Convolutional Neural Networks

Nov 3rd ,2016

Applied Deep Learning

Yun-Nung (Vivian) Chen X Mark Chang

www.csie.ntu.edu.tw/~yvchen/f105-adl

臺灣大學
National Taiwan University

Slide credit from Mark Chang

Convolutional Neural Networks

- We need a course to talk about this topic
  - http://cs231n.stanford.edu/syllabus.html
- However, we only have a lecture

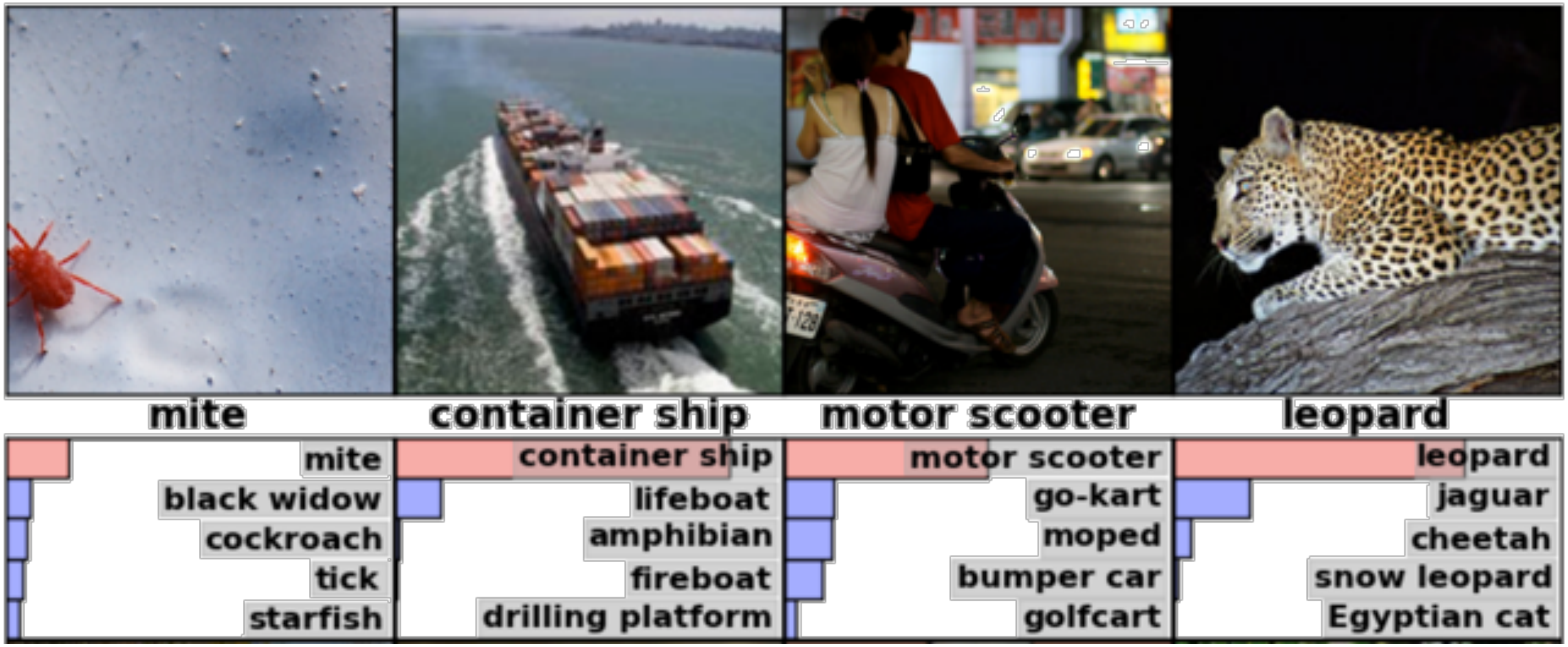
Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- Sentiment Analysis by CNN

Outline

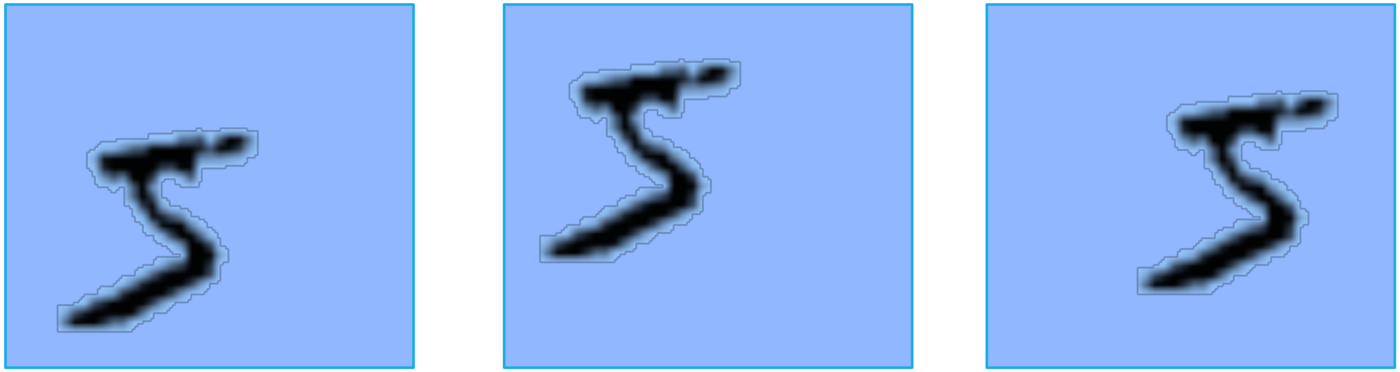
- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- Sentiment Analysis by CNN

