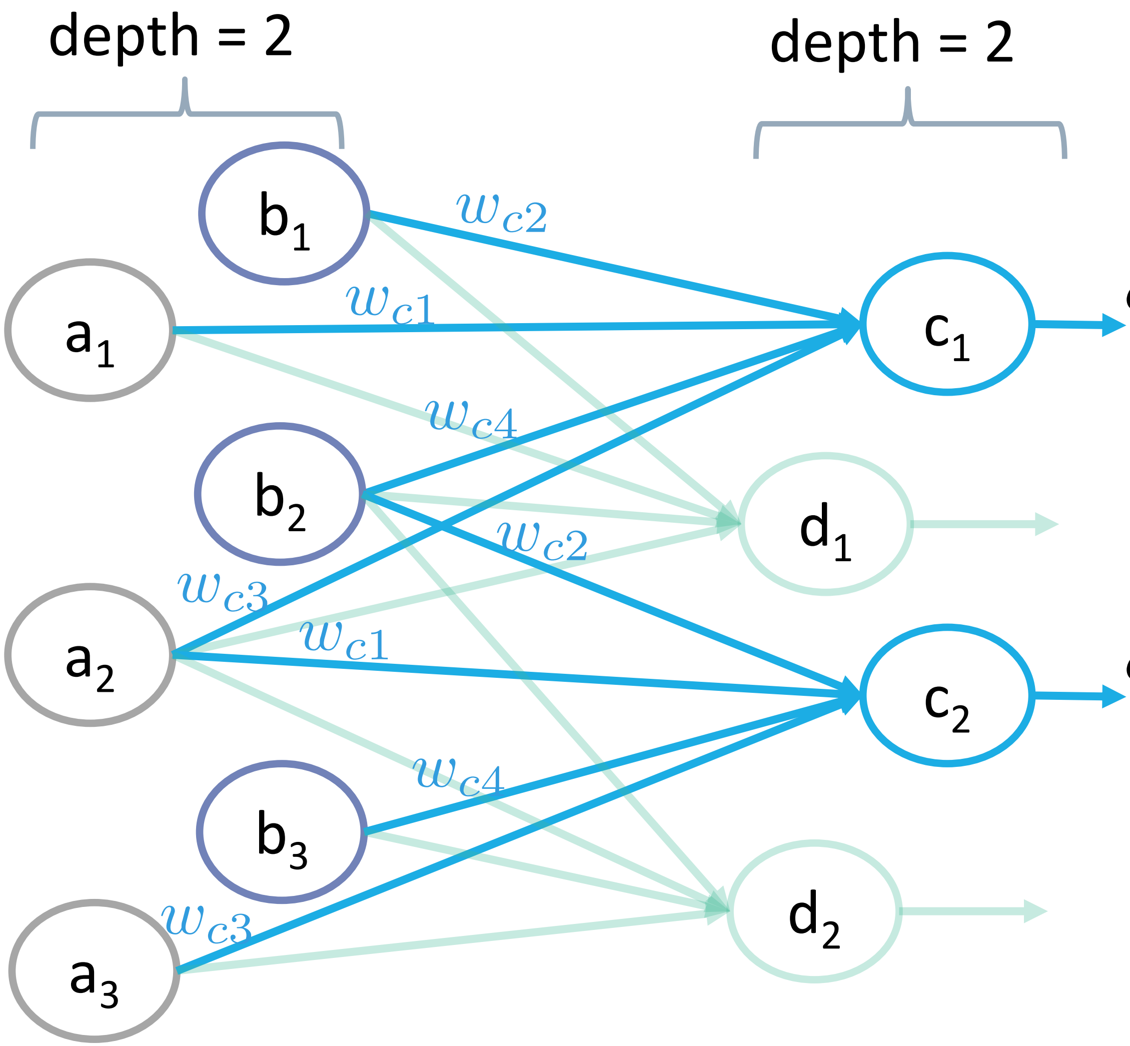
$$b_1 = w_{b1}a_1 + w_{b2}a_2$$

$$c_1 = w_{c1}a_1 + w_{c2}a_2$$

$$b_2 = w_{b1}a_2 + w_{b2}a_3$$

$$c_2 = w_{c1}a_2 + w_{c2}a_3$$

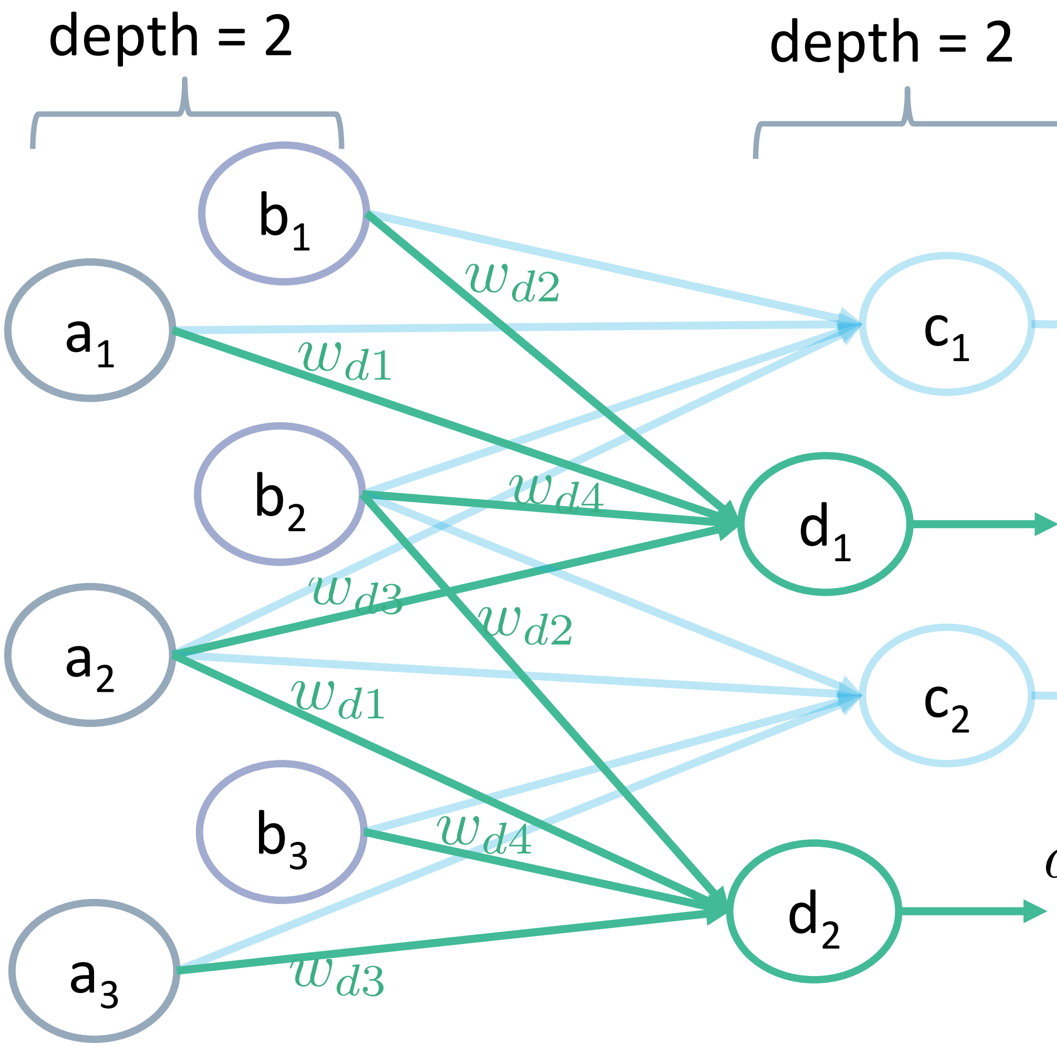
Convolutional Layers



$$c_1 = a_1 w_{c1} + b_1 w_{c2} + a_2 w_{c3} + b_2 w_{c4}$$

$$c_2 = a_2 w_{c1} + b_2 w_{c2} + a_3 w_{c3} + b_3 w_{c4}$$

Convolutional Layers



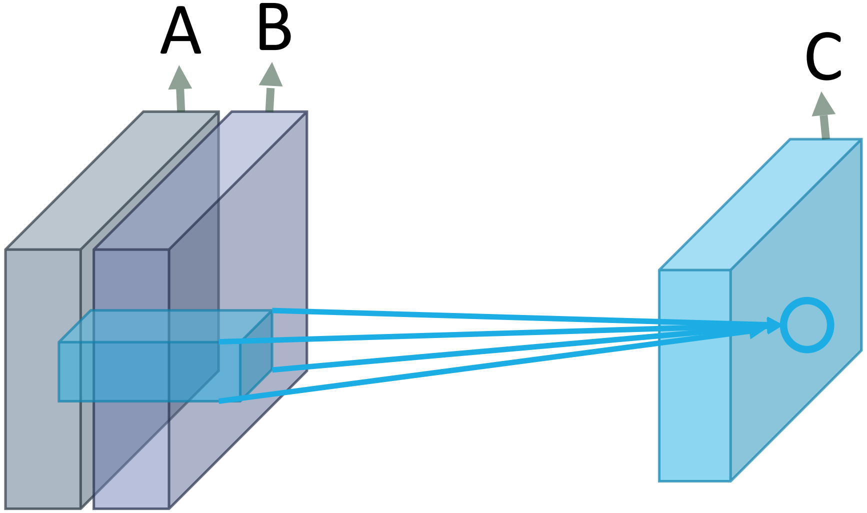
$$c_1 = a_1 w_{c1} + b_1 w_{c2} + a_2 w_{c3} + b_2 w_{c4}$$

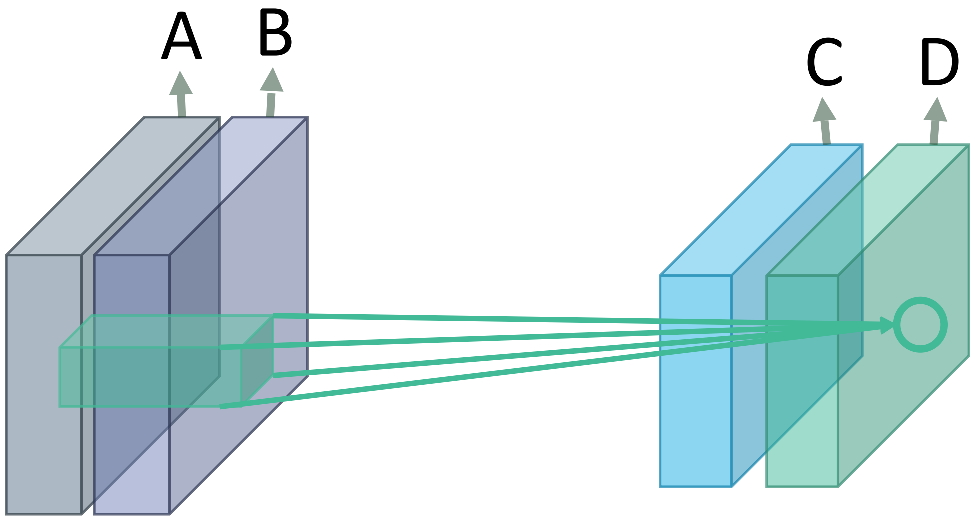
$$d_1 = a_1 w_{d1} + b_1 w_{d2} + a_2 w_{d3} + b_2 w_{d4}$$

$$c_2 = a_2 w_{c1} + b_2 w_{c2} + a_3 w_{c3} + b_3 w_{c4}$$

$$d_2 = a_2 w_{d1} + b_2 w_{d2} + a_3 w_{d3} + b_3 w_{d4}$$

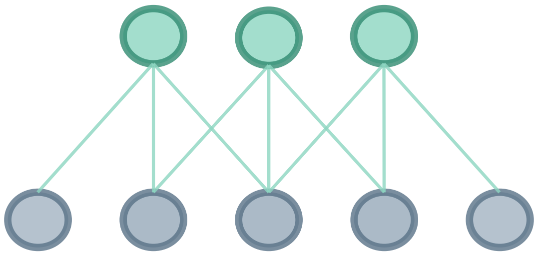
Convolutional Layers



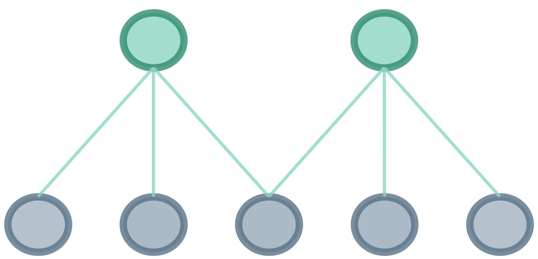


Hyper-parameters of CNN

- Stride

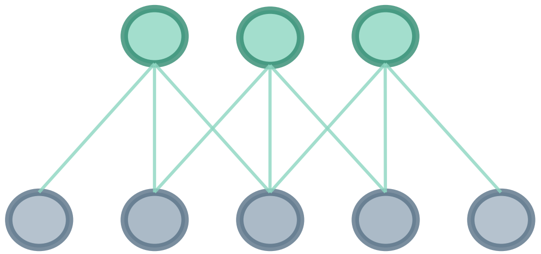


Stride = 1

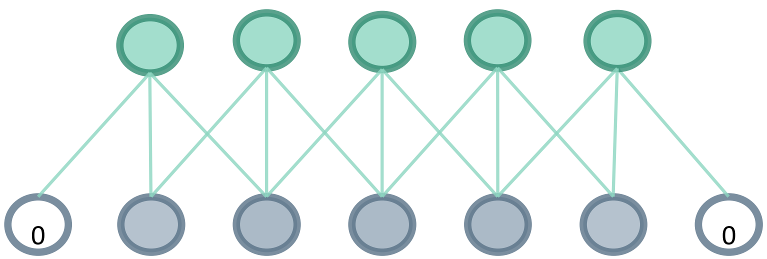


Stride = 2

- Padding

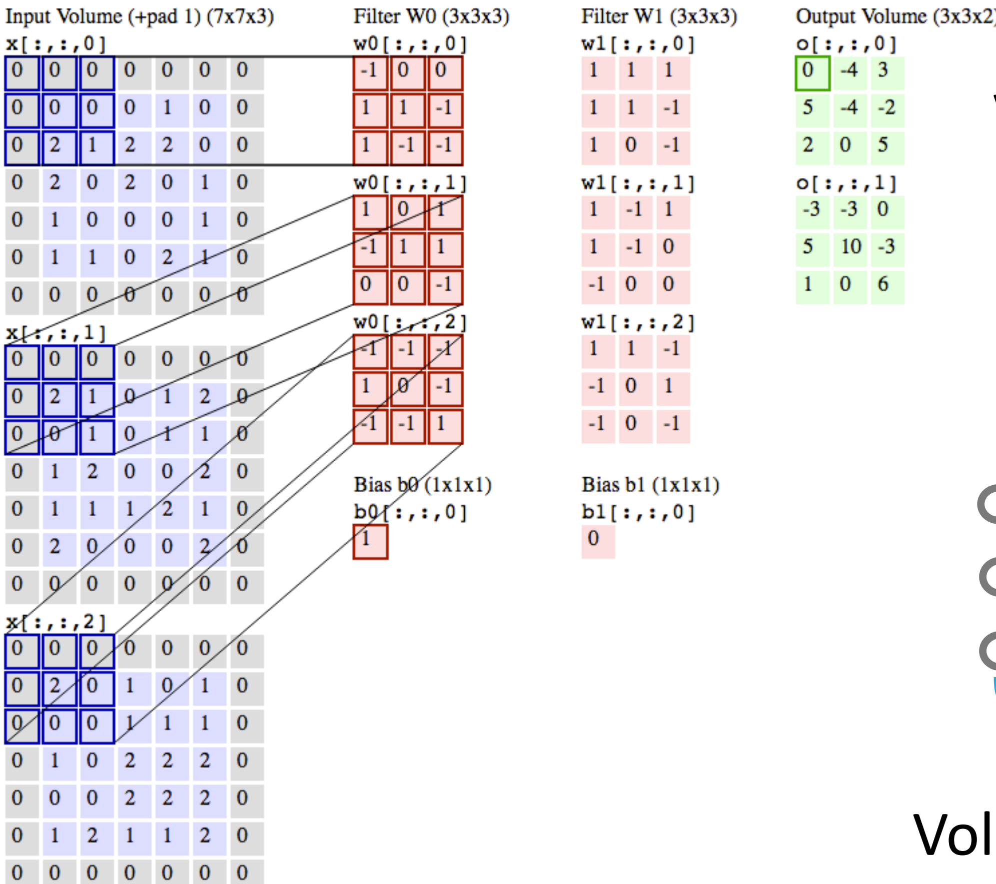


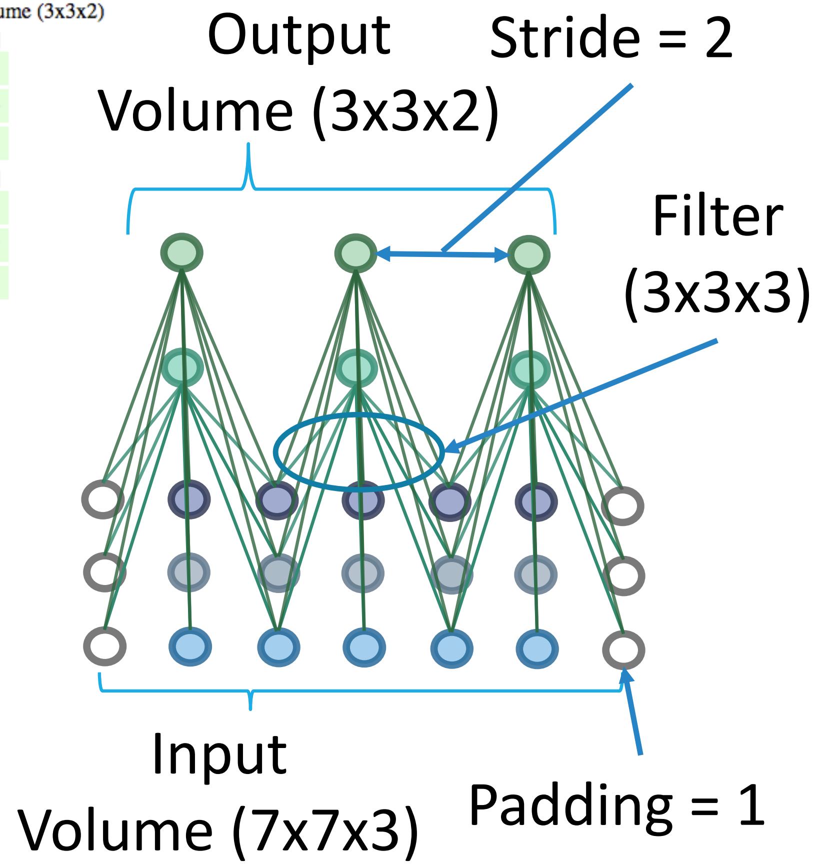
Padding = 0



Padding = 1

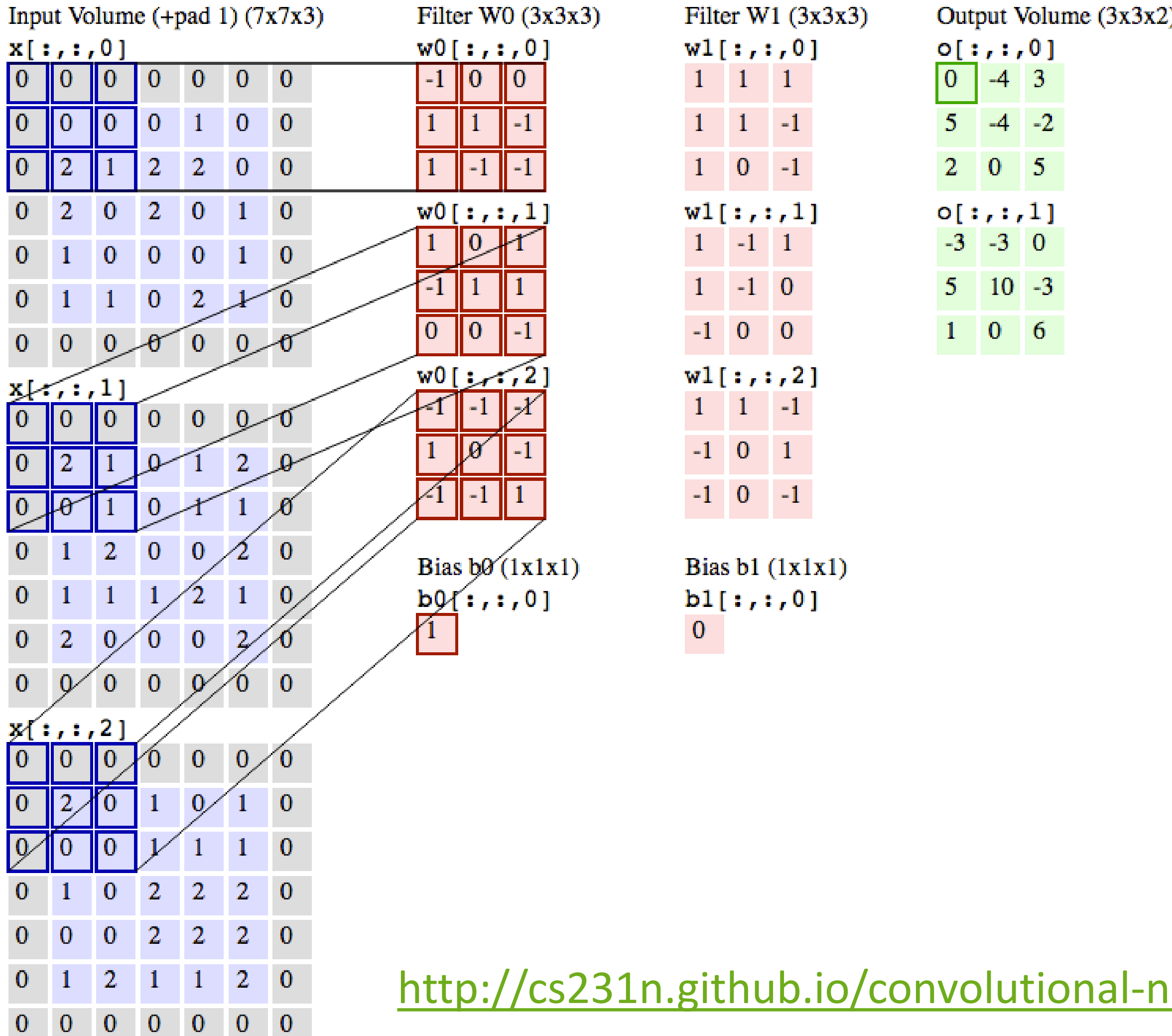
Example





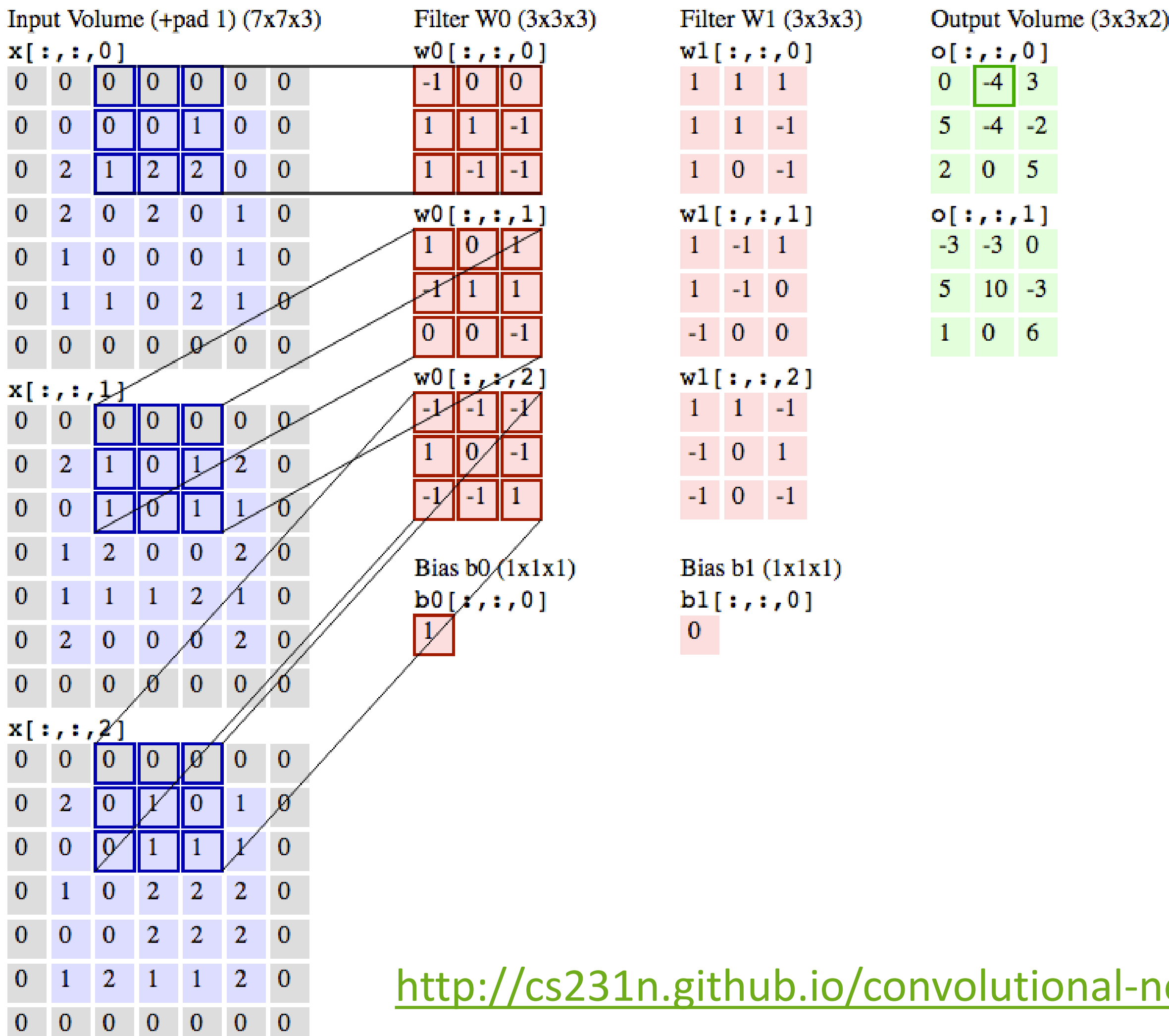
http://cs231n.github.io/convolutional-networks/

Convolutional Layers



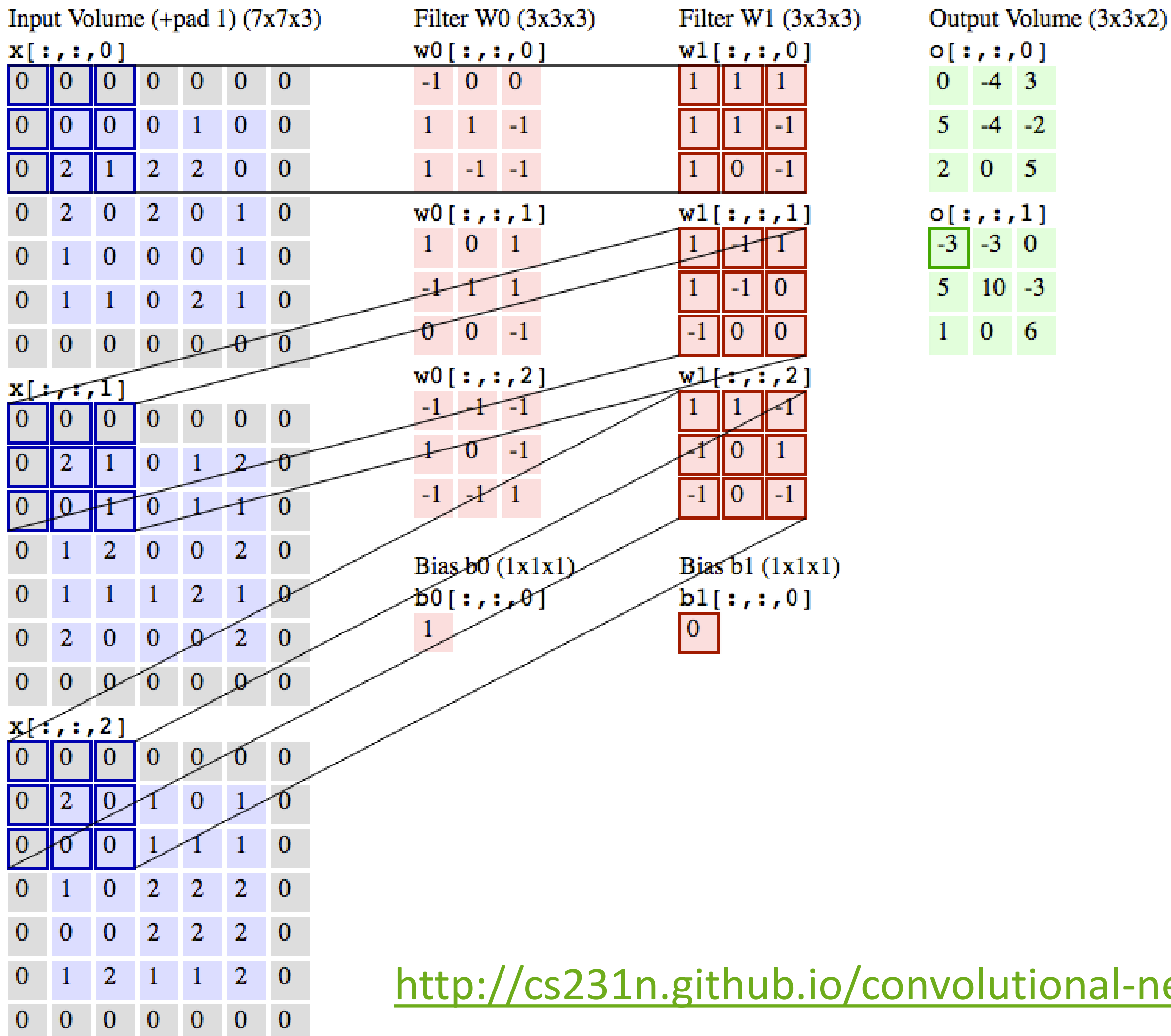
http://cs231n.github.io/convolutional-networks/

Convolutional Layers



http://cs231n.github.io/convolutional-networks/

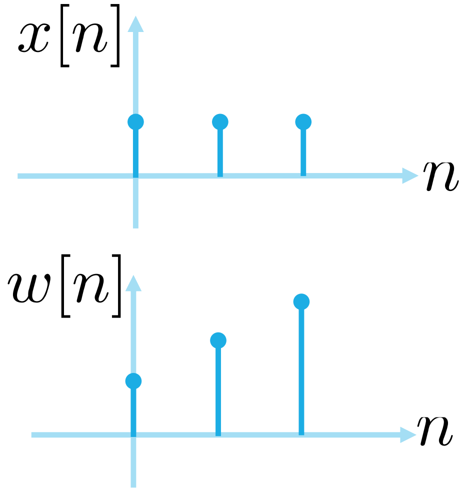
Convolutional Layers

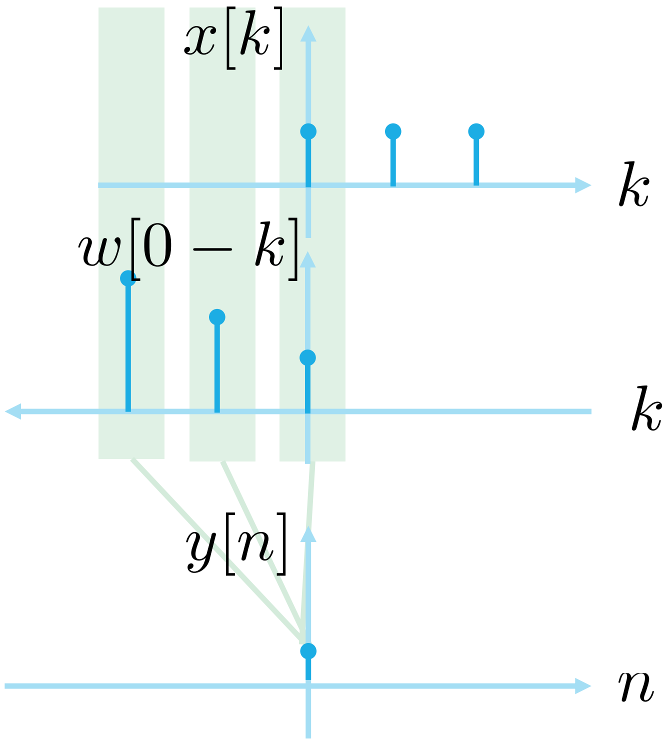


http://cs231n.github.io/convolutional-networks/

Relationship with Convolution

$$y[n] = \sum_k x[k]w[n-k]$$

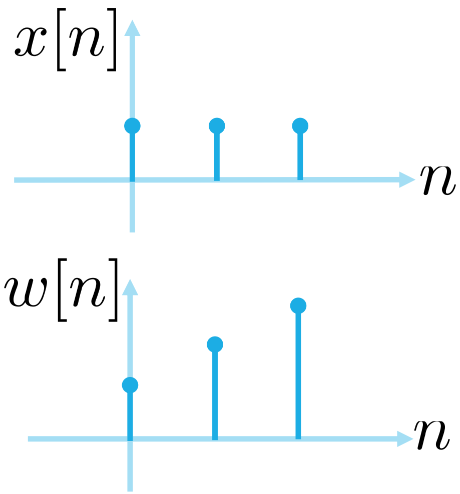


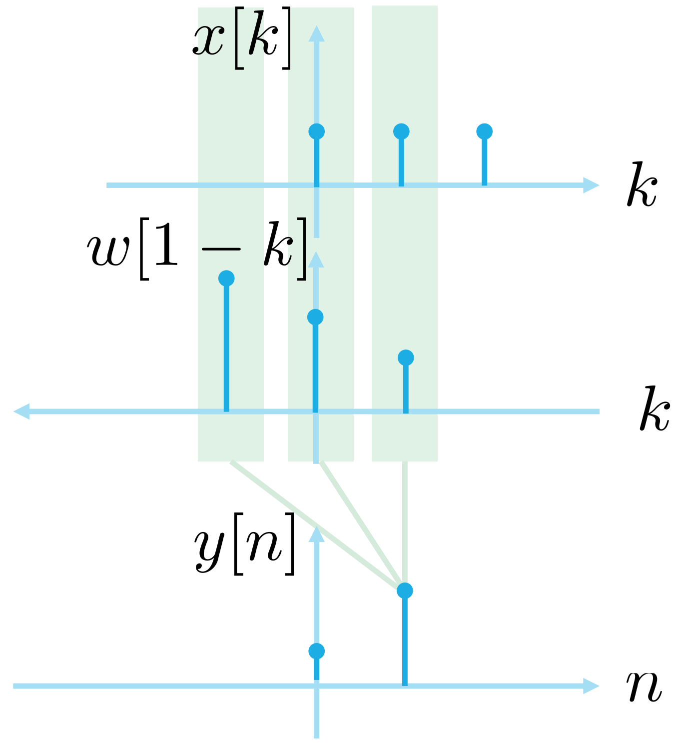


$$y[0] = x[-2]w[2] + x[-1]w[1] + x[0]w[0]$$

Relationship with Convolution

$$y[n] = \sum_k x[k]w[n-k]$$

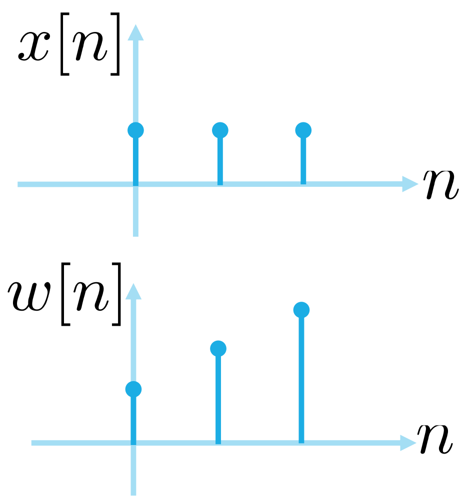


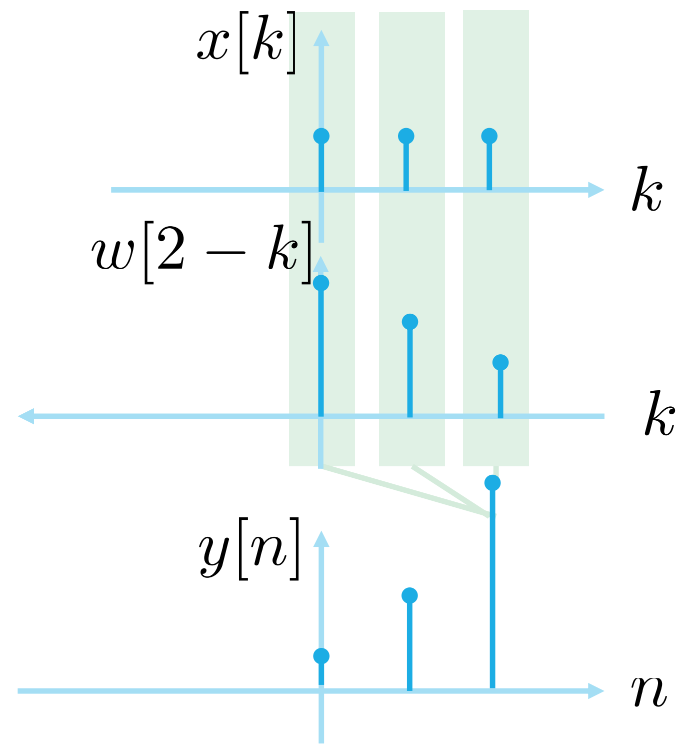


$$y[1] = x[-1]w[2] + x[0]w[1] + x[2]w[0]$$

Relationship with Convolution

$$y[n] = \sum_k x[k]w[n-k]$$

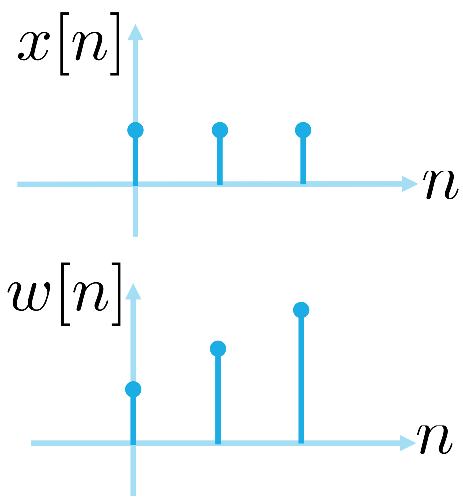


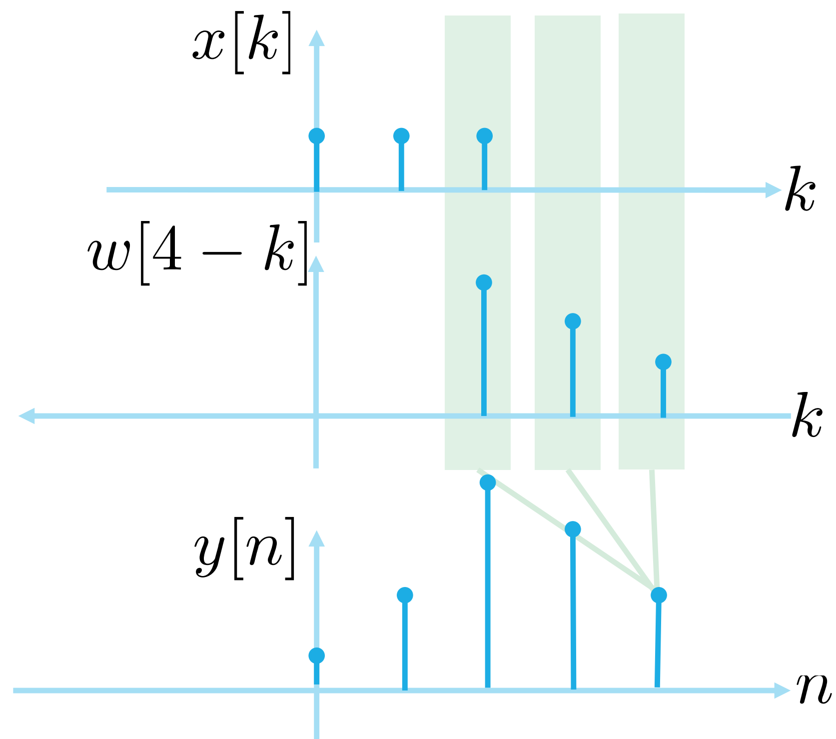


$$y[2] = x[0]w[2] + x[1]w[1] + x[2]w[0]$$

Relationship with Convolution

$$y[n] = \sum_k x[k]w[n-k]$$

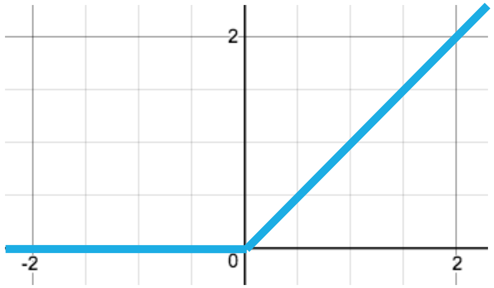


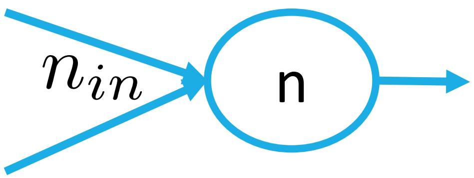


$$y[4] = x[2]w[2] + x[3]w[1] + x[4]w[0]$$

Nonlinearity

- Rectified Linear (ReLU)