Image Recognition

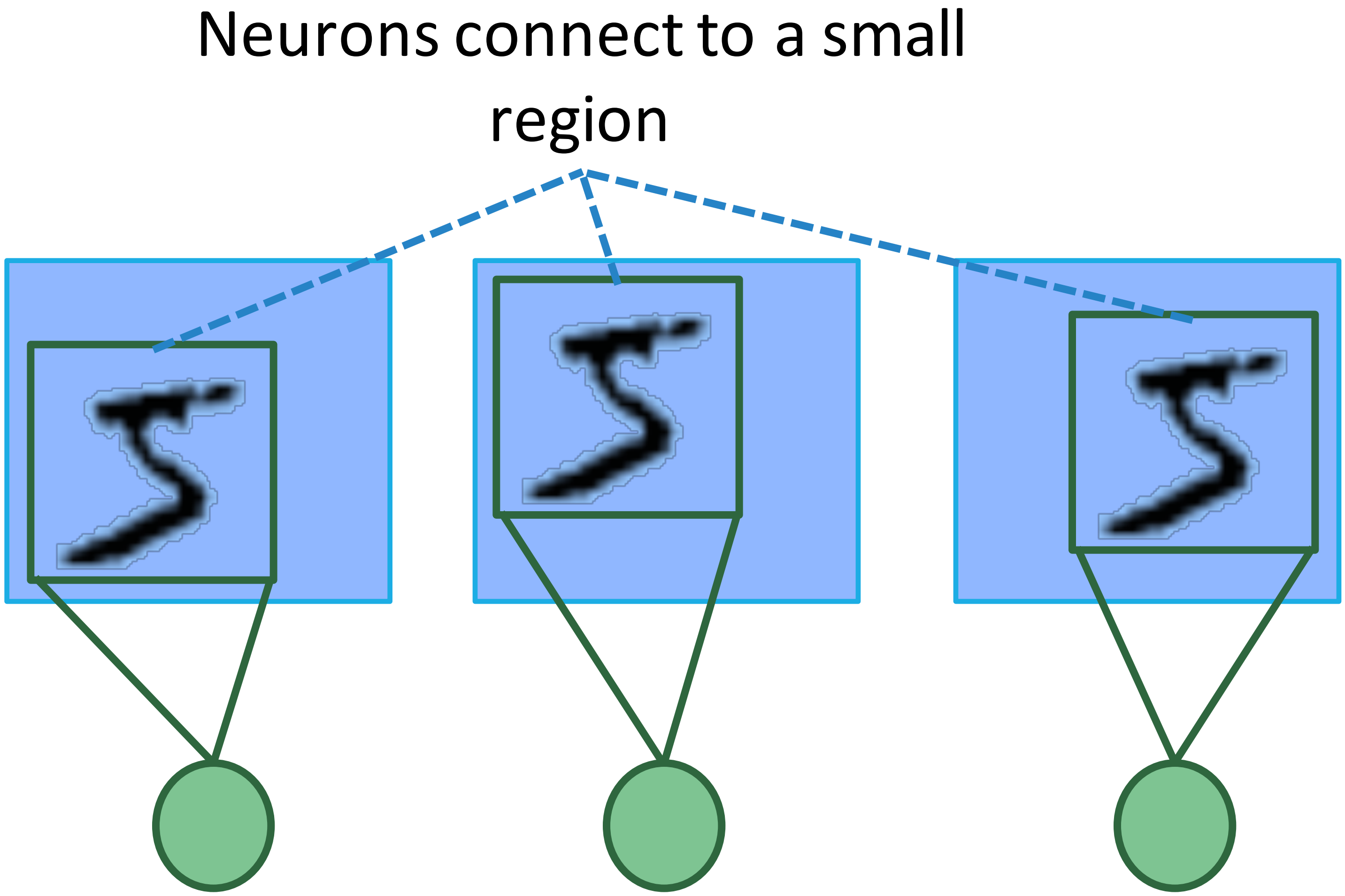


http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

Image Recognition

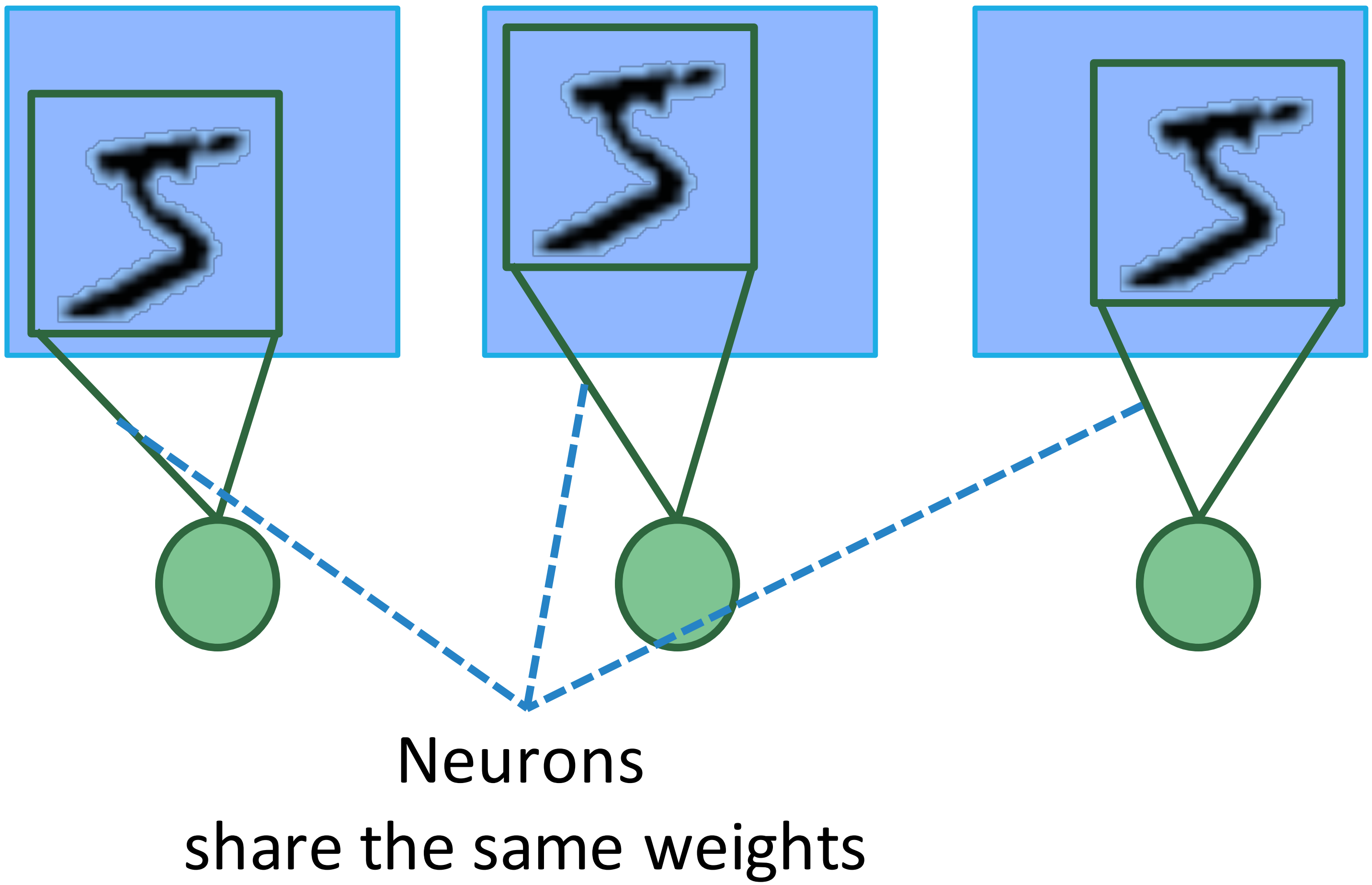


Local Connectivity



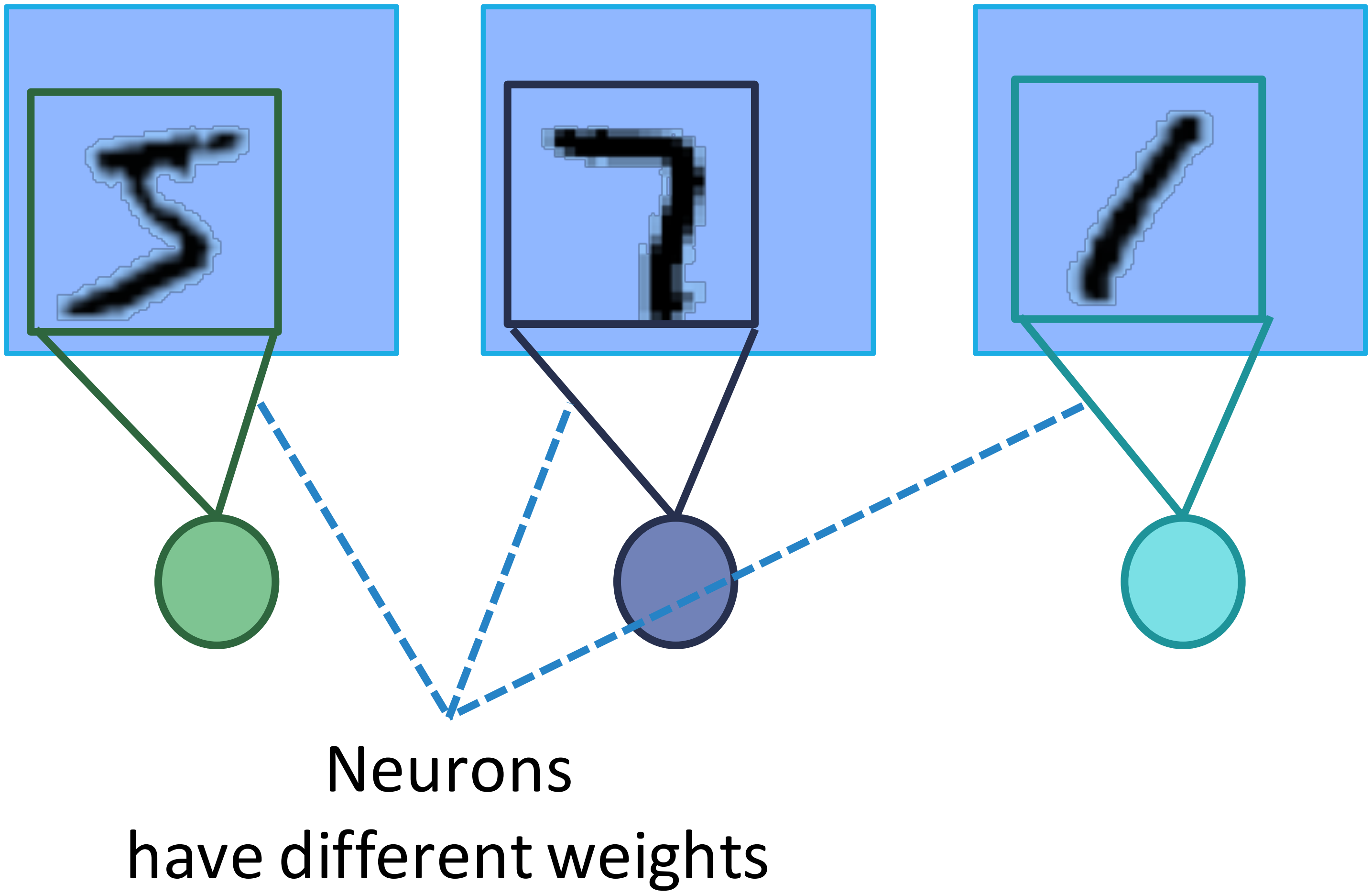
Parameter Sharing

- The same feature in different positions

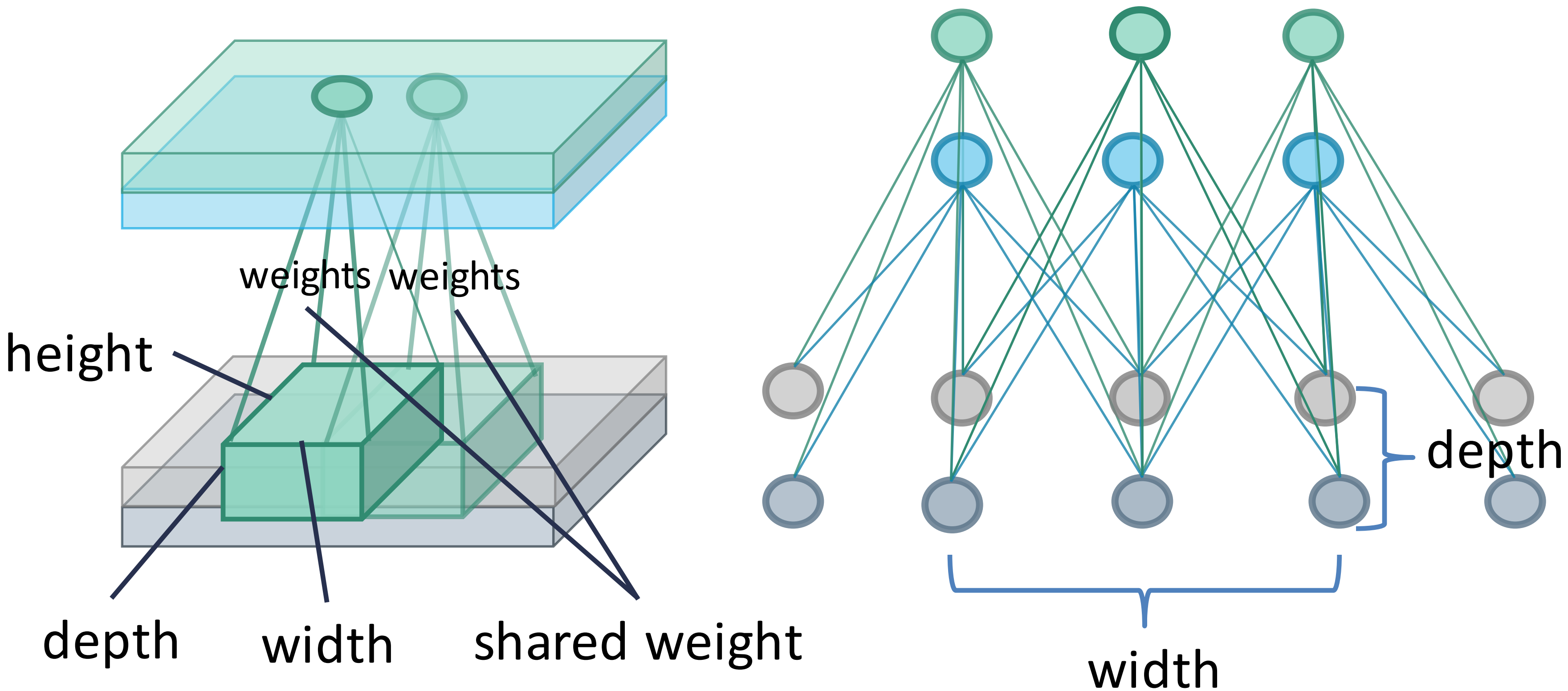


Parameter Sharing

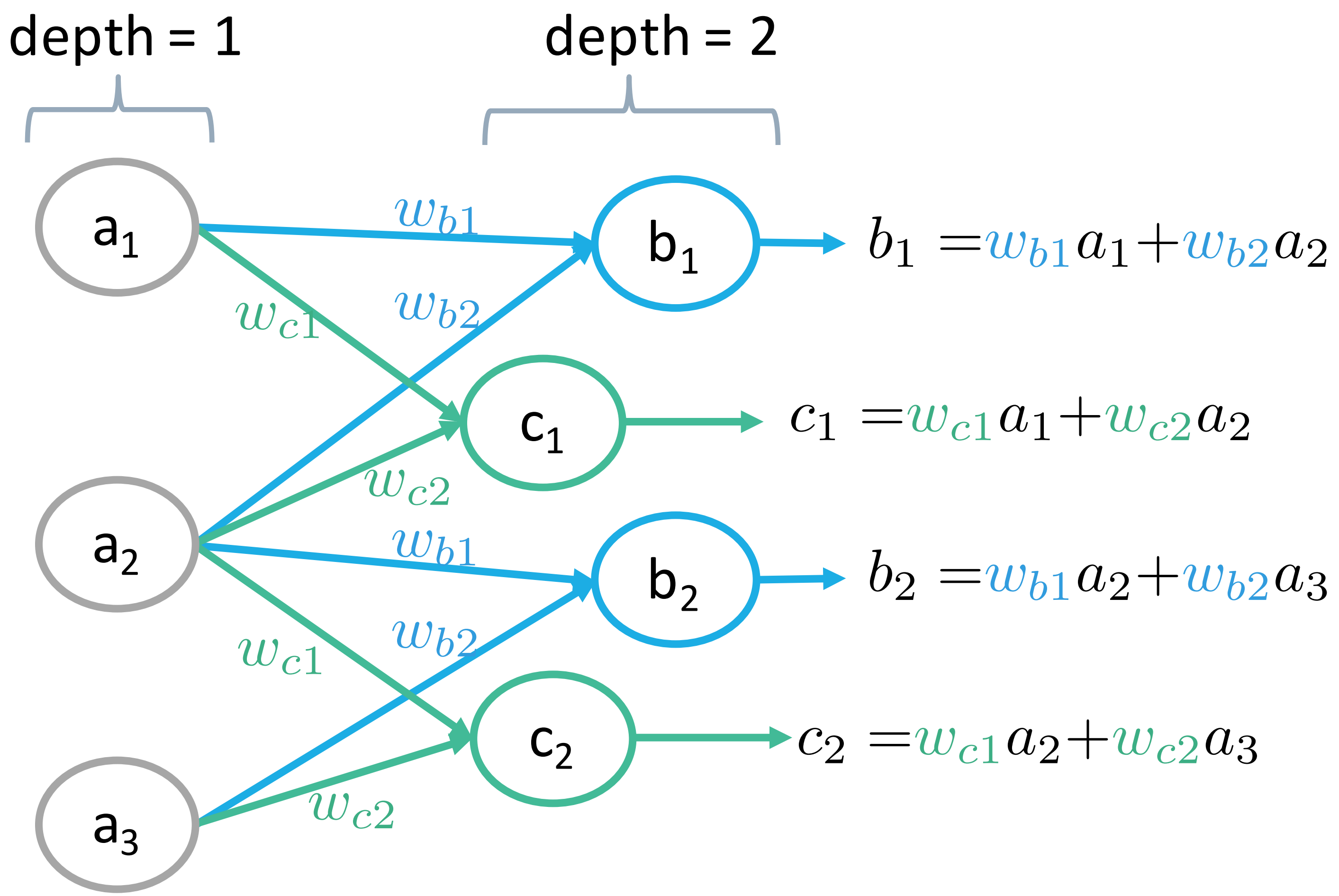
- Different features in the same position



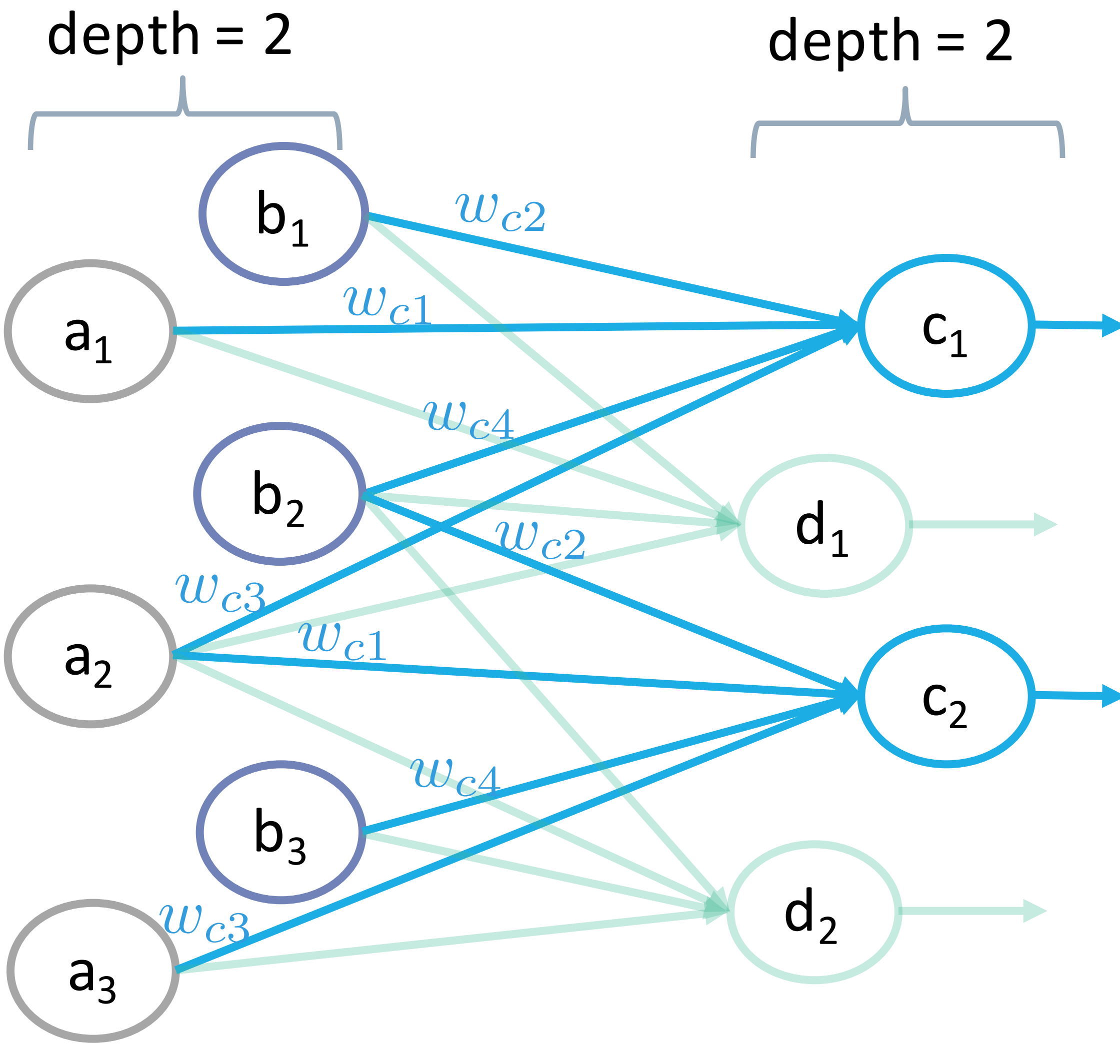
Convolutional Layers



Convolutional Layers



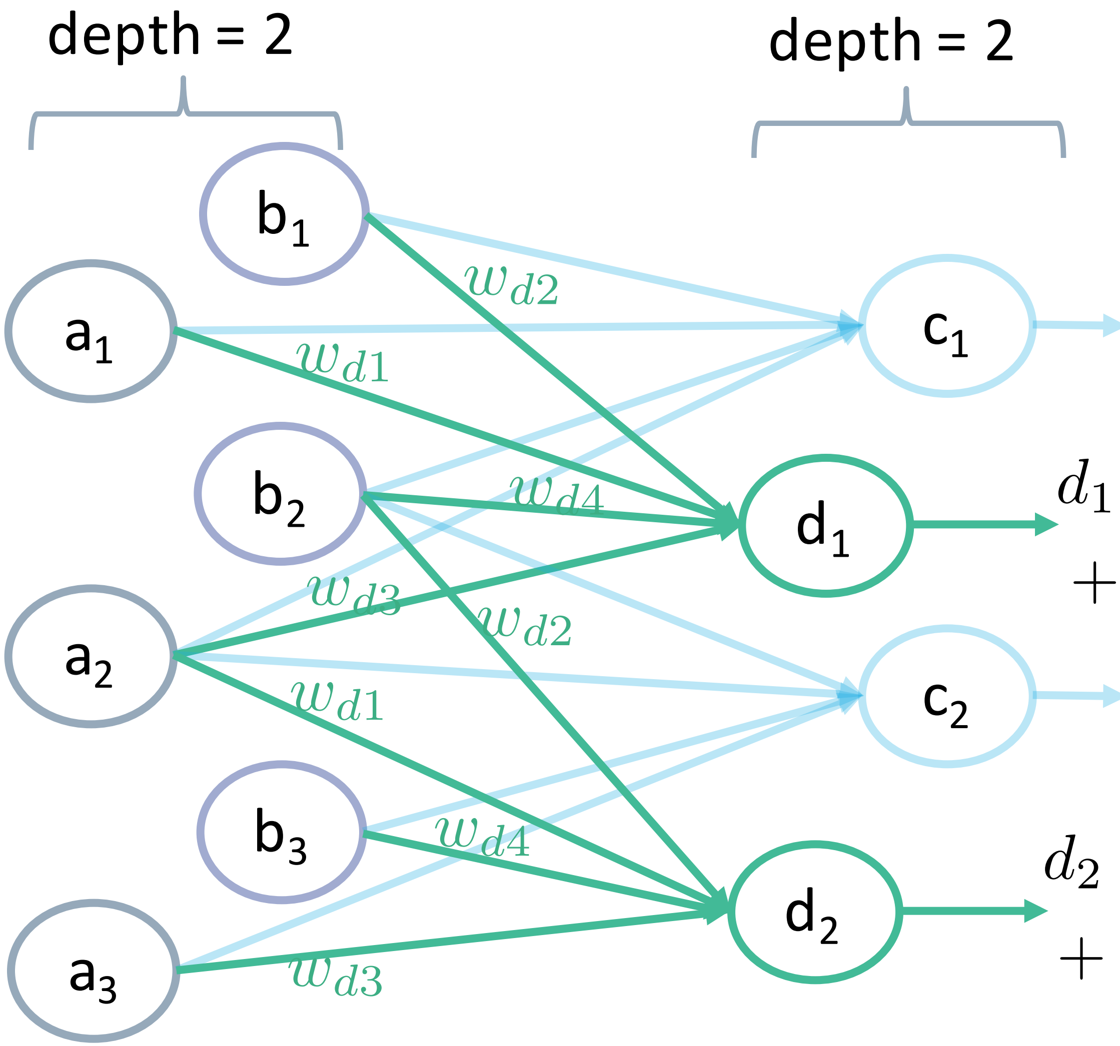
Convolutional Layers



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

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

Convolutional Layers



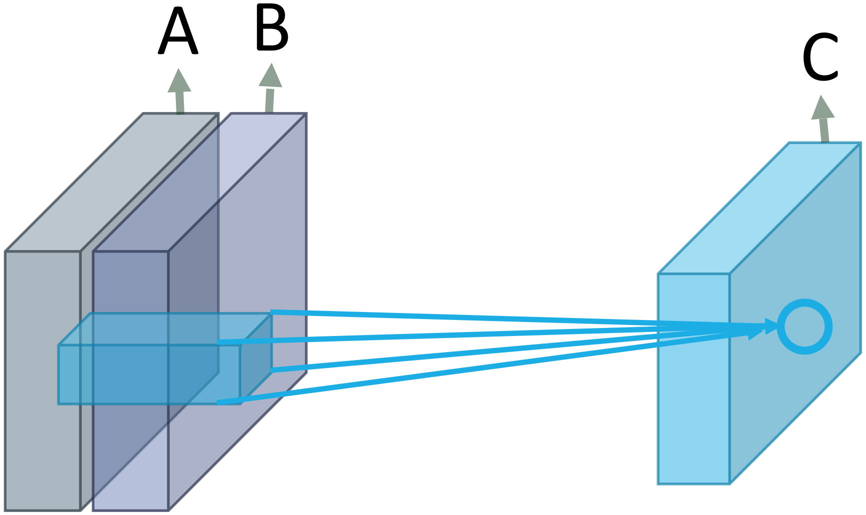
$$c_1 = a_1 w_{c1} + b_1 w_{c2} + a_2 w_{c3} + b_2 w_{c4}$$

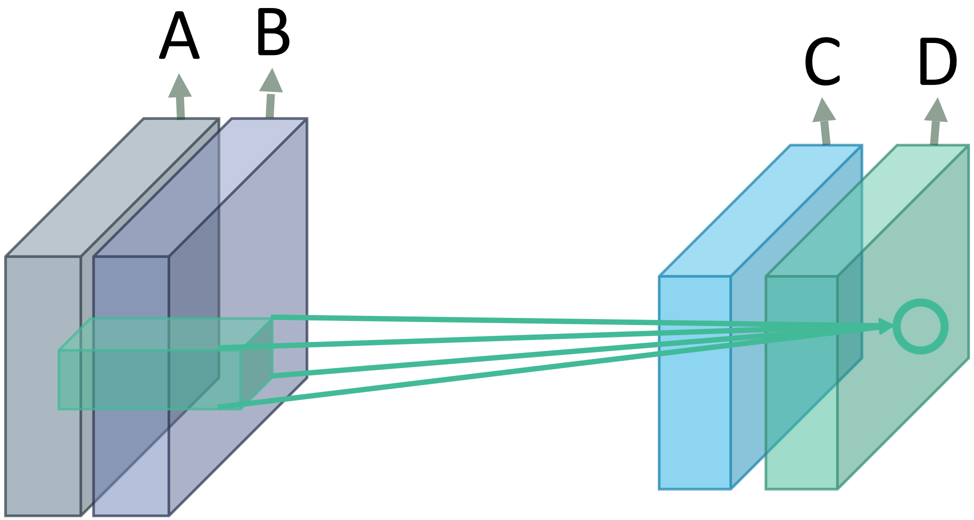
$$d_1 = a_1 w_{d1} + b_1 w_{d2} + a_2 w_{d3} + b_2 w_{d4}$$

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

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

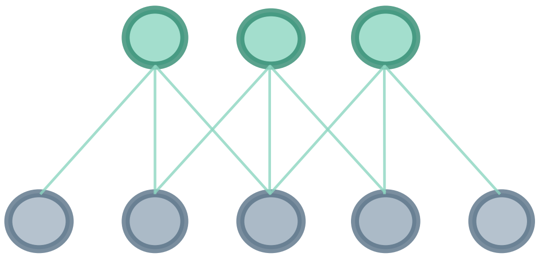
Convolutional Layers



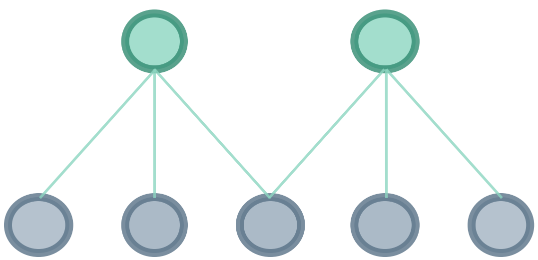


Hyper-parameters of CNN

- Stride

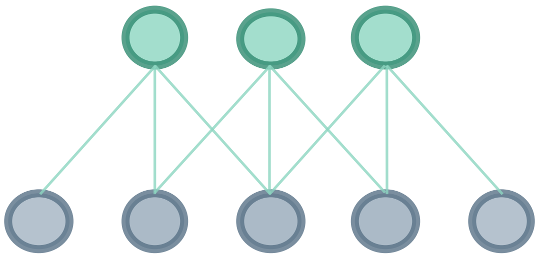


Stride = 1

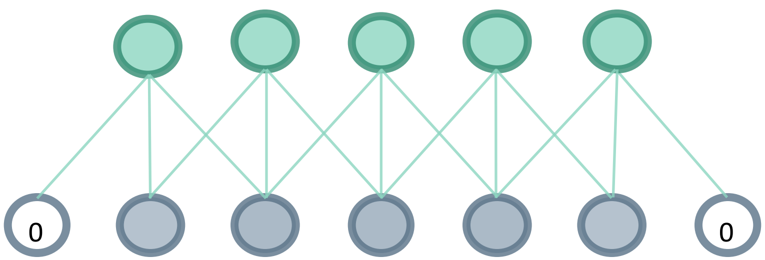


Stride = 2

- Padding

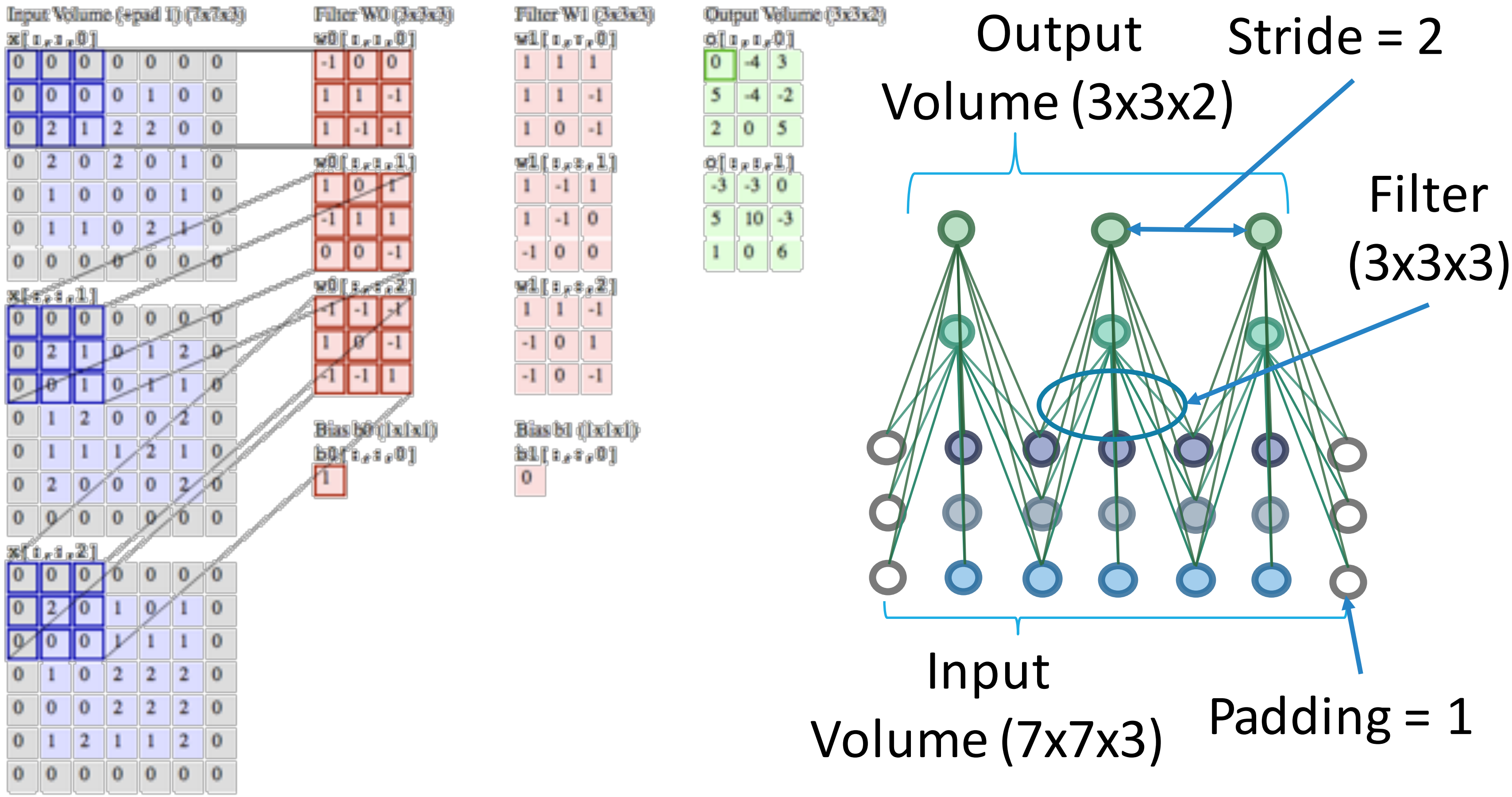


Padding = 0



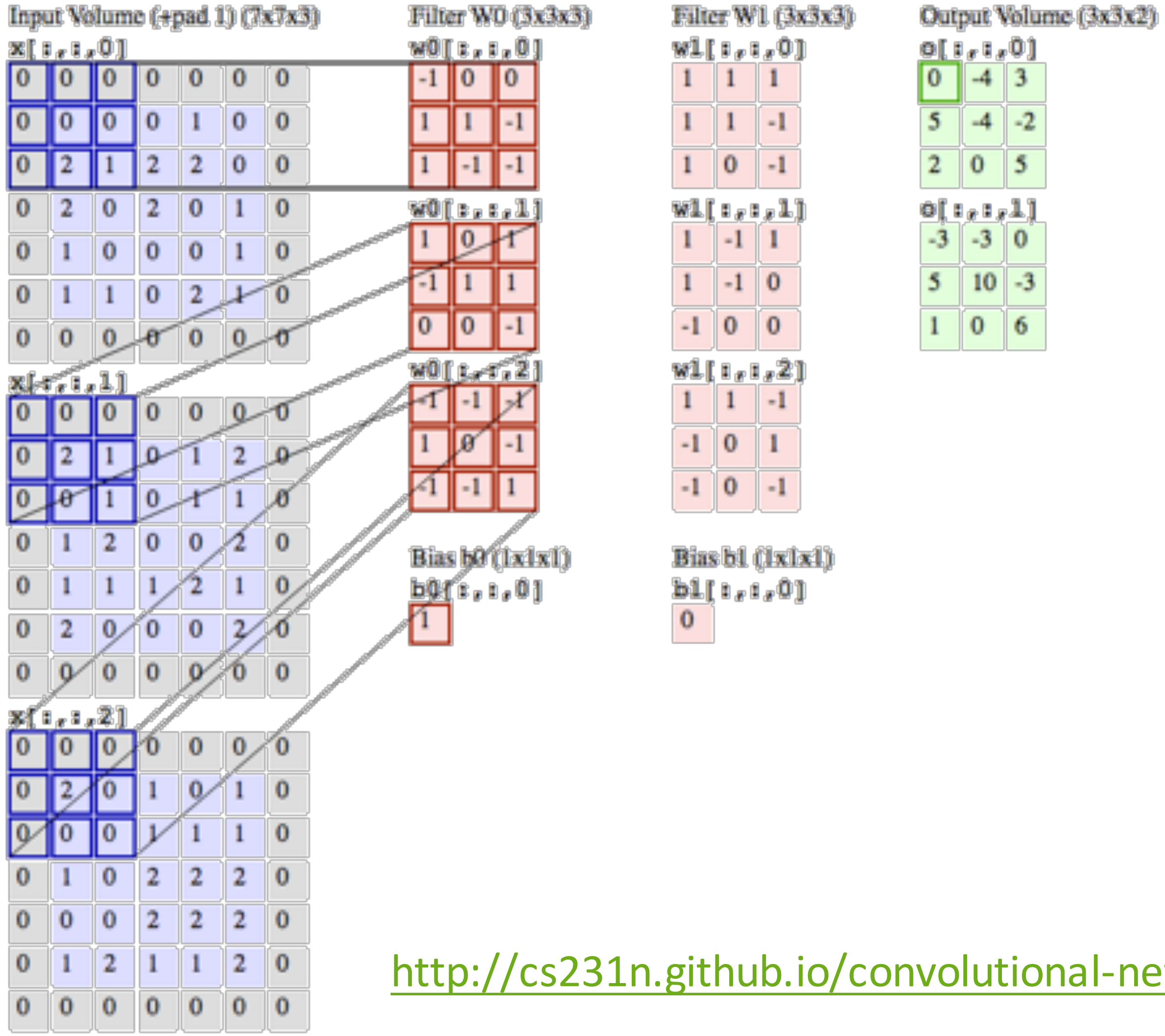
Padding = 1

Example



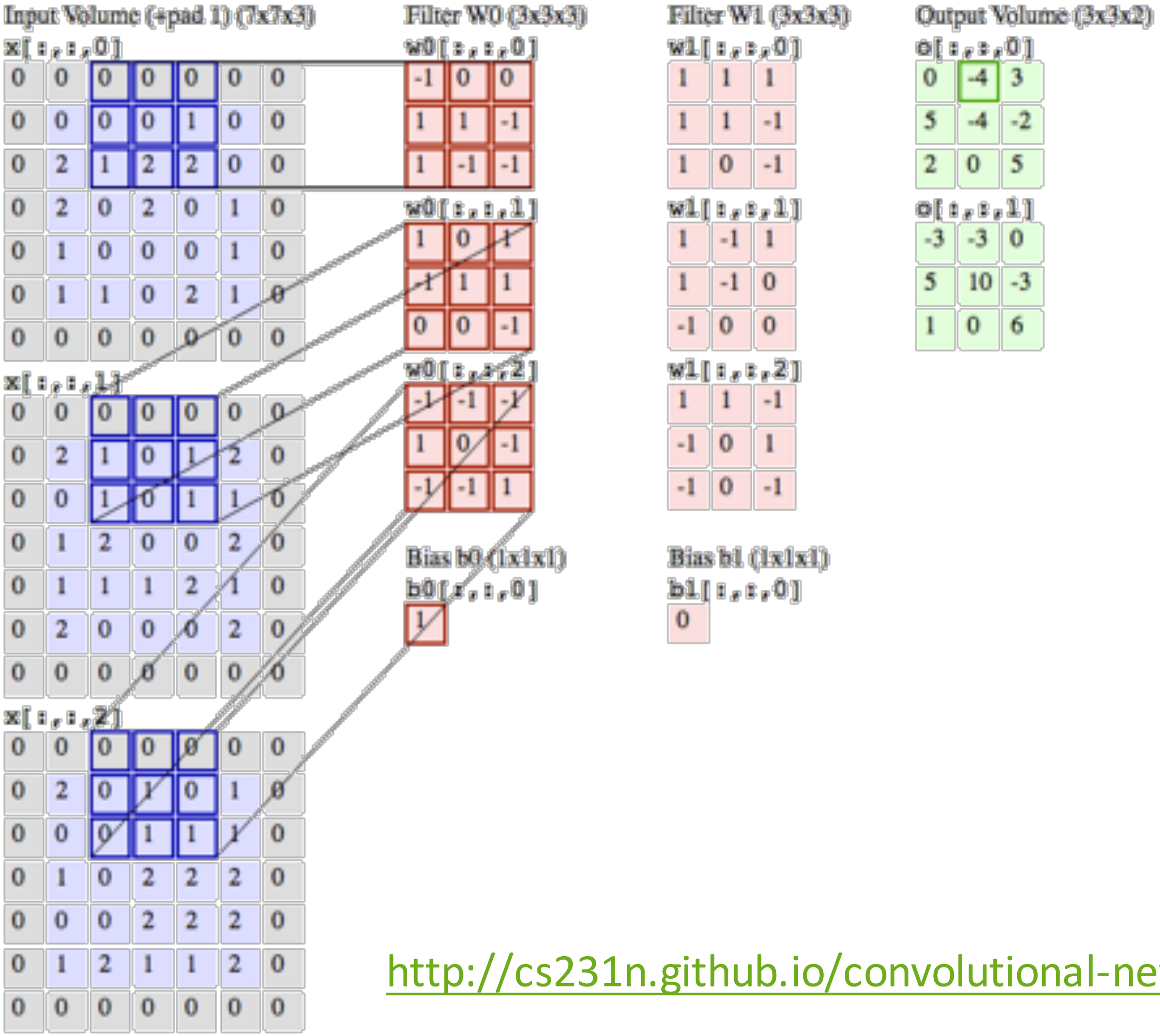
http://cs231n.github.io/convolutional-networks/

Convolutional Layers



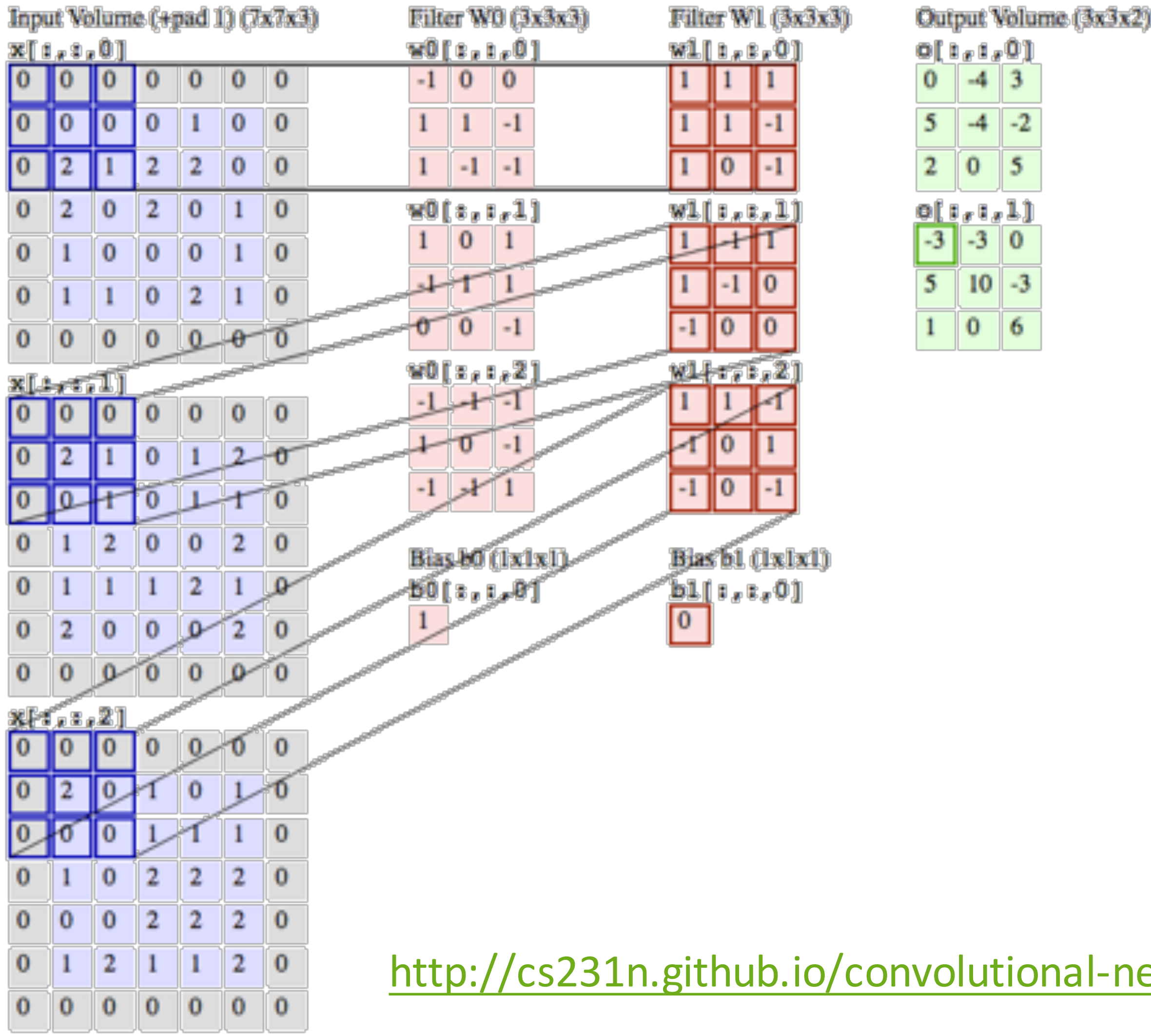
http://cs231n.github.io/convolutional-networks/

Convolutional Layers



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

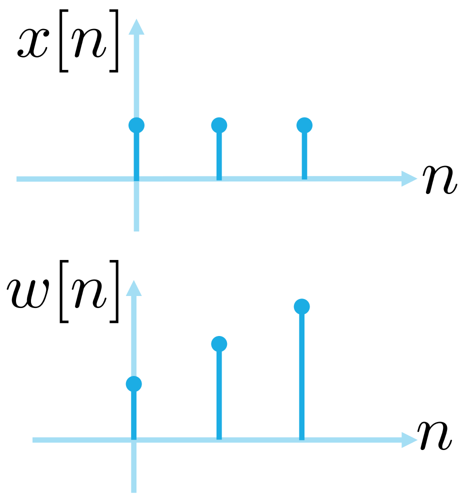
Convolutional Layers

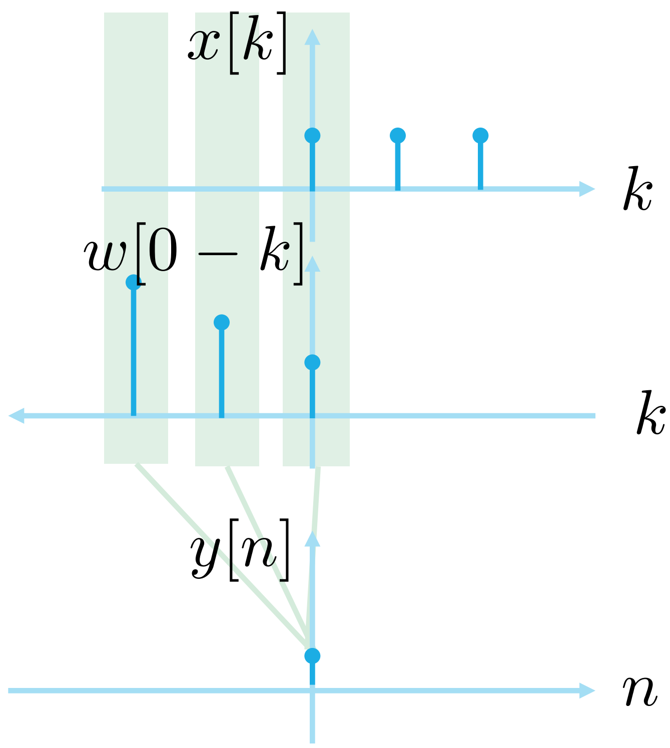


http://cs231n.github.io/convolutional-networks/

Relationship with Convolution

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

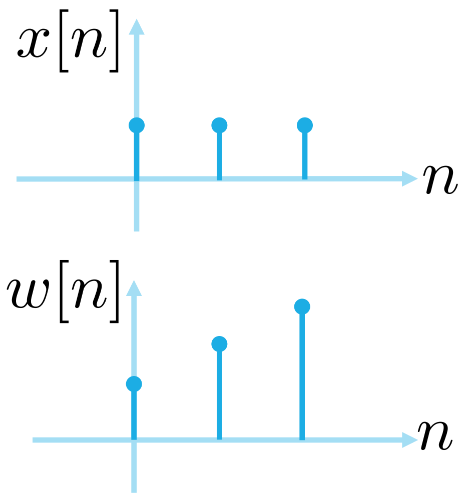


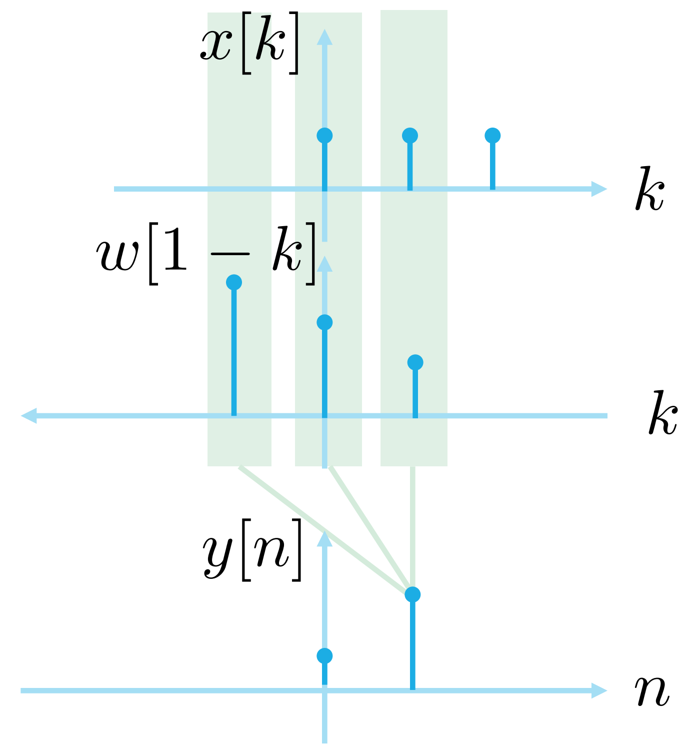


$$y[0] = x[-2]w[2] + x[-1]w[1] + x[0]w[0]$$

Relationship with Convolution

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

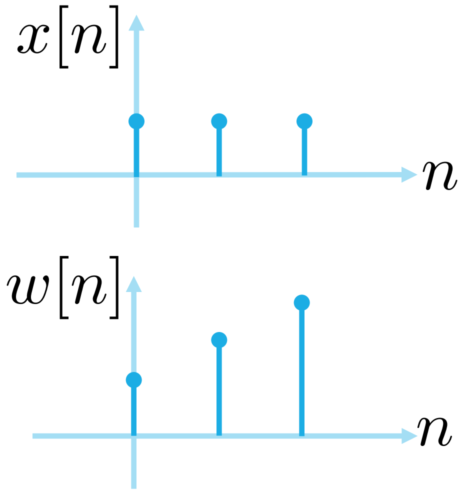


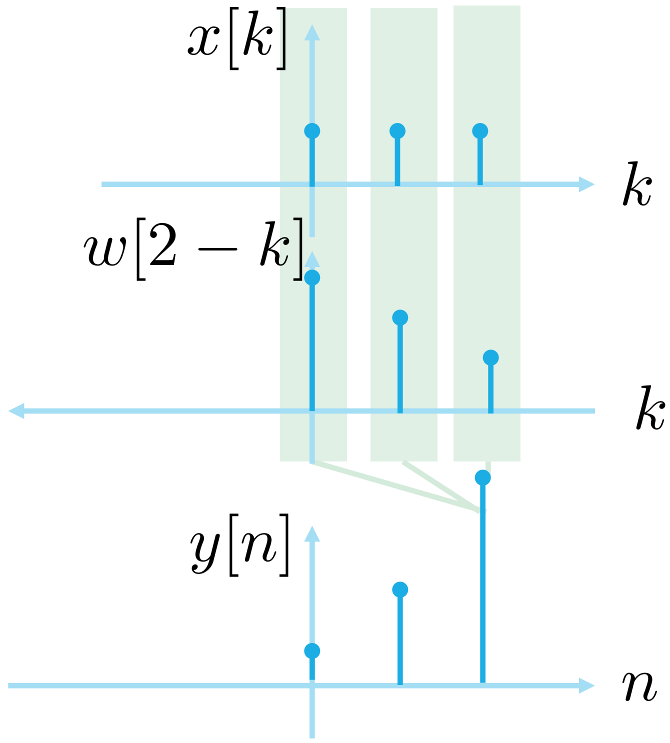


$$y[1] = x[-1]w[2] + x[0]w[1] + x[2]w[0]$$

Relationship with Convolution

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

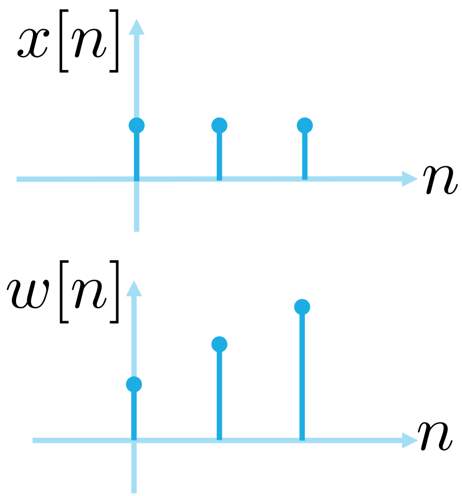


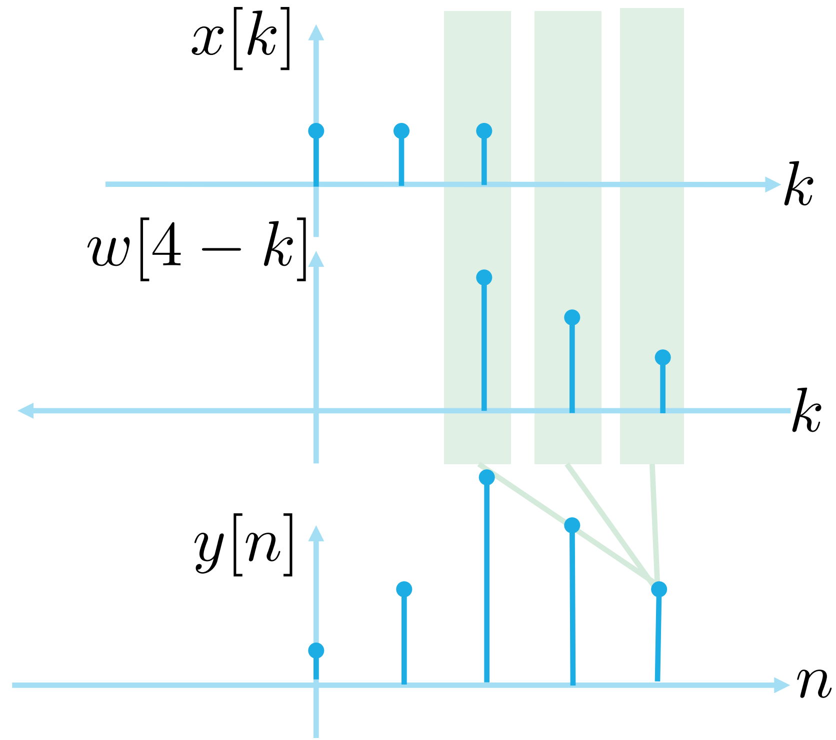


$$y[2] = x[0]w[2] + x[1]w[1] + x[2]w[0]$$

Relationship with Convolution

$$y[n] = \sum_{k} x[k]w[n-k]$$

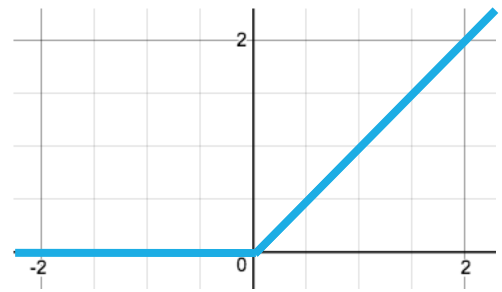


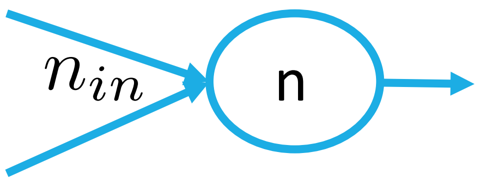


$$y[4] = x[2]w[2] + x[3]w[1] + x[4]w[0]$$

Nonlinearity

- Rectified Linear (ReLU)



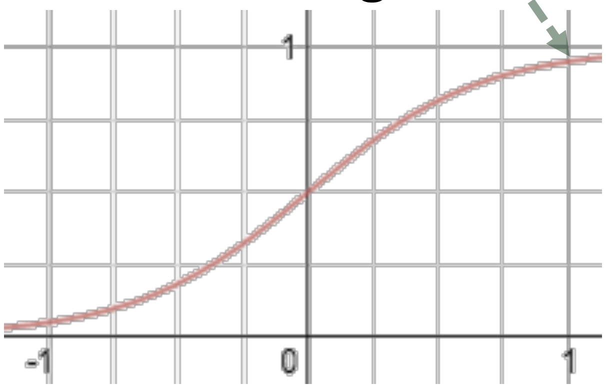


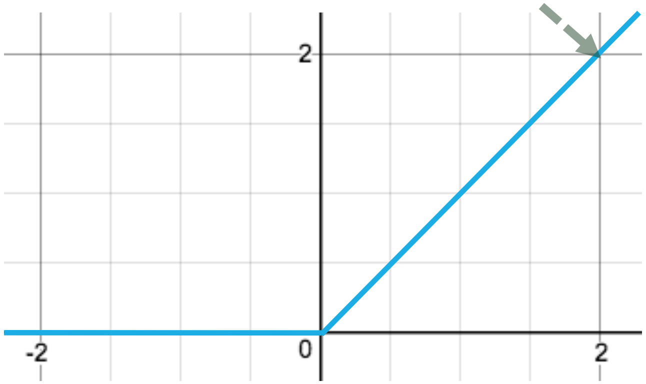
$$n_{out} = \begin{cases} n_{in} \text{ if } n_{in} > 0 \\ 0 \text{ otherwise} \end{cases}$$

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

Why ReLU?