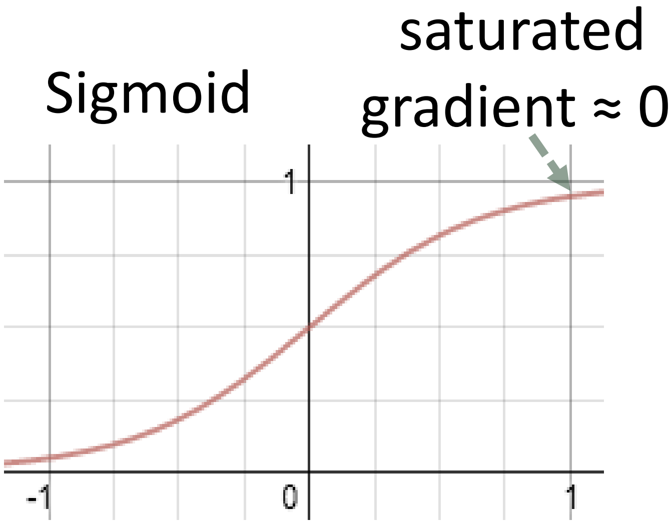


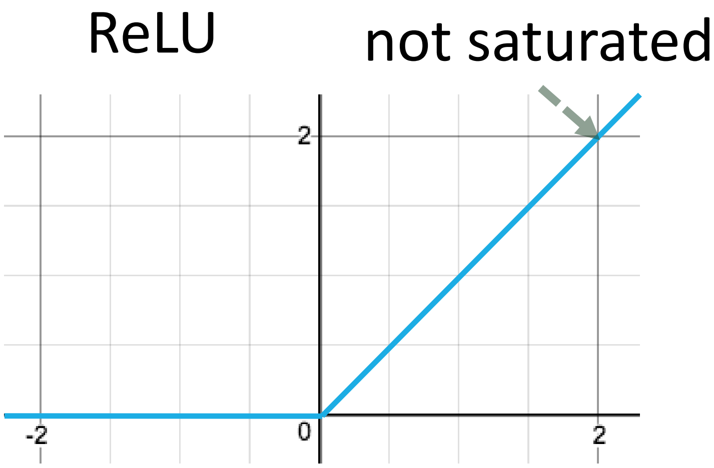
$$n_{out} = \begin{cases} n_{in} \text{ if } n_{in} > 0 \\ 0 \text{ otherwise} \end{cases}$$

$$\begin{bmatrix} 1 \\ 4 \\ -3 \\ 1 \end{bmatrix} \Rightarrow \boxed{\text{ReLU}} \Rightarrow \begin{bmatrix} 1 \\ 4 \\ 0 \\ 1 \end{bmatrix}$$

Why ReLU?

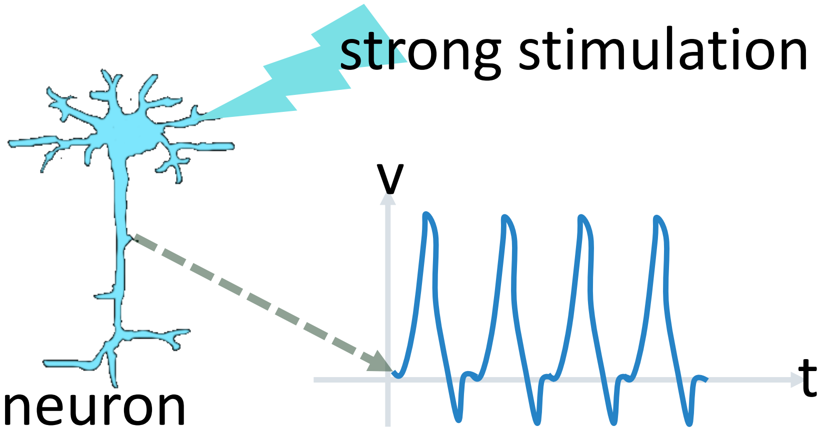
- Easy to train
- Avoid gradient vanishing problem

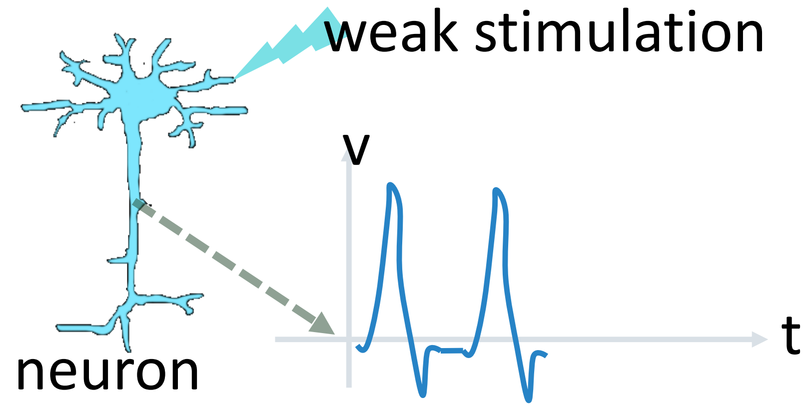


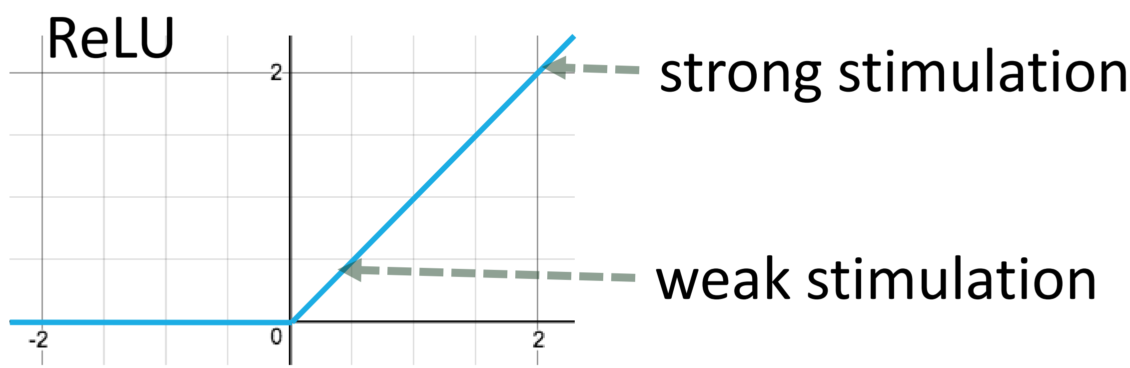


Why ReLU?

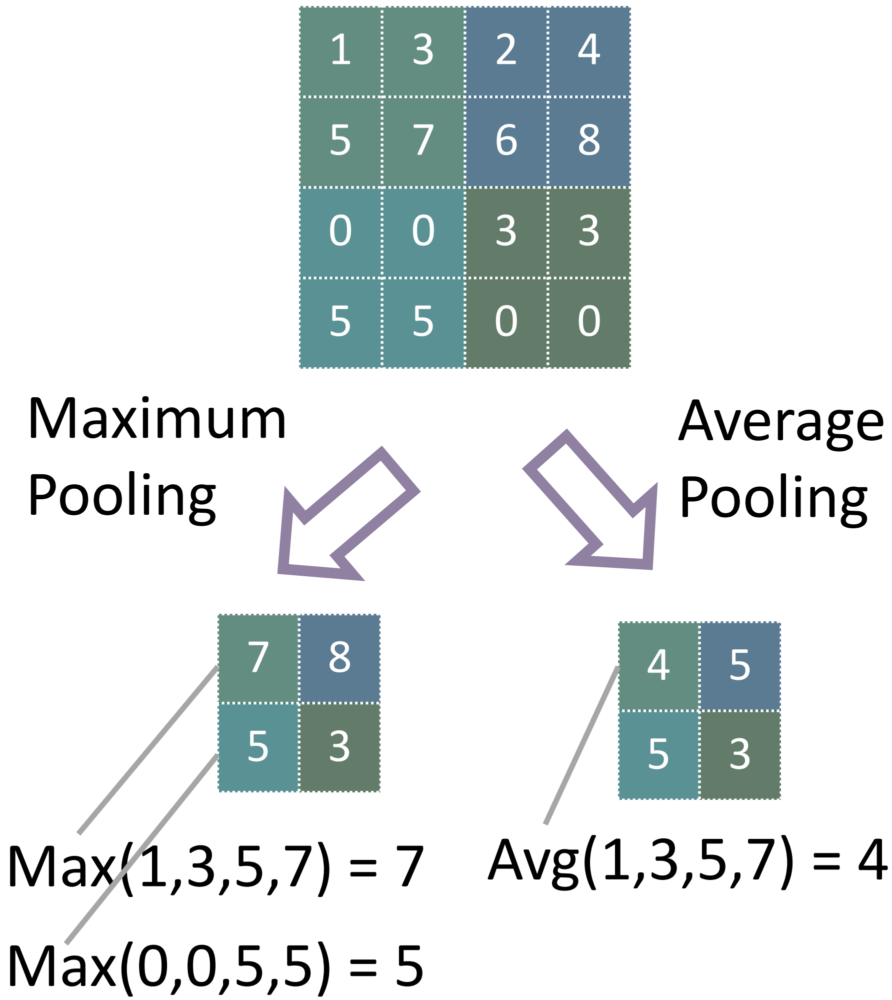
- Biological reason

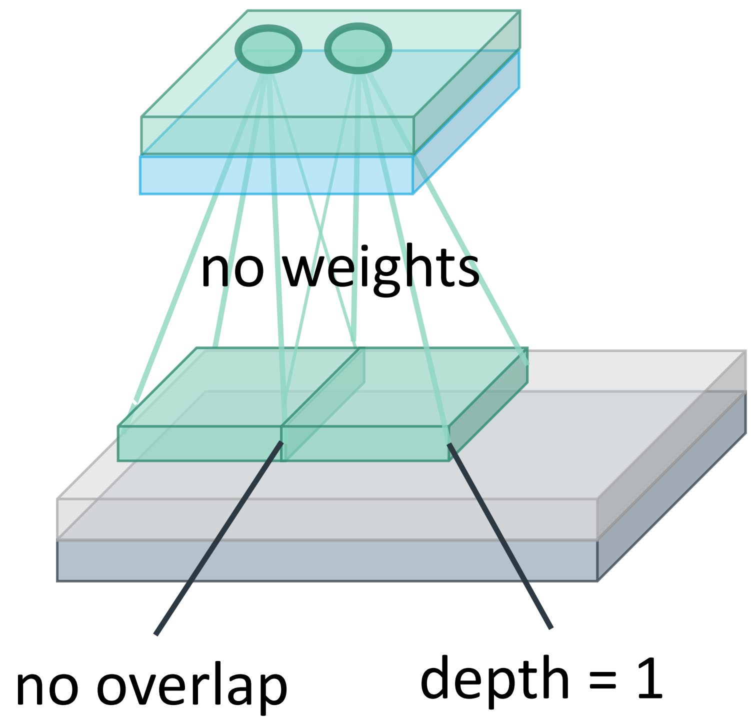




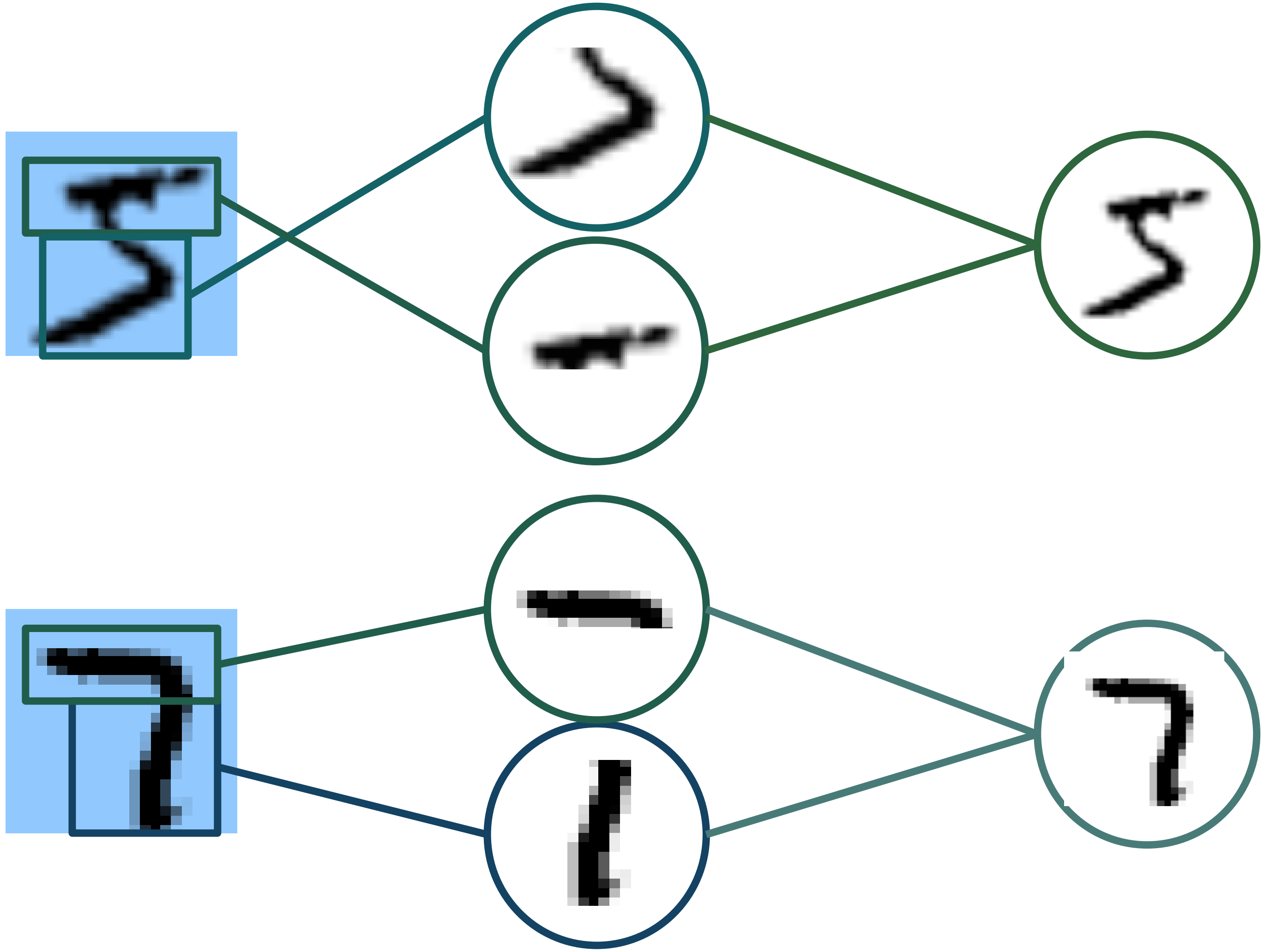


Pooling Layer

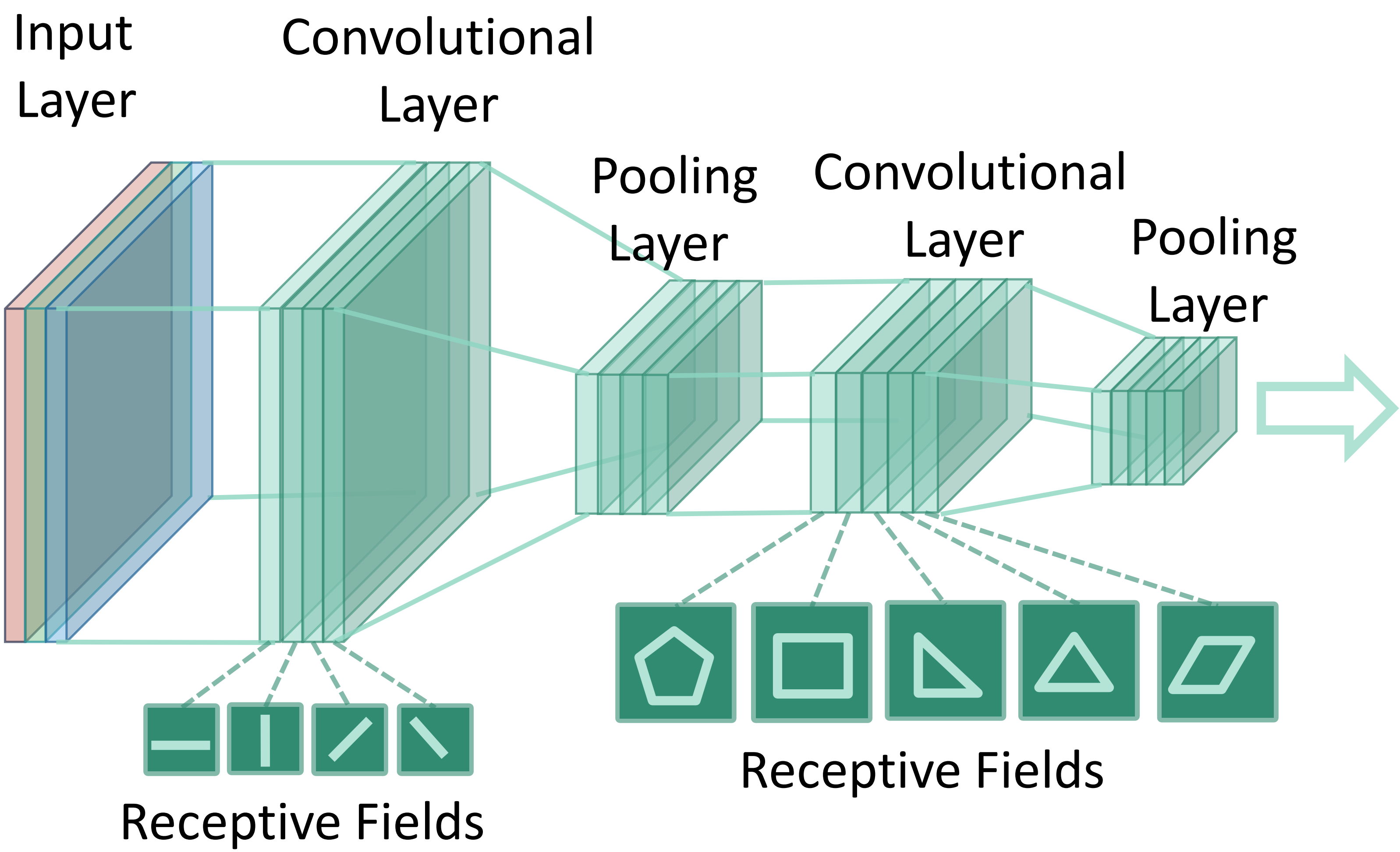




Why “Deep” Learning?

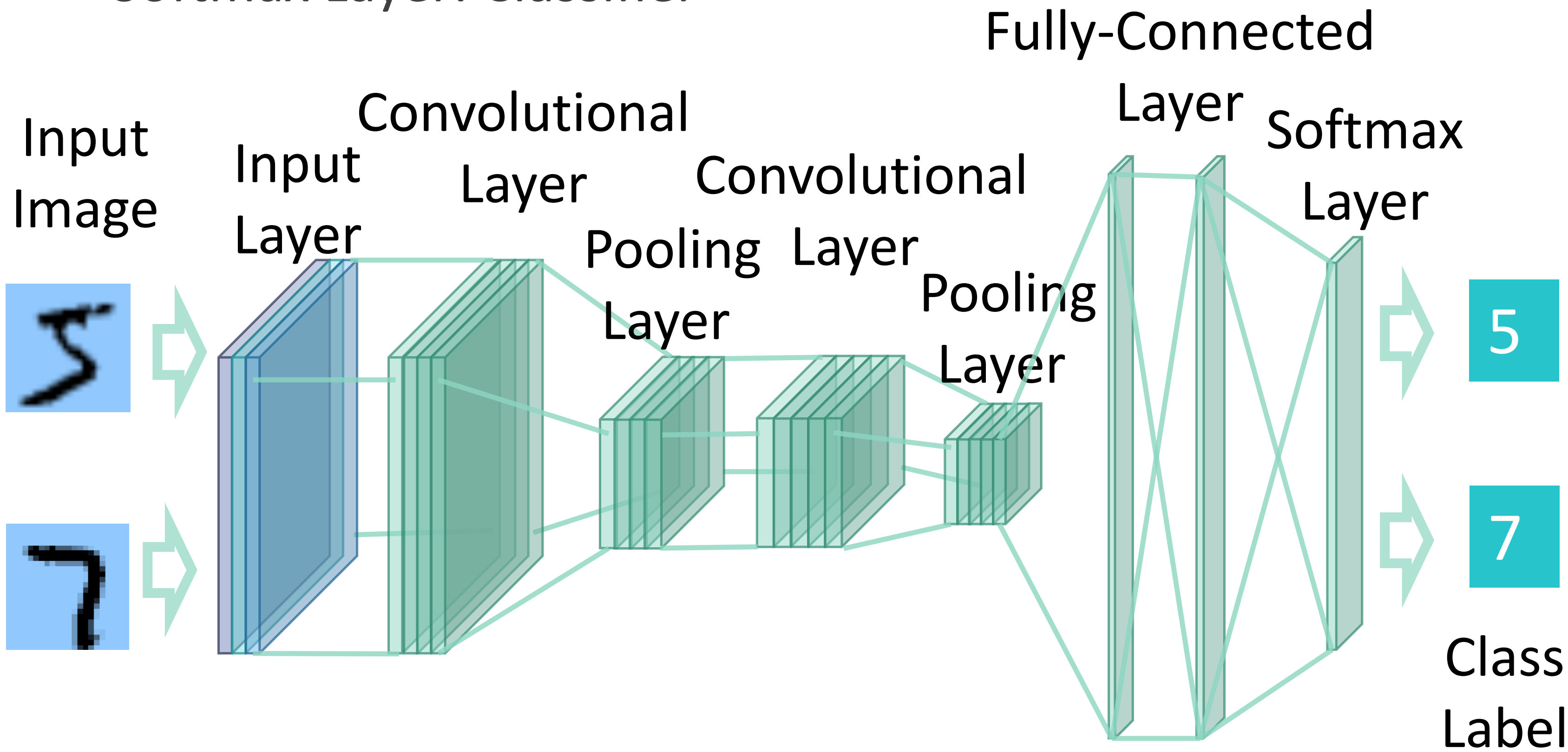


Visual Perception of Computer



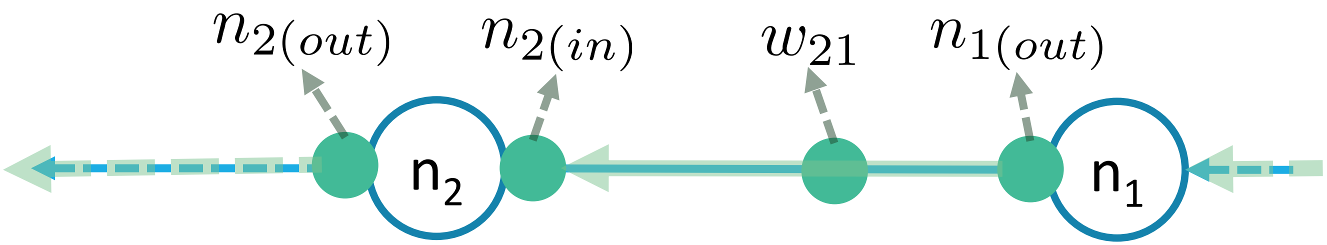
Fully-Connected Layer

- Fully-Connected Layers : Global feature extraction
- Softmax Layer: Classifier



Training

- Forward Propagation

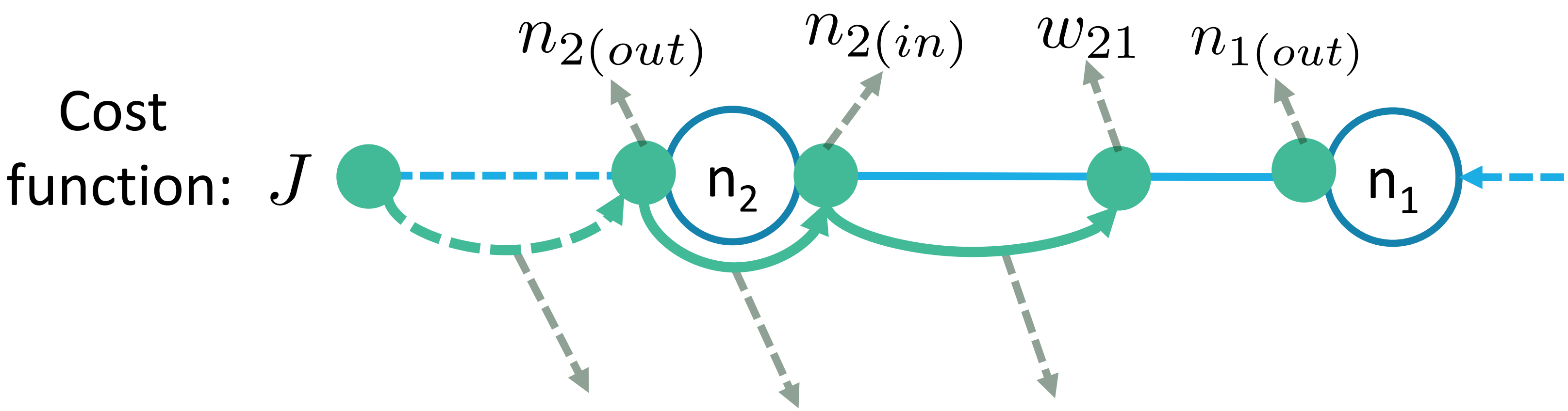


$$n_{2(in)} = w_{21} n_{1(out)}$$

$$n_{2(out)} = g(n_{2(in)}), \quad g \text{ is activation function}$$

Training

- Update weights



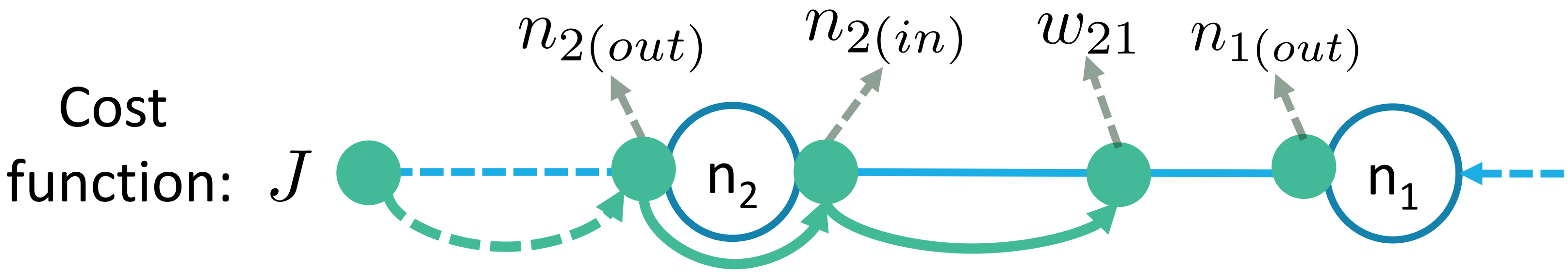
$$\frac{\partial J}{\partial w_{21}} = \frac{\partial J}{\partial n_{2(out)}} \frac{\partial n_{2(out)}}{\partial n_{2(in)}} \frac{\partial n_{2(in)}}{\partial w_{21}}$$

$$w_{21} \leftarrow w_{21} - \eta \frac{\partial J}{\partial w_{21}}$$

$$\Rightarrow w_{21} \leftarrow w_{21} - \eta \frac{\partial J}{\partial n_{2(out)}} \frac{\partial n_{2(out)}}{\partial n_{2(in)}} \frac{\partial n_{2(in)}}{\partial w_{21}}$$

Training

- Update weights



$$n_{2(out)} = g(n_{2(in)}), n_{2(in)} = w_{21} n_{1(out)}$$

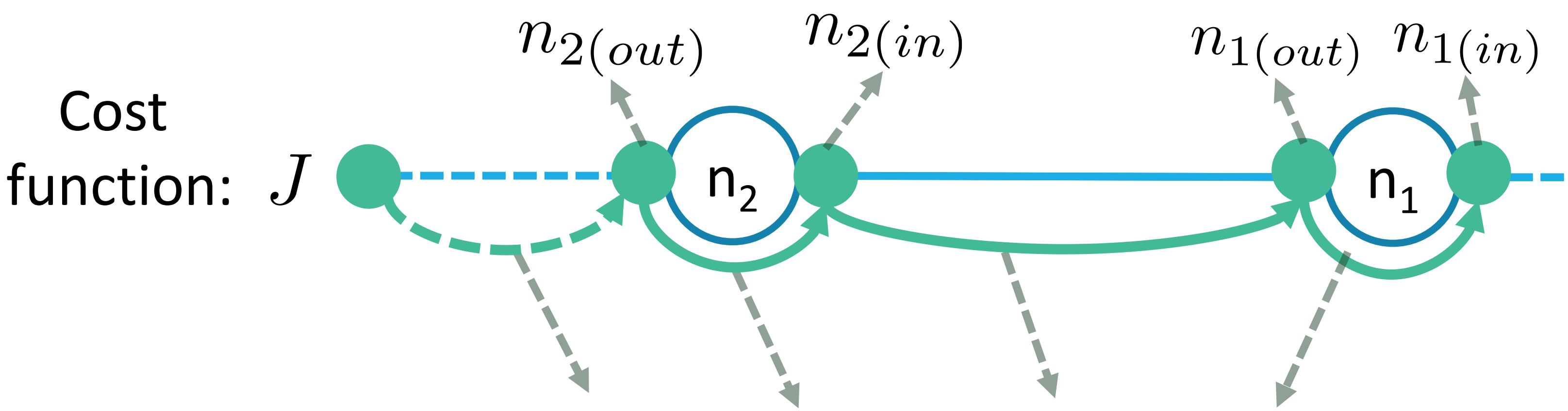
$$\Rightarrow \frac{\partial n_{2(out)}}{\partial n_{2(in)}} = g'(n_{2(in)}), \frac{\partial n_{2(in)}}{\partial w_{21}} = n_{1(out)}$$

$$w_{21} \leftarrow w_{21} - \eta \frac{\partial J}{\partial n_{2(out)}} \frac{\partial n_{2(out)}}{\partial n_{2(in)}} \frac{\partial n_{2(in)}}{\partial w_{21}}$$

$$\Rightarrow w_{21} \leftarrow w_{21} - \eta \frac{\partial J}{\partial n_{2(out)}} g'(n_{2(in)}) n_{1(out)}$$