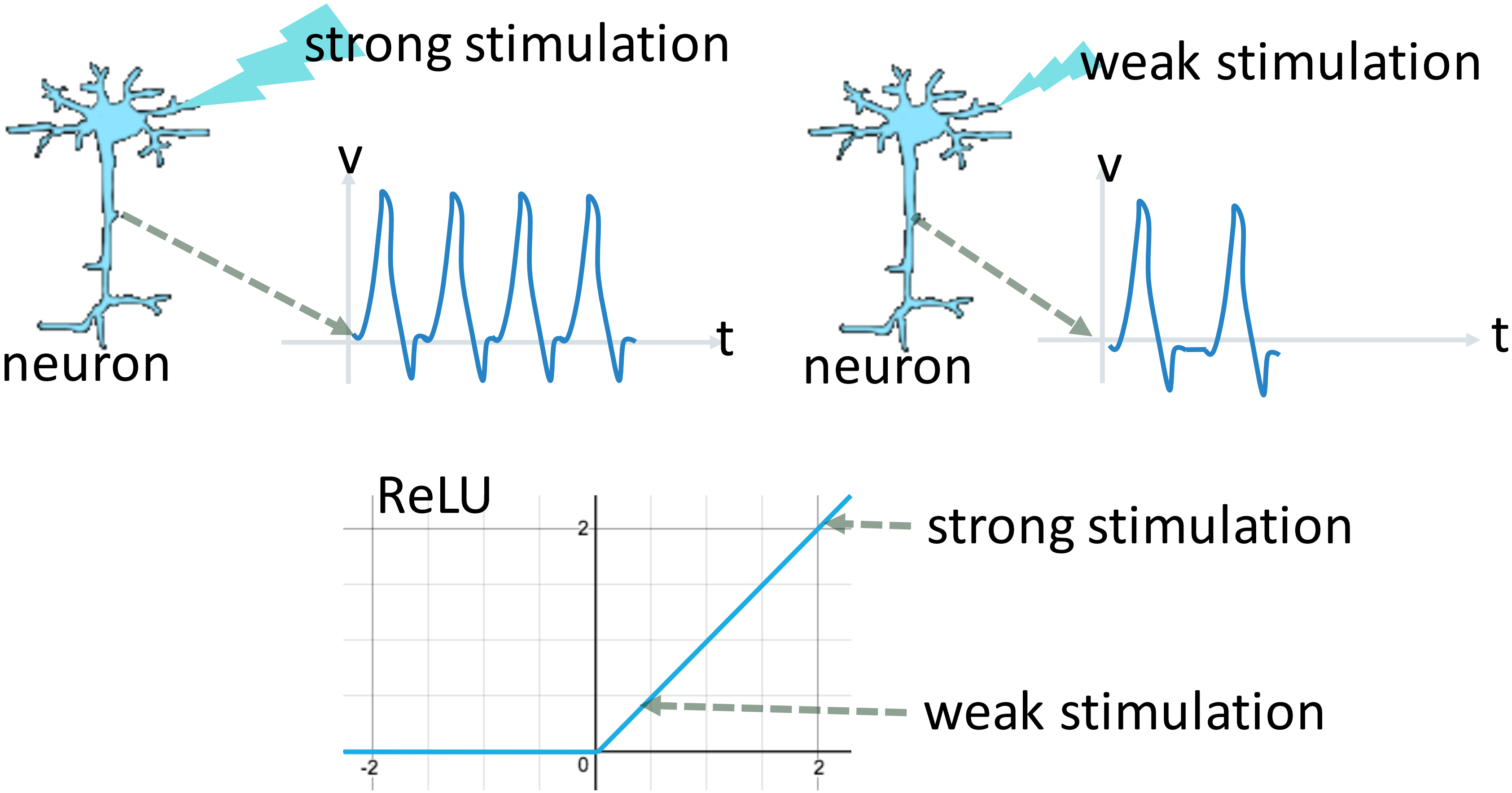
- Easy to train
- Avoid gradient vanishing problem



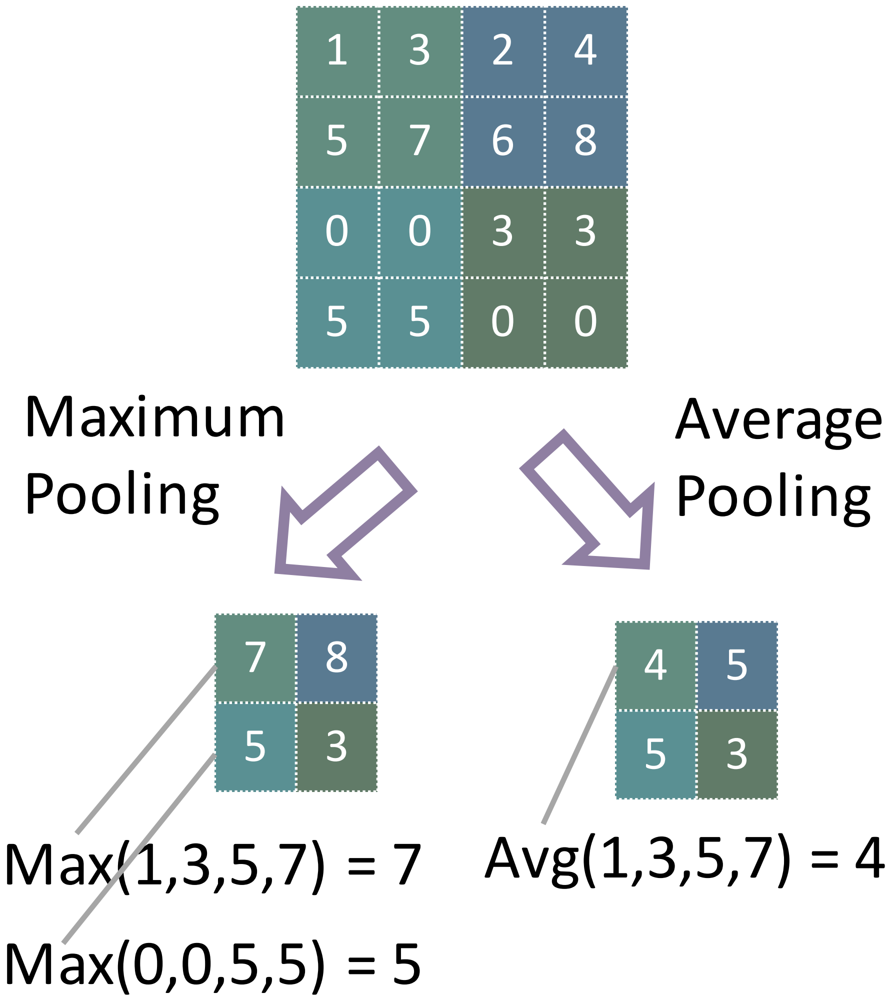


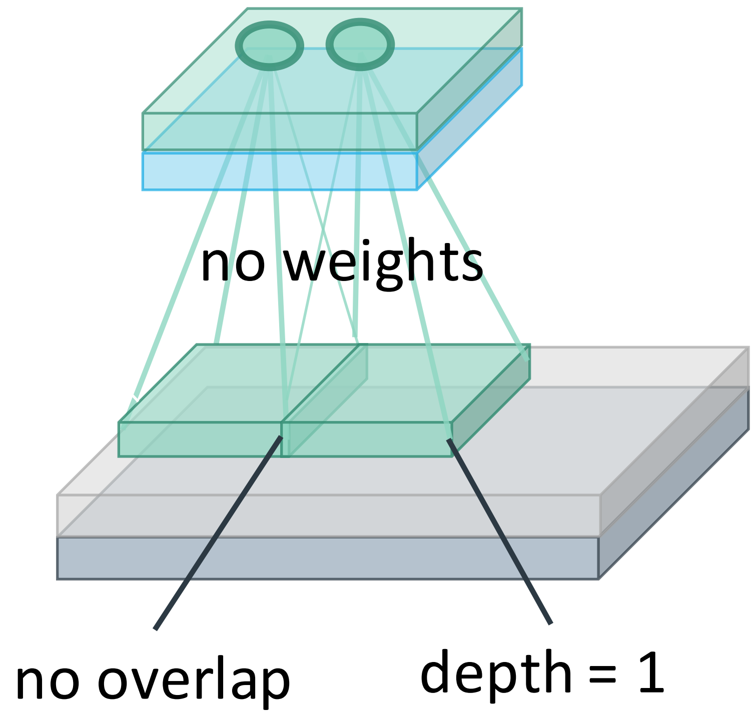
Why ReLU?

- Biological reason

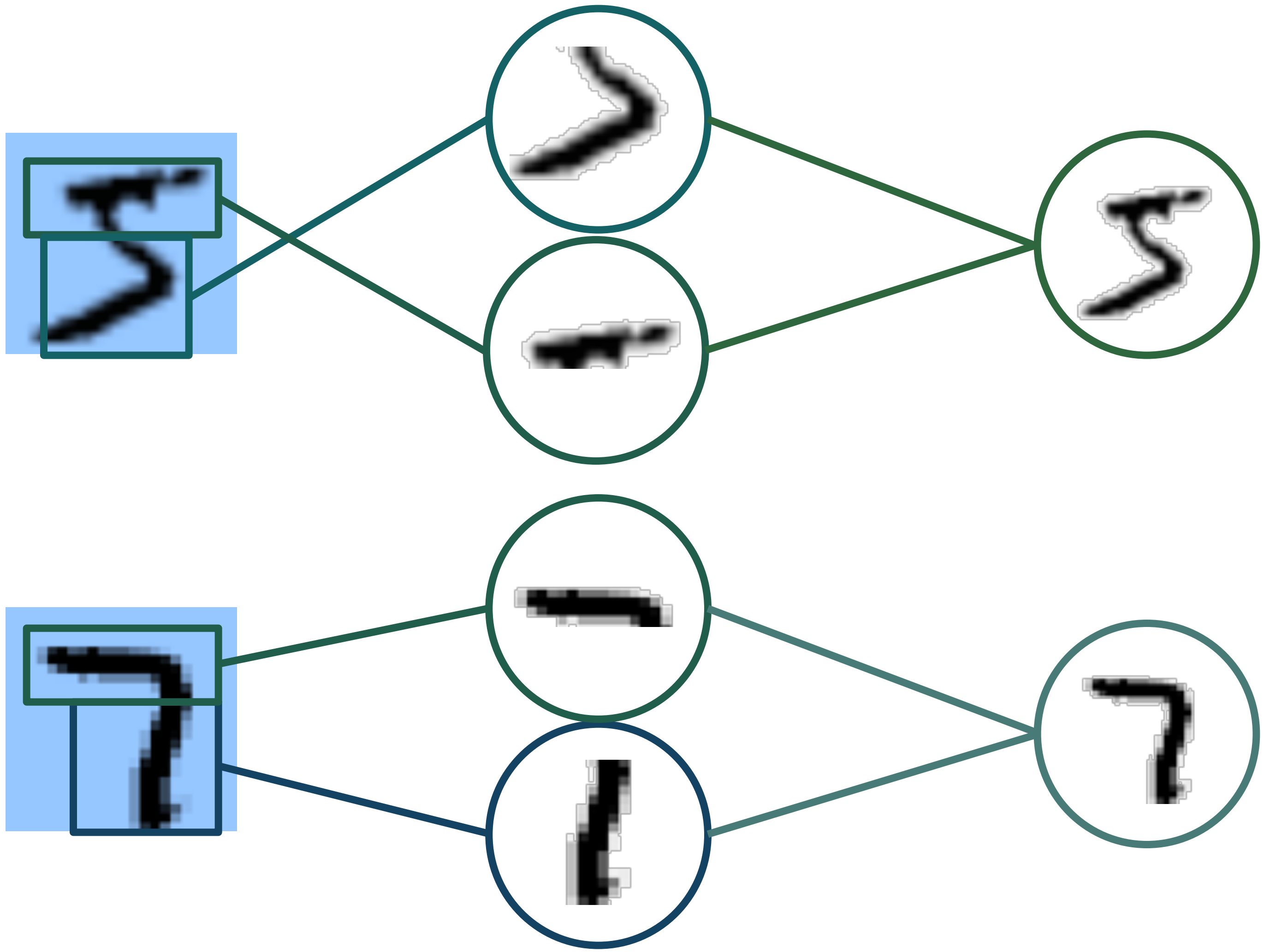


Pooling Layer

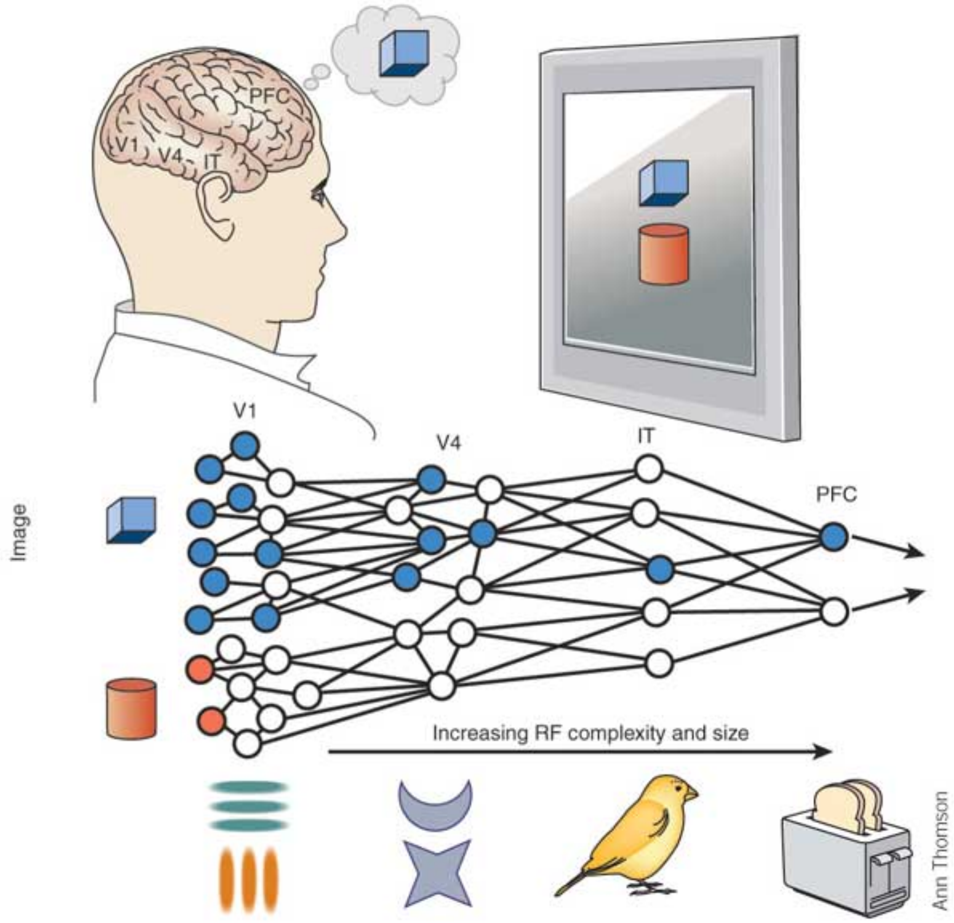




Why "Deep" Learning?

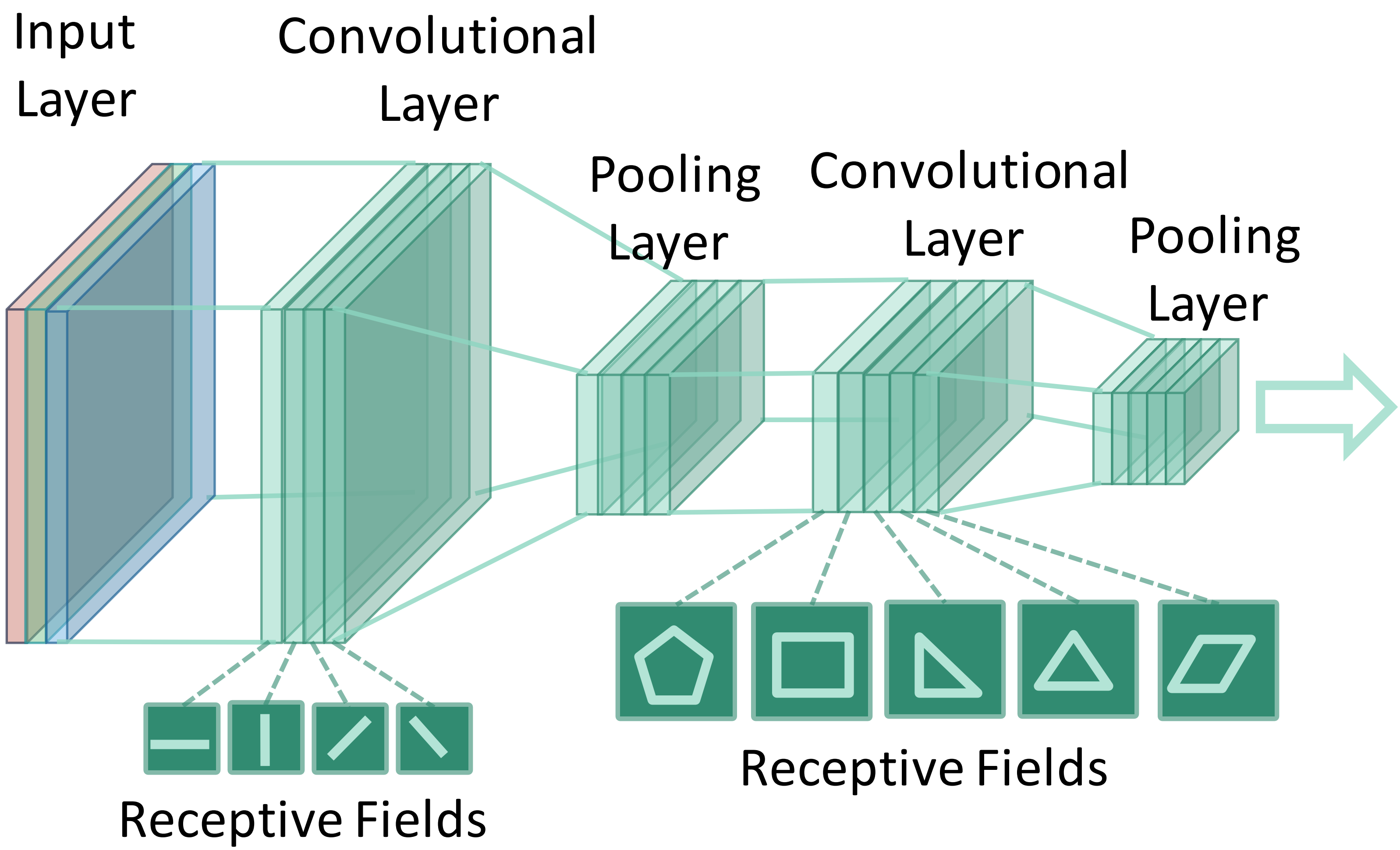


Visual Perception of Human

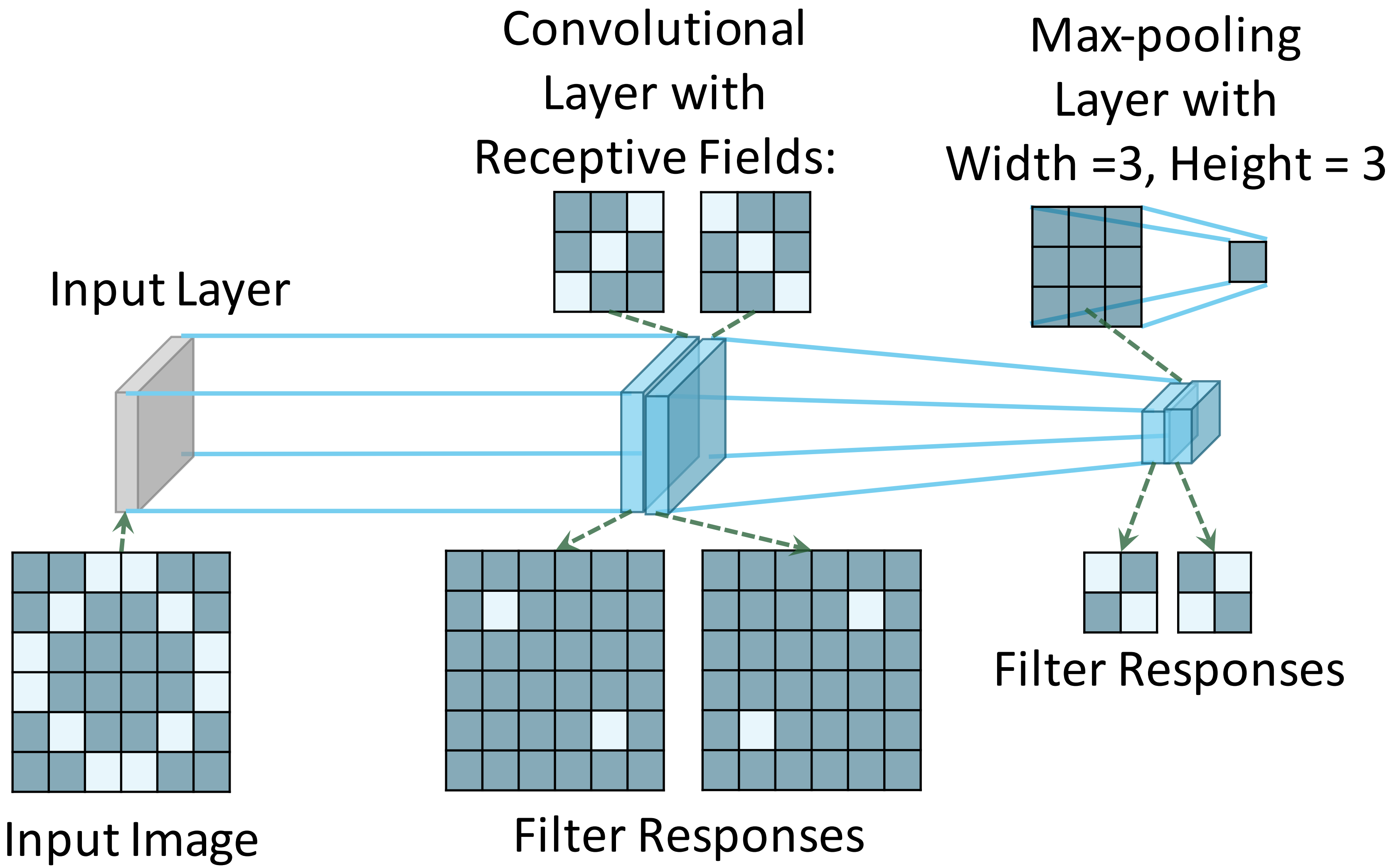


http://www.nature.com/neuro/journal/v8/n8/images/nn0805-975-F1.jpg

Visual Perception of Computer

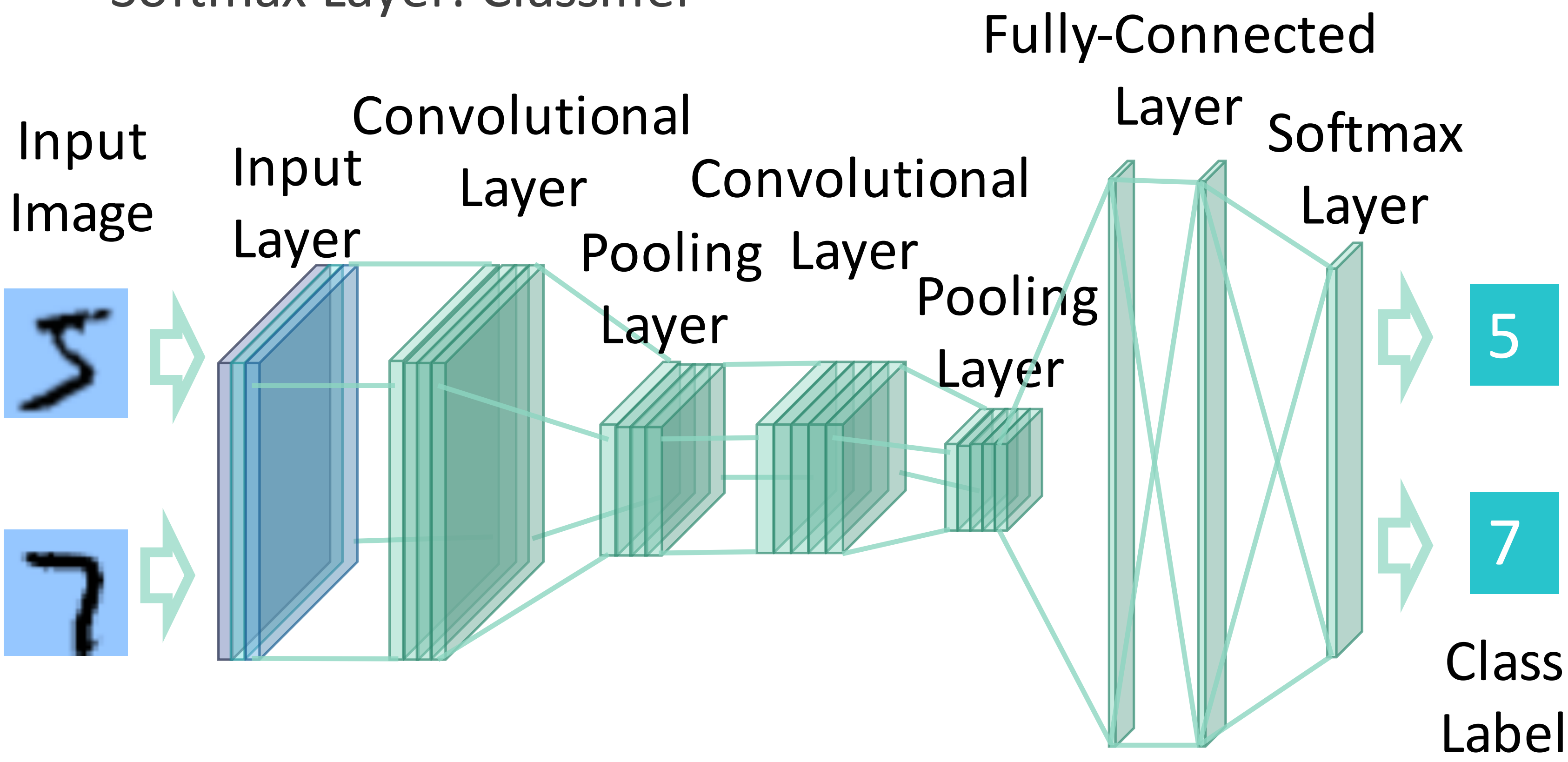


Visual Perception of Computer



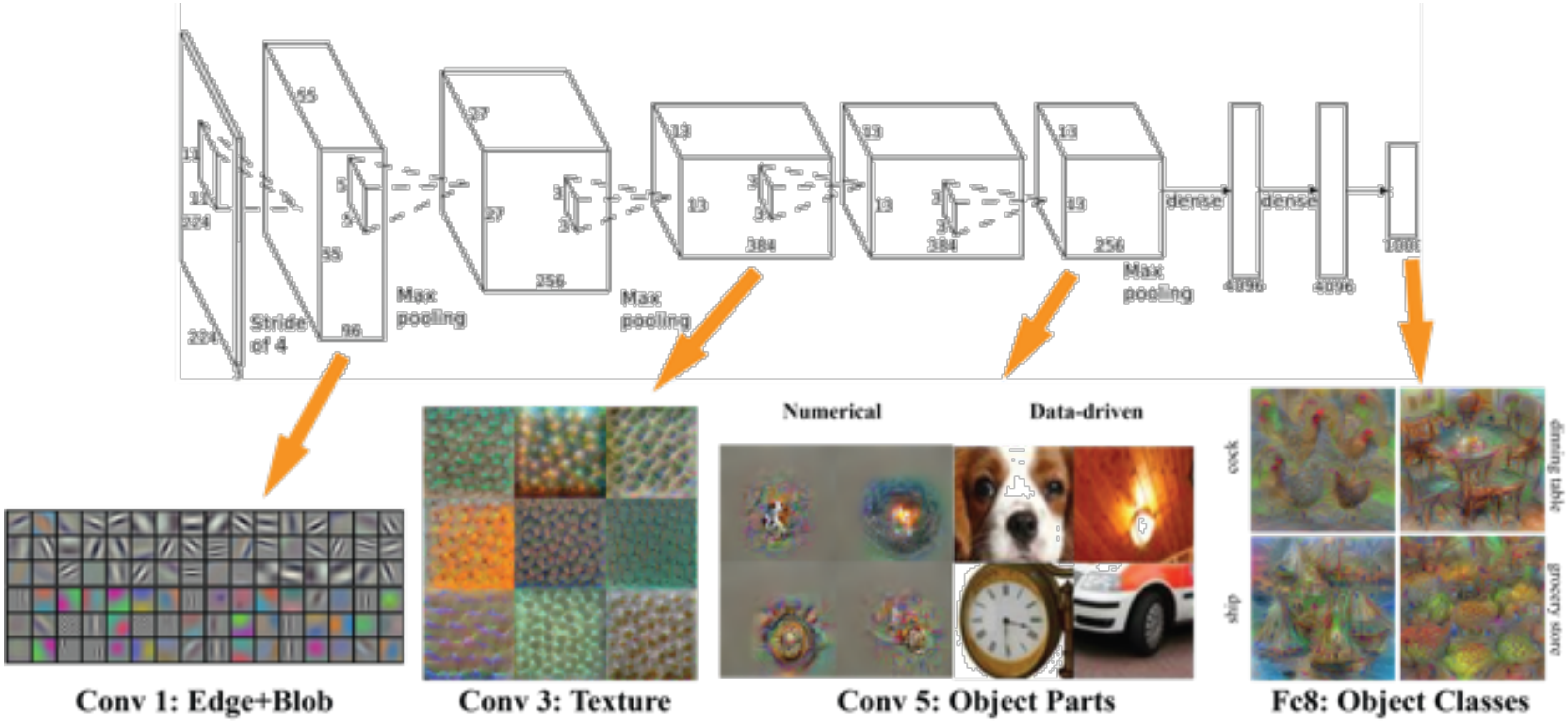
Fully-Connected Layer

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



Visual Perception of Computer

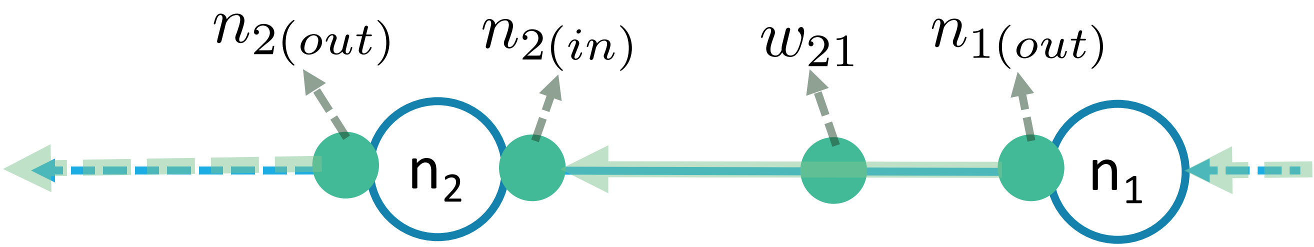
- Alexnet
  http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf



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

Training

- Forward Propagation

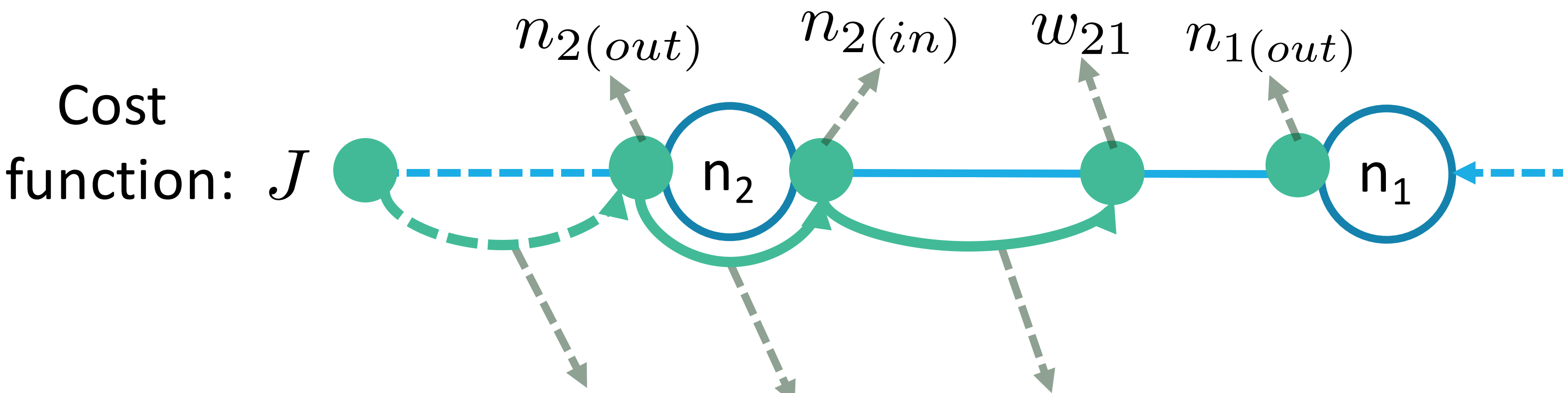


$$n_{2(in)} = w_{21} n_{1(out)}$$

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

Training

- Update weights



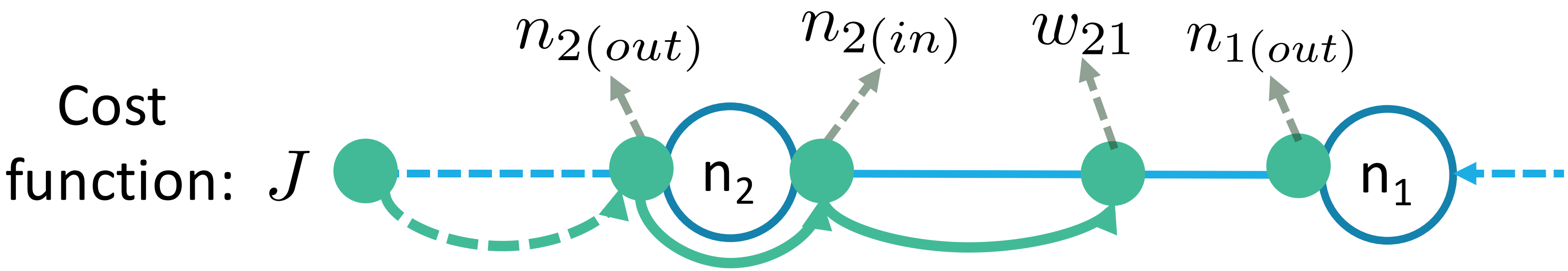
$$\frac{\partial J}{\partial w_{21}} = \frac{\partial J}{\partial n_{2(out)}} \frac{\partial n_{2(out)}}{\partial n_{2(in)}} \frac{\partial n_{2(in)}}{\partial w_{21}}$$

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

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

Training

- Update weights



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

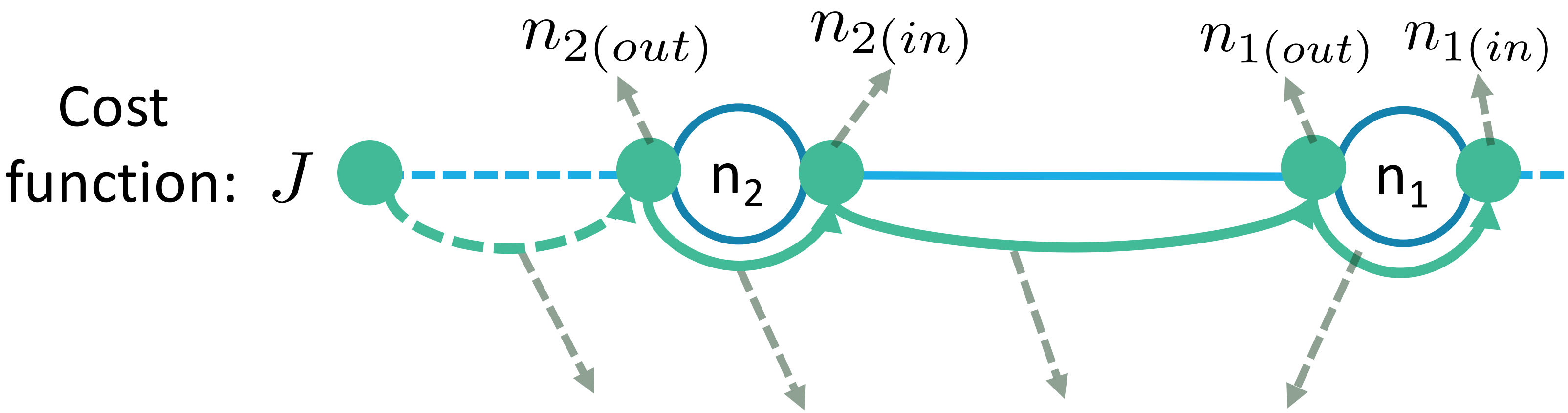
$$\Rightarrow \frac{\partial n_{2(out)}}{\partial n_{2(in)}} = g'(n_{2(in)}), \frac{\partial n_{2(in)}}{\partial w_{21}} = n_{1(out)}$$