Training

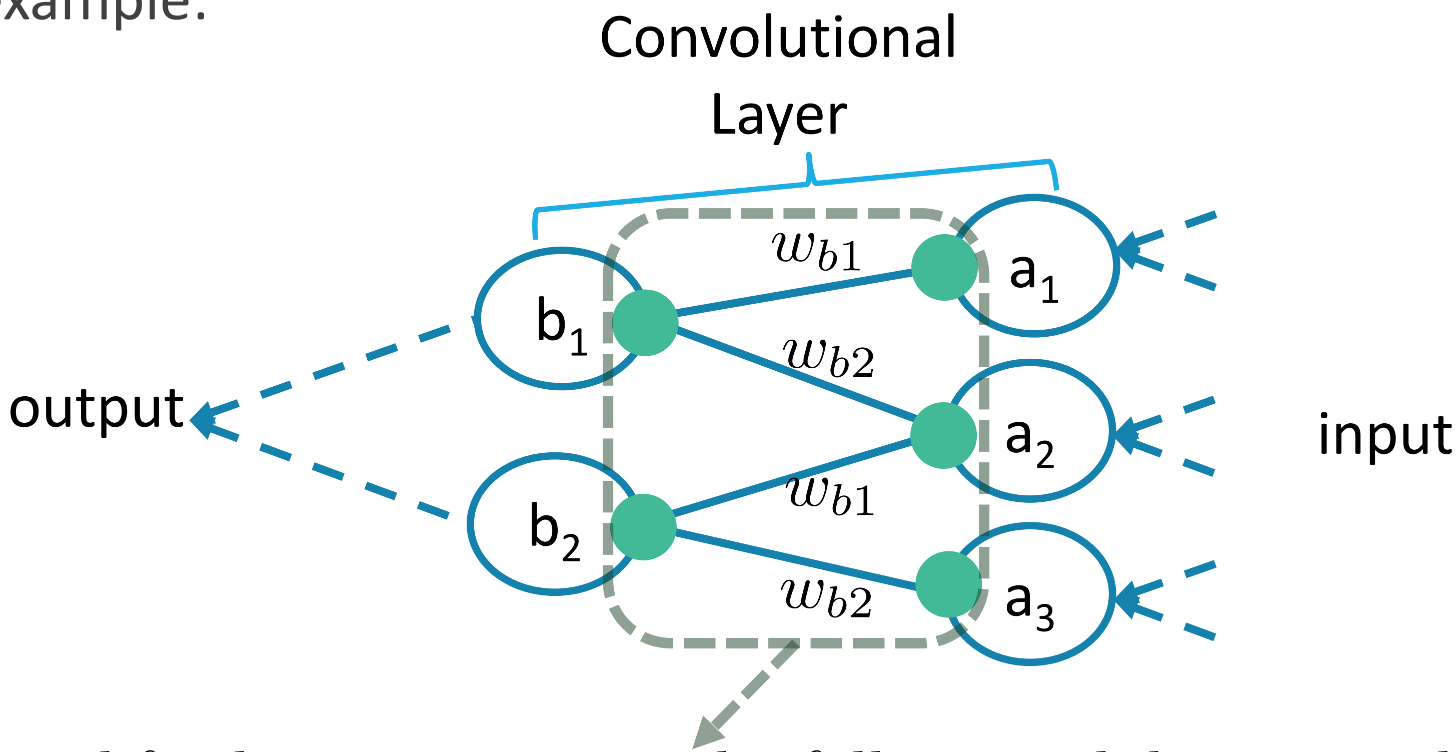
- Propagate to the previous layer



$$\frac{\partial J}{\partial n_{1(in)}} = \frac{\partial J}{\partial n_{2(out)}} \frac{\partial n_{2(out)}}{\partial n_{2(in)}} \frac{\partial n_{2(in)}}{\partial n_{1(out)}} \frac{\partial n_{1(out)}}{\partial n_{1(in)}}$$

Training Convolutional Layers

- example:



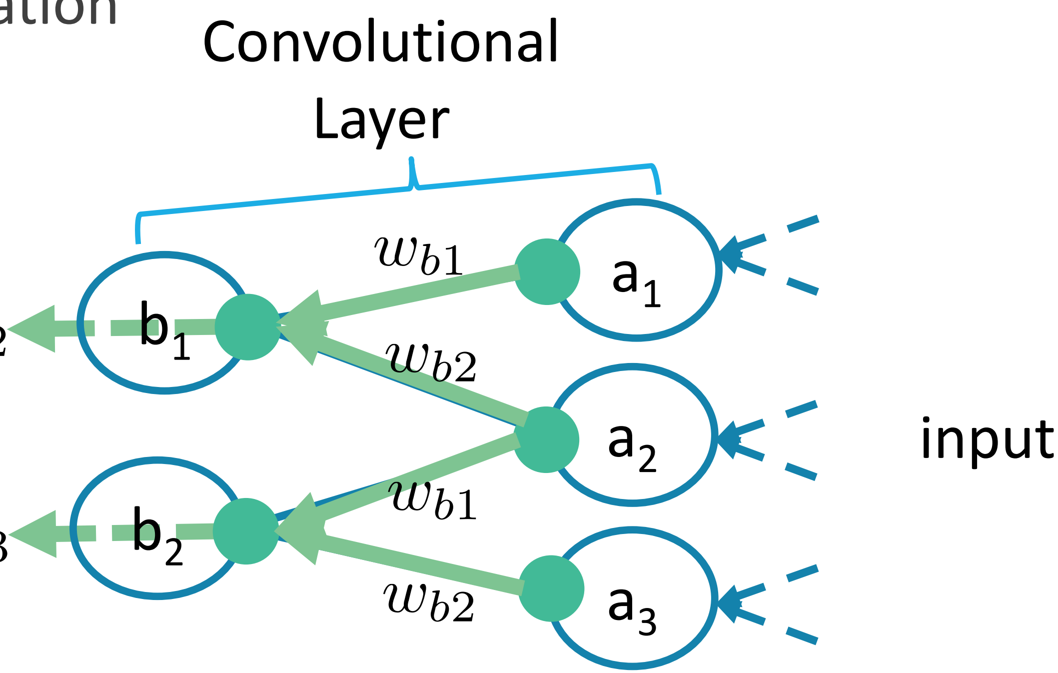
To simplify the notations, in the following slides, we make:
b_1 means $b_{1(in)}$, a_1 means $a_{1(out)}$, and so on.

Training Convolutional Layers

- Forward propagation

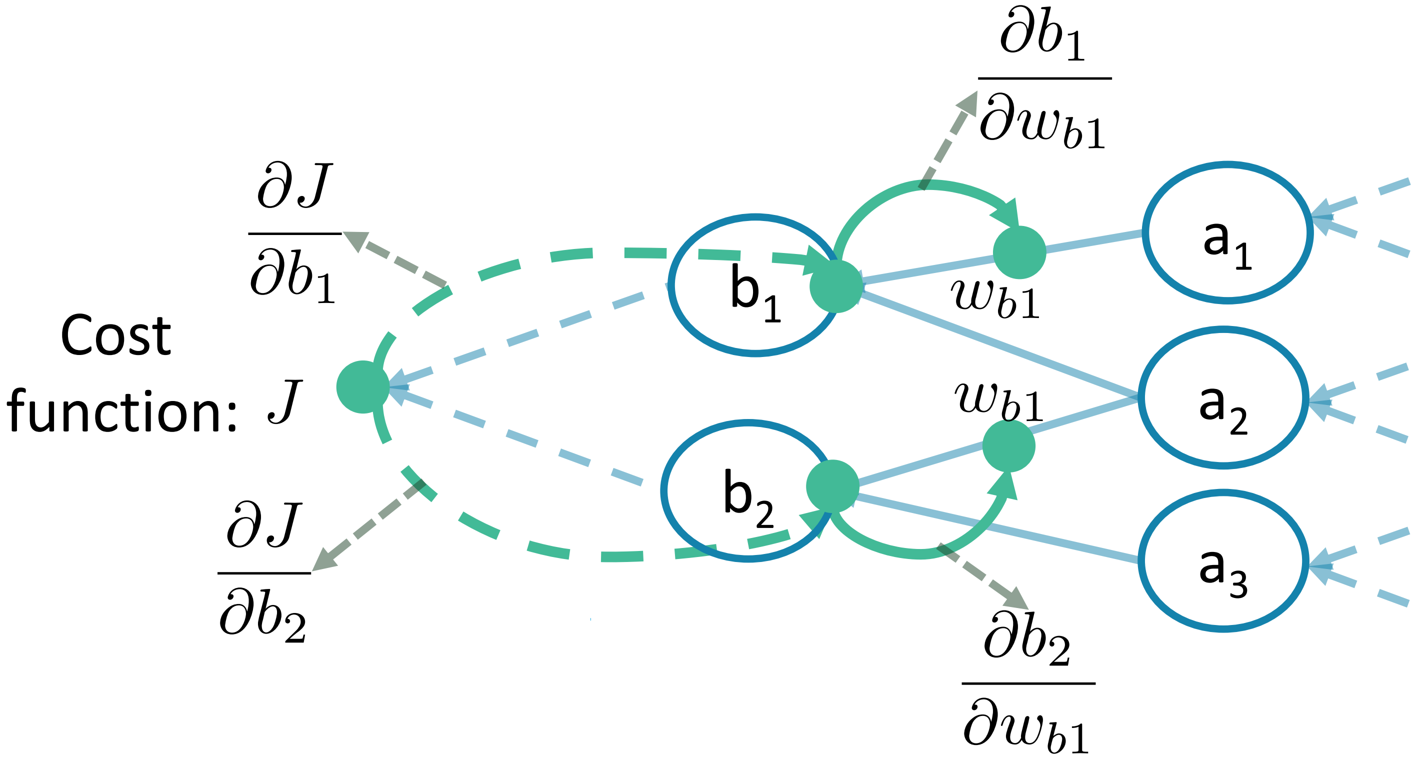
$$b_1 = w_{b1}a_1 + w_{b2}a_2$$

$$b_2 = w_{b1}a_2 + w_{b2}a_3$$



Training Convolutional Layers

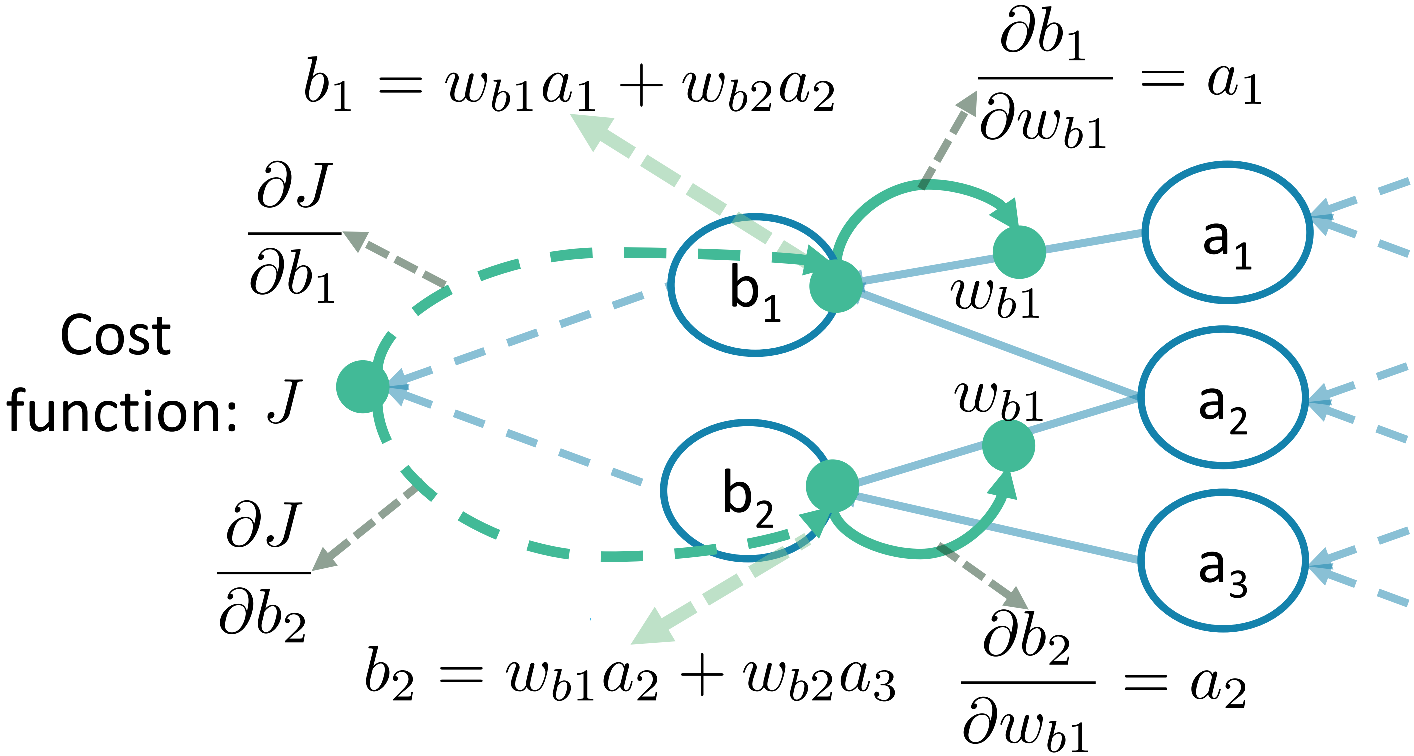
- Update weights



$$w_{b1} \leftarrow w_{b1} - \eta\left(\frac{\partial J}{\partial b_1}\frac{\partial b_1}{\partial w_{b1}} + \frac{\partial J}{\partial b_2}\frac{\partial b_2}{\partial w_{b1}}\right)$$

Training Convolutional Layers

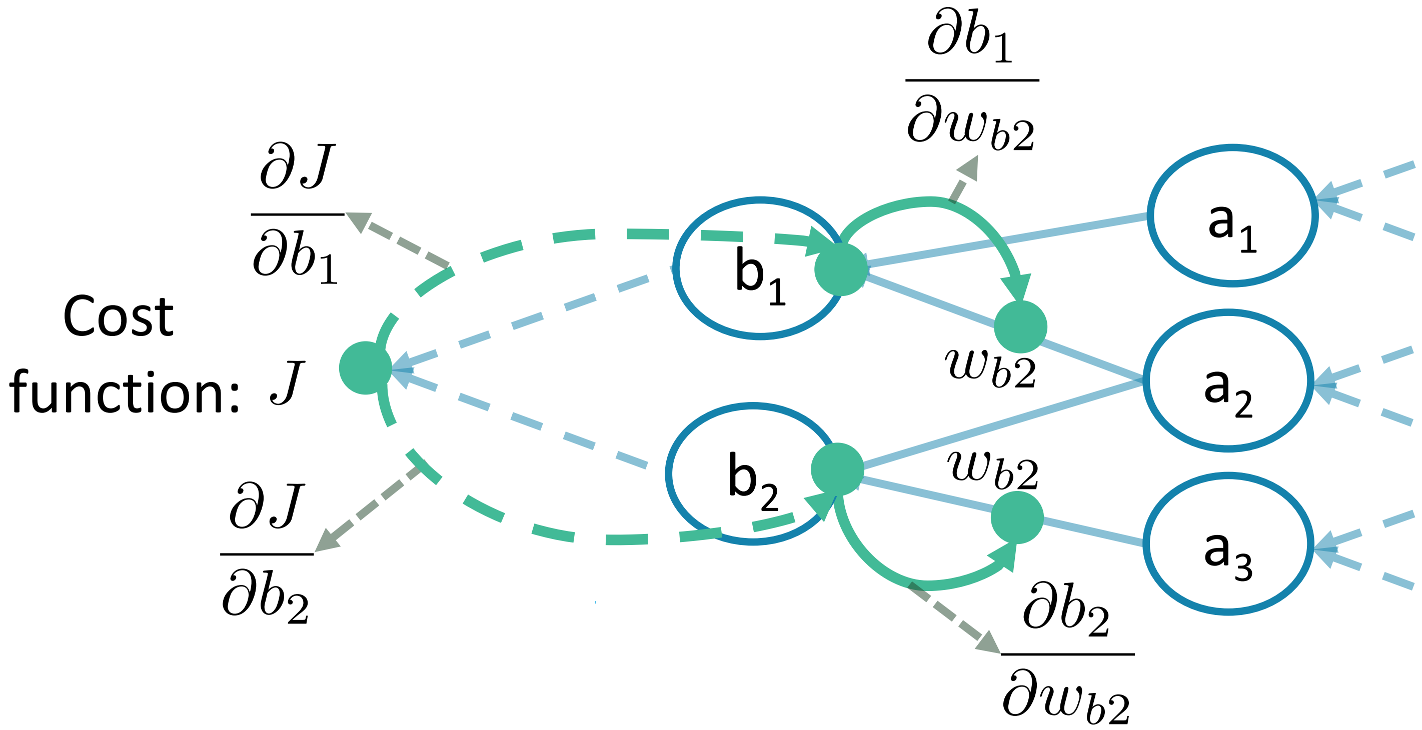
- Update weights



$$w_{b1} \leftarrow w_{b1} - \eta\left(\frac{\partial J}{\partial b_1}a_1 + \frac{\partial J}{\partial b_2}a_2\right)$$

Training Convolutional Layers

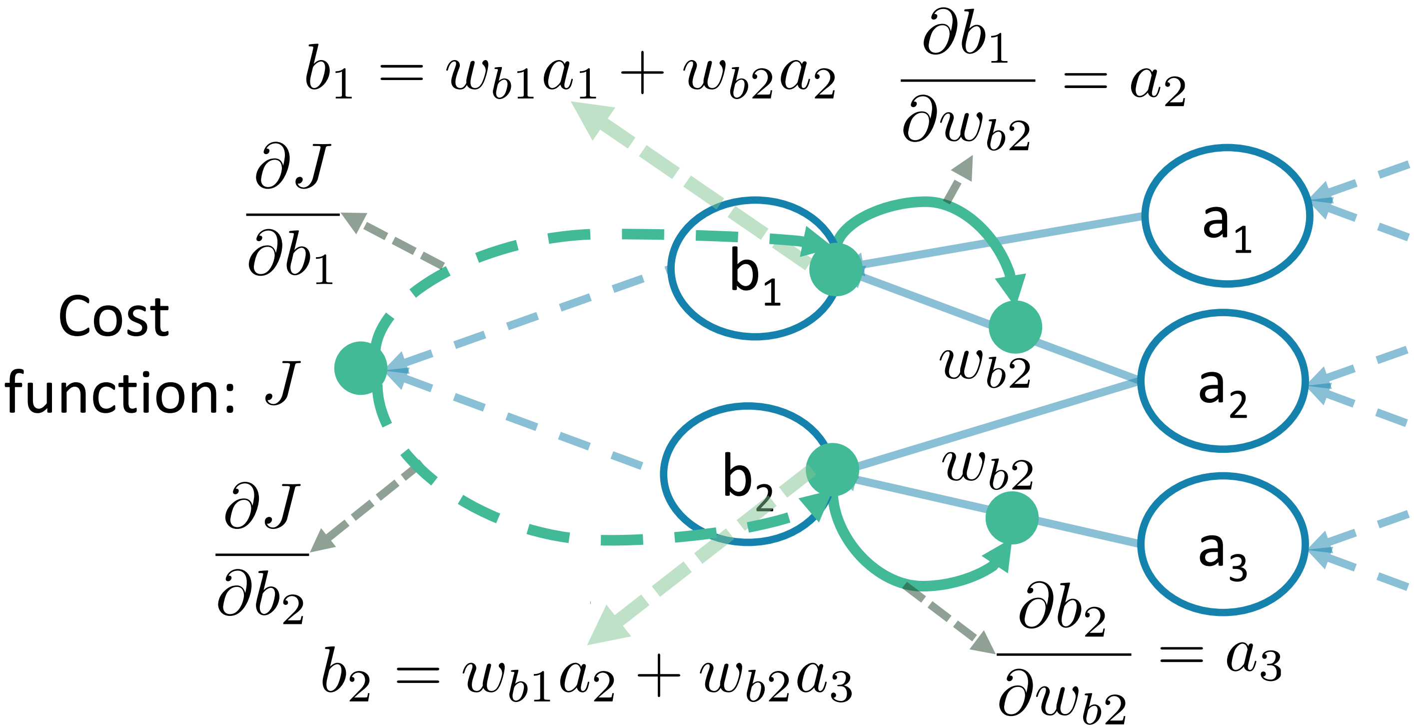
- Update weights



$$w_{b2} \leftarrow w_{b2} - \eta\left(\frac{\partial J}{\partial b_1}\frac{\partial b_1}{\partial w_{b2}} + \frac{\partial J}{\partial b_2}\frac{\partial b_2}{\partial w_{b2}}\right)$$

Training Convolutional Layers

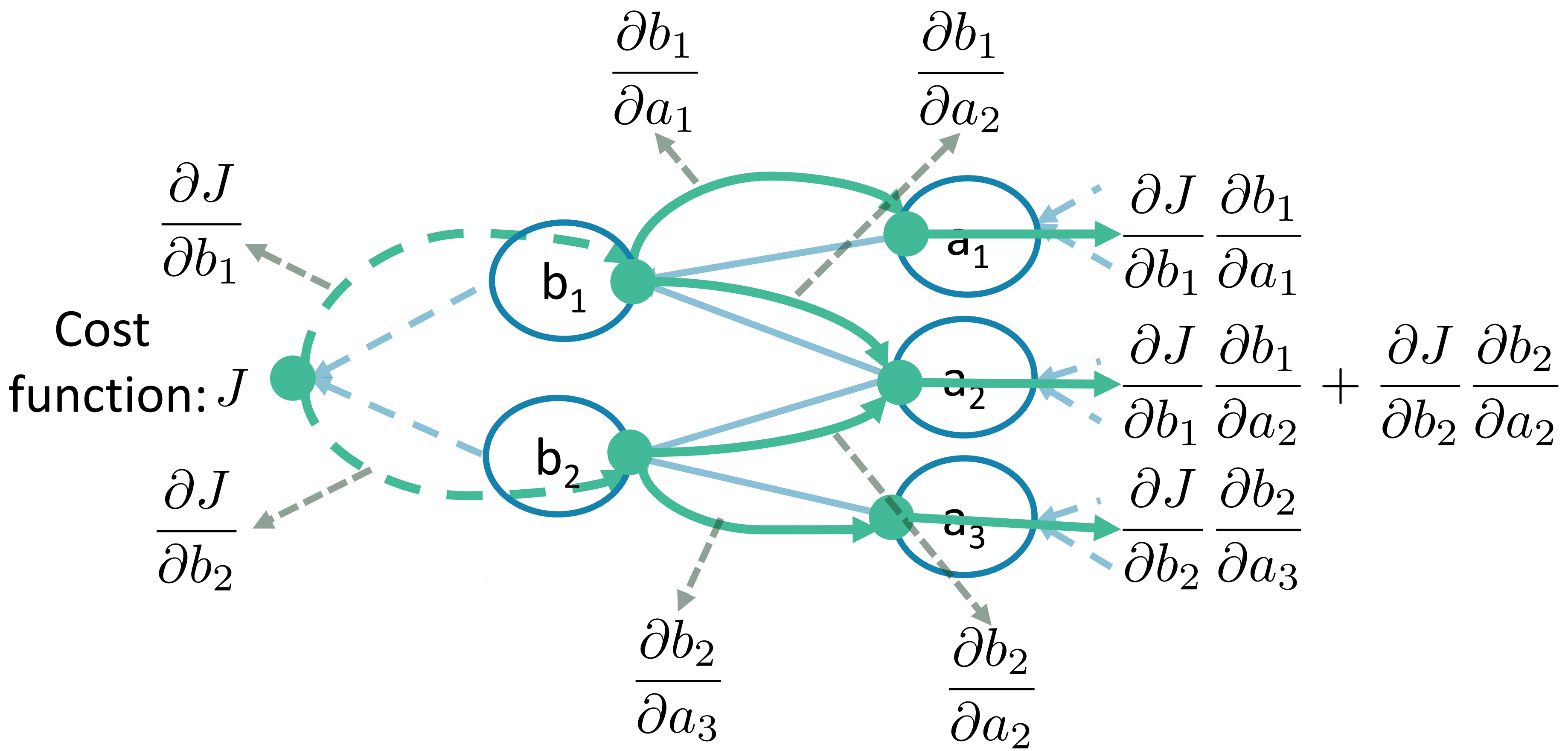
- Update weights



$$w_{b2} \leftarrow w_{b2} - \eta\left(\frac{\partial J}{\partial b_1}a_2 + \frac{\partial J}{\partial b_2}a_3\right)$$

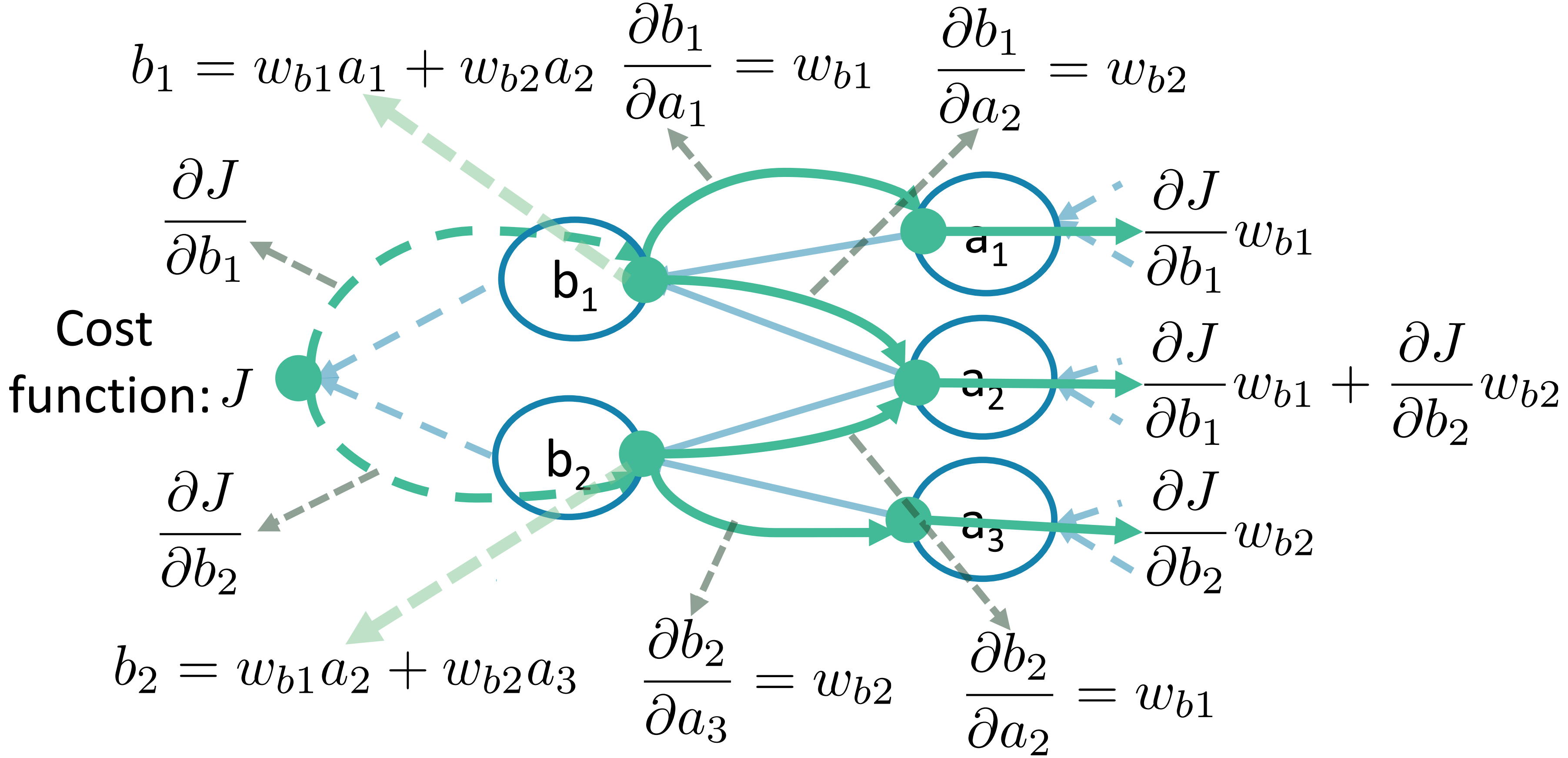
Training Convolutional Layers

- Propagate to the previous layer



Training Convolutional Layers

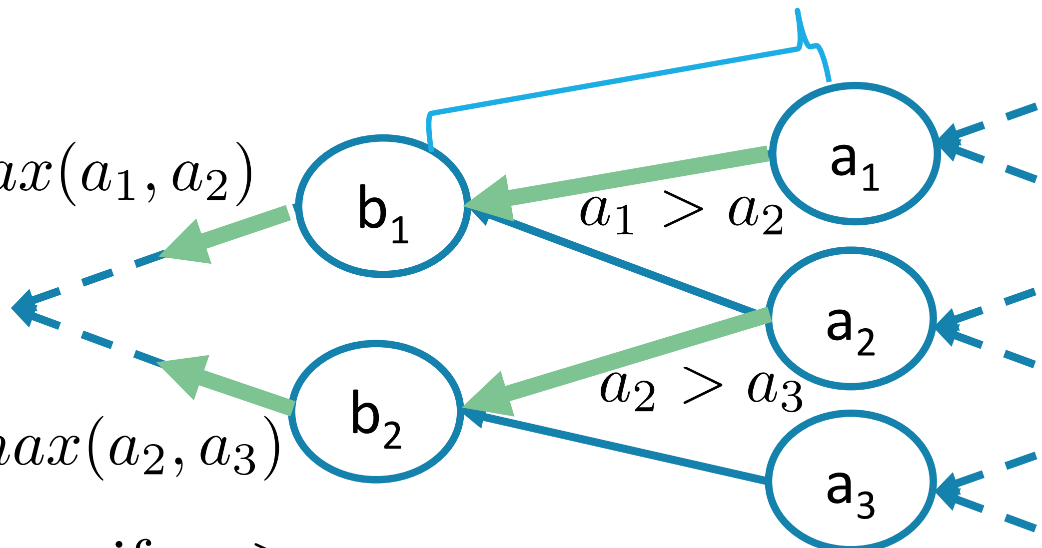
- Propagate to the previous layer



Max-Pooling Layers during Training

- Pooling layers have no weights
- No need to update weights

$$b_1 = max(a_1, a_2)$$



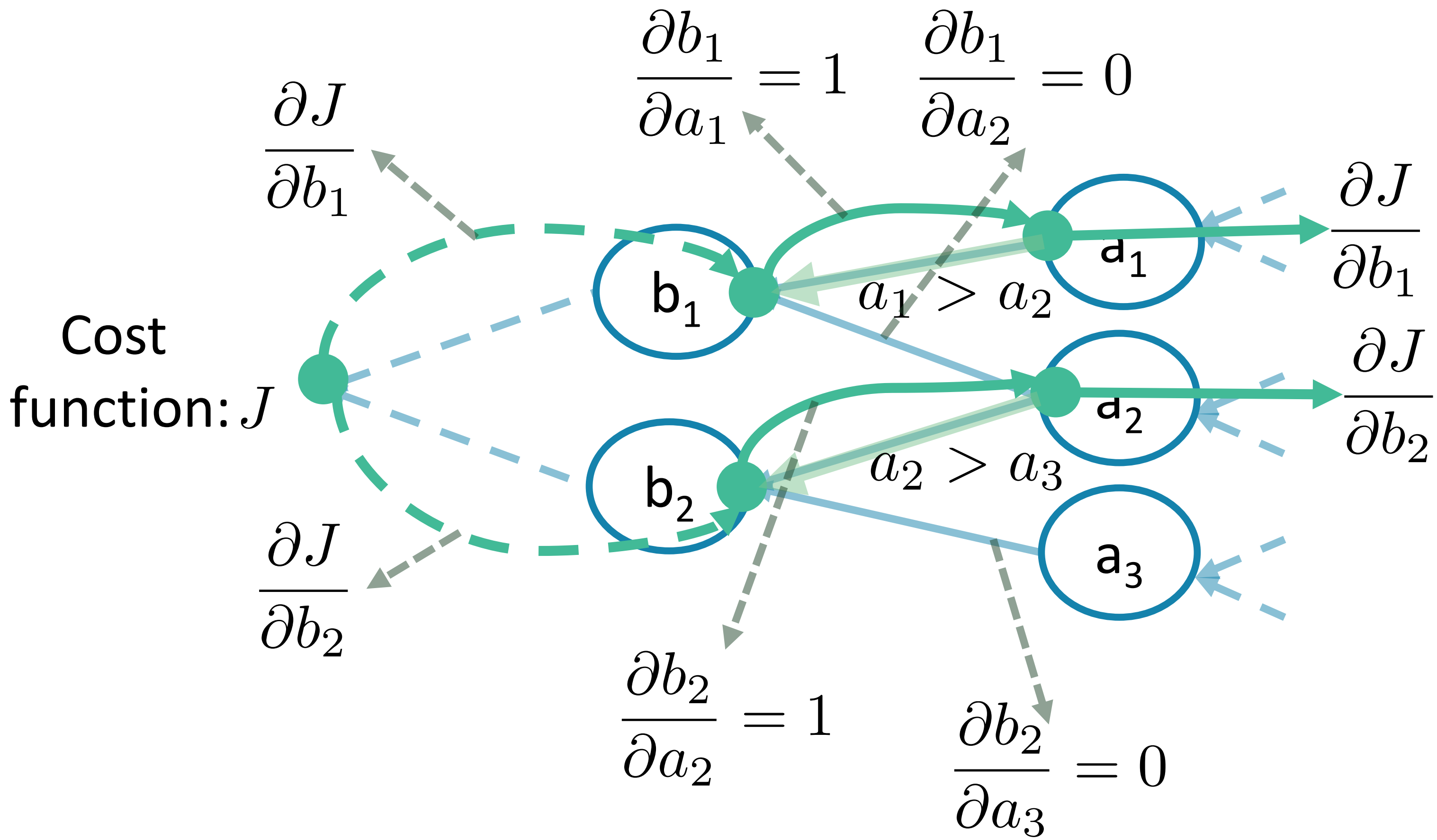
$$b_2 = max(a_2, a_3)$$

$$= \begin{cases} a_2 \text{ if } a_2 \geq a_3 \\ a_3 \text{ otherwise} \end{cases}$$

$$\frac{\partial b_2}{\partial a_2} = \begin{cases} 1 \text{ if } a_2 \geq a_3 \\ 0 \text{ otherwise} \end{cases}$$