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

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

Training

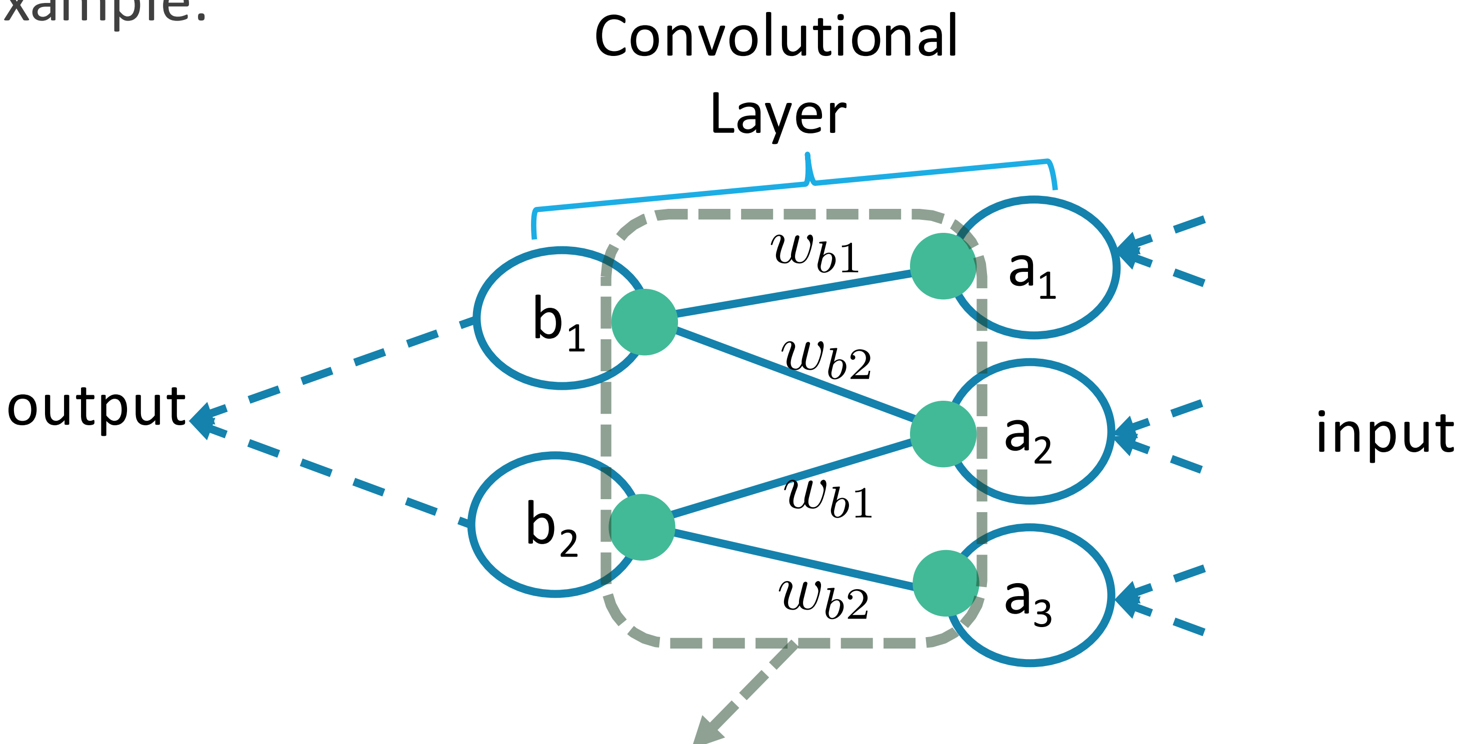
- Propagate to the previous layer



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

Training Convolutional Layers

- example:



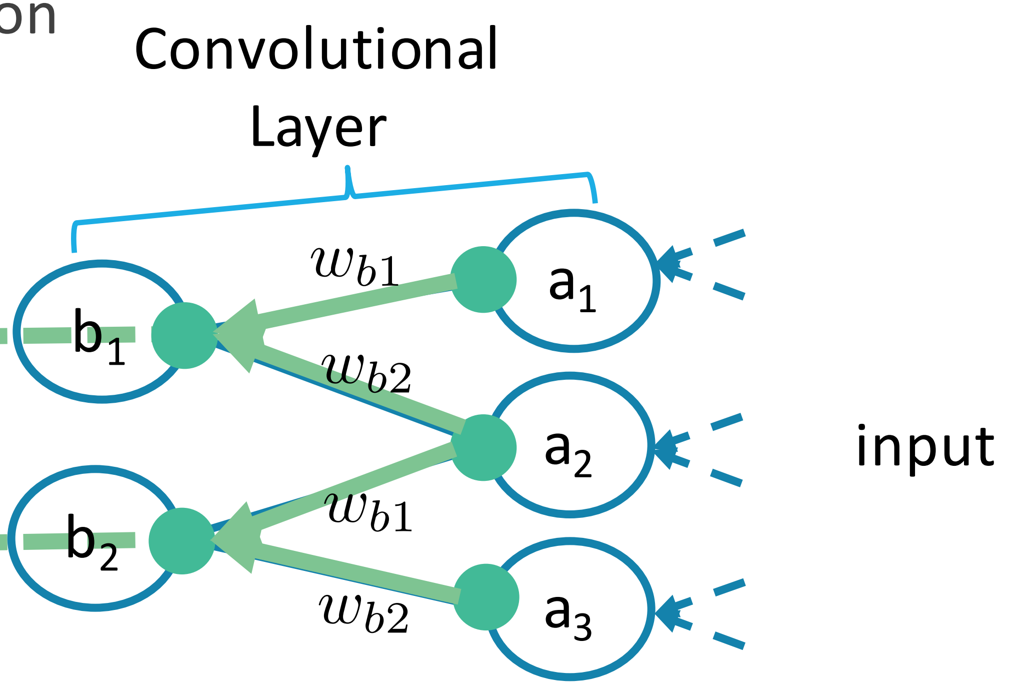
To simplify the notations, in the following slides, we make: b_1 means $b_{1(in)}$, a_1 means $a_{1(out)}$, and so on.

Training Convolutional Layers

- Forward propagation

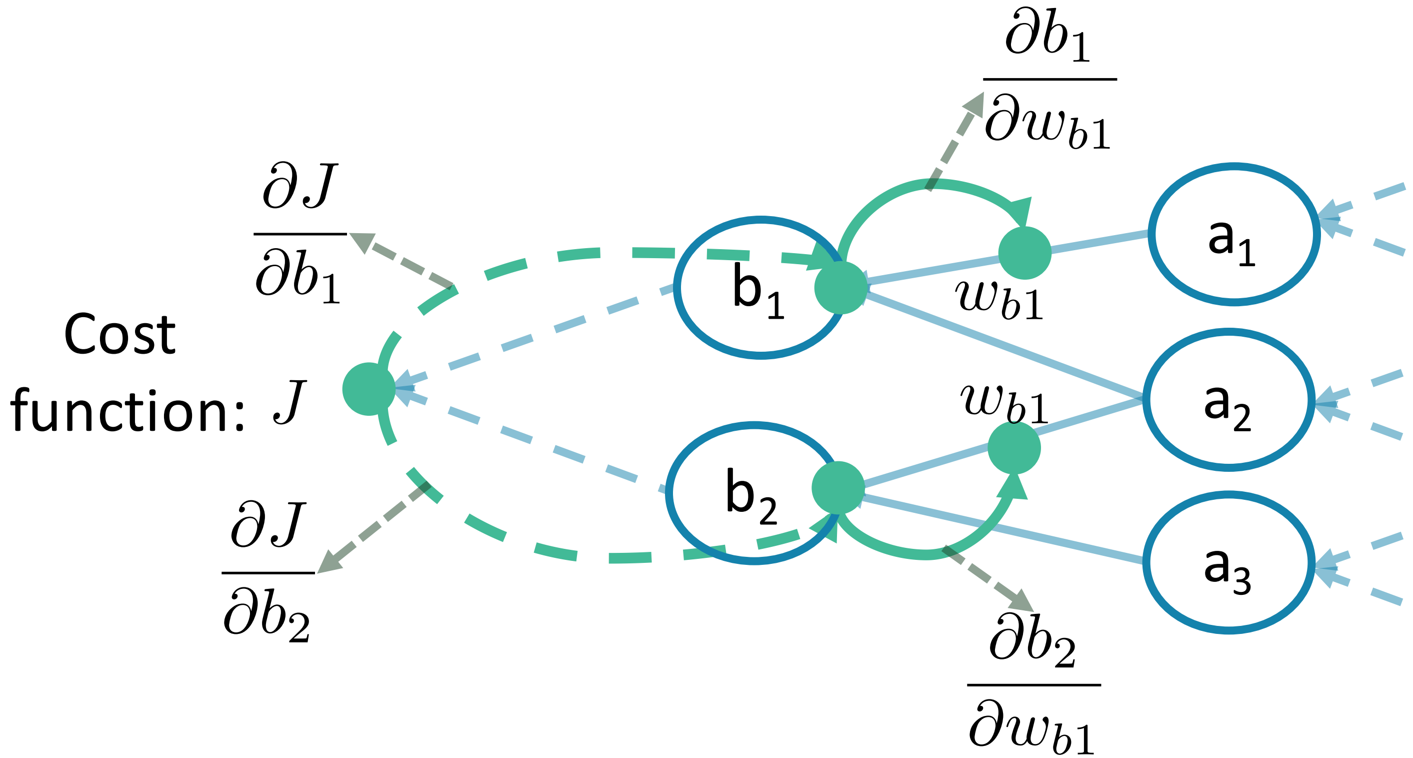
$$b_1 = w_{b1}a_1 + w_{b2}a_2$$

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



Training Convolutional Layers

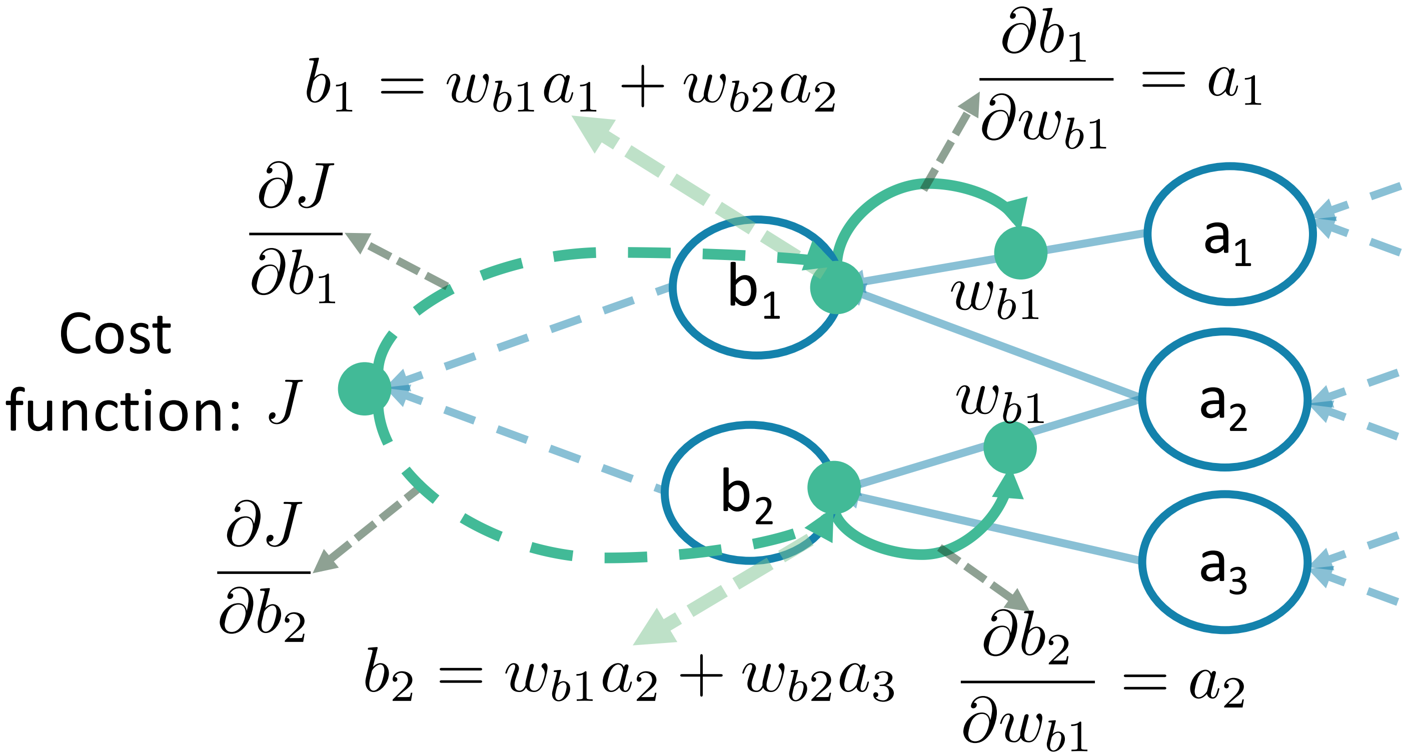
- Update weights



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

Training Convolutional Layers

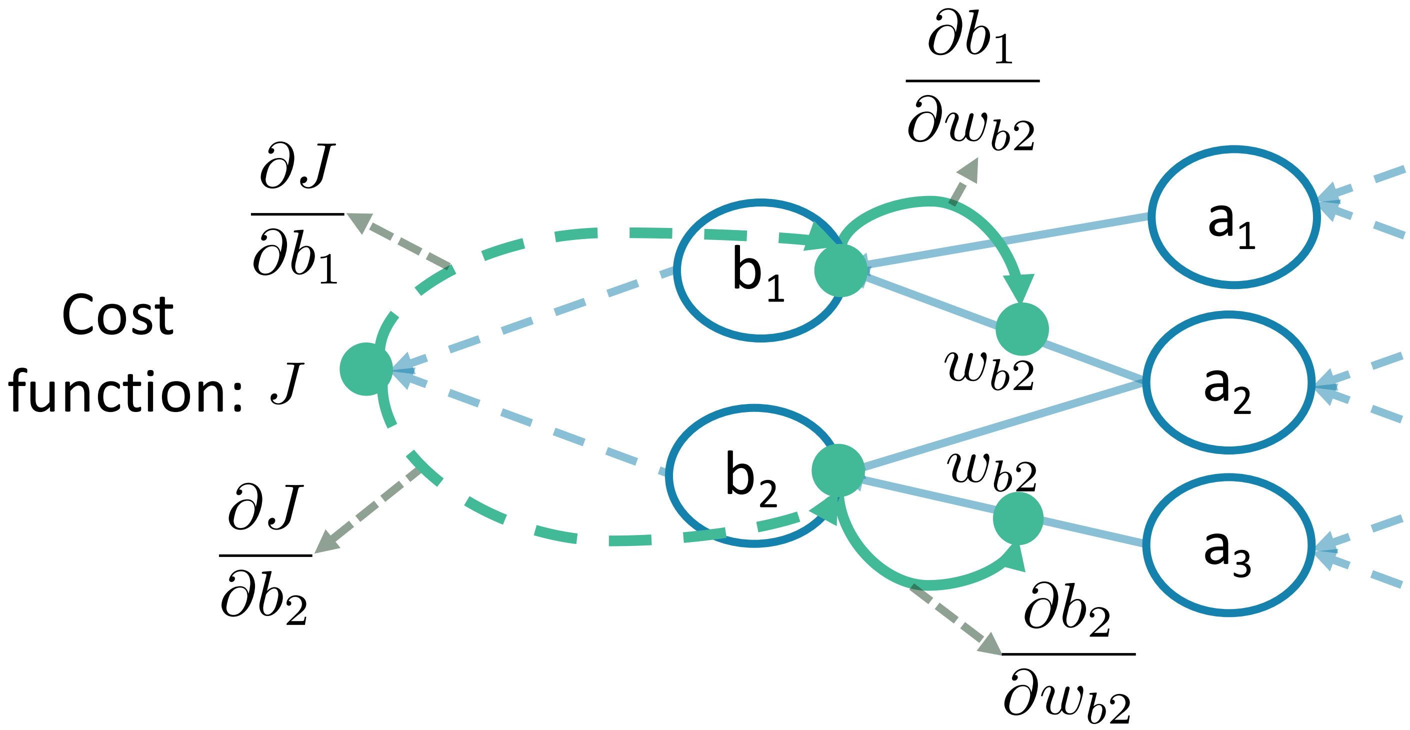
- Update weights



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

Training Convolutional Layers

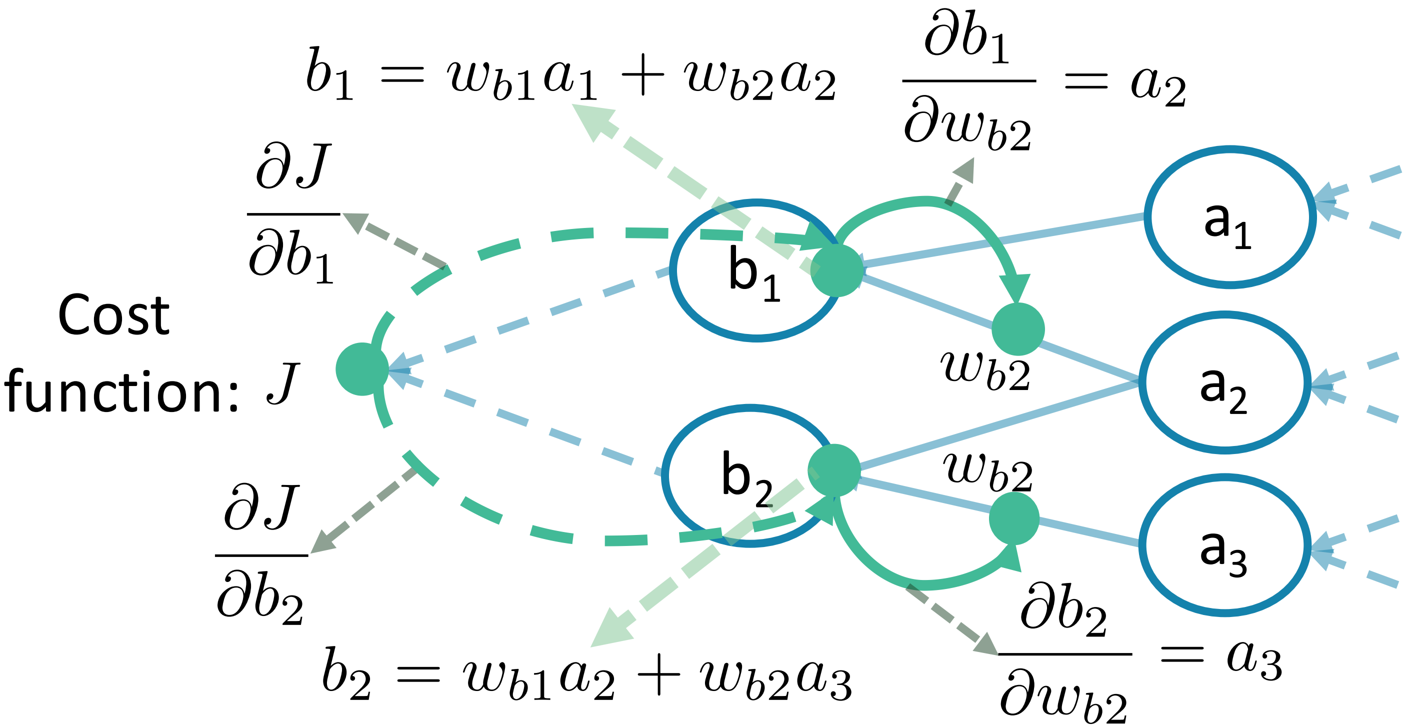
- Update weights



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

Training Convolutional Layers

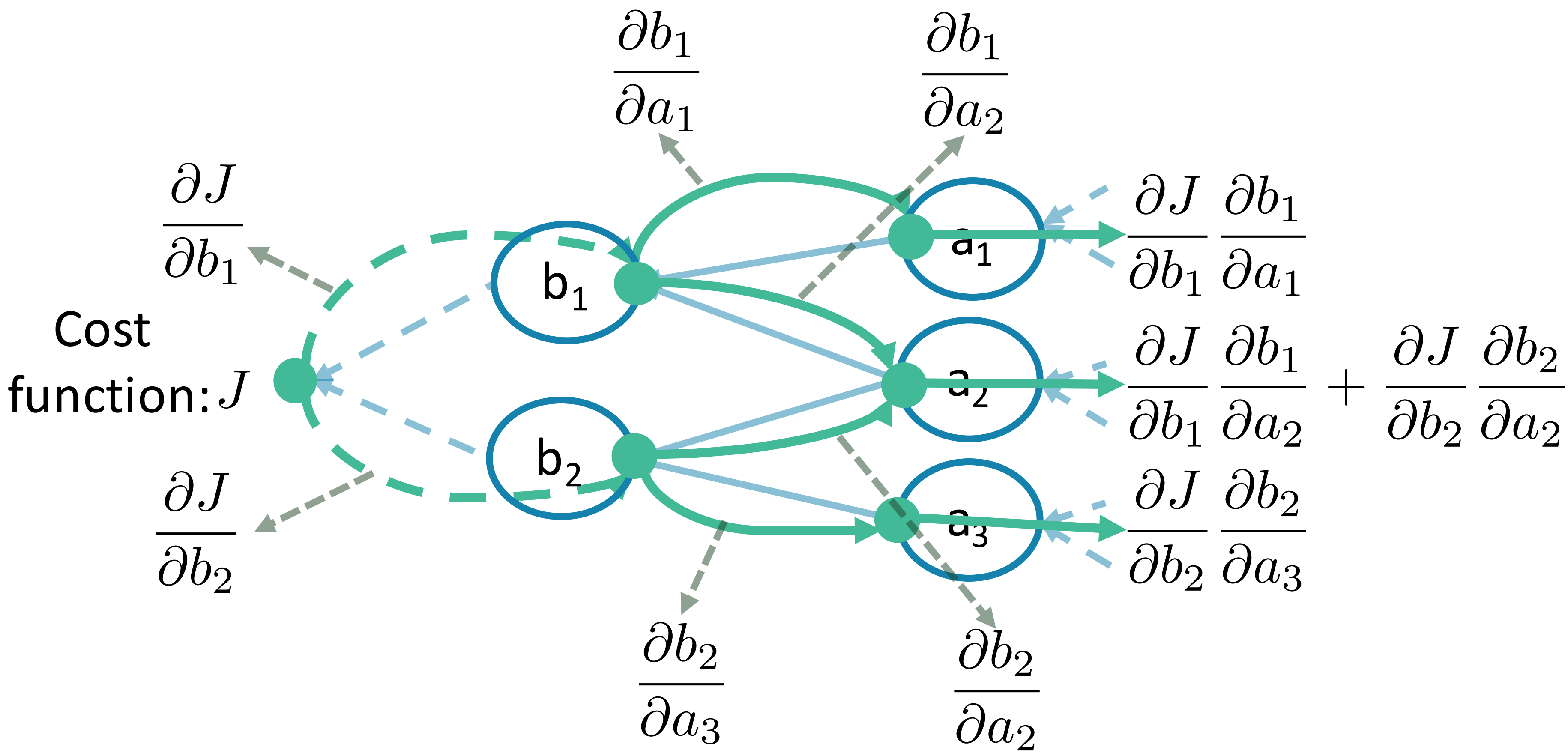
- Update weights



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

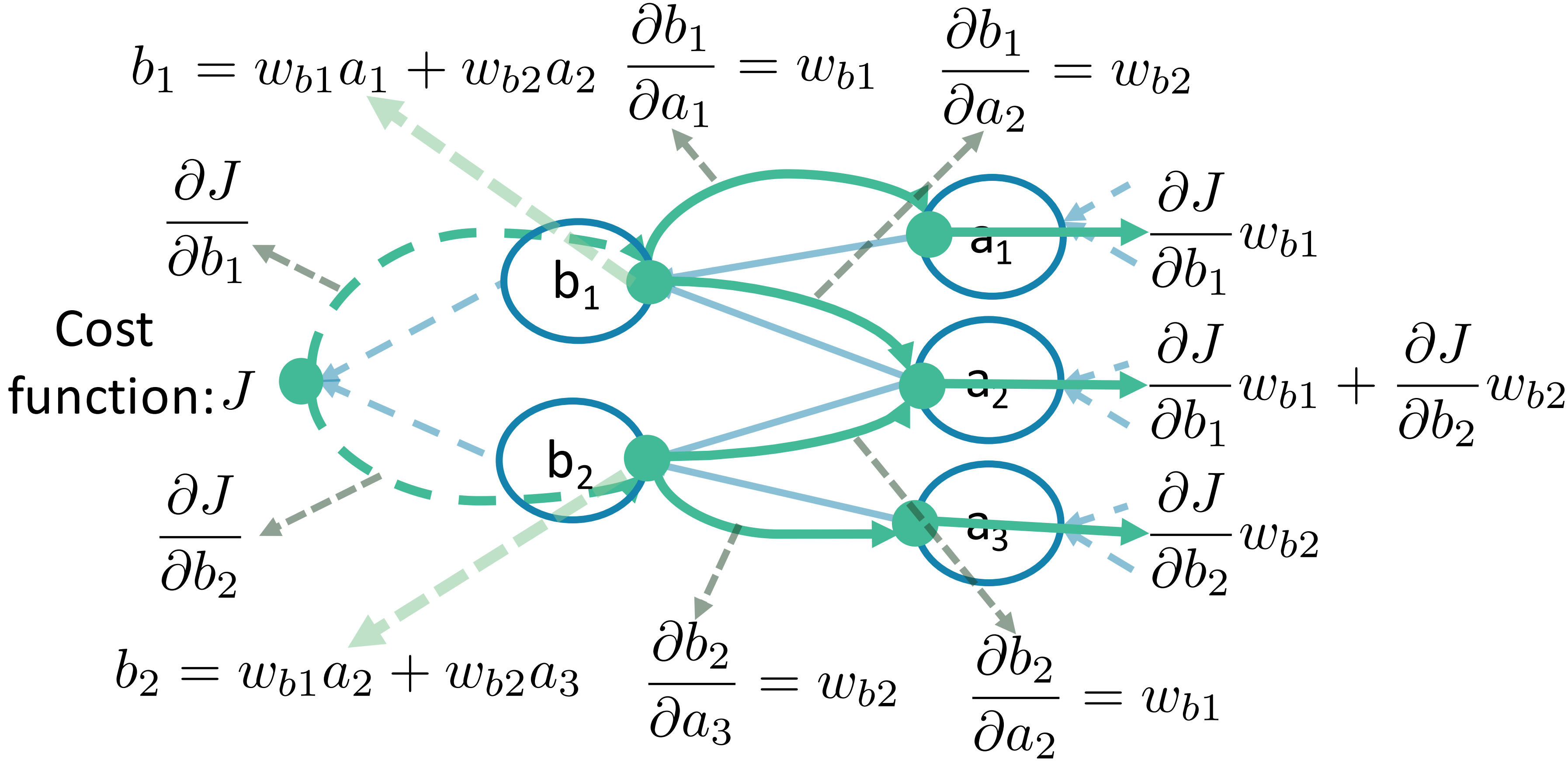
Training Convolutional Layers

- Propagate to the previous layer



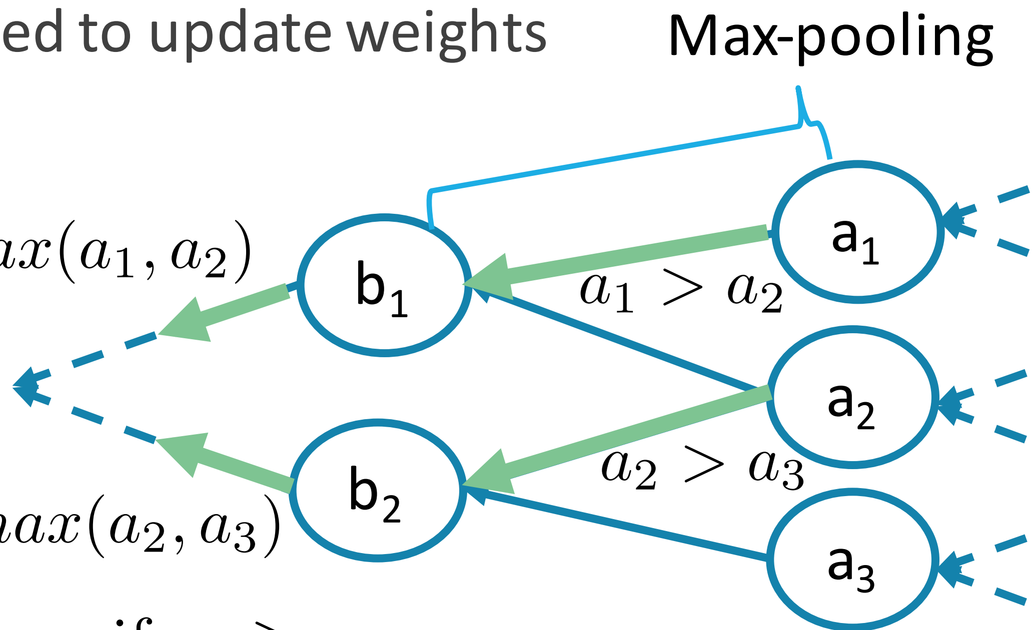
Training Convolutional Layers

- Propagate to the previous layer



Max-Pooling Layers during Training

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



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

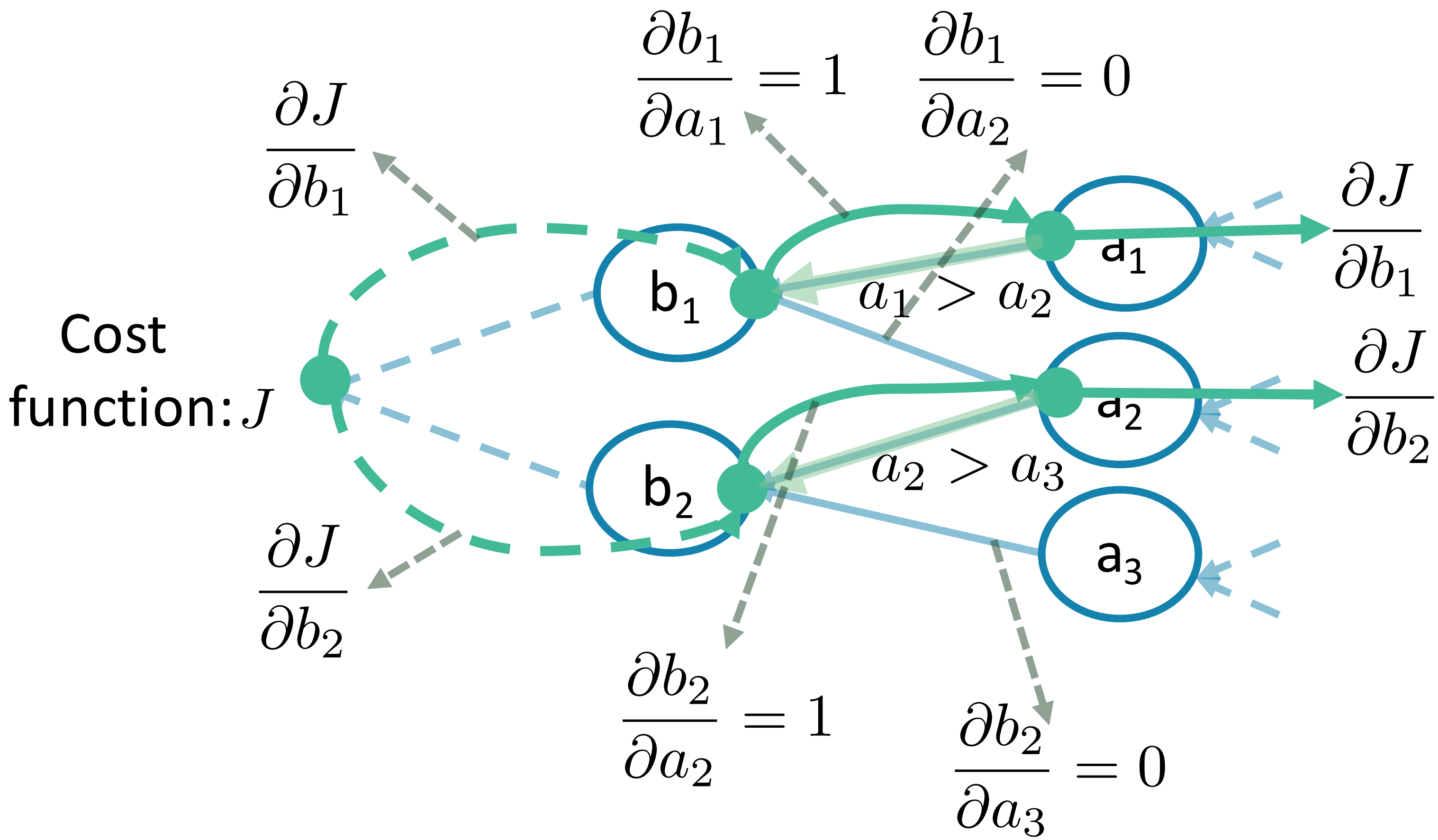
$$b_2 = max(a_2, a_3)$$

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

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

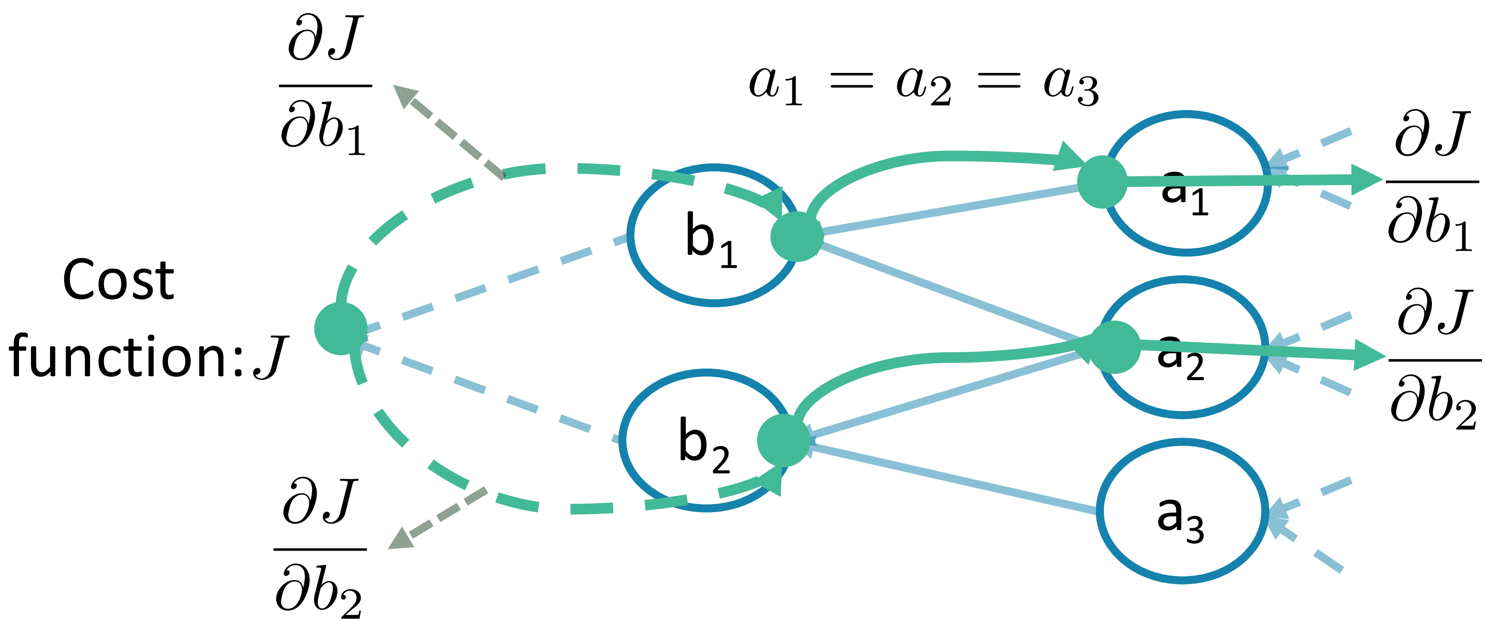
Max-Pooling Layers during Training

- Propagate to the previous layer



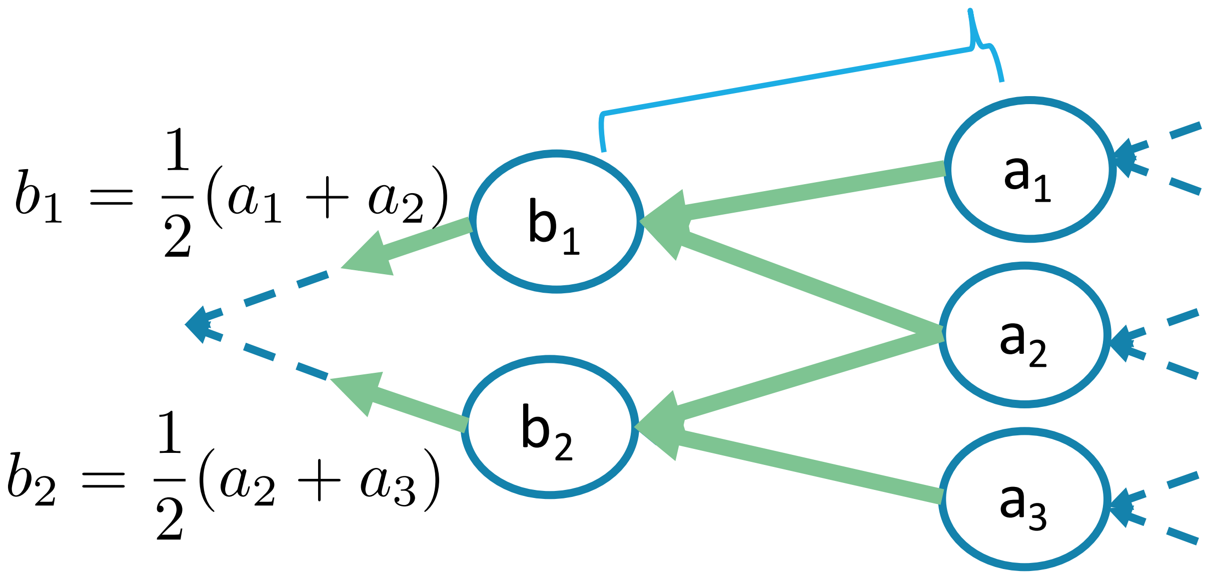
Max-Pooling Layers during Training

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



Avg-Pooling Layers during Training

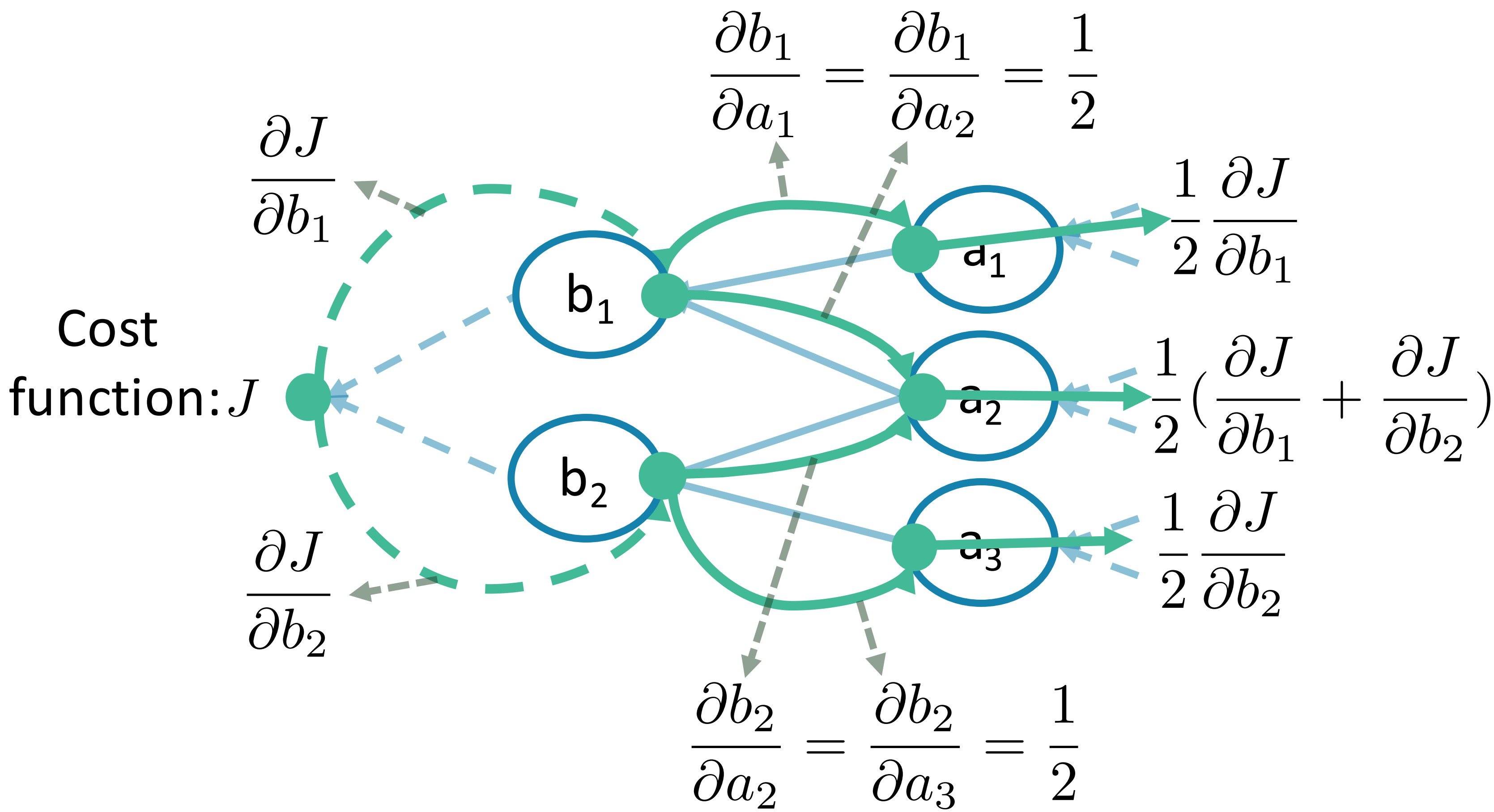
- Pooling layers have no weights
- No need to update weights



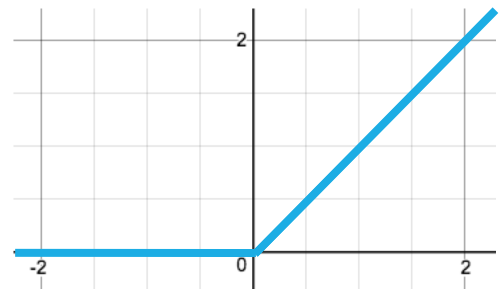
$$\frac{\partial b_2}{\partial a_2} = \frac{1}{2} \qquad \frac{\partial b_2}{\partial a_3} = \frac{1}{2}$$