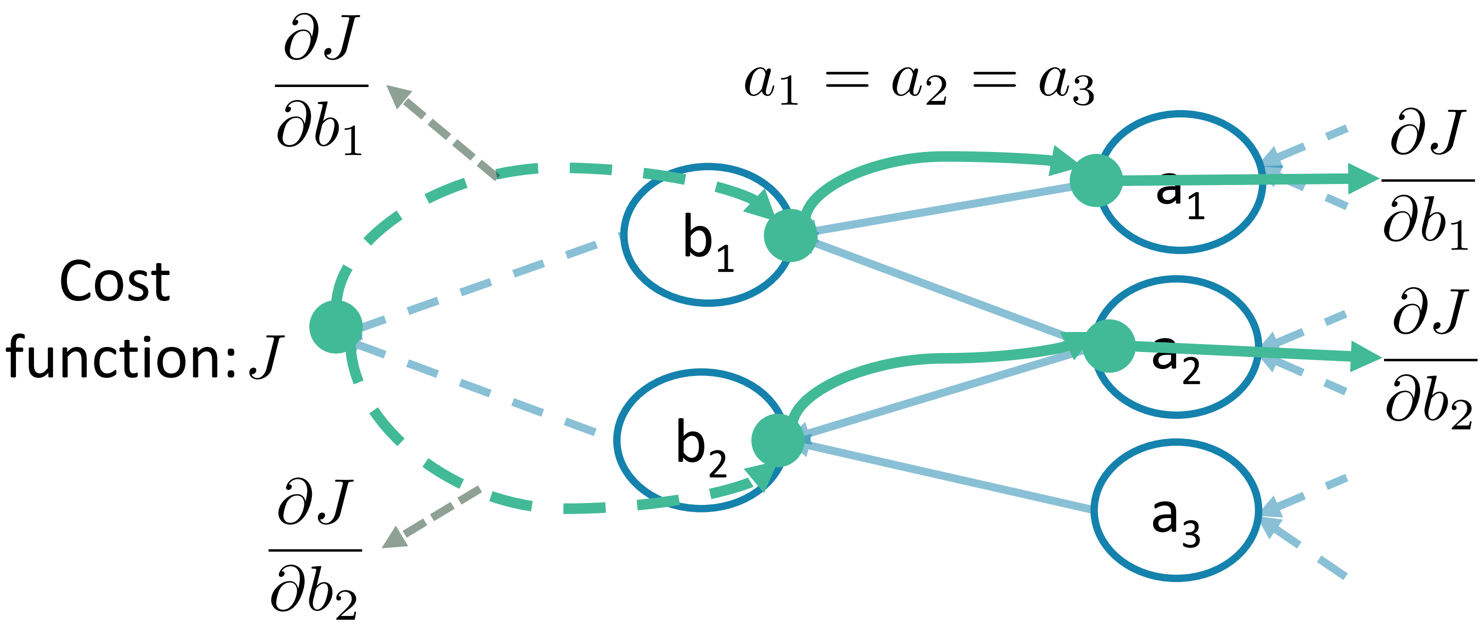
Max-Pooling Layers during Training

- Propagate to the previous layer



Max-Pooling Layers during Training

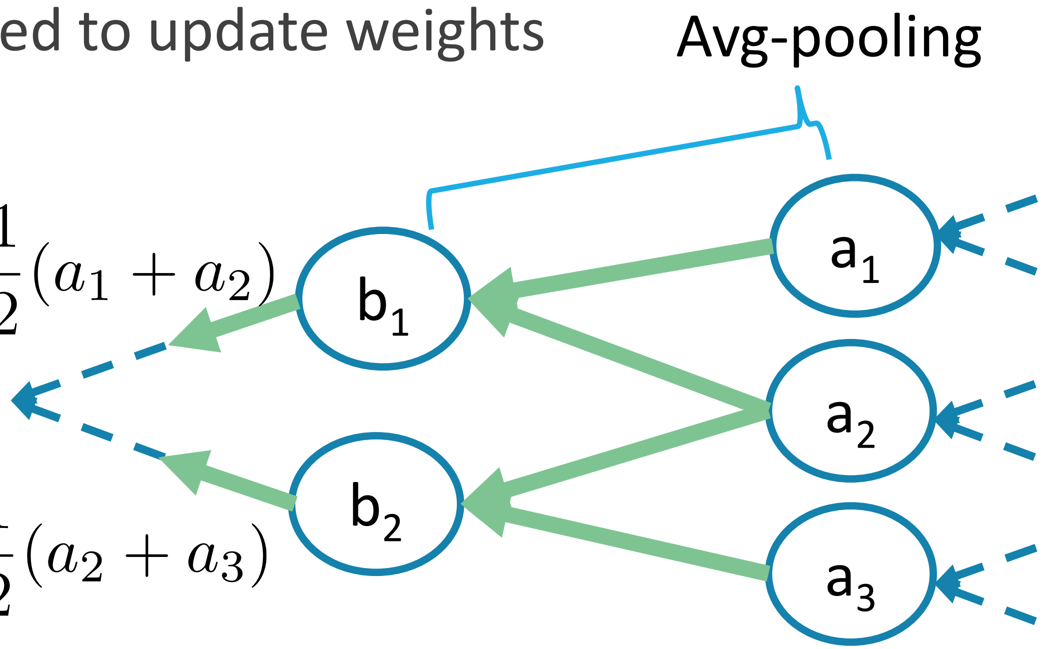
- if $a_1 = a_2$??
  - Choose the node with **smaller index**



Avg-Pooling Layers during Training

- Pooling layers have no weights
- No need to update weights

$$b_1 = \frac{1}{2}(a_1 + a_2)$$

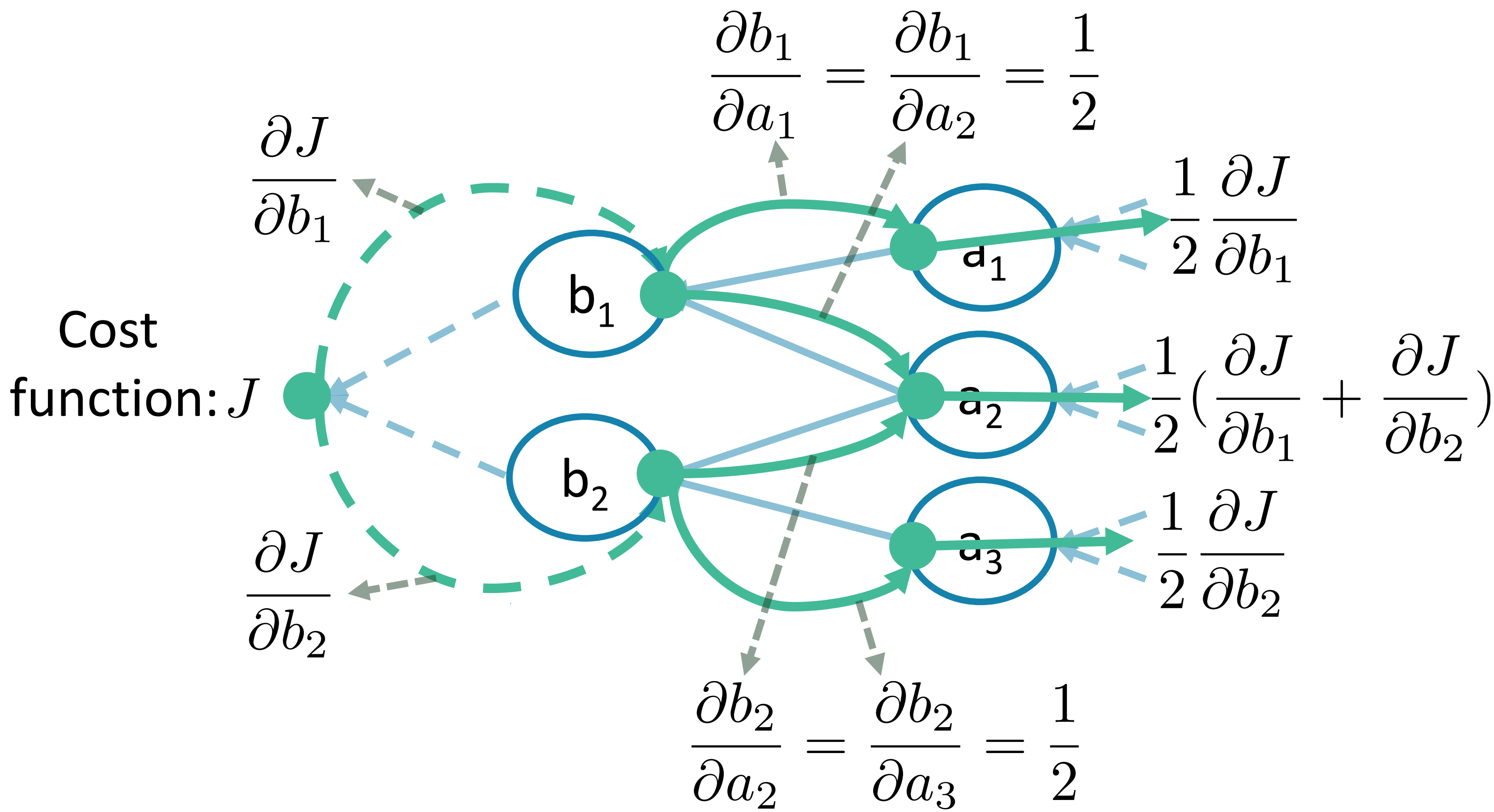


$$b_2 = \frac{1}{2}(a_2 + a_3)$$

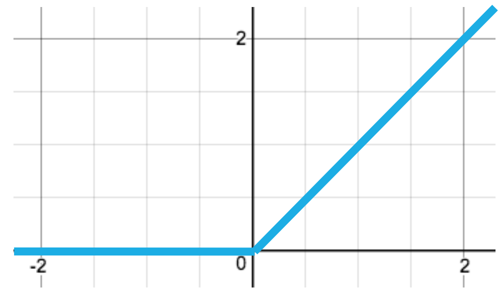
$$\frac{\partial b_2}{\partial a_2} = \frac{1}{2} \qquad \frac{\partial b_2}{\partial a_3} = \frac{1}{2}$$

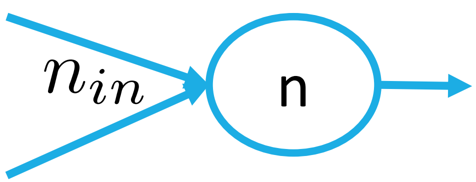
Avg-Pooling Layers during Training

- Propagate to the previous layer



ReLU during Training





$$n_{out} = \begin{cases} n_{in} \text{ if } n_{in} > 0 \\ 0 \text{ otherwise} \end{cases}$$

$$\frac{\partial n_{out}}{\partial n_{in}} = \begin{cases} 1 \text{ if } n_{in} > 1 \\ 0 \text{ otherwise} \end{cases}$$

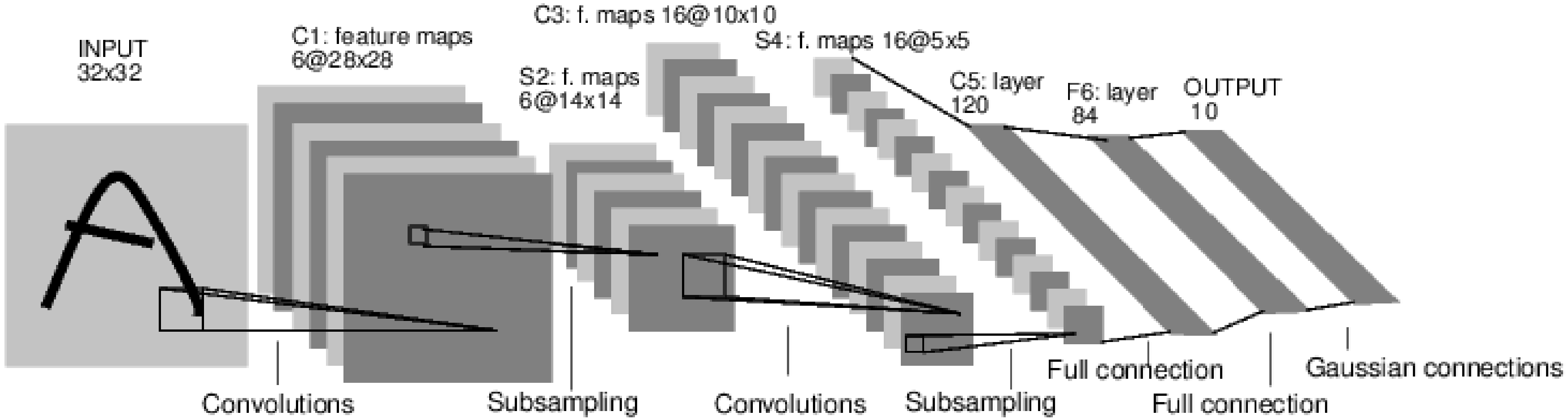
Training CNN

是怎樣傳過來
就怎樣傳回去

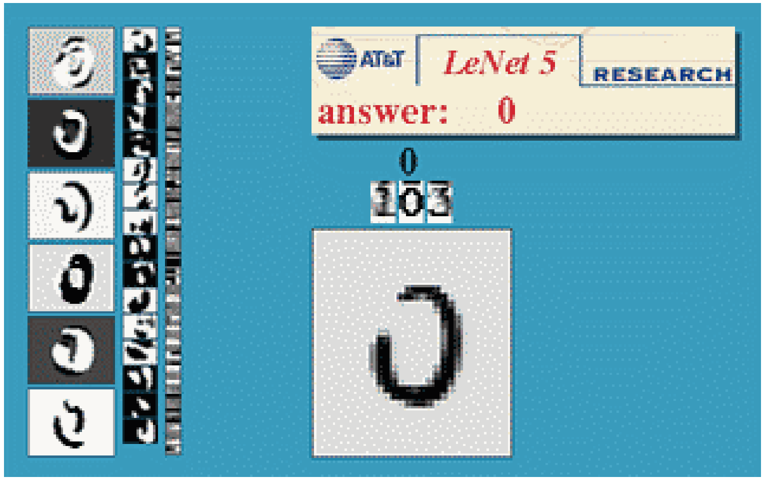
Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- More Applications

LeNet (1998)



Yann LeCun



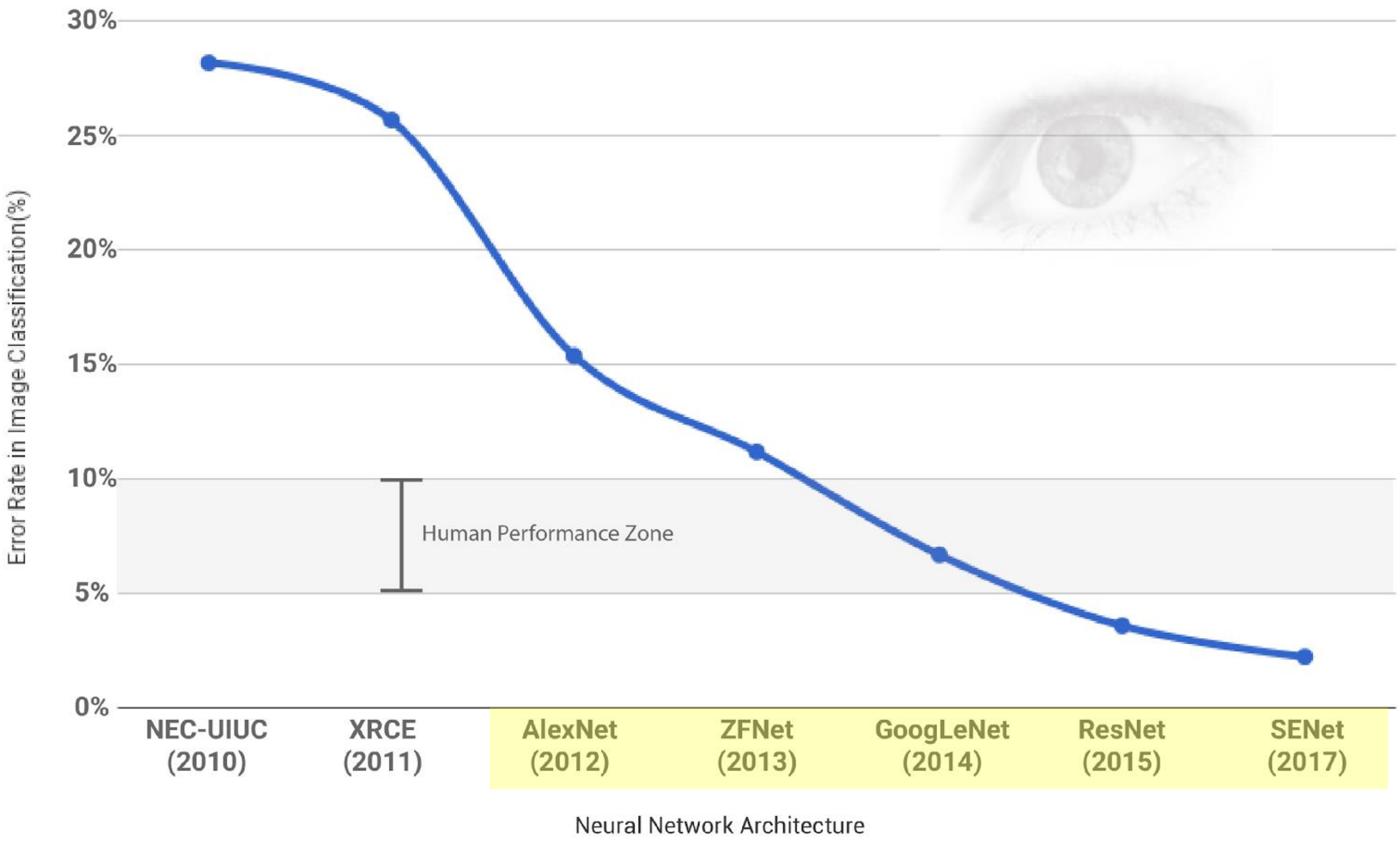
http://yann.lecun.com/exdb/lenet/

ImageNet Challenge (2010-2017)

- ImageNet Large Scale Visual Recognition Challenge
  - 1000 categories
  - Training: 1,200,000
  - Validation: 50,000
  - Testing: 100,000

http://vision.stanford.edu/Datasets/collage_s.png

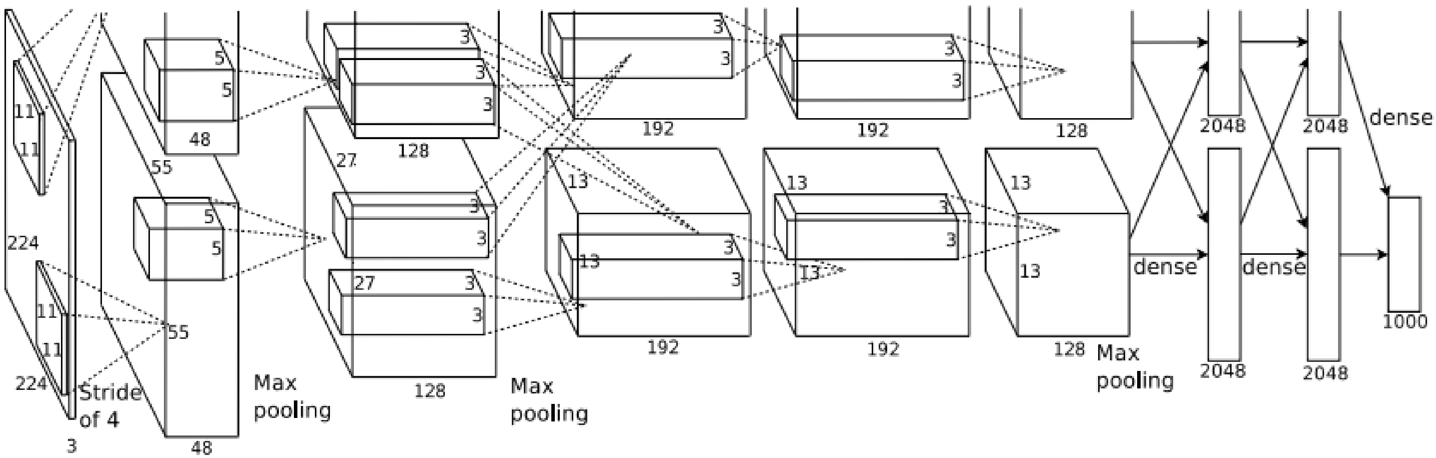
ImageNet Challenge (2010-2017)



https://chtseng.wordpress.com/2017/11/20/ilsvrc-歷屆的深度學習模型

AlexNet (2012)

- The resurgence of Deep Learning
  - ReLU, dropout, image augmentation, max pooling




Alex Krizhevsky


Geoffrey Hinton

VGGNet (2014)

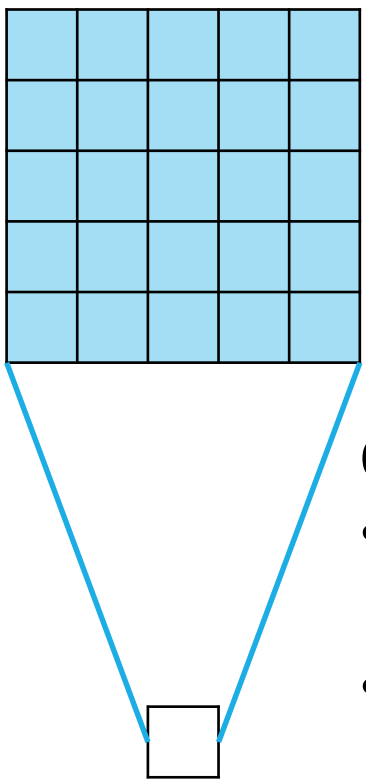
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

D: VGG16
E: VGG19
All filters are 3x3

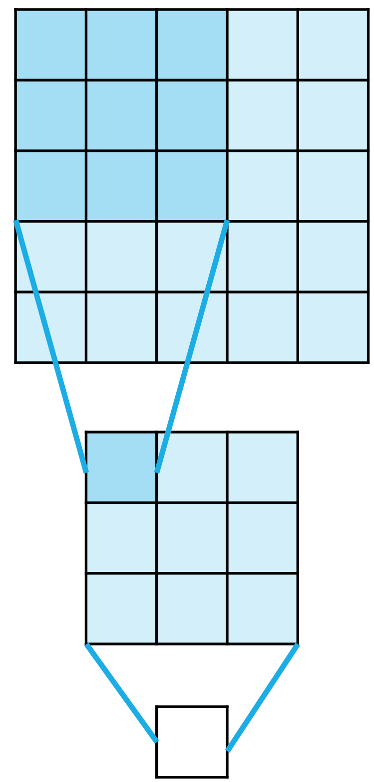
https://arxiv.org/abs/1409.1556

VGGNet

- More layers & smaller filters (3x3) is better
- More non-linearity, fewer parameters

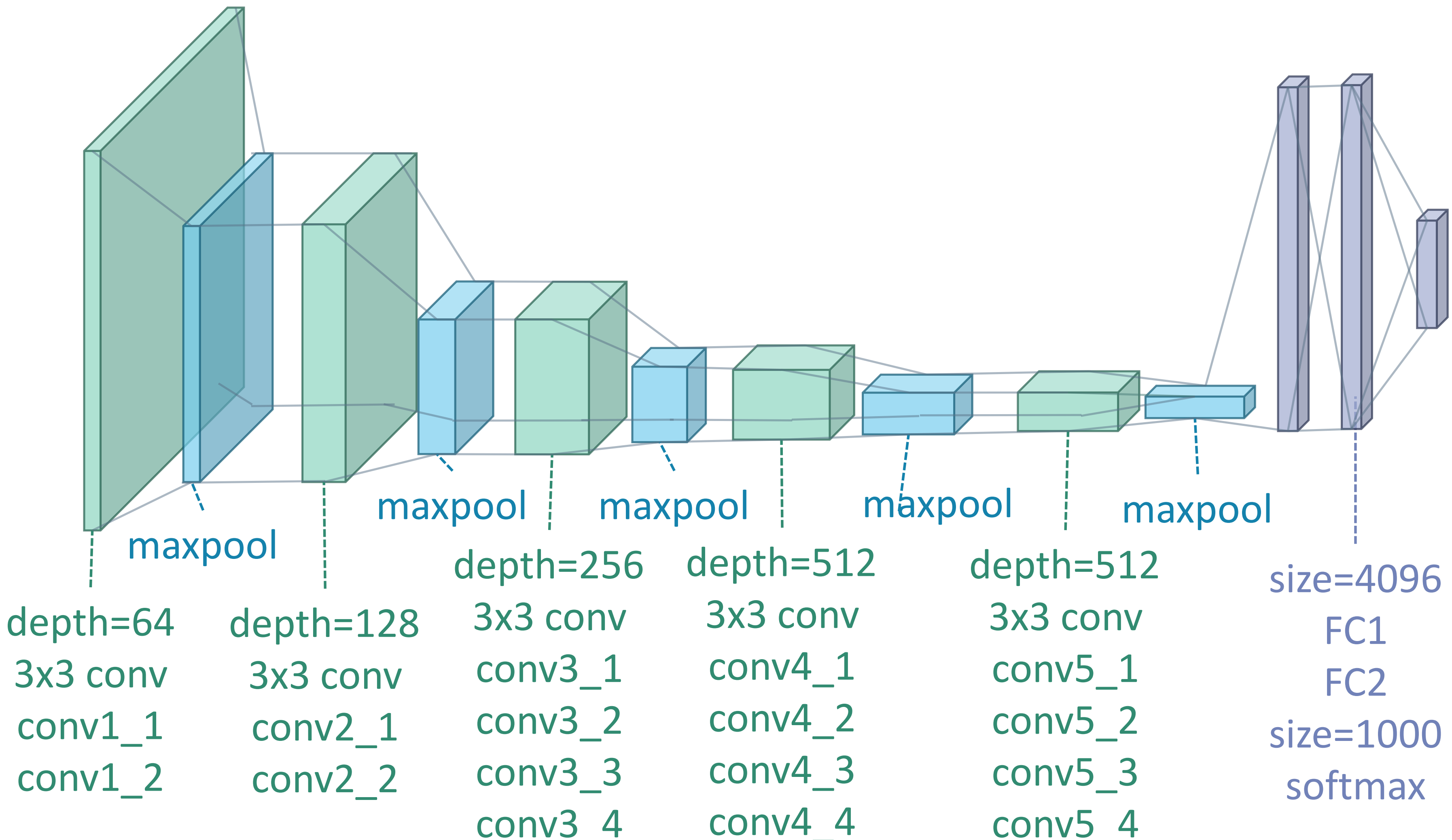


One 5x5 filter
- Parameters: 5x5 = 25
- Non-linear:1



Two 3x3 filters
- Parameters: 3x3x2 = 18
- Non-linear:2

VGG 19

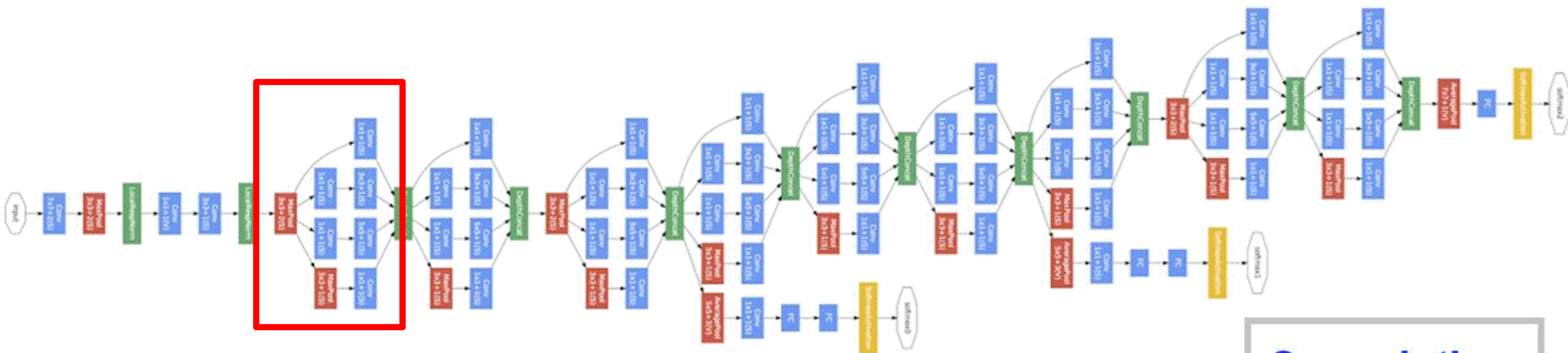


maxpool
maxpool
maxpool
maxpool
maxpool
depth=64
3x3 conv
conv1_1
conv1_2
depth=128
3x3 conv
conv2_1
conv2_2
depth=256
3x3 conv
conv3_1
conv3_2
conv3_3
conv3_4
depth=512
3x3 conv
conv4_1
conv4_2
conv4_3
conv4_4
depth=512
3x3 conv
conv5_1
conv5_2
conv5_3
conv5_4
size=4096
FC1
FC2
size=1000
softmax

GoogLeNet (2014)

- Paper: http://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

22 layers deep network

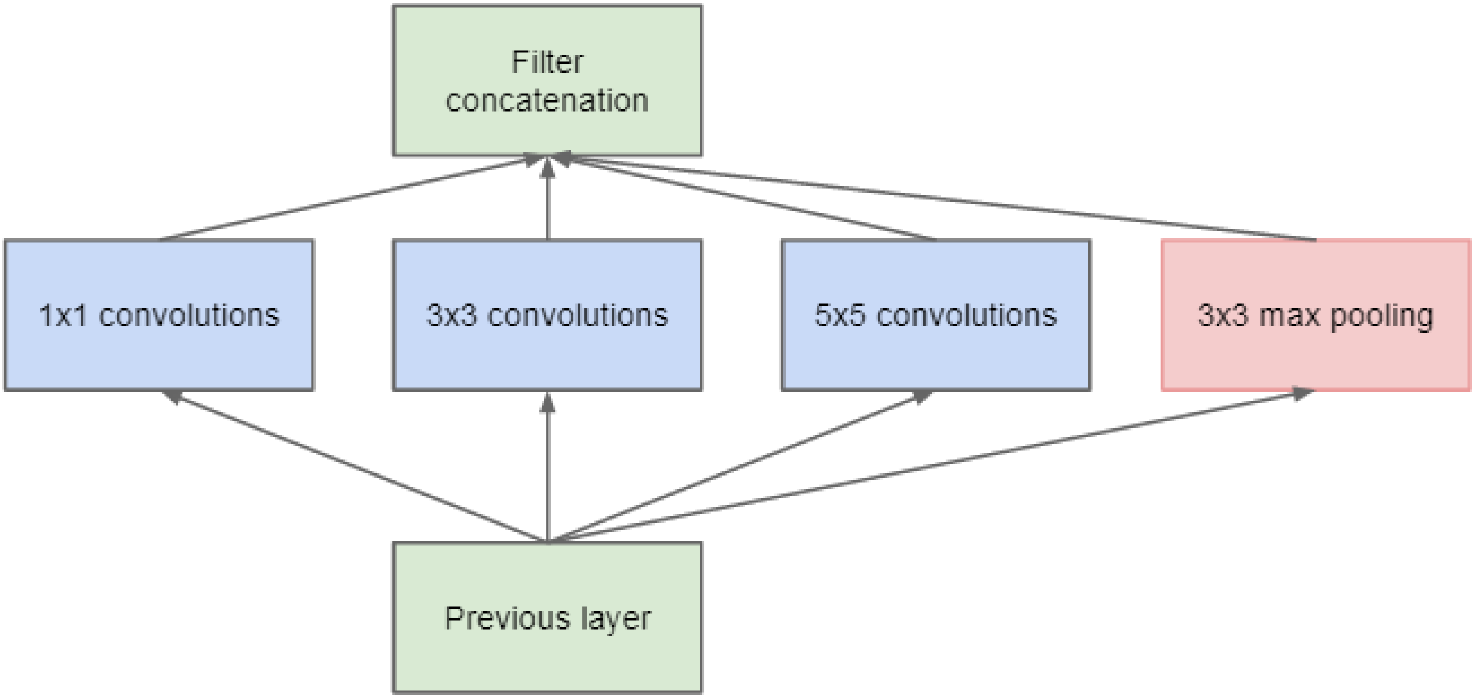


Inception Module

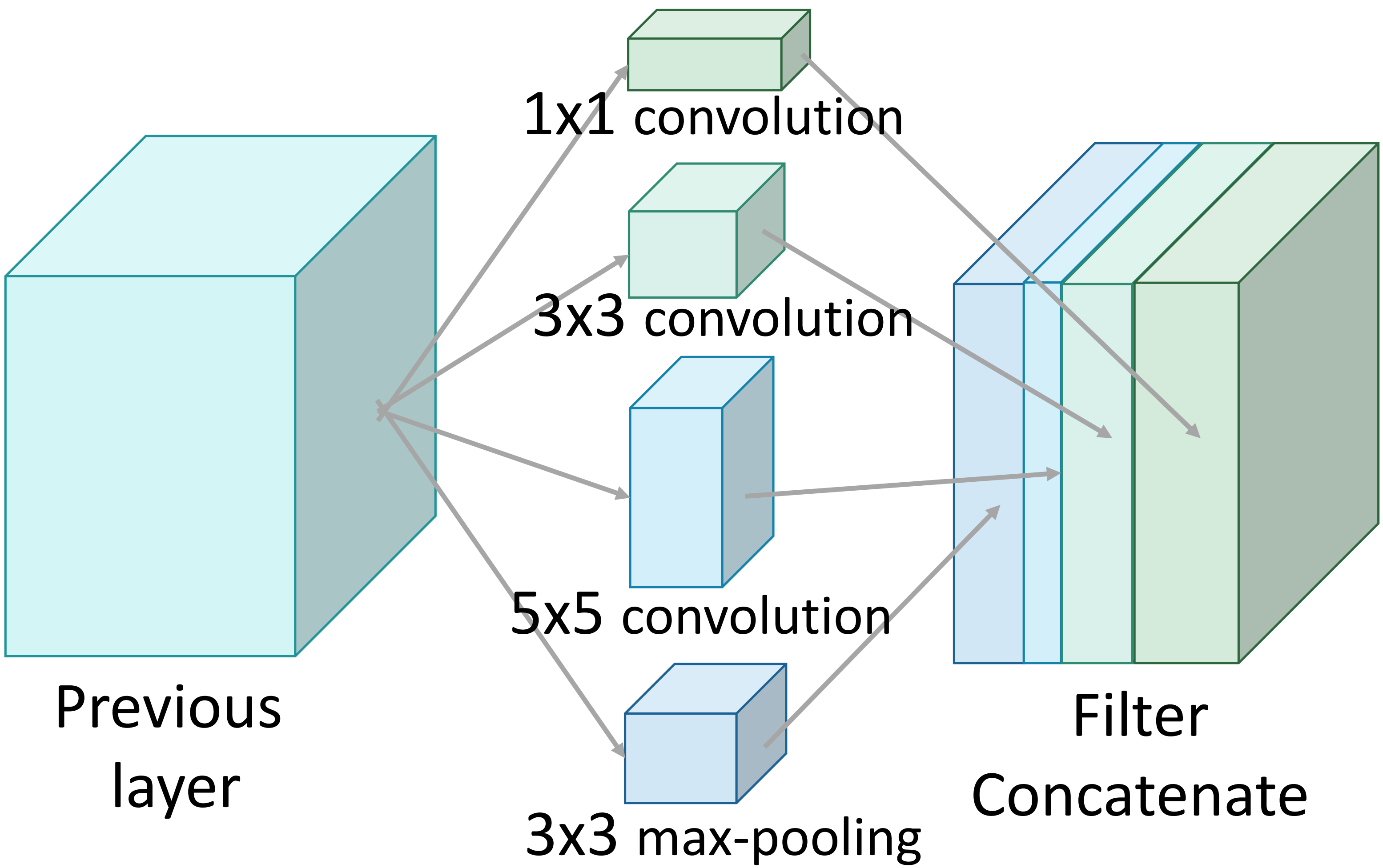
Convolution
Pooling
Softmax
Other

Inception Module

- Best size?
  - 3x3? 5x5?
- Use them all, and combine

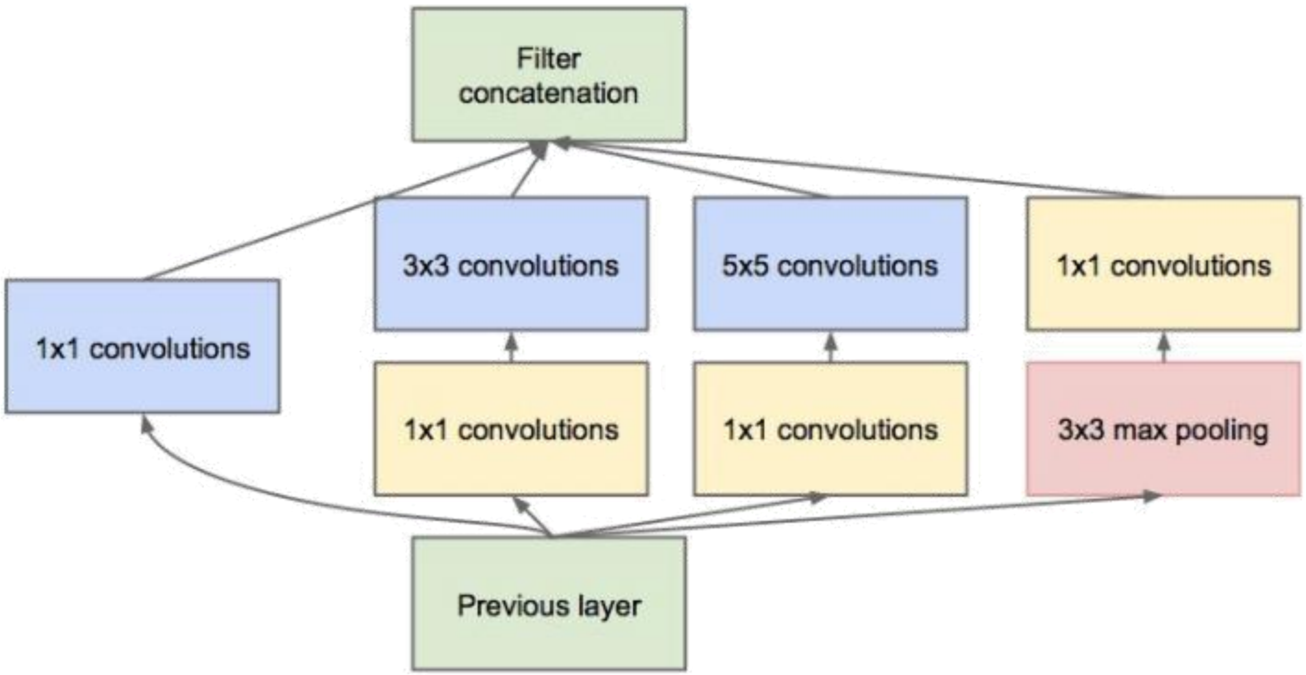


Inception Module

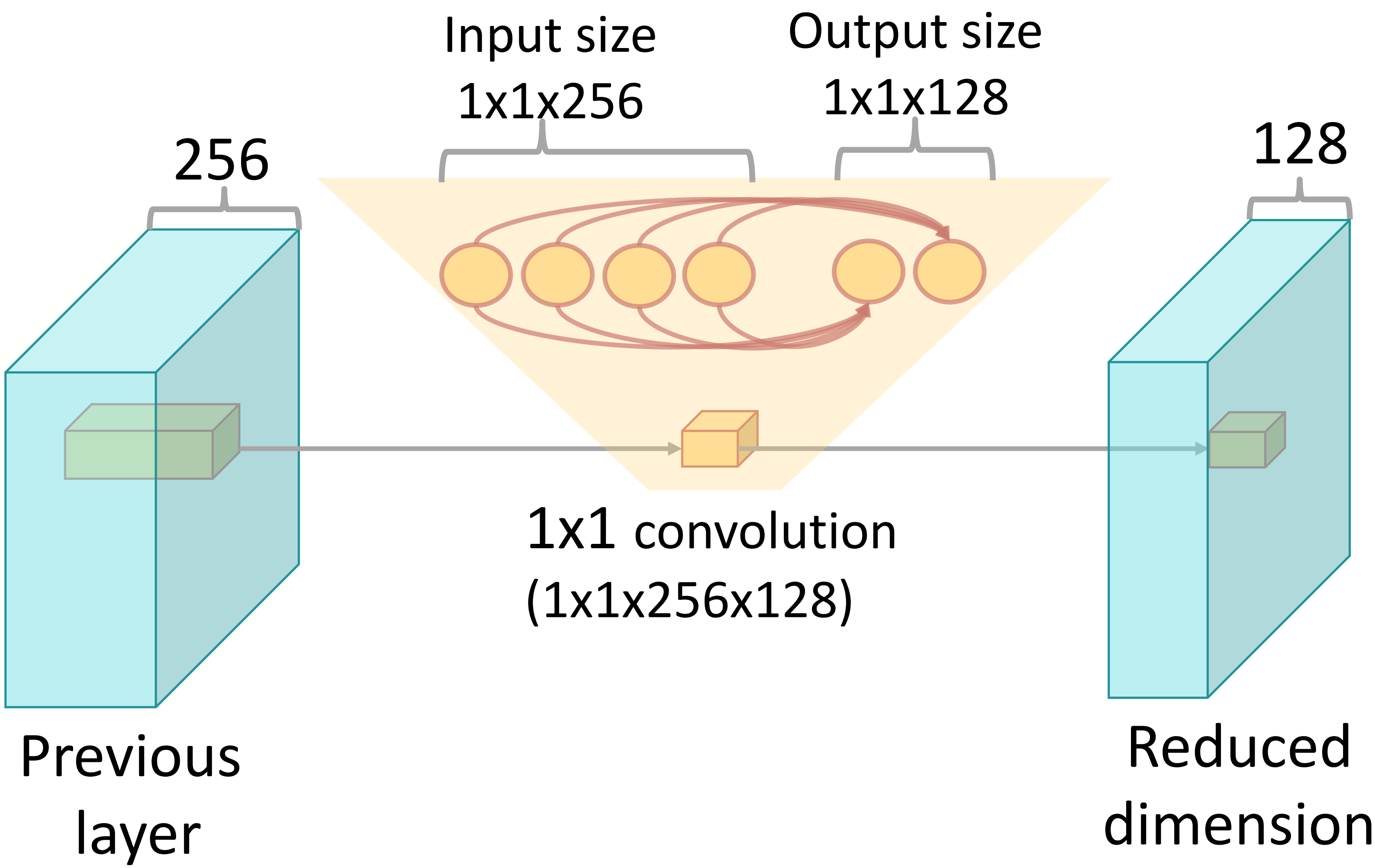


Inception Module with Dimension Reduction

- Use 1x1 filters to reduce dimension



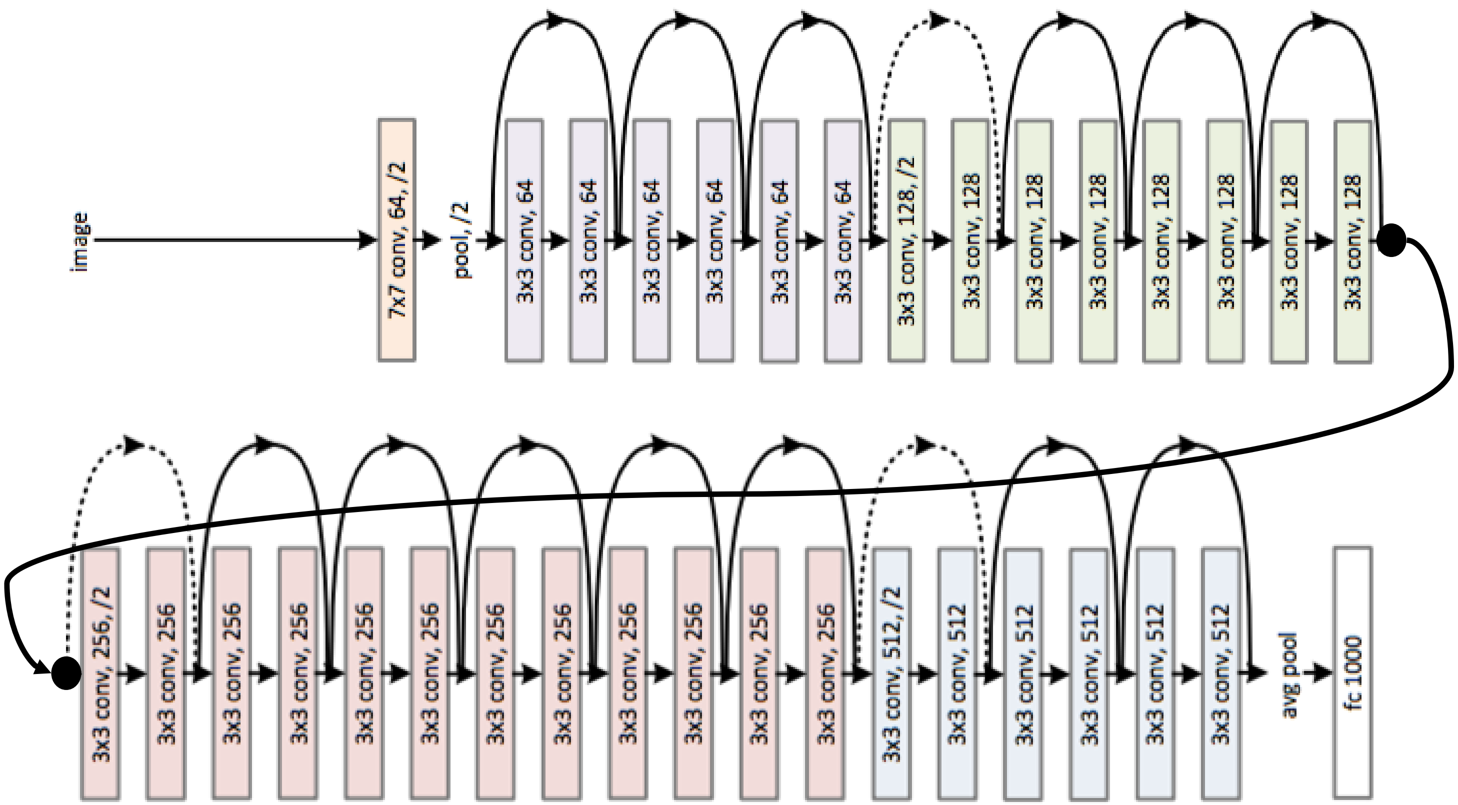
Inception Module with Dimension Reduction



IMAGENET

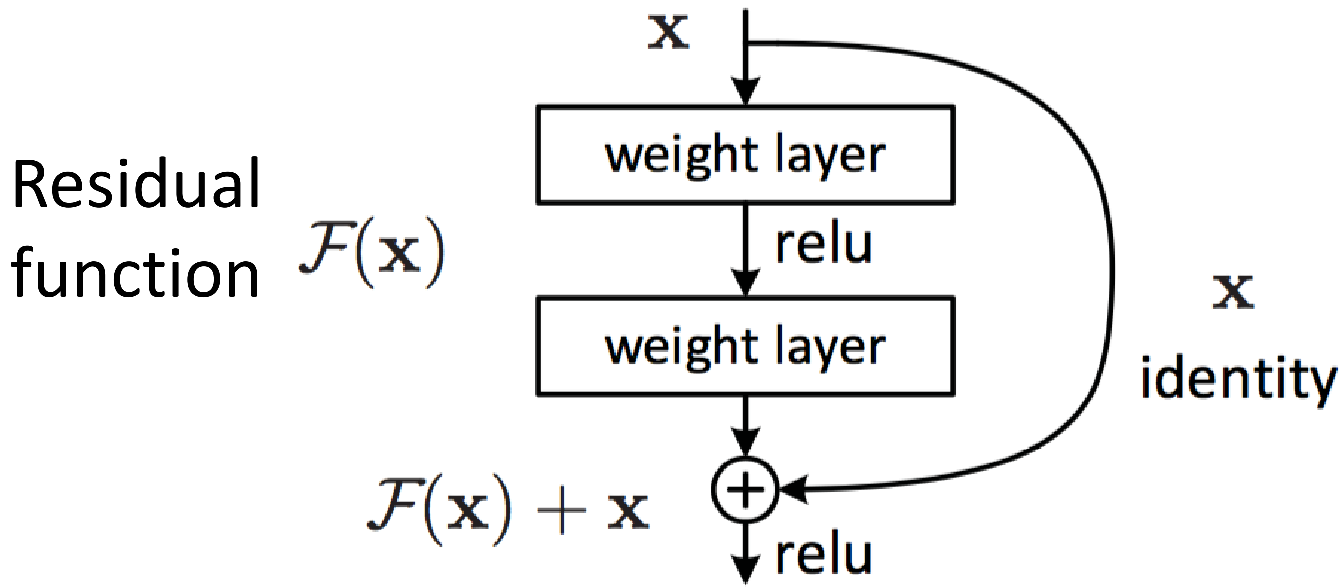
ResNet (2015)

- Residual Networks with 152 layers



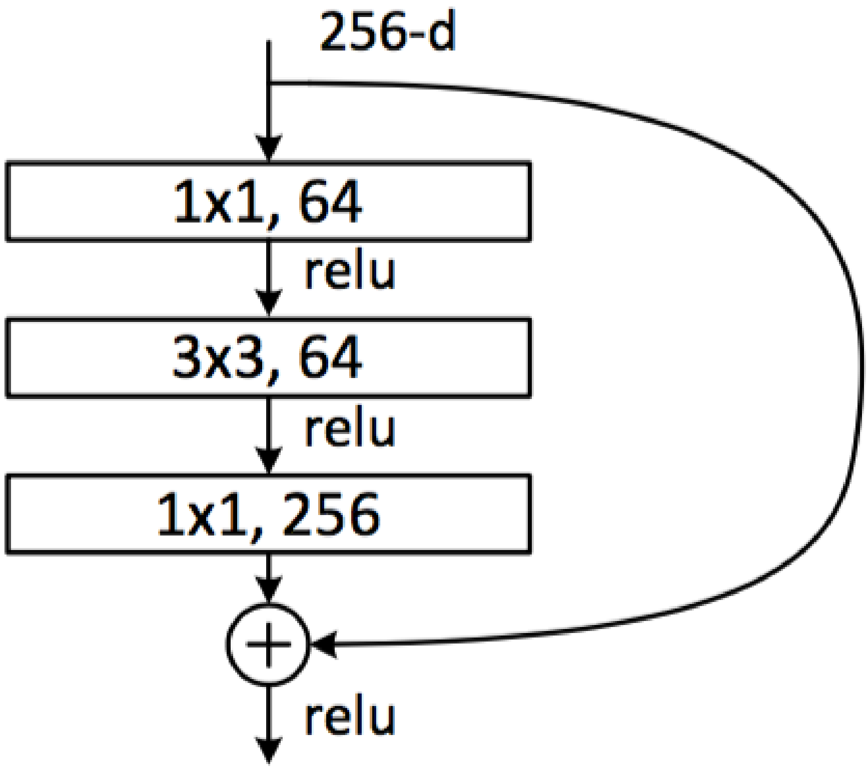
ResNet

- Residual learning: a building block

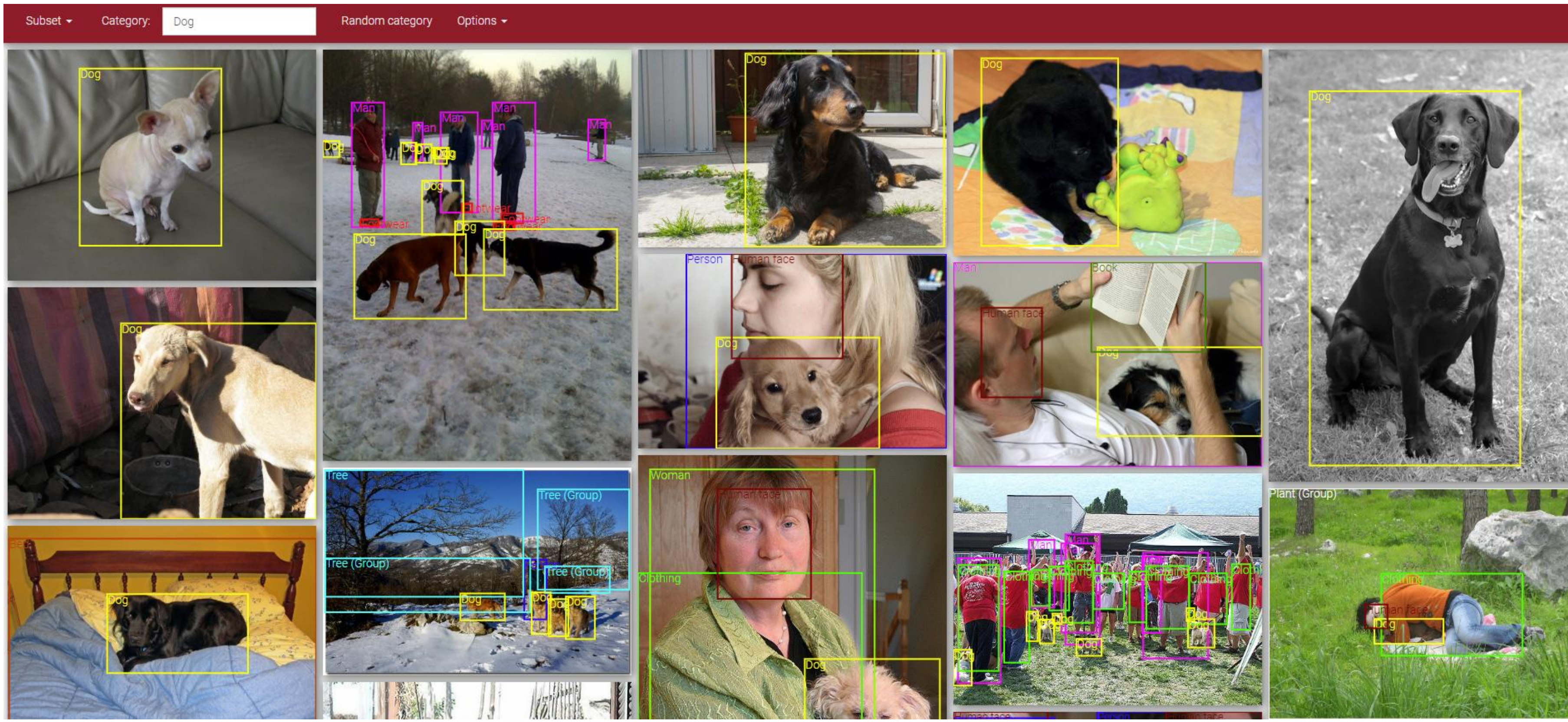


Residual Learning with Dimension Reduction

- using 1x1 filters



Open Images Extended - Crowdsourced



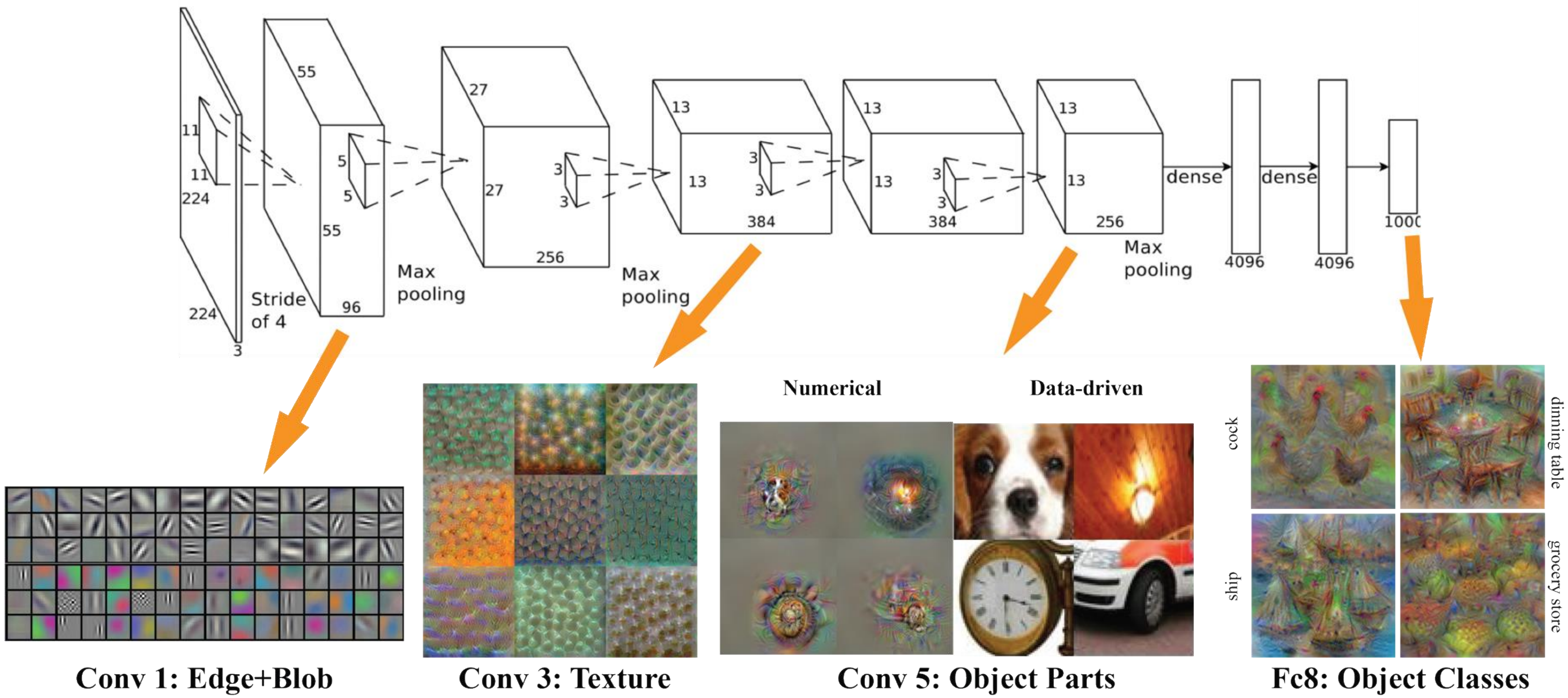
https://ai.google/tools/datasets/open-images-extended-crowdsourced/?fbclid=IwAR1K8mvT68l6JH8Ff8MtBmIAE0JpYX3d_VFiChzKeSECATng4yV1xdANGTg

Pretrained Model Download

- http://www.vlfeat.org/matconvnet/pretrained/
  - Alexnet:
    - http://www.vlfeat.org/matconvnet/models/imagenet-matconvnet-alex.mat
  - VGG19:
    - http://www.vlfeat.org/matconvnet/models/imagenet-vgg-verydeep-19.mat
  - GoogLeNet:
    - http://www.vlfeat.org/matconvnet/models/imagenet-googlenet-dag.mat
  - ResNet
    - http://www.vlfeat.org/matconvnet/models/imagenet-resnet-152-dag.mat

Using Pretrained Model

- Lower layers：edge, blob, texture (more general)
- Higher layers : object part (more specific)

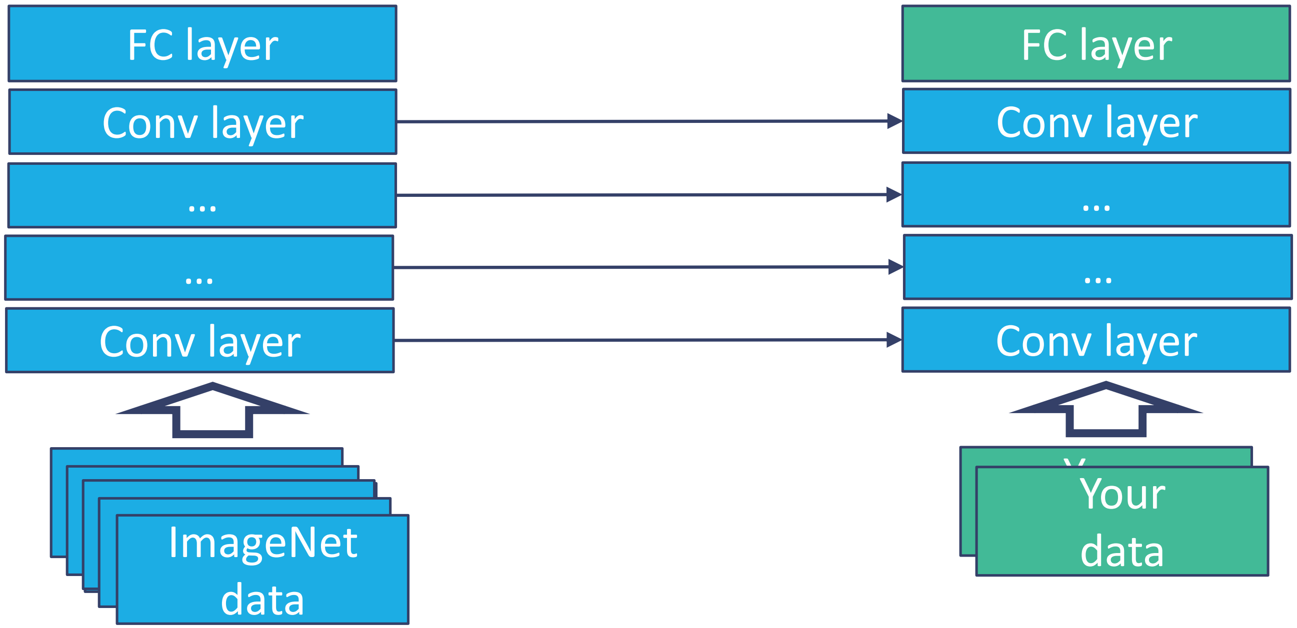


http://vision03.csail.mit.edu/cnn_art/data/single_layer.png

Transfer Learning

- The pretrained model is trained on ImageNet

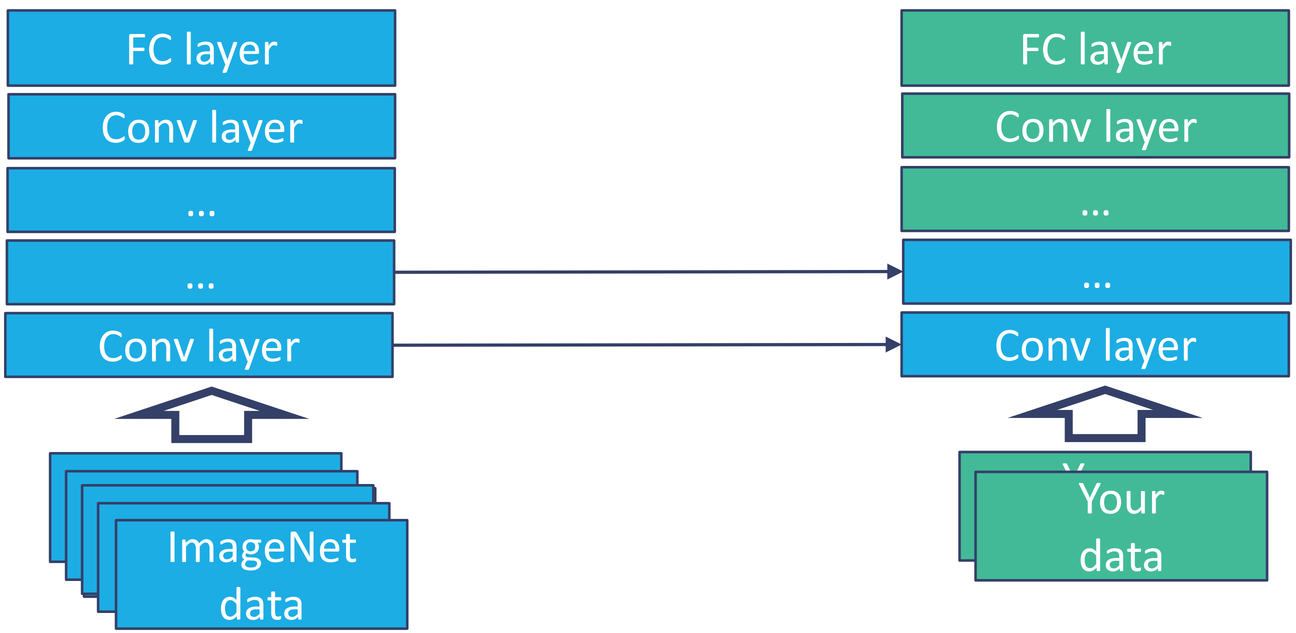
- If your data is similar to the ImageNet data
  - Fix all CNN Layers
  - Train FC layer



Transfer Learning

- The pretrained model is trained on ImageNet

- If your data is far different from the ImageNet data
  - Fix lower CNN Layers
  - Train higher CNN and FC layers



Transfer Learning Example

daisy
634
photos

dandelion
899
photos

roses
642
photos

tulips
800
photos

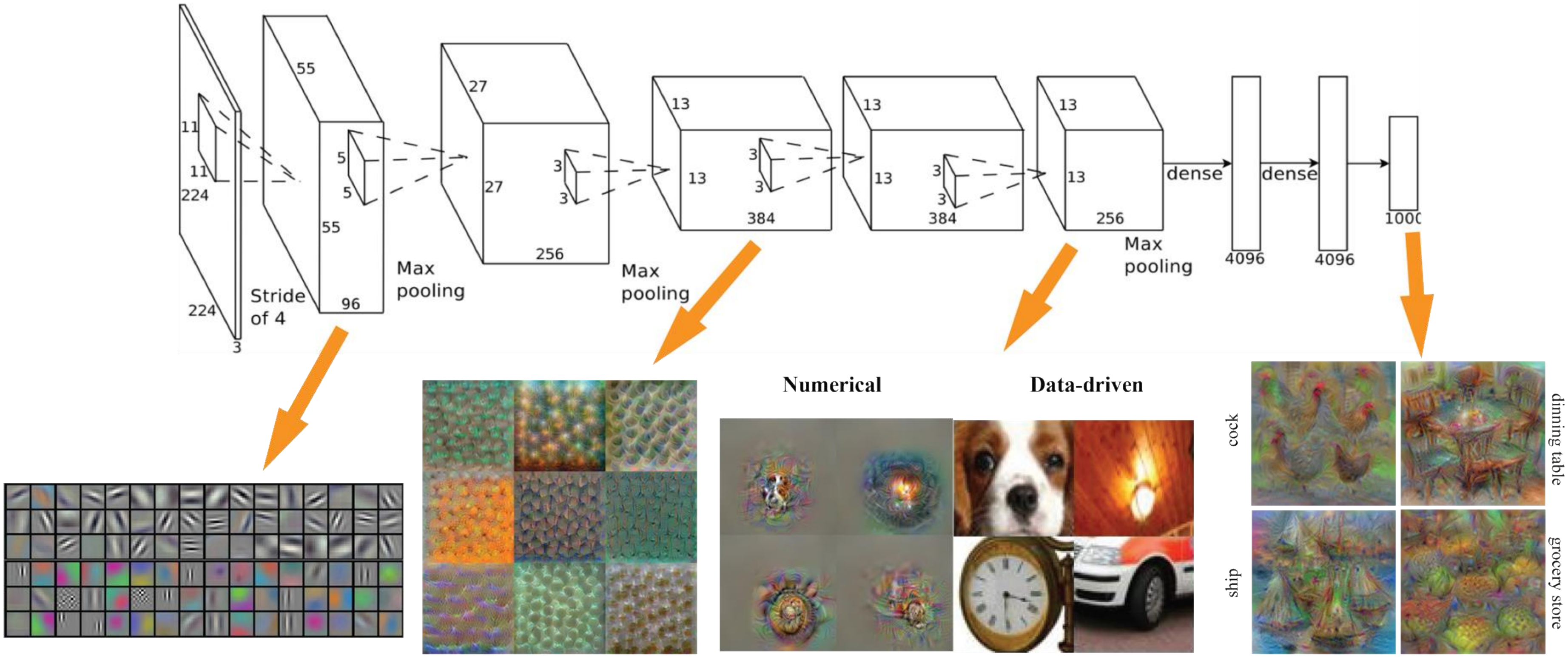
sunflowers
700
photos

http://download.tensorflow.org/example_images/flower_photos.tgz

Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- More Applications

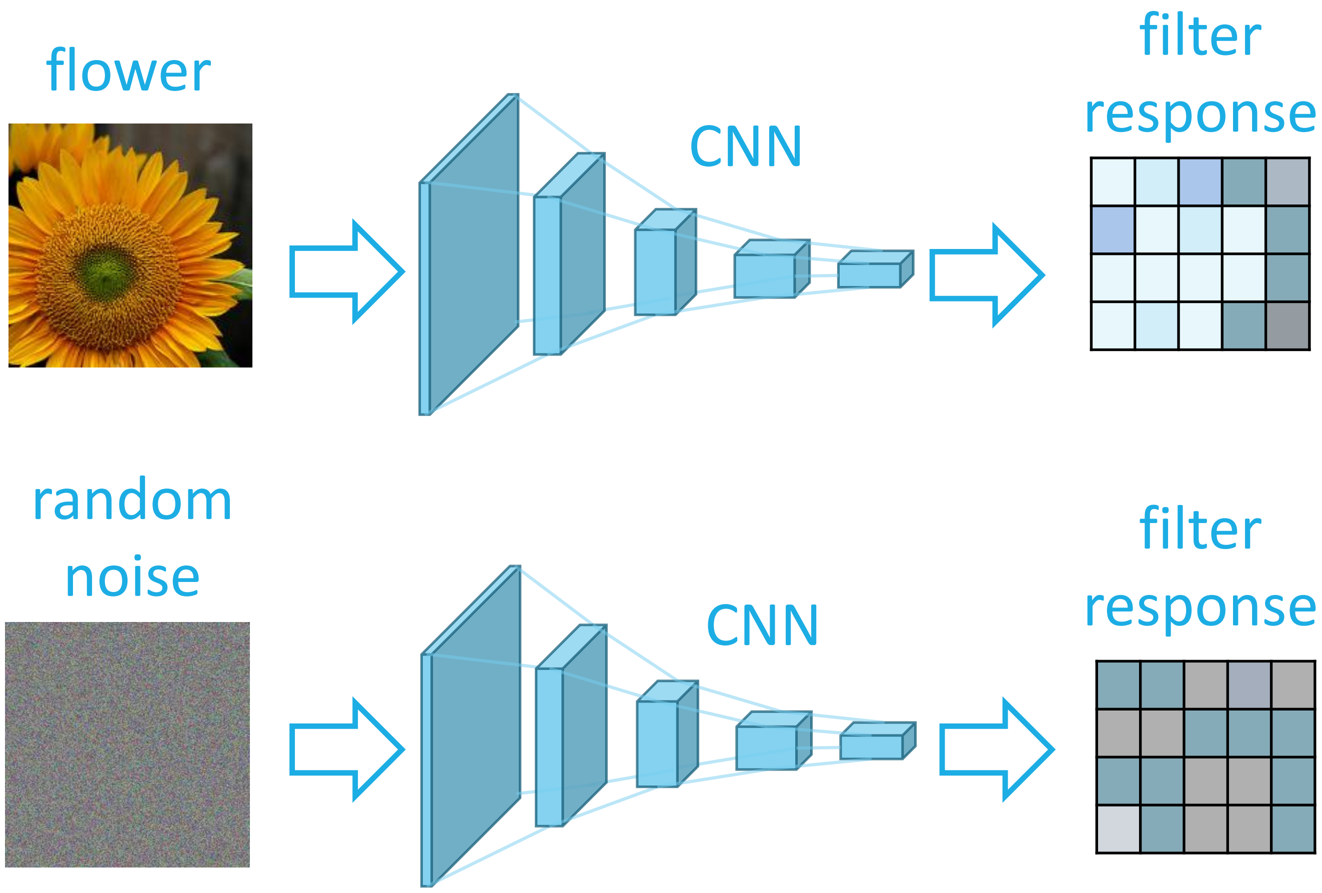
Visualizing CNN



Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes

http://vision03.csail.mit.edu/cnn_art/data/single_layer.png

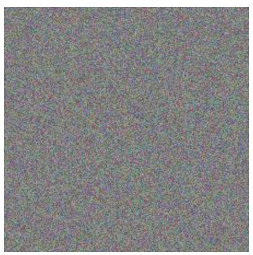
Visualizing CNN



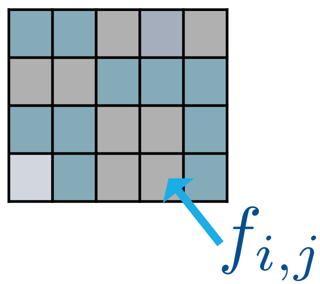
Gradient Ascent

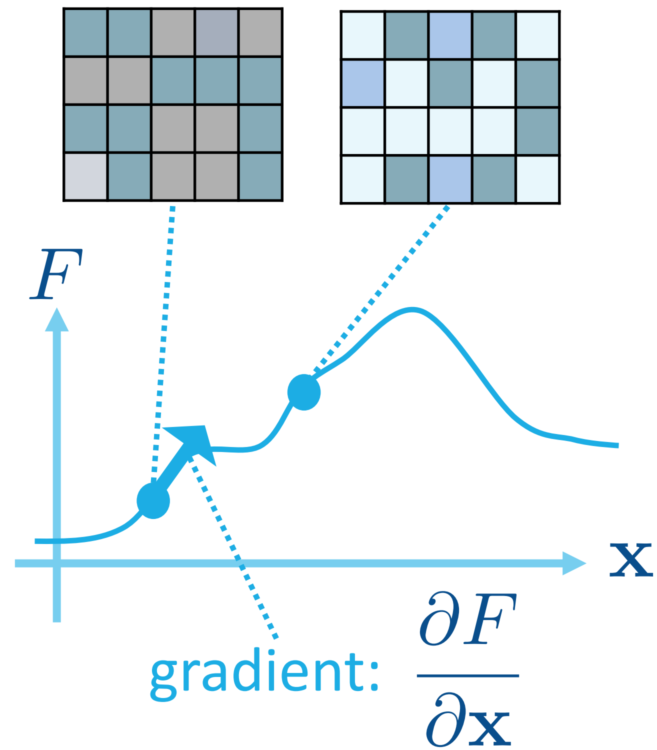
- Magnify the filter response

random noise: $\mathbf{x}$



filter response: $\mathbf{f}$



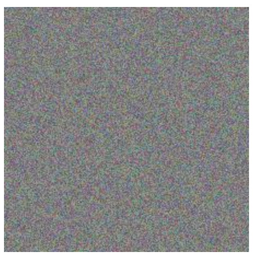


score: $$F = \sum_{i,j} f_{i,j}$$

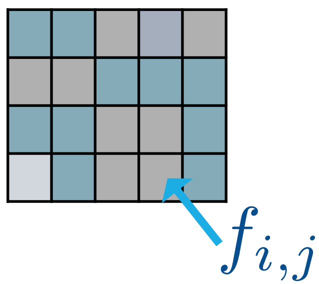
Gradient Ascent

- Magnify the filter response

random noise: $\mathbf{x}$

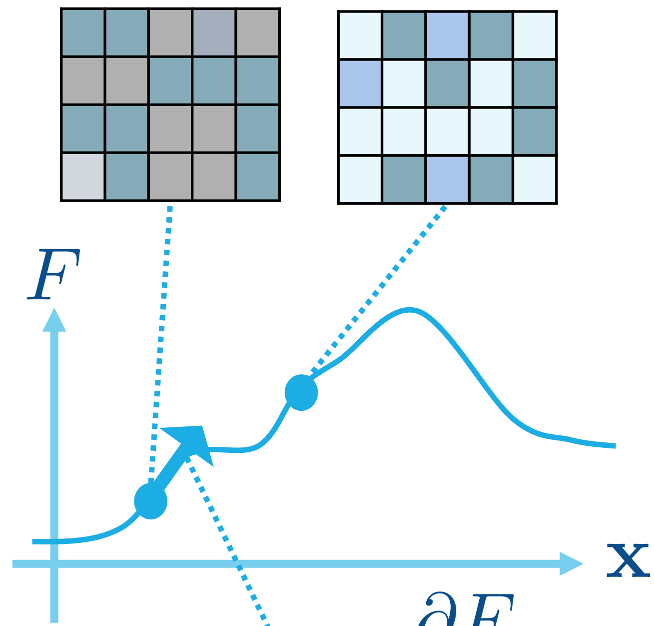


filter response: $\mathbf{f}$



lower score

higher score



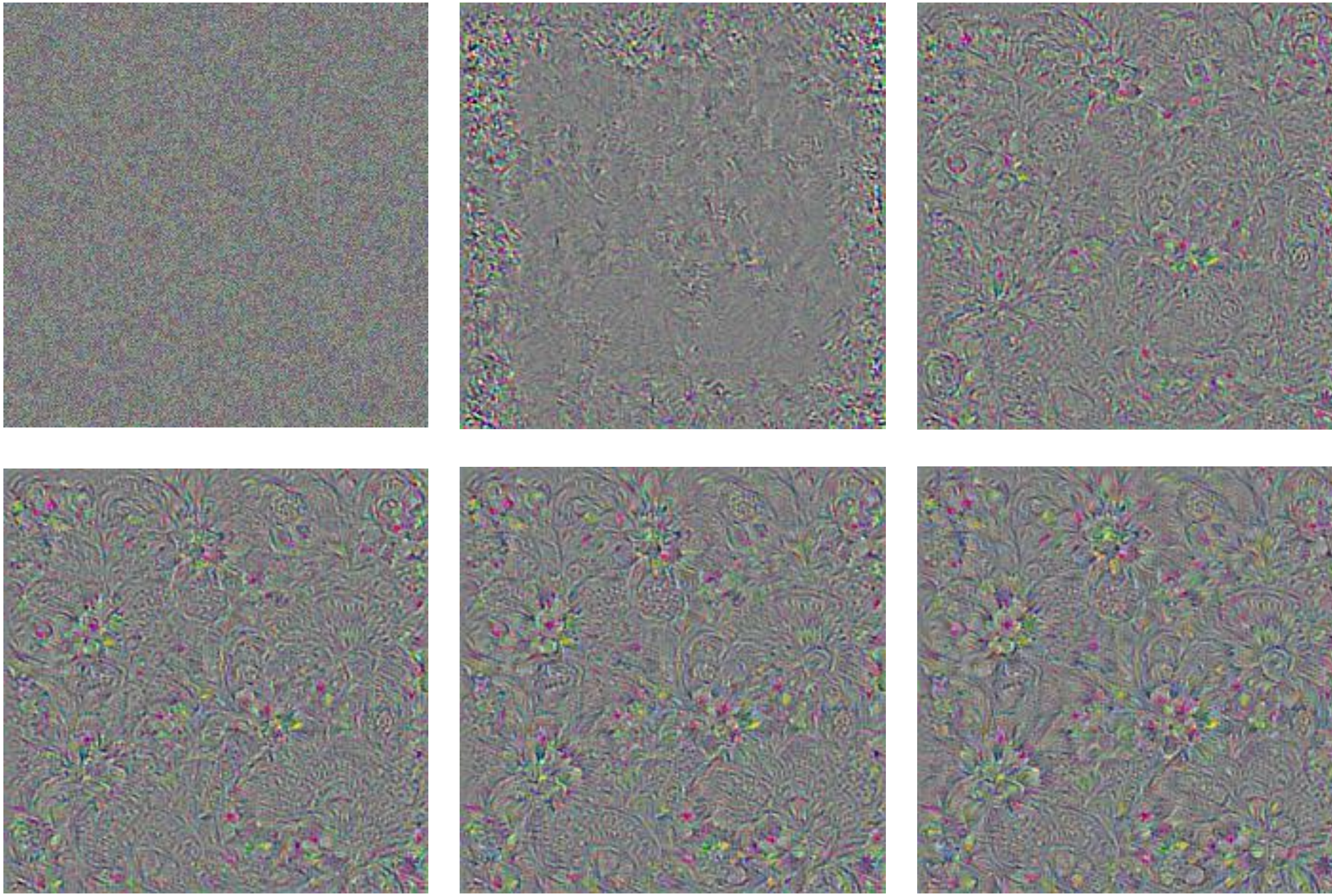
gradient: $\frac{\partial F}{\partial \mathbf{x}}$

update $\mathbf{x}$

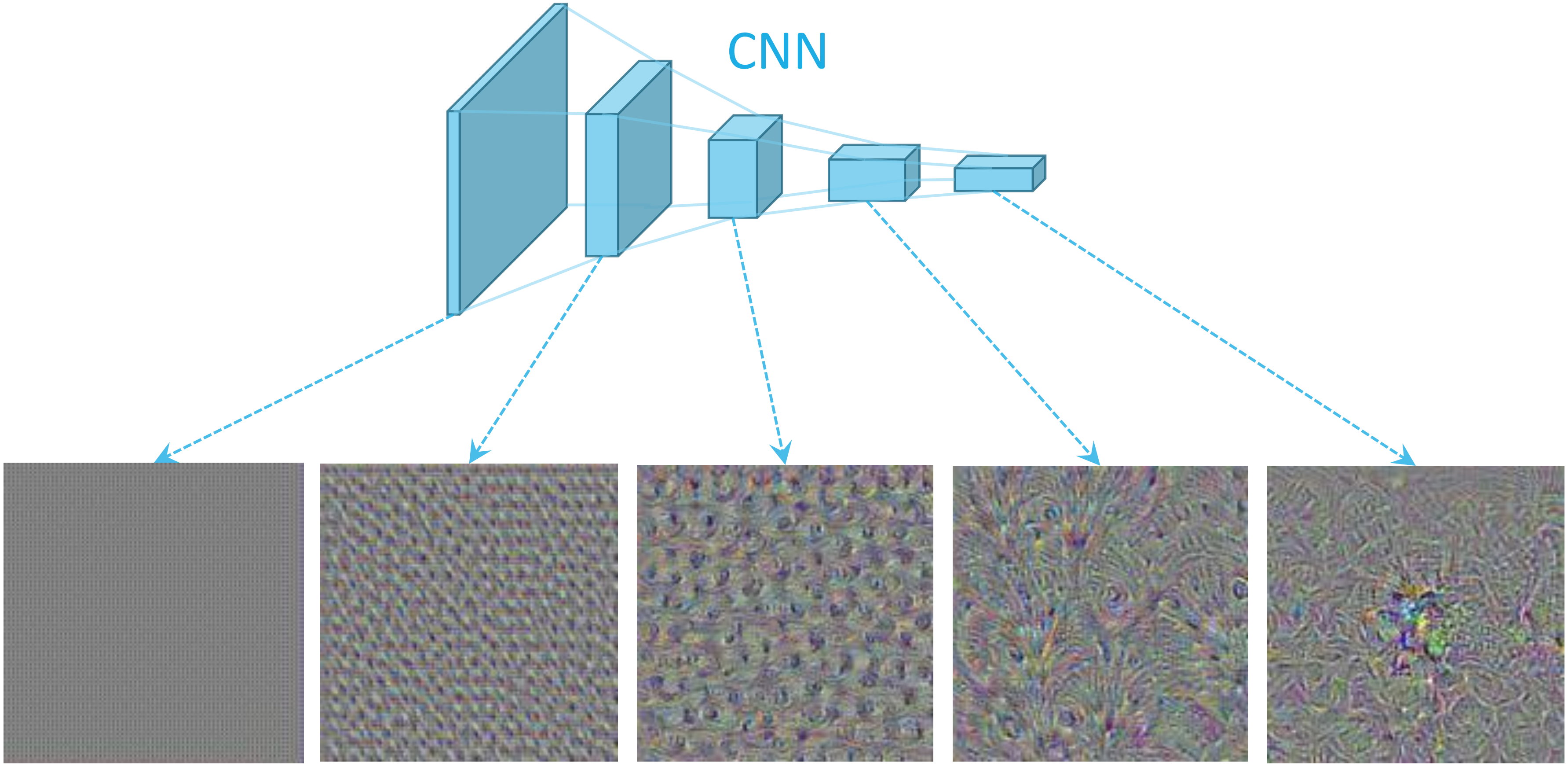
$$\mathbf{x} \leftarrow \mathbf{x} + \eta \frac{\partial F}{\partial \mathbf{x}}$$

learning rate

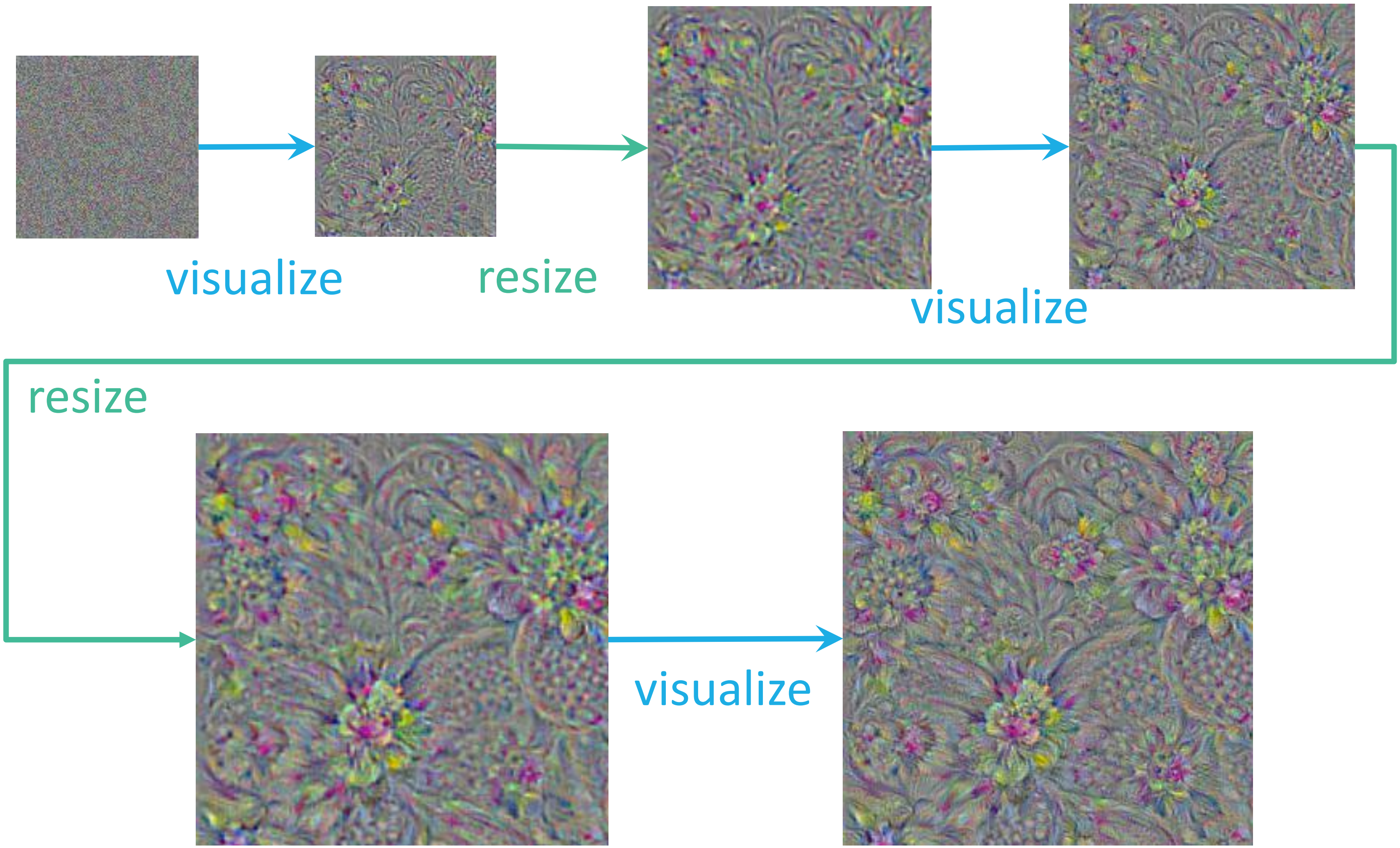
Gradient Ascent



Different Layers of Visualization



Multiscale Image Generation



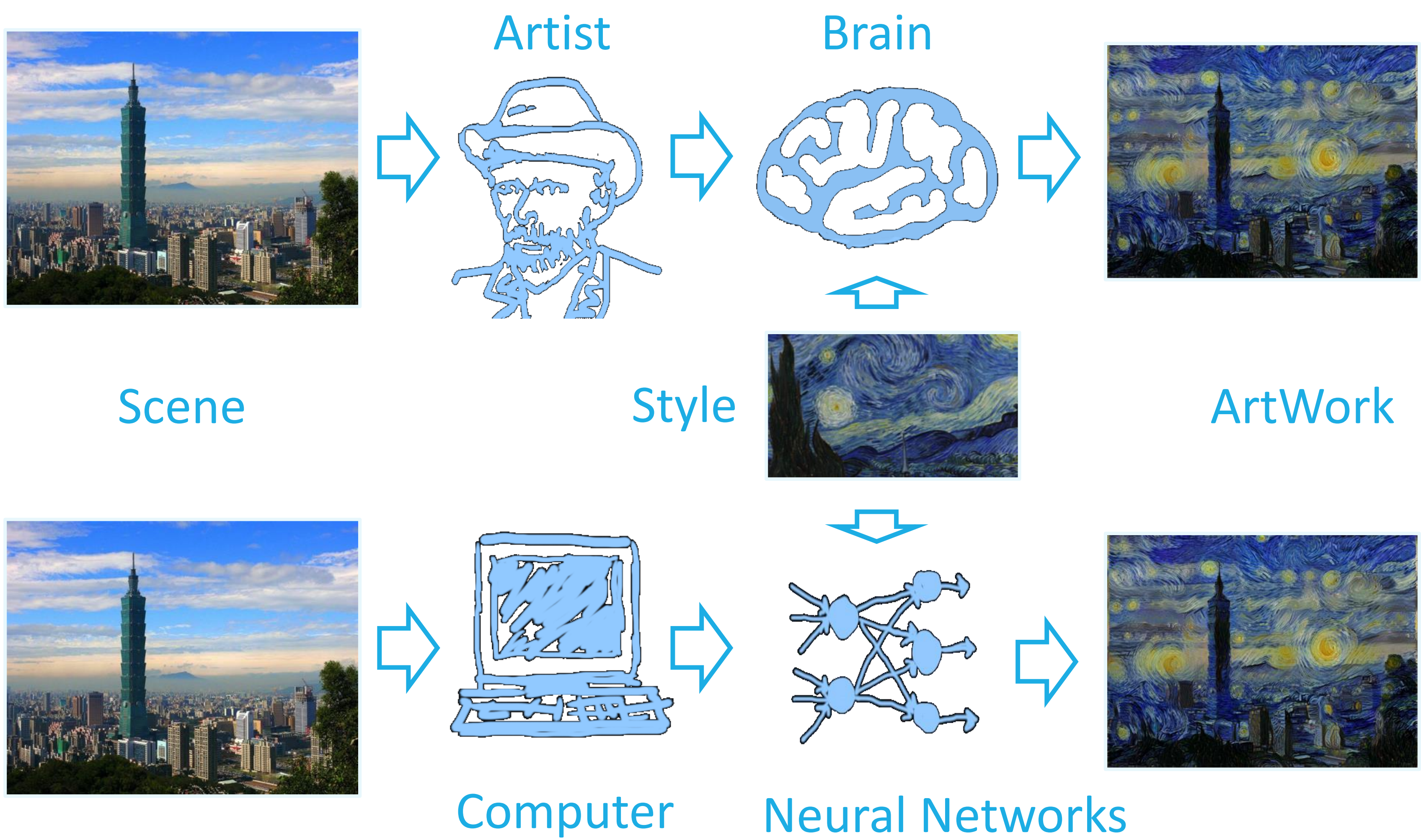
Deep Dream

- Given a photo, machine adds what it sees ……

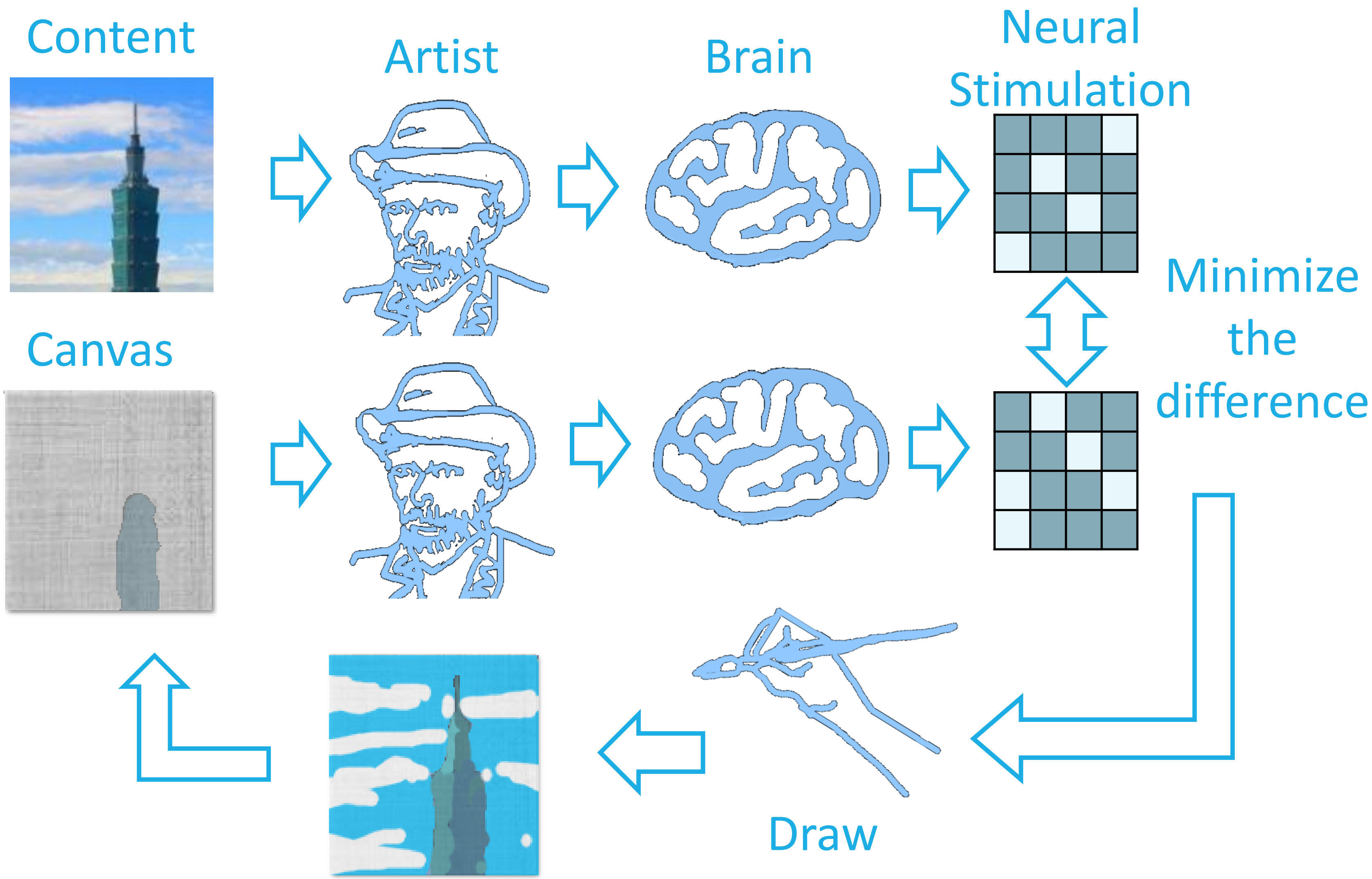
Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- More Applications

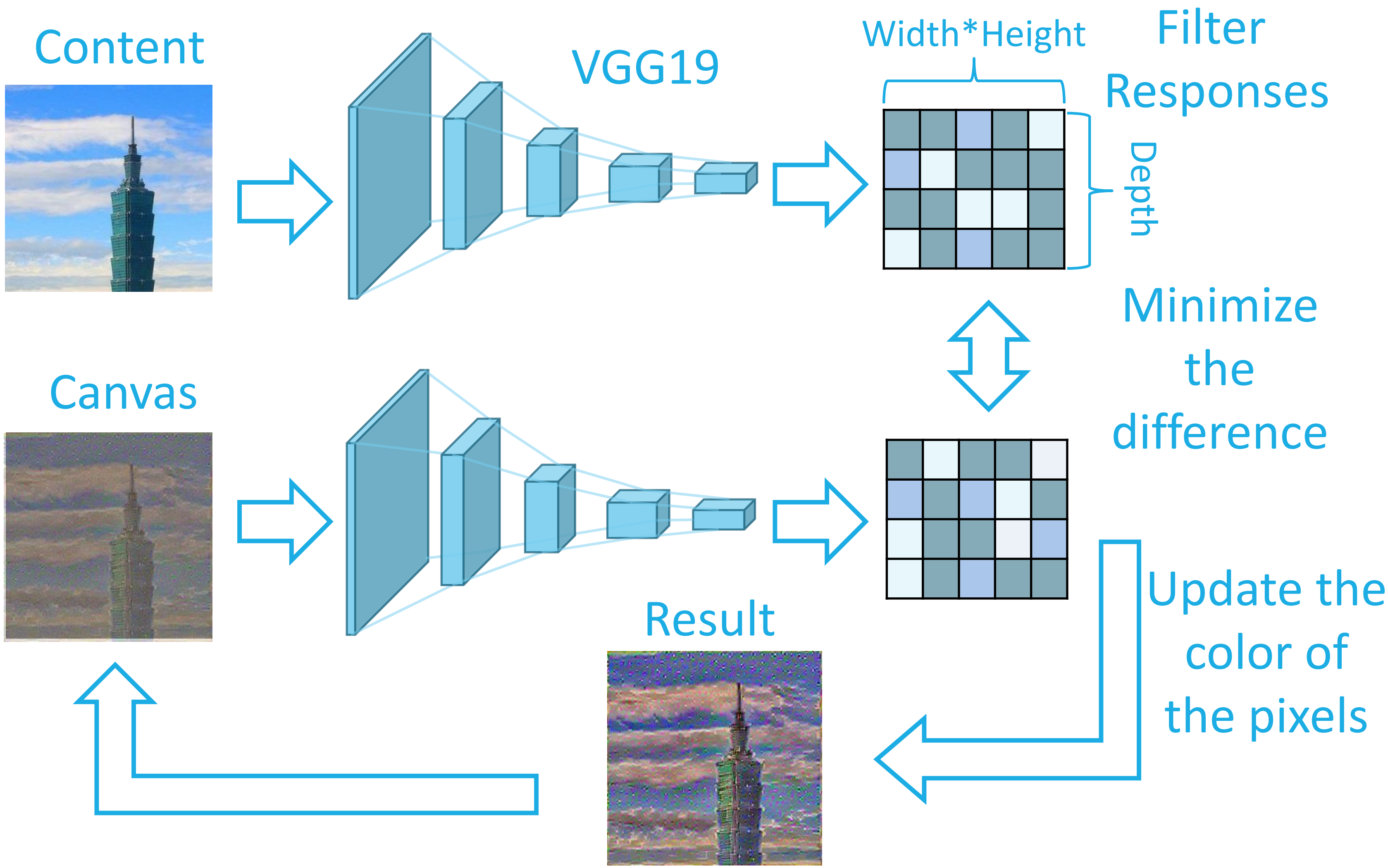
The Mechanism of Painting



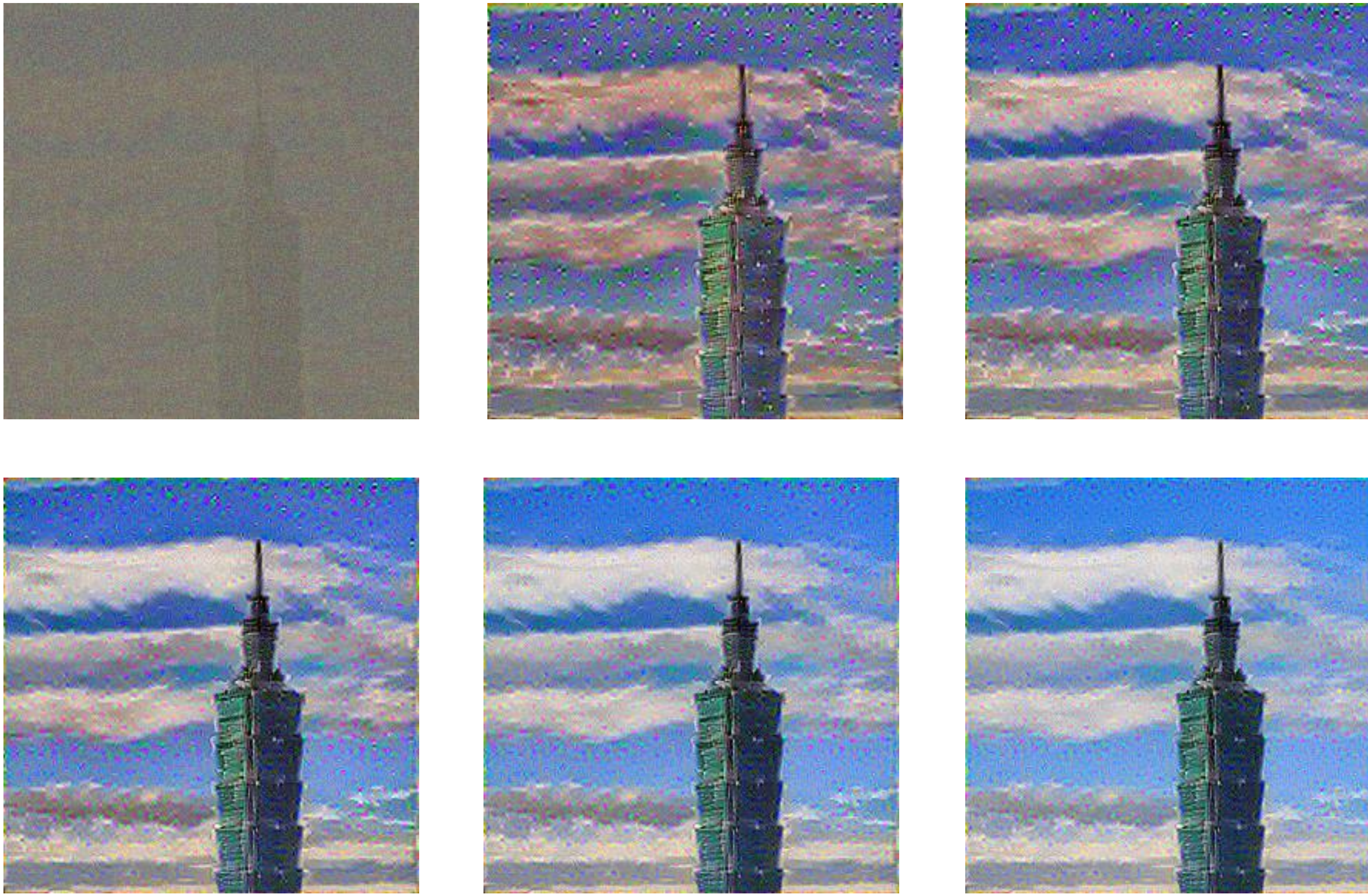
Content Generation



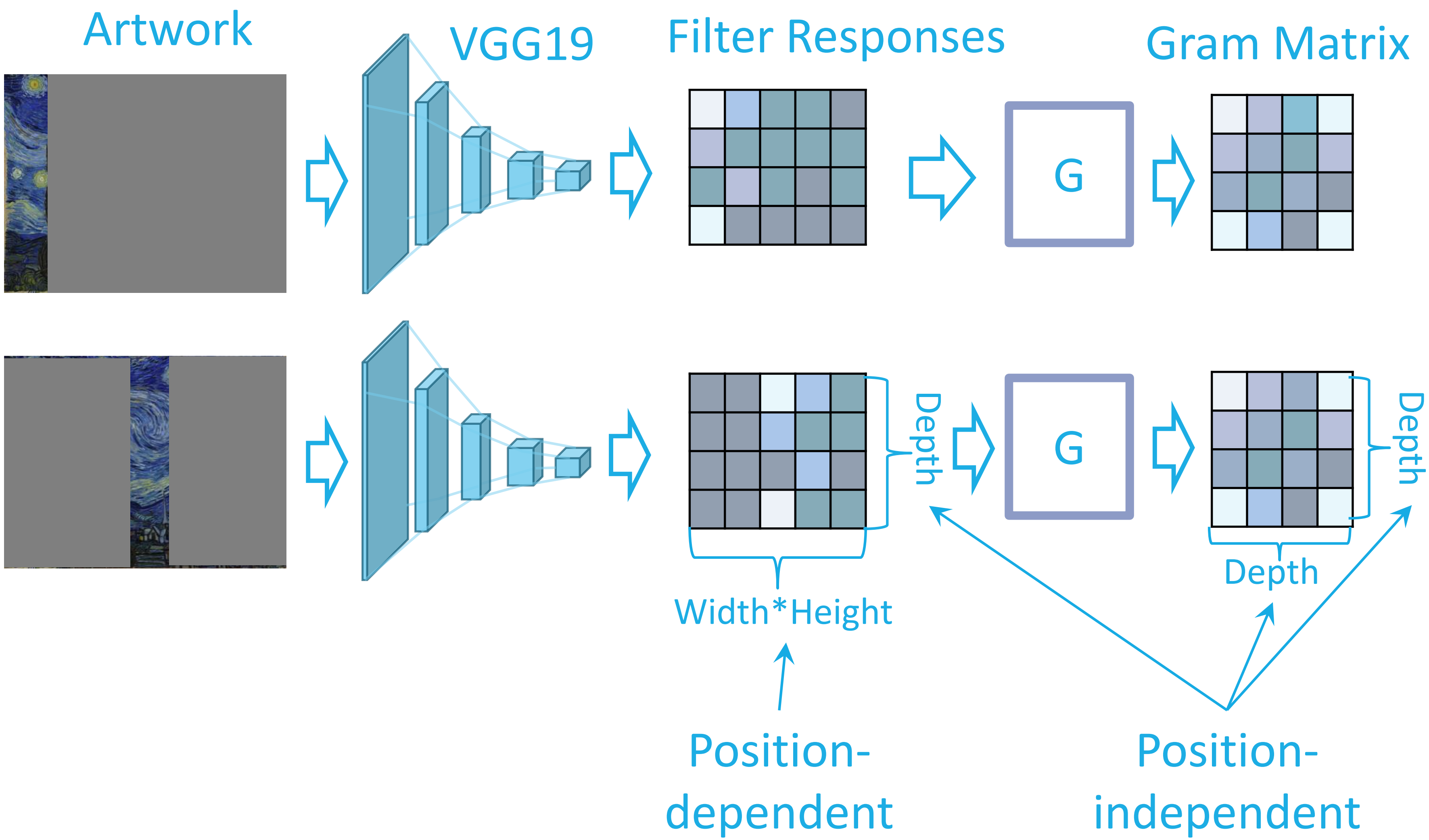
Content Generation



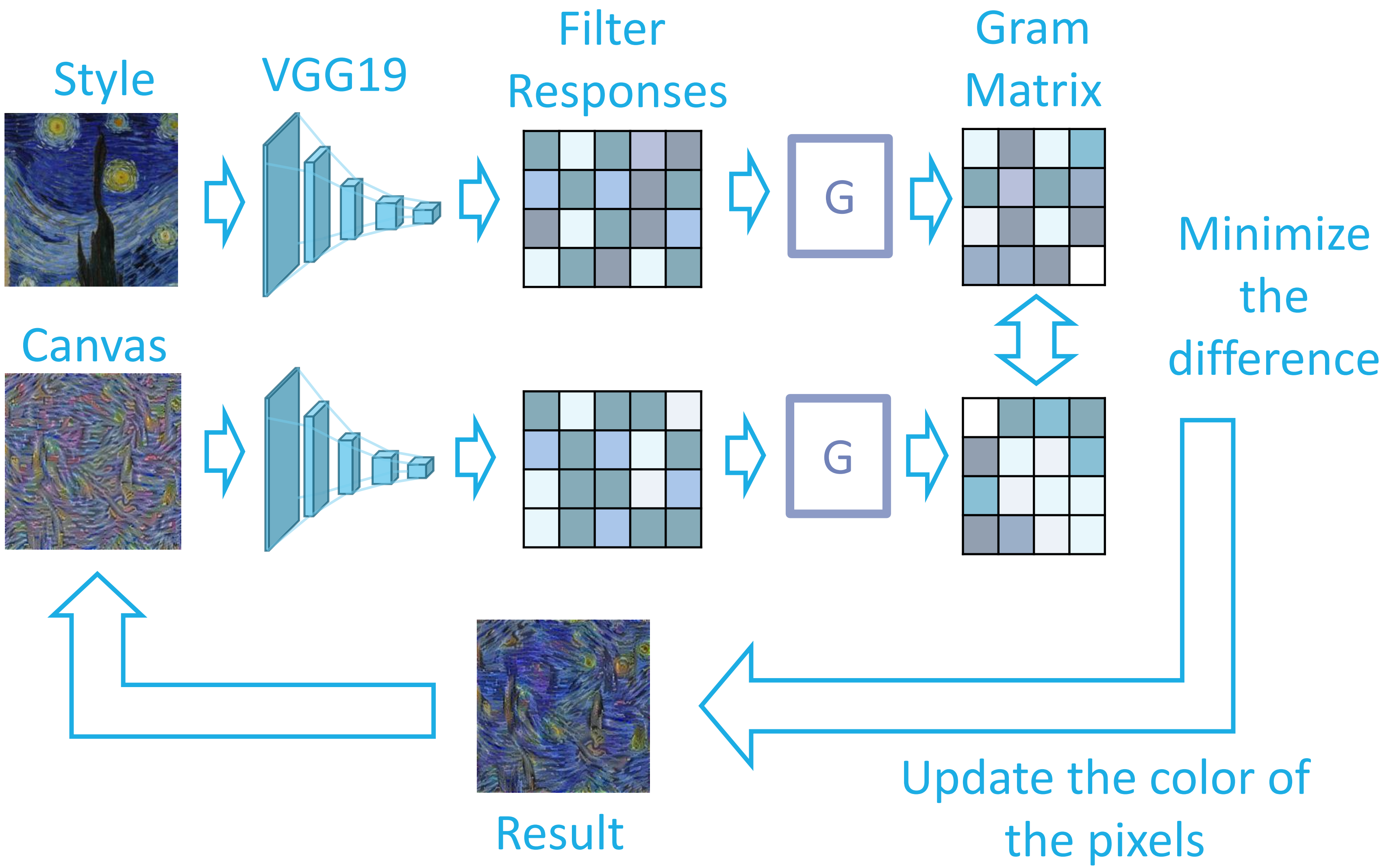
Content Generation



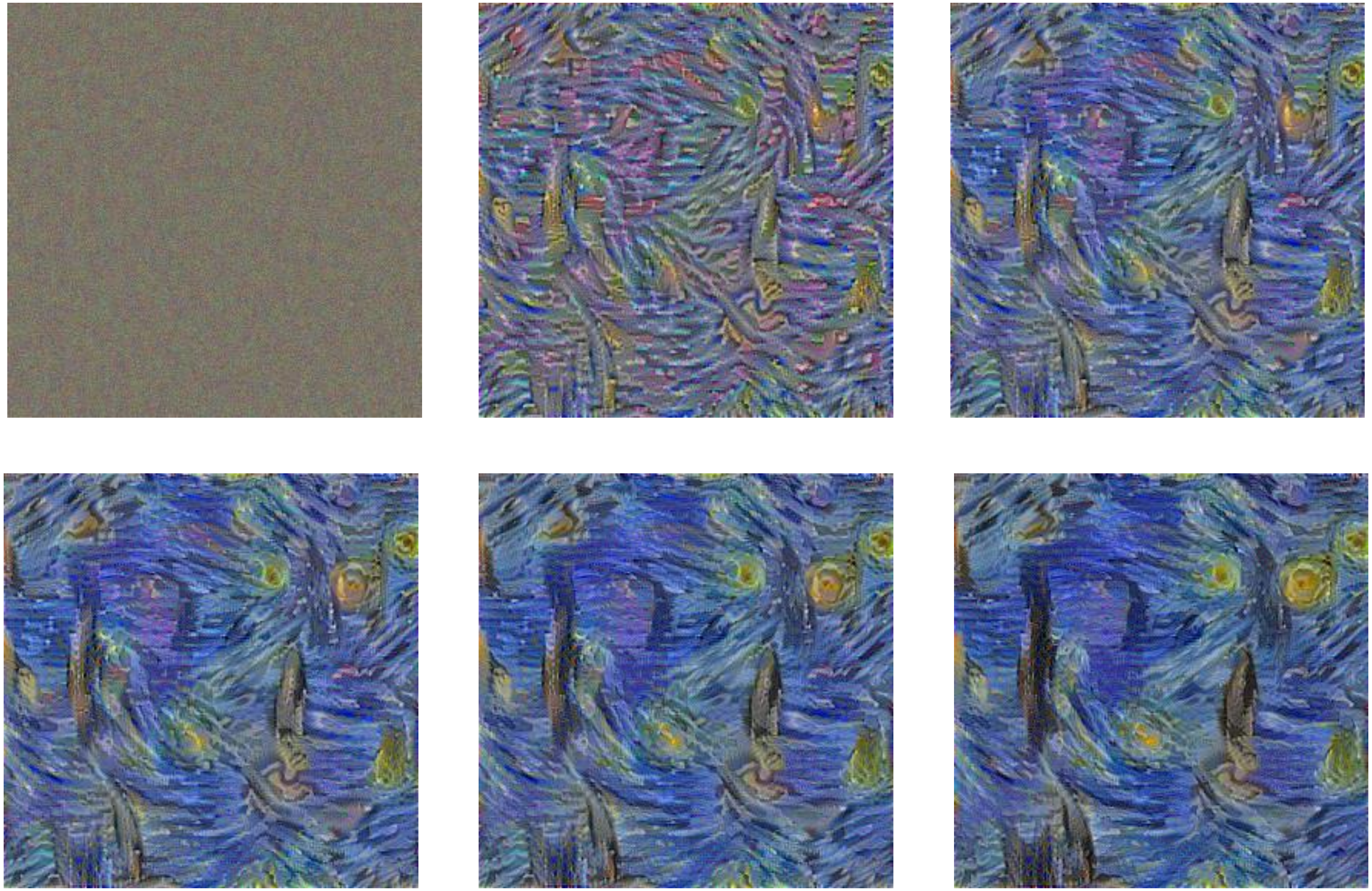
Style Generation



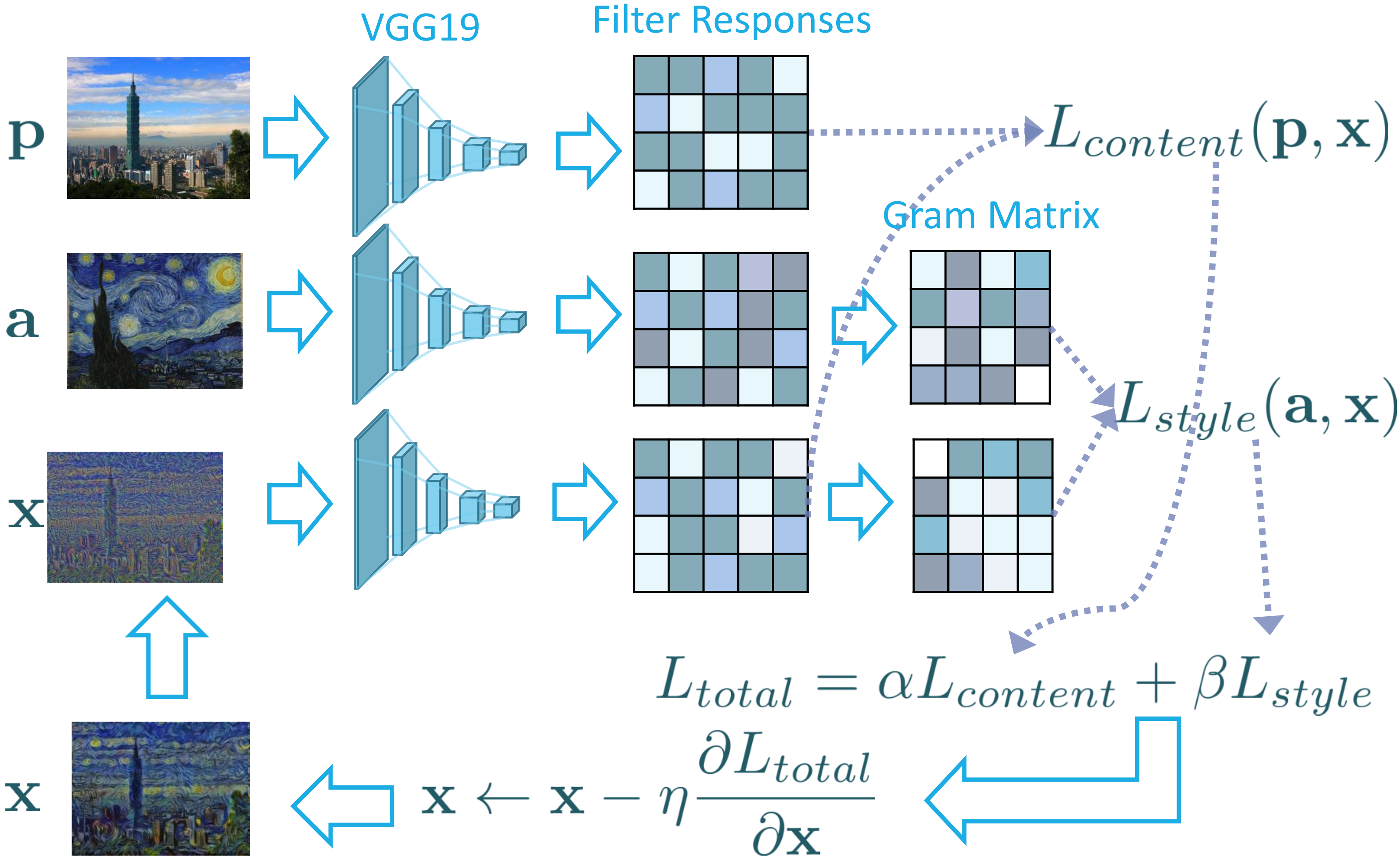
Style Generation



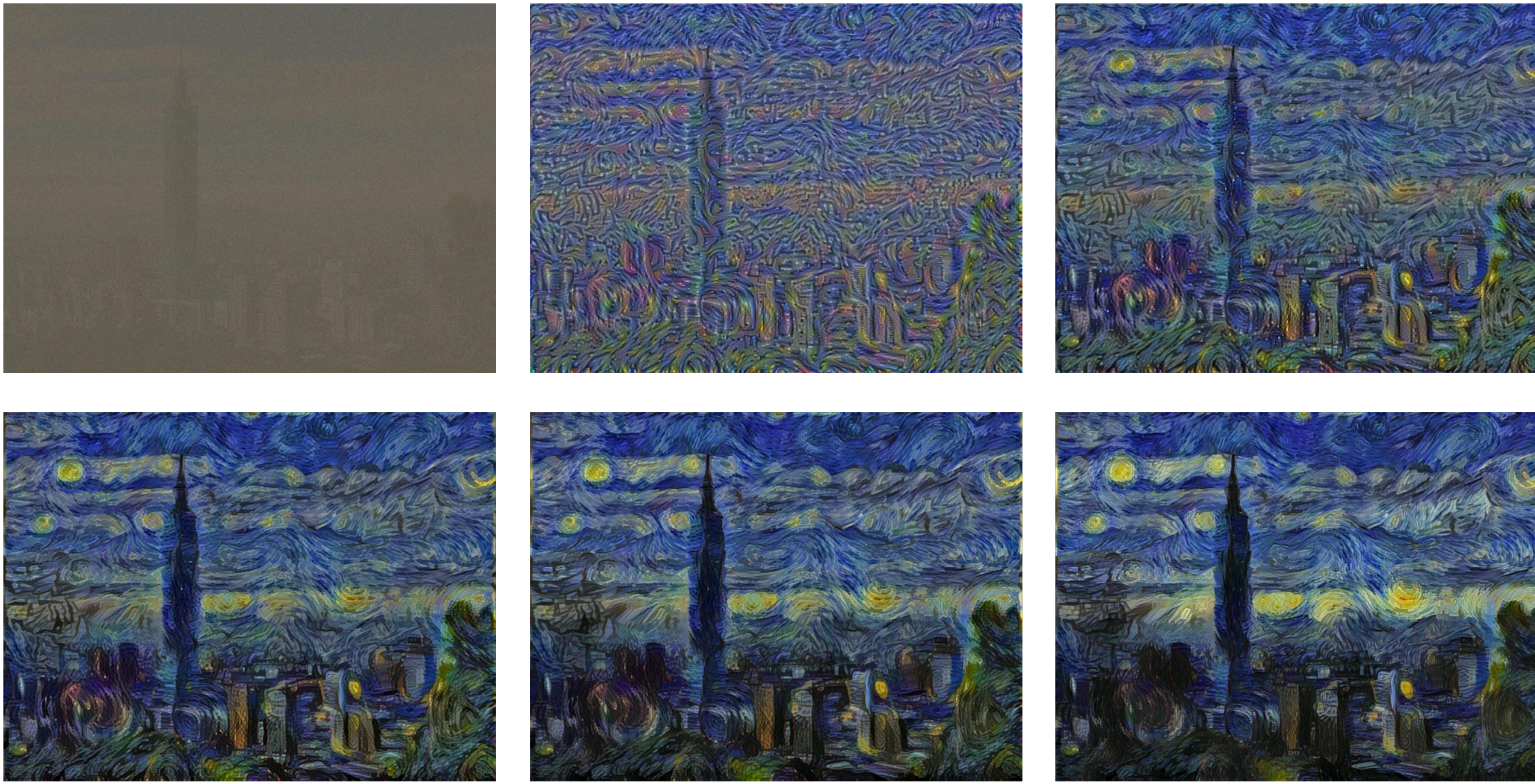
Style Generation



Artwork Generation

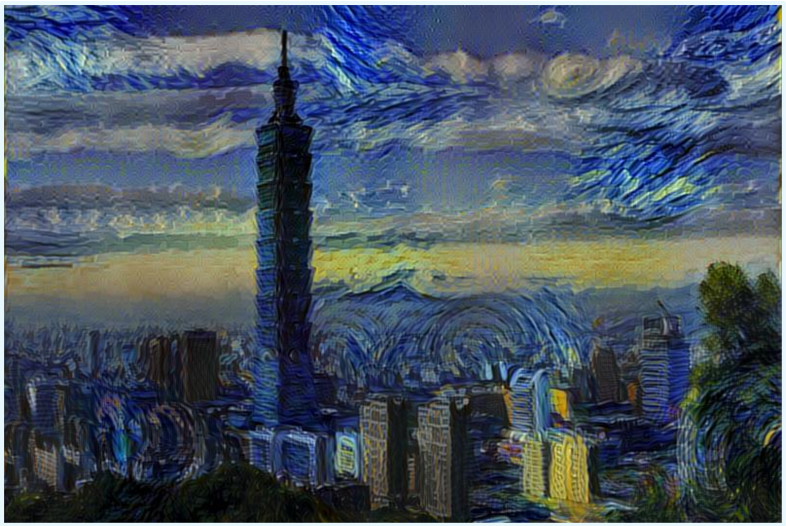


Artwork Generation

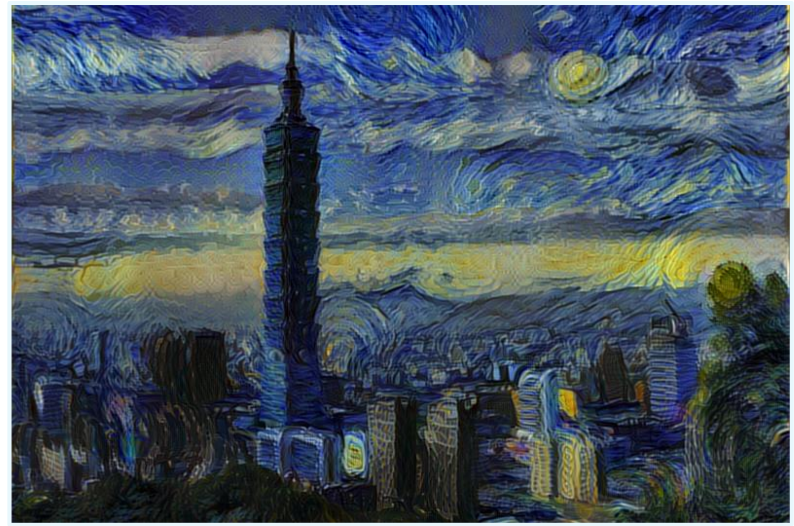


Content v.s. Style

$\frac{\alpha}{\beta}$



0.15



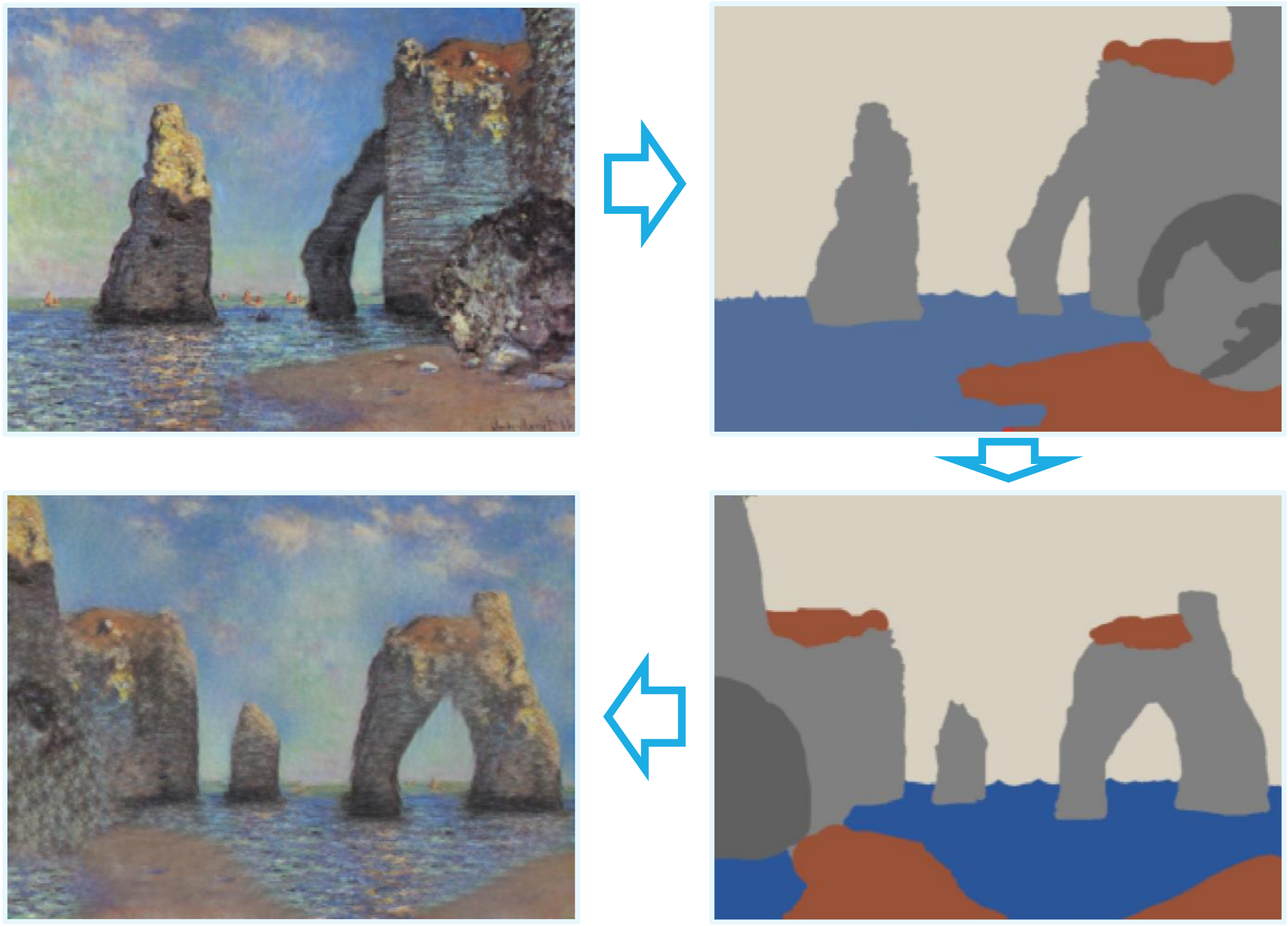
0.05

0.02

0.007

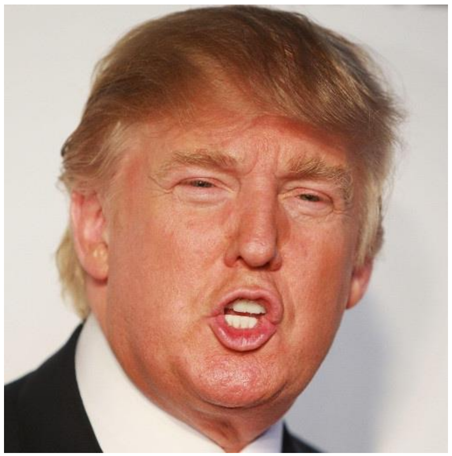
Neural Doodle

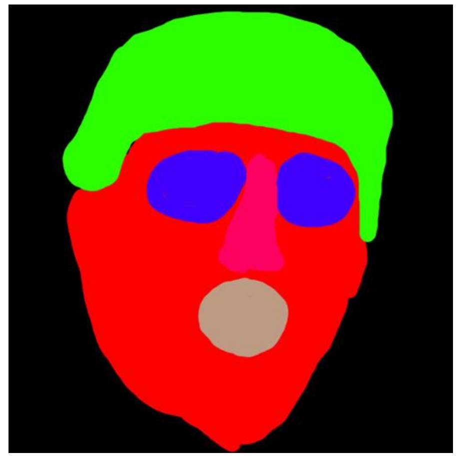
- Image analogy

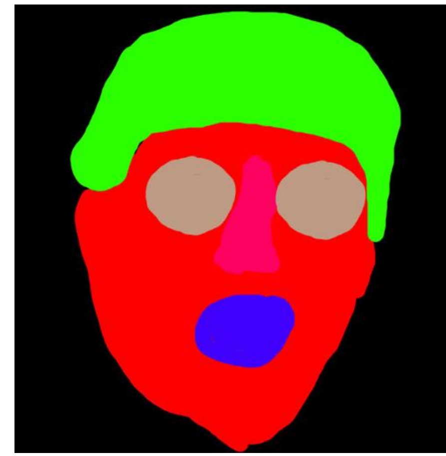


Neural Doodle

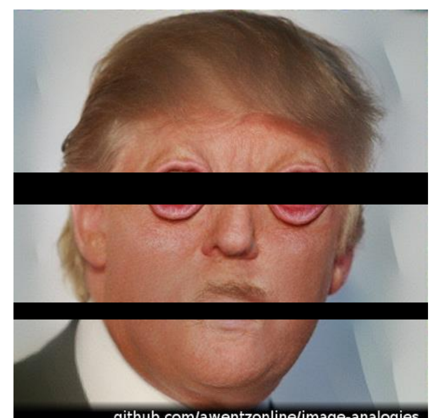
- Image analogy



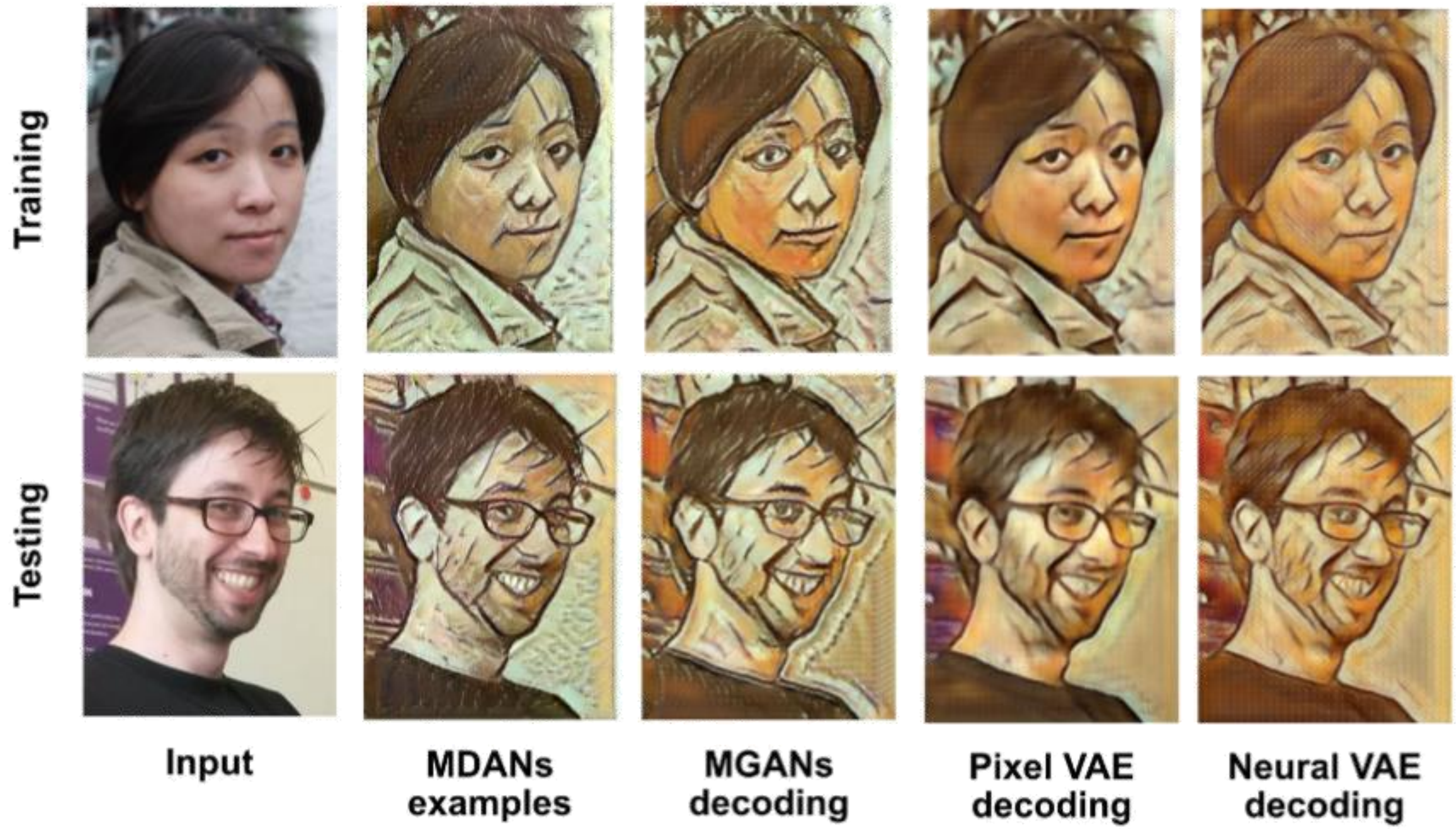




恐怖連結，慎入！
https://raw.githubusercontent.com/awentzonline/image-analogies/master/examples/images/trump-image-analogy.jpg



Real-time Texture Synthesis



https://arxiv.org/pdf/1604.04382v1.pdf

Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- More Applications

More Application: Playing Go

Training: record of previous plays 黑: 5之五 → 白: 天元 → 黑: 五之5 ...

CNN

Target:
"天元" = 1
else = 0

CNN

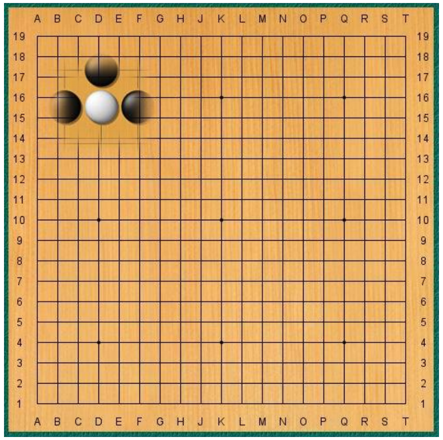
Target:
"五之 5" = 1
else = 0

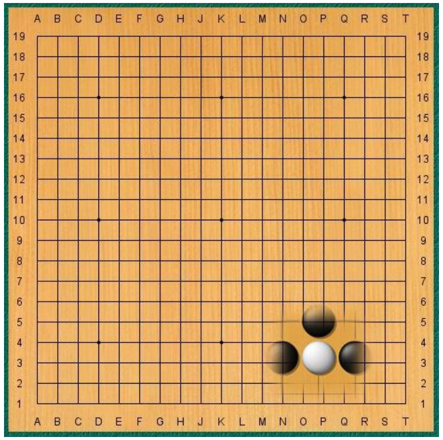
Why CNN for Go?

- Some patterns are much smaller than the whole image

- The same patterns appear in different regions.

AlphaGo uses 5 x 5 for first layer





Why CNN for Go?

- Subsampling the pixels will not change the object

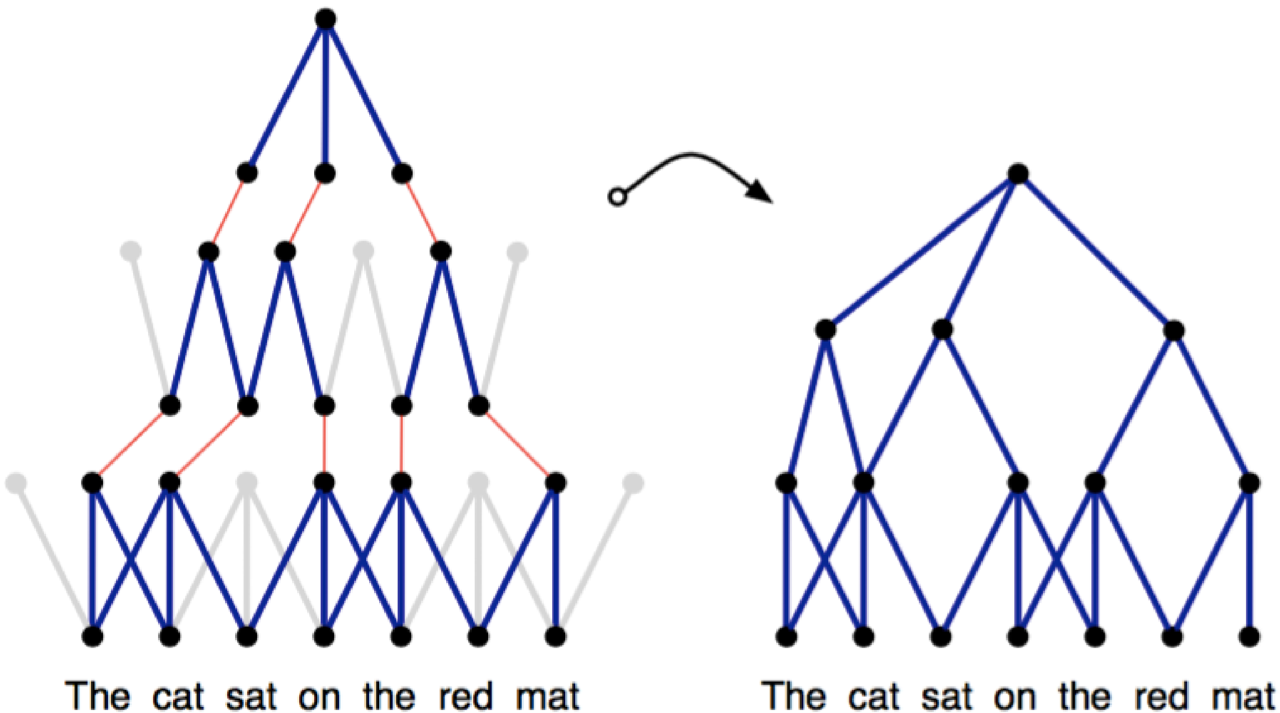
Max Pooling

How to explain this???

Neural network architecture. The input to the policy network is a 19 × 19 × 48 image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with stride 1, with a different bias for each position, and applies a softmax function. The [illegible] Extended Data Table 3 additionally show the results of training with *k* = 128, 256 and 384 filters.

Alpha Go does not use Max Pooling ……

More Applications: Sentence Encoding



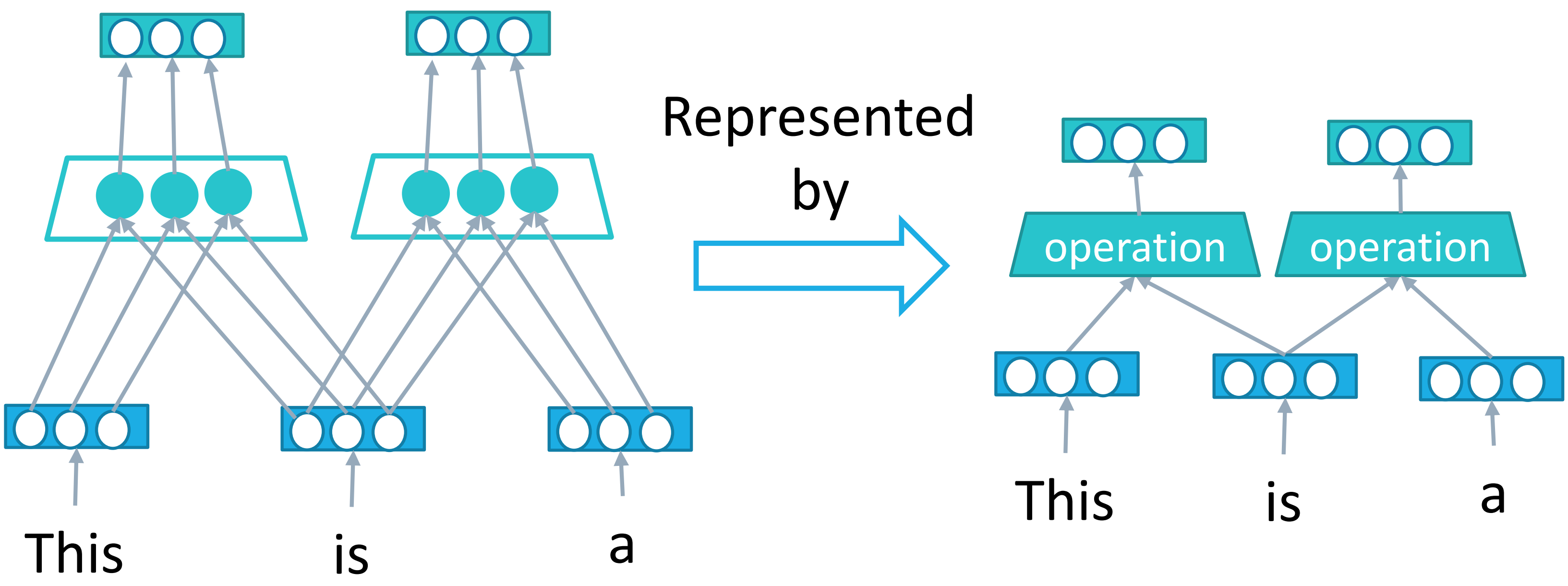
Ambiguity in Natural Language

http://3rd.mafengwo.cn/travels/info_weibo.php?id=2861280

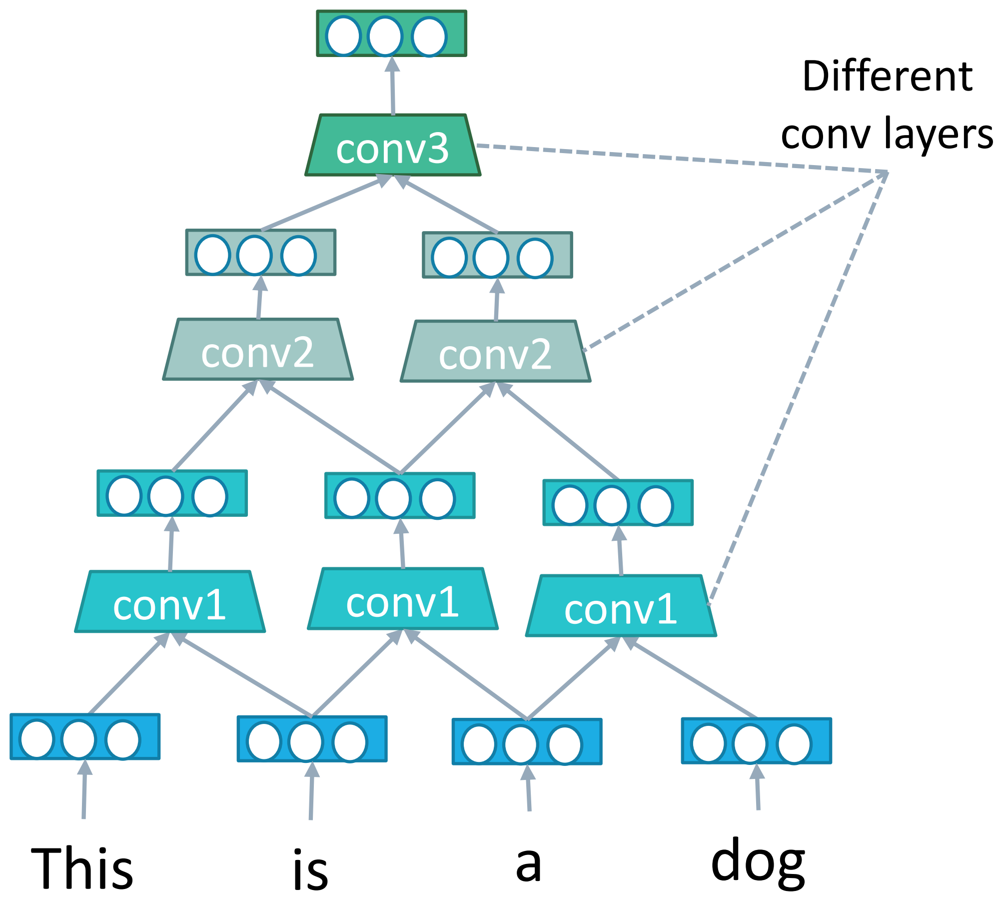
http://www.appledaily.com.tw/realtimenews/article/new/20151006/705309/

Element-wise 1D Operations on Word Vectors

- 1D Convolution or 1D Pooling

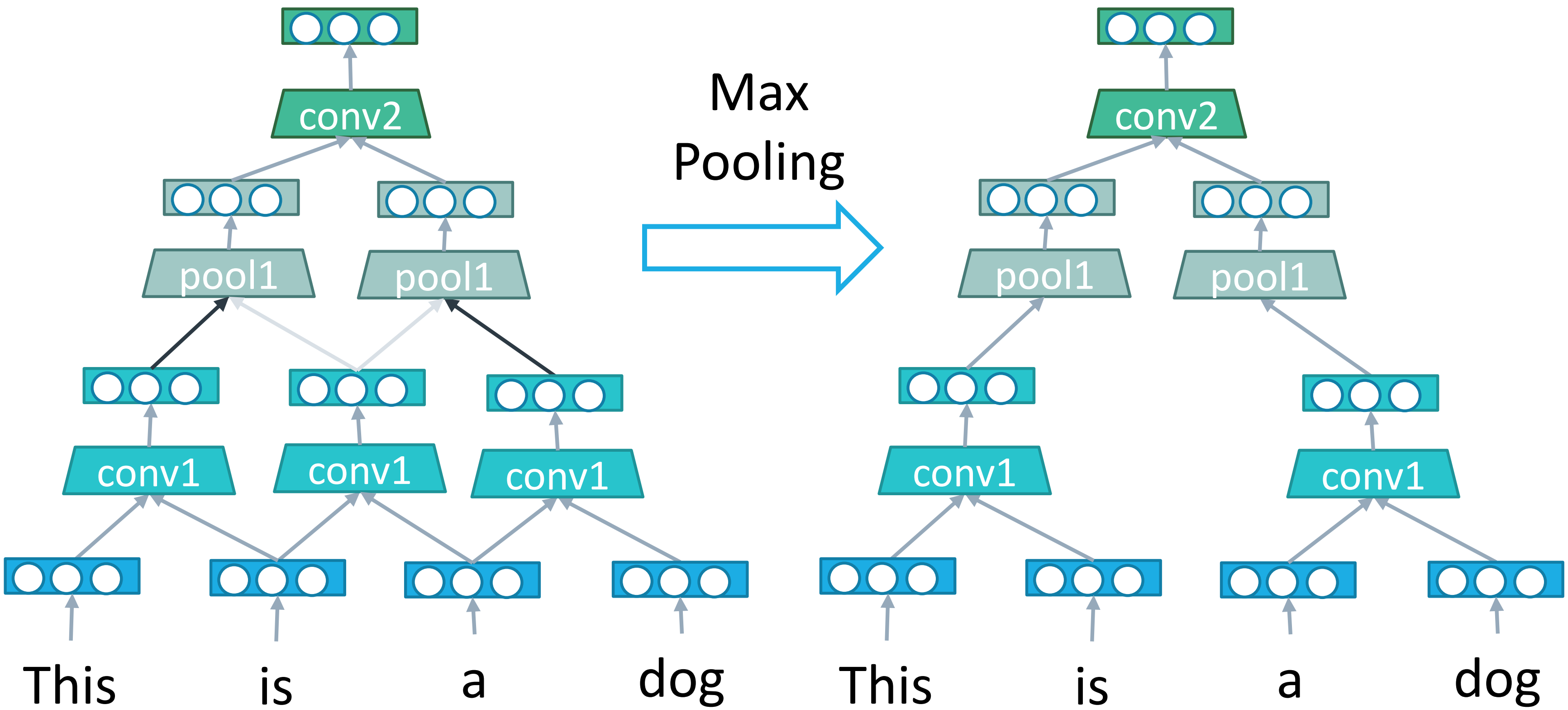


CNN Model


Different
conv layers
conv3
conv2
conv2
conv1
conv1
conv1
This
is
a
dog

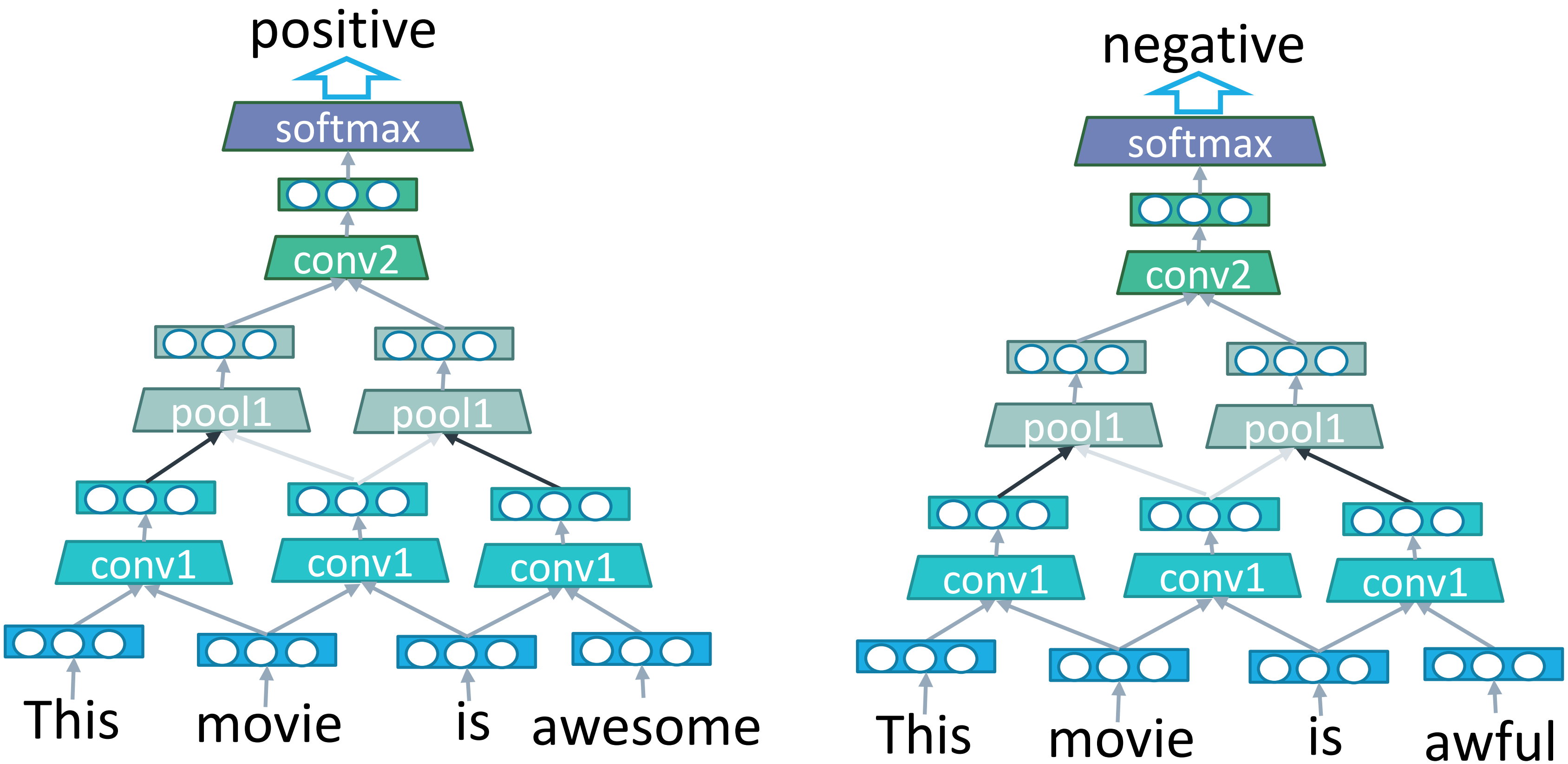
CNN with Max-Pooling Layers

- Similar to syntax tree
- But human-labeled syntax tree is not needed



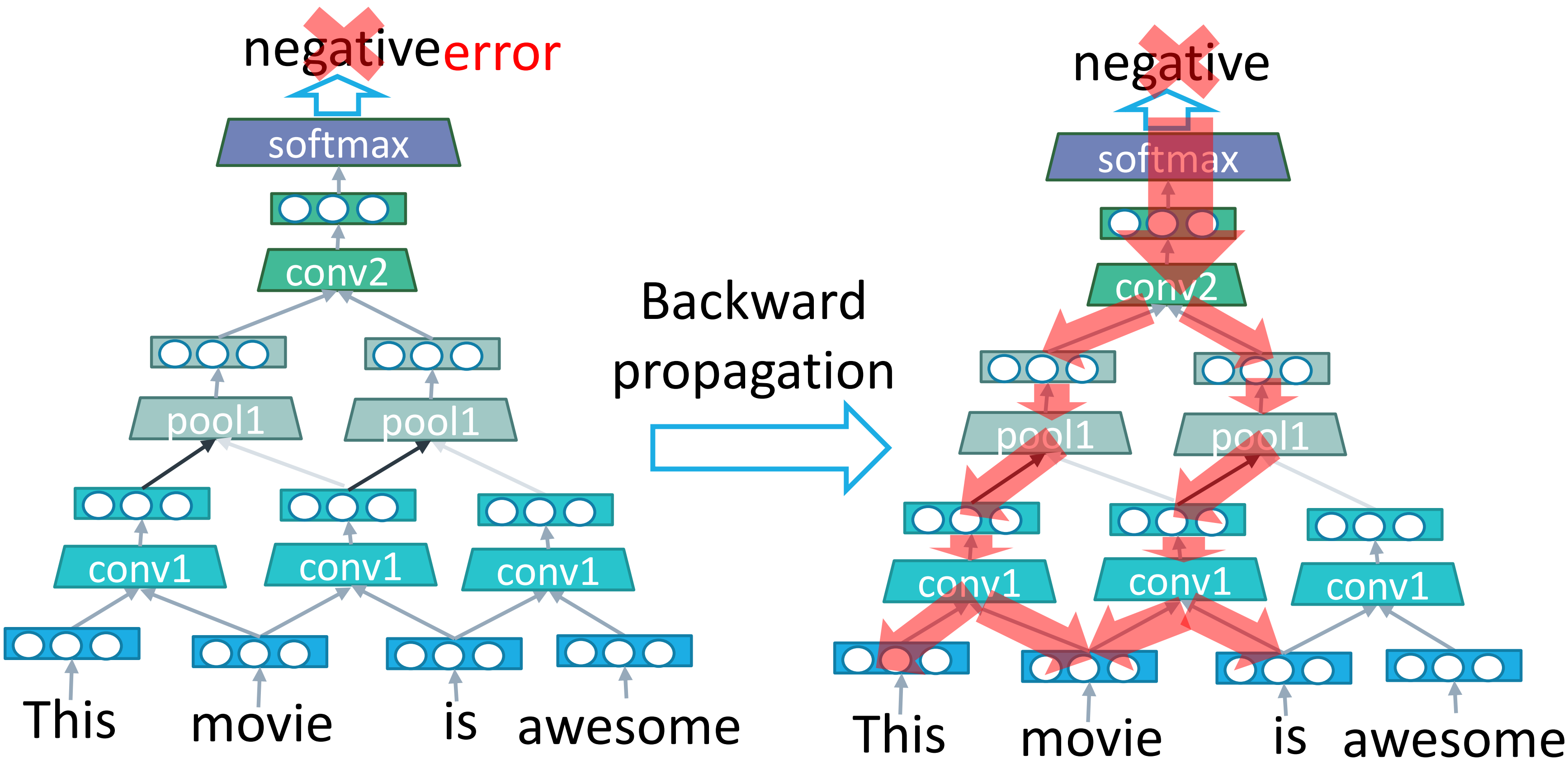
Sentiment Analysis by CNN

- Use softmax layer to classify the sentiments



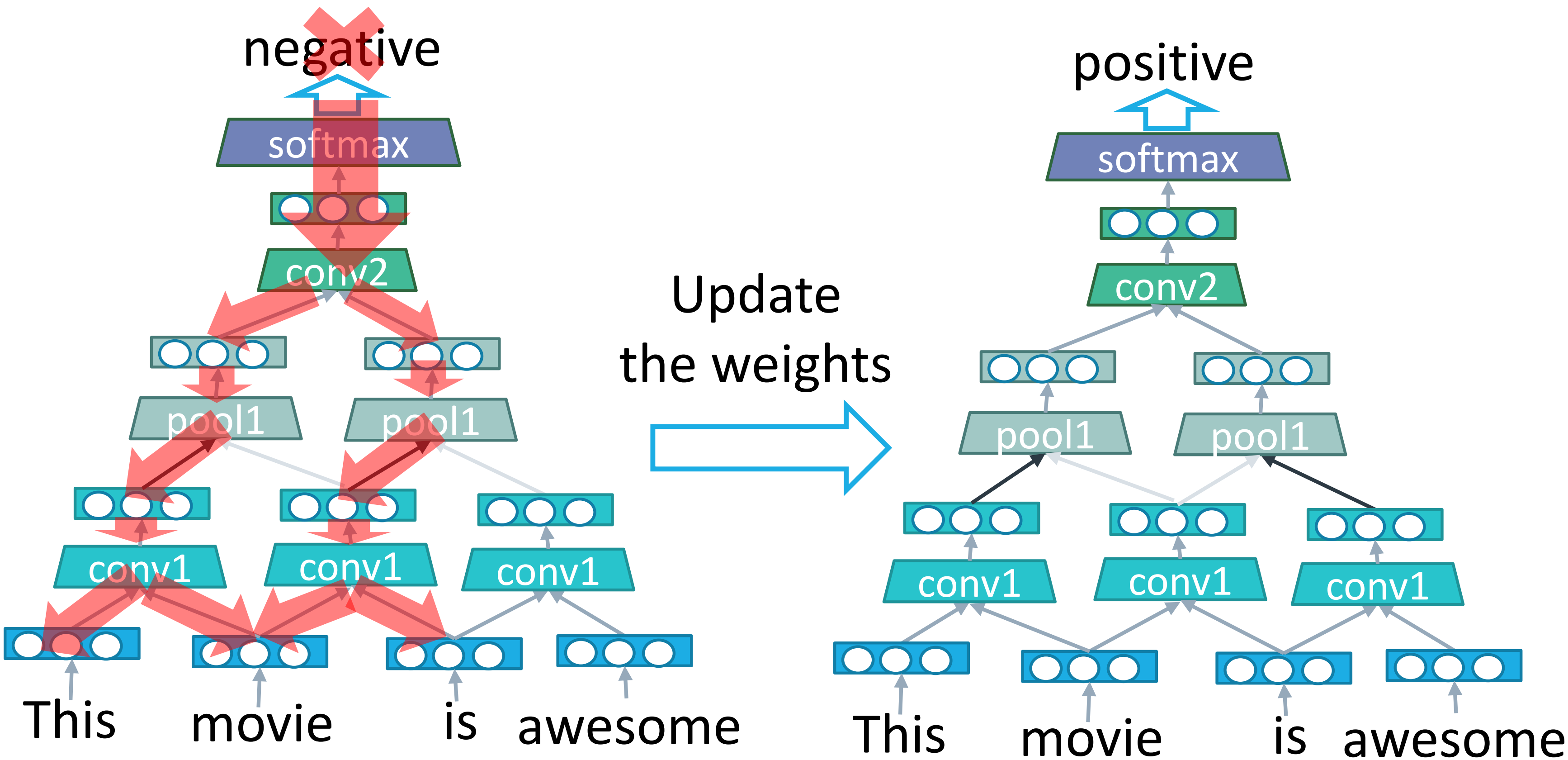
Sentiment Analysis by CNN

- Build the "correct syntax tree" by training



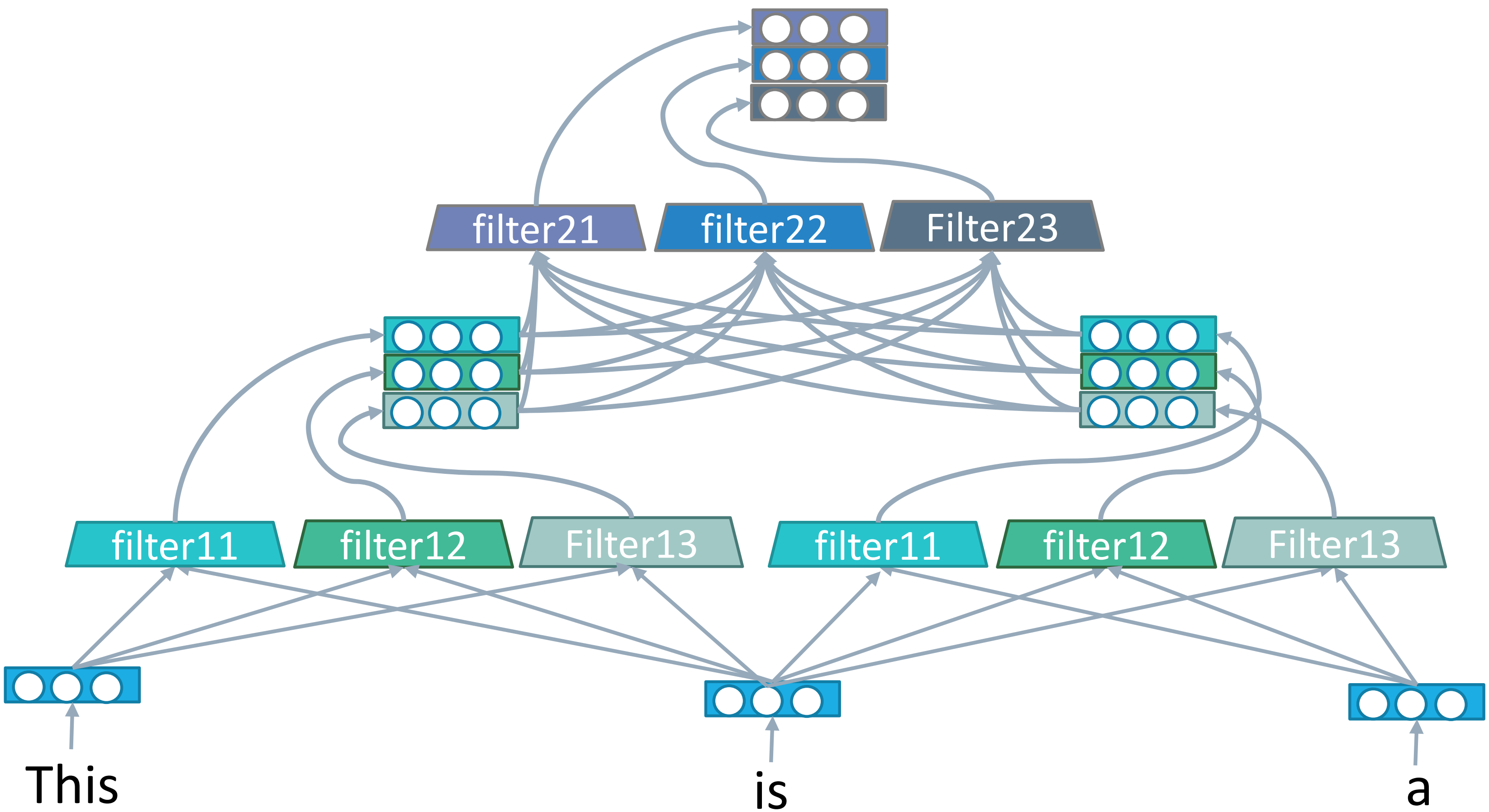
Sentiment Analysis by CNN

- Build the "correct syntax tree" by training



Multiple Filters

- Richer features than RNN



Resizing Sentence

- Image can be easily resized
- Sentence can't be easily resized

resize

全台灣最高樓在台北市

resize

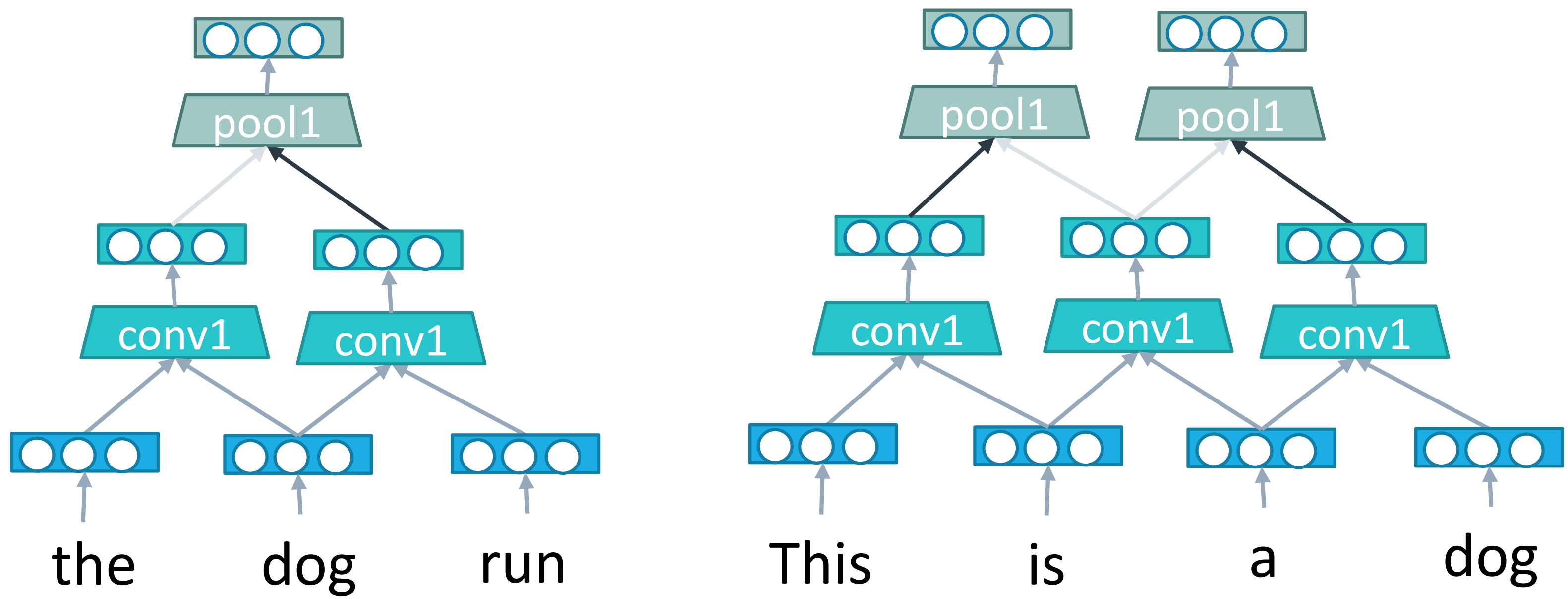
全台灣最高的高樓在台北市

全台灣最高樓在台北

台灣最高樓在台北

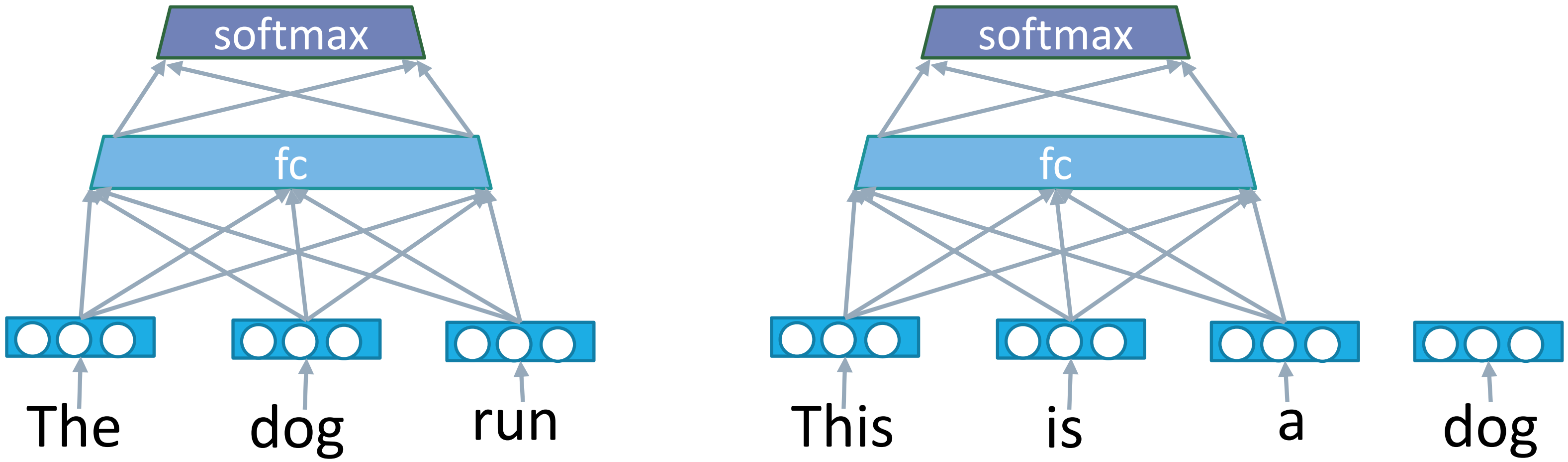
Various Input Size

- Convolutional layers and pooling layers
  - can handle input with various size



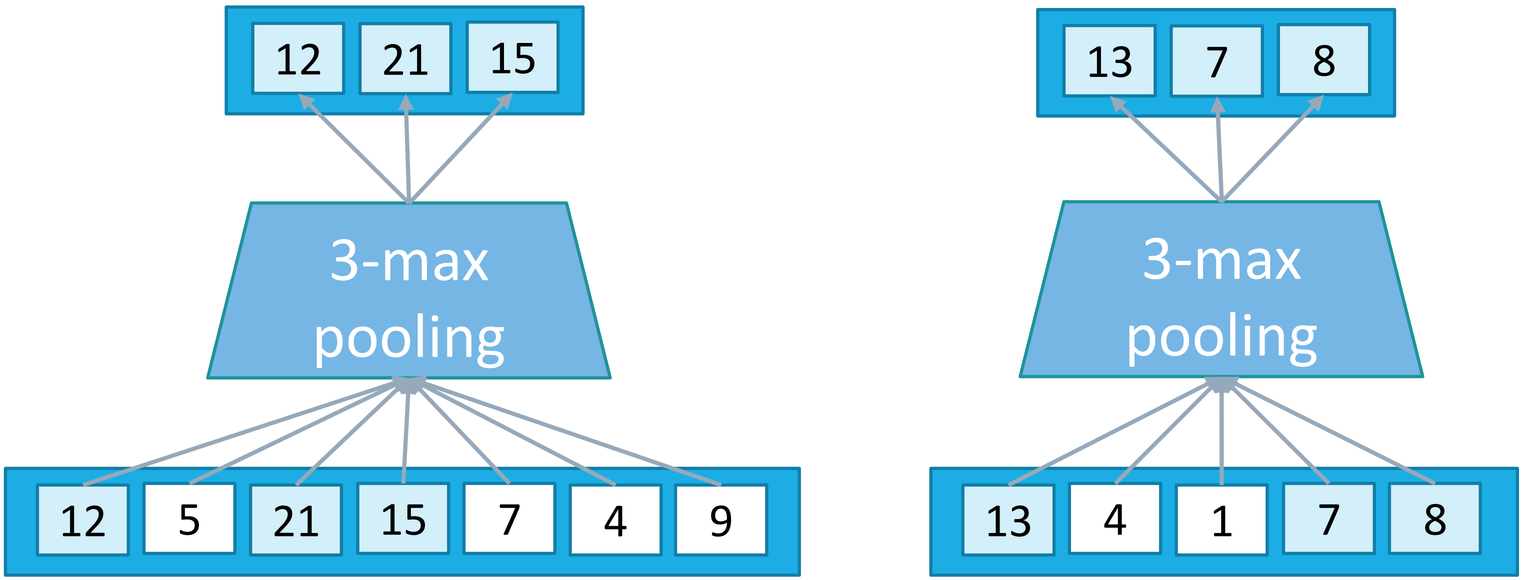
Various Input Size

- Fully-connected layer and softmax layer
  - need fixed-size input



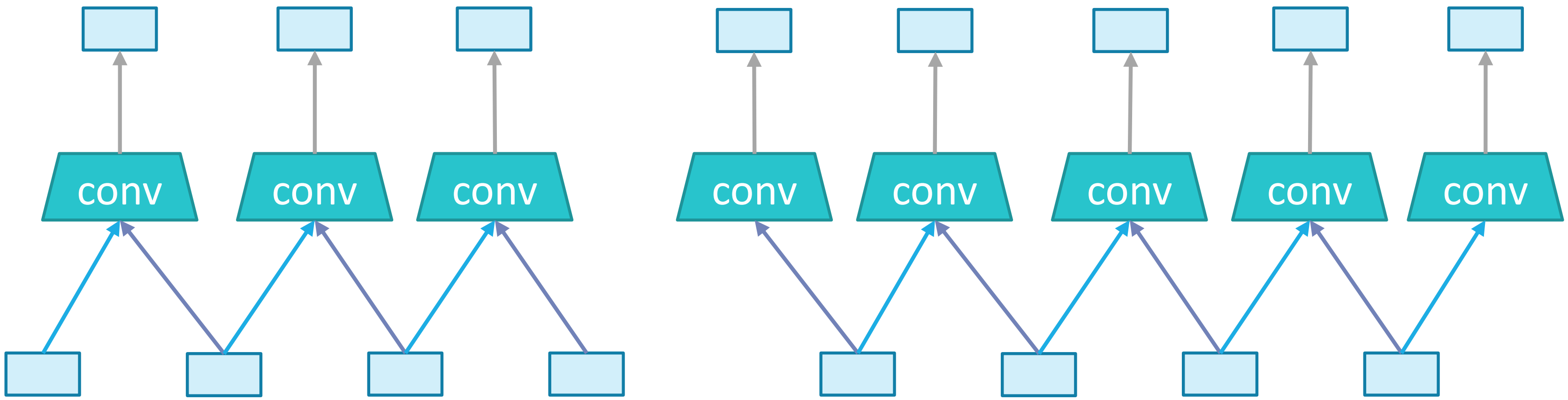
k-max Pooling

- choose the k-max values
- preserve the order of input values
- variable-size input, fixed-size output



Wide Convolution

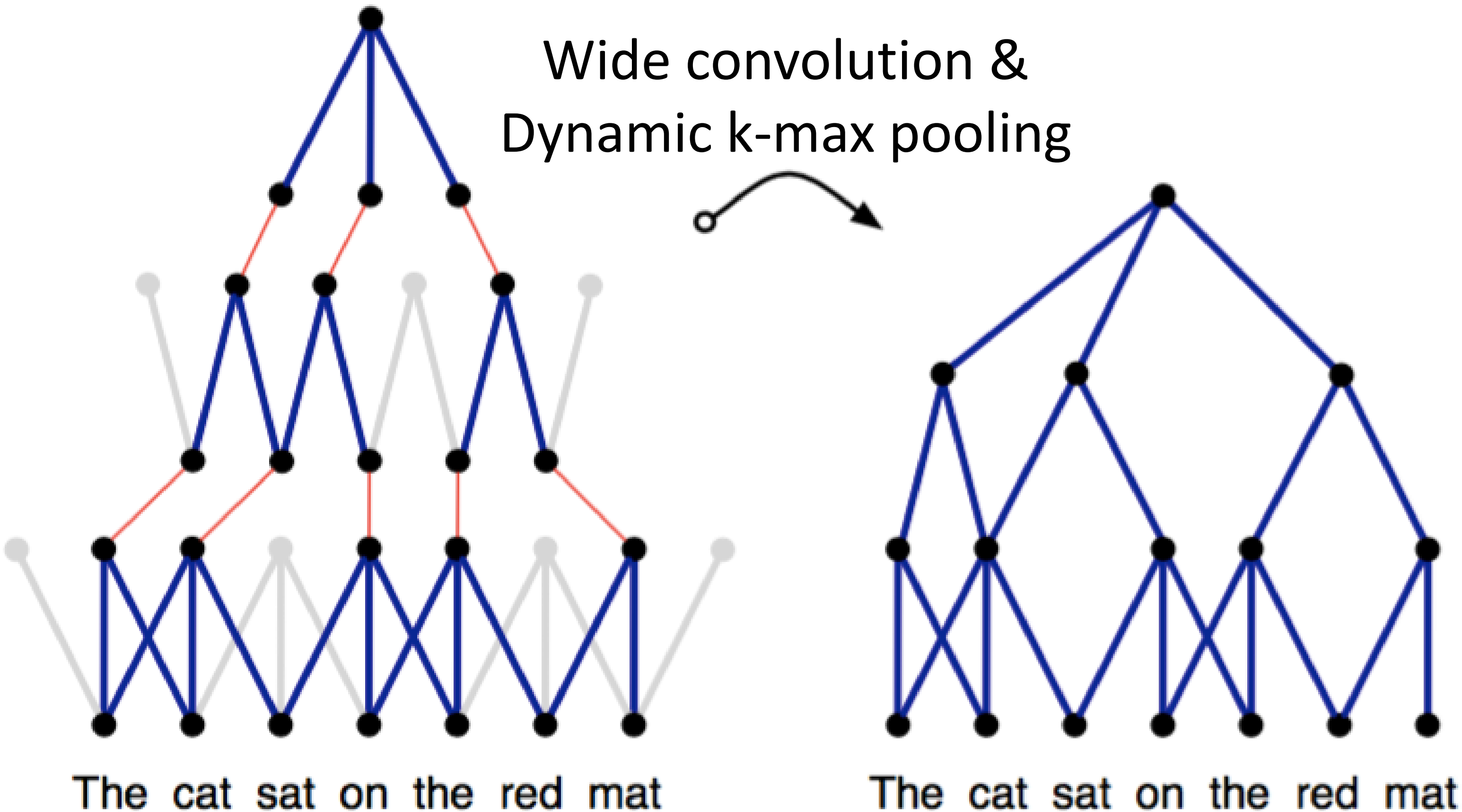
- Ensures that all weights reach the entire sentence



Narrow convolution

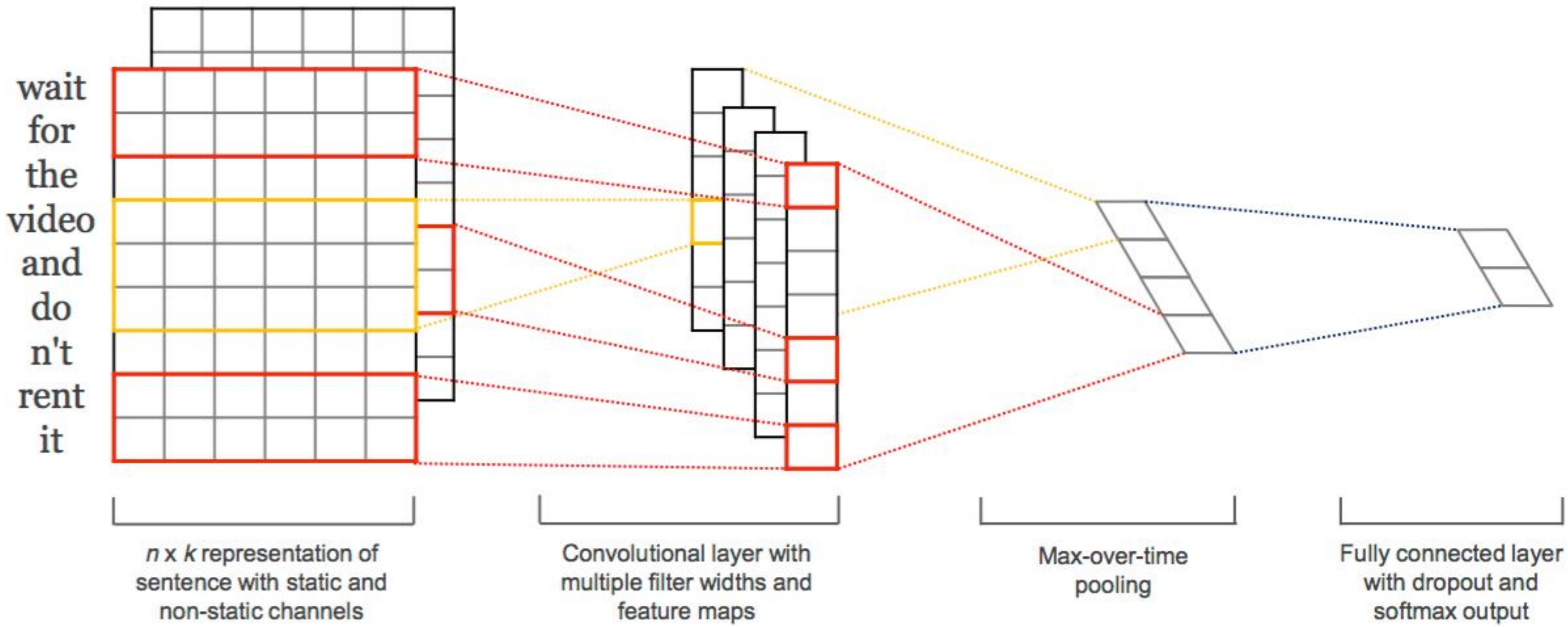
Wide convolution

Dynamic k-max Pooling



CNN for Sentence Classification

- Pretrained by word2vec
- Static & non-static channels
  - Static: fix the values during training
  - Non-Static: update the values during training



Concluding Remarks