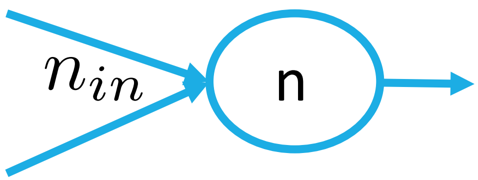
Avg-Pooling Layers during Training

- 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

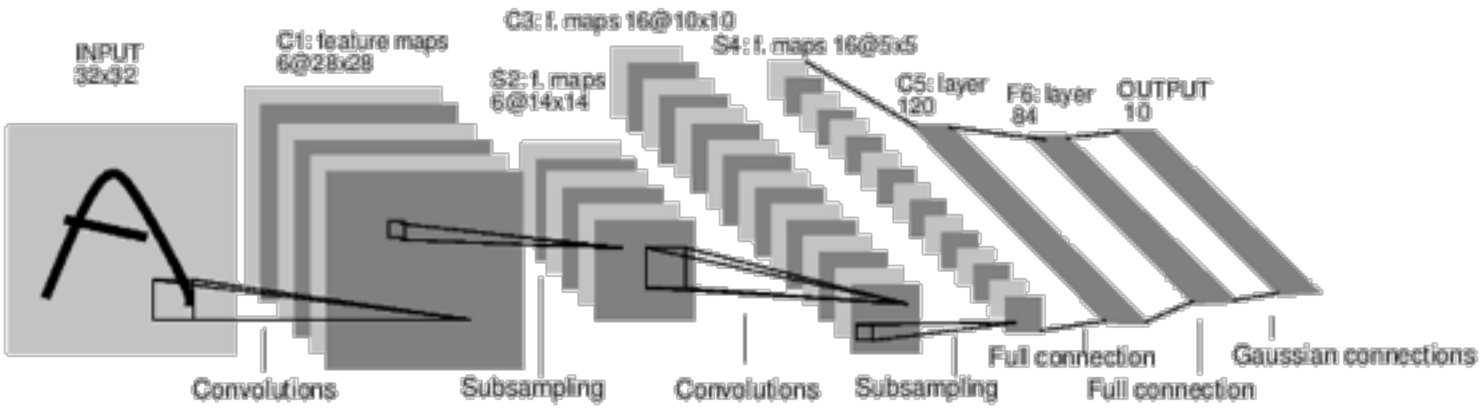
是怎樣傳過來
就怎樣傳回去

Outline

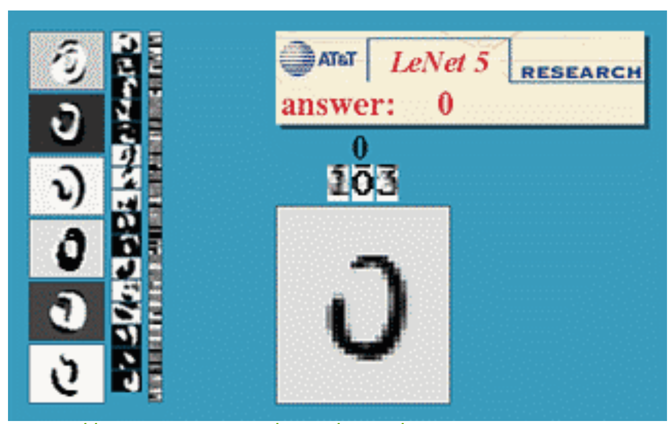
- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- Sentiment Analysis by CNN

LeNet

- Paper: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



Yann LeCun



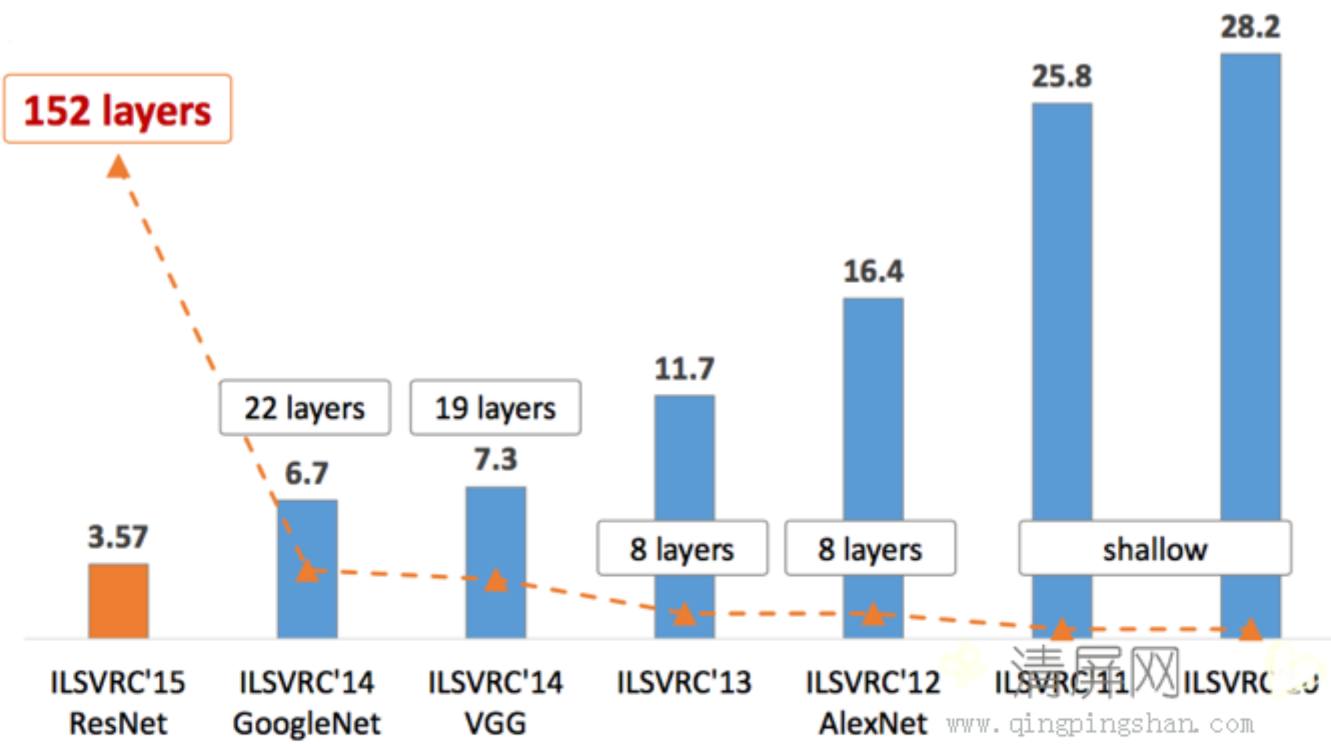
http://yann.lecun.com/exdb/lenet/

ImageNet Challenge

- ImageNet Large Scale Visual Recognition Challenge
  - http://image-net.org/challenges/LSVRC/
- Dataset :
  - 1000 categories
  - Training: 1,200,000
  - Validation: 50,000
  - Testing: 100,000

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

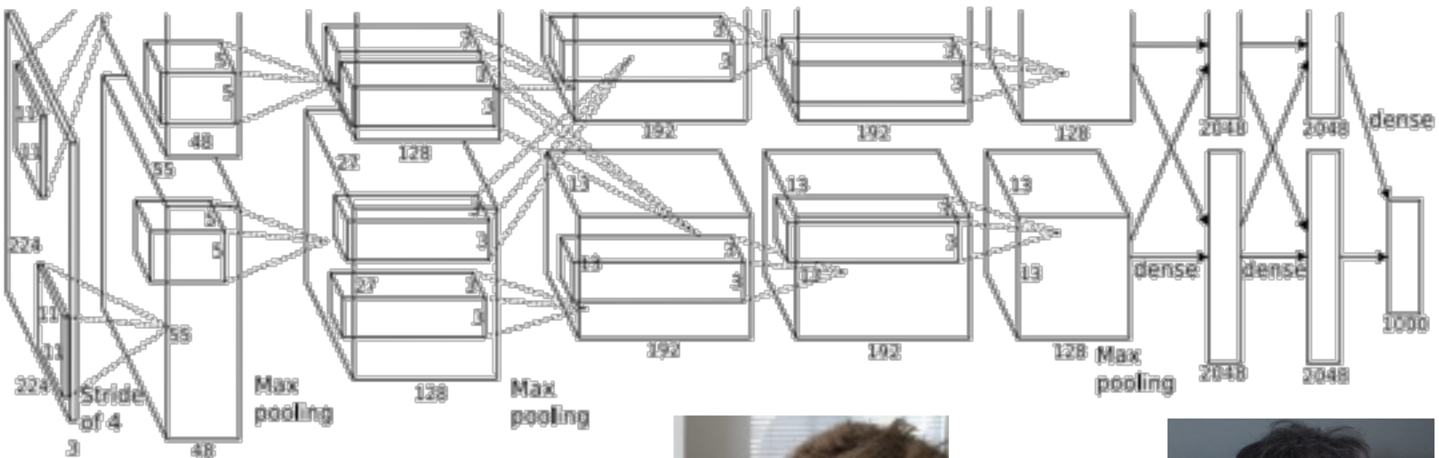
ImageNet Challenge



http://www.qingpingshan.com/uploads/allimg/160818/1J22QI5-0.png

AlexNet (2012)

- Paper: http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf
- The resurgence of Deep Learning




Alex Krizhevsky


Geoffrey Hinton

VGGNet (2014)

- Paper: https://arxiv.org/abs/1409.1556

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					

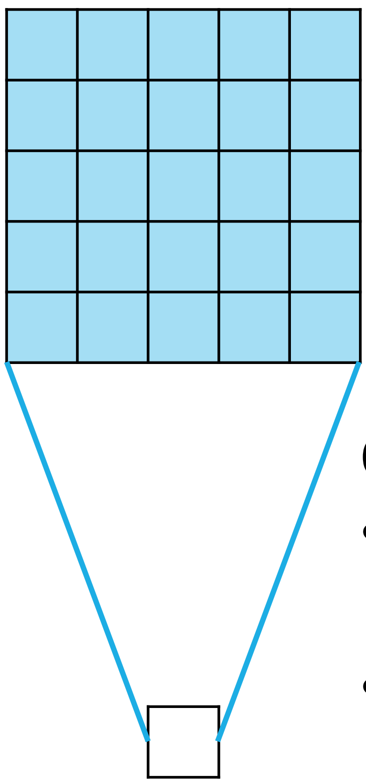
D: VGG16

E: VGG19

All filters are 3x3

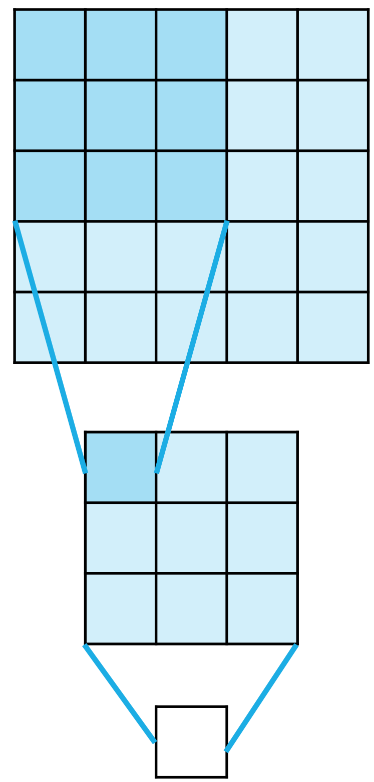
VGGNet

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



One 5x5 filter

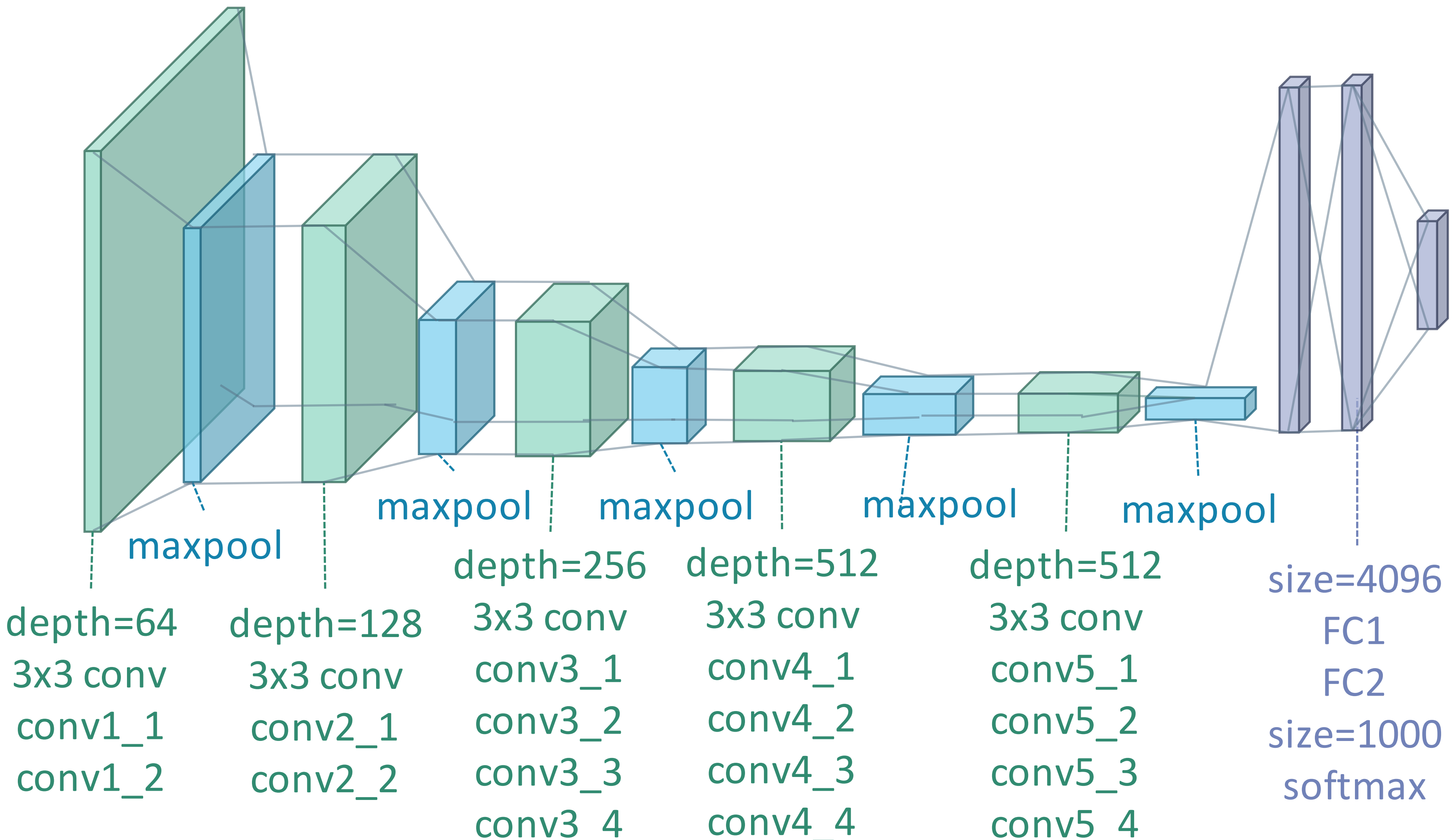
- Parameters: 5x5 = 25
- Non-linear:1



Two 3x3 filters

- Parameters: 3x3x2 = 18
- Non-linear:2

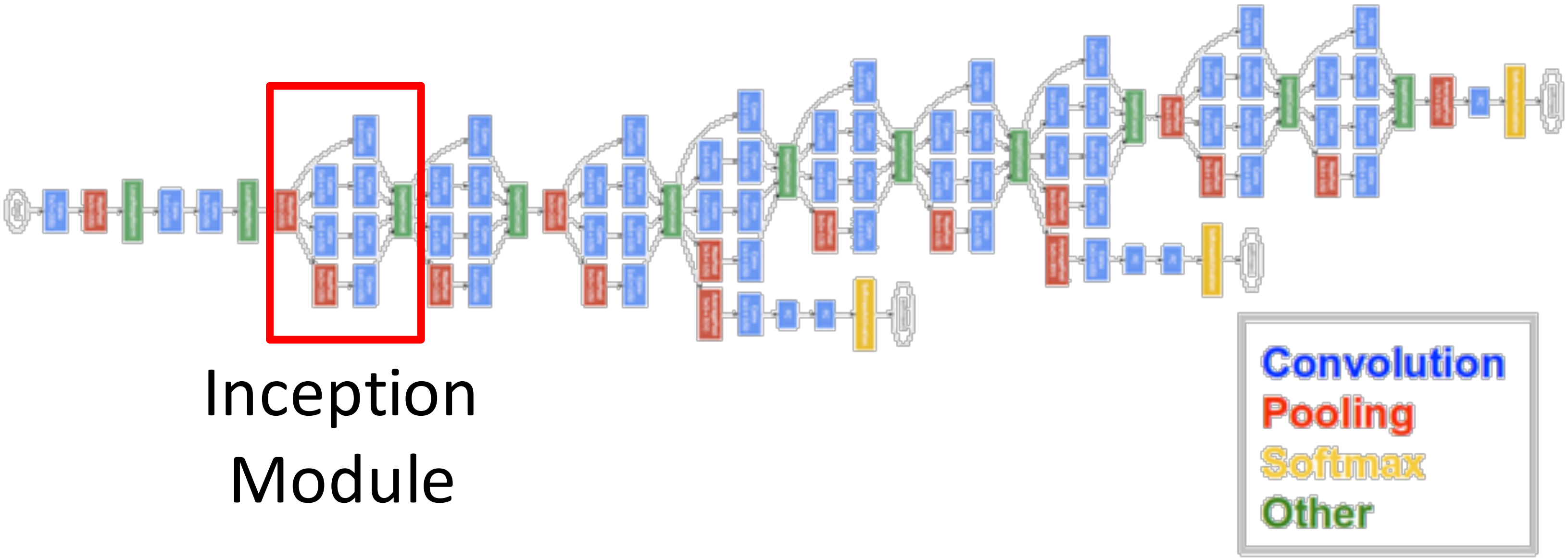
VGG 19


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)

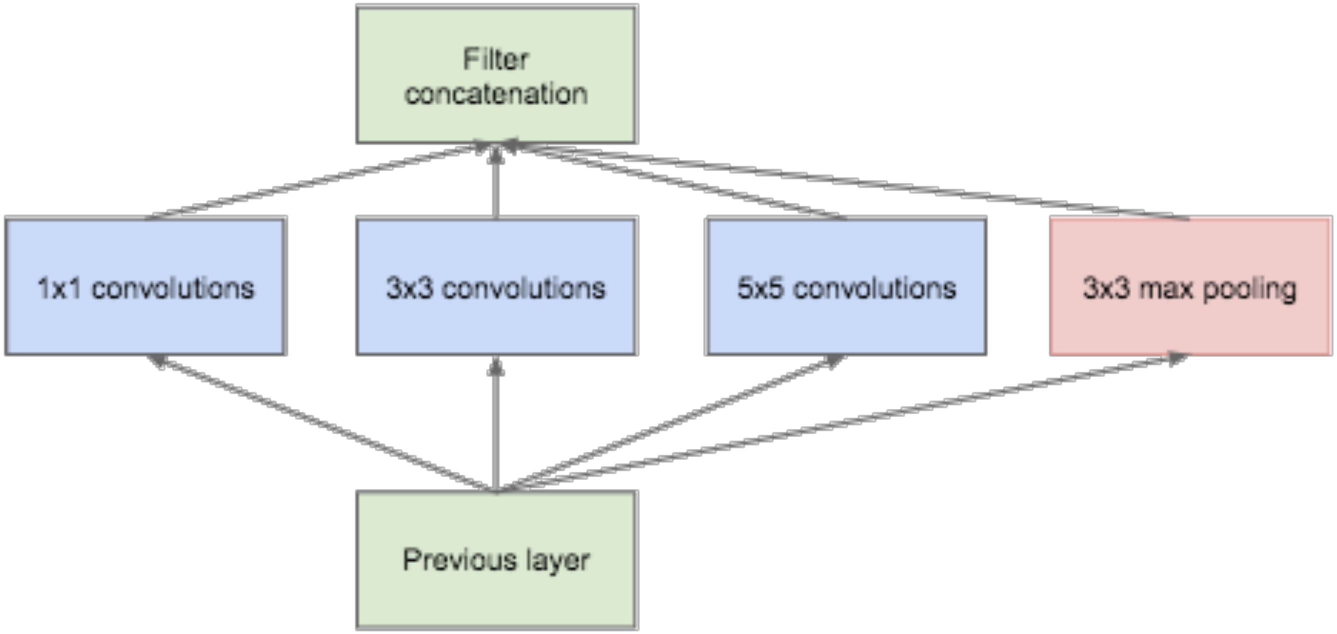
- Paper: http://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

22 layers deep network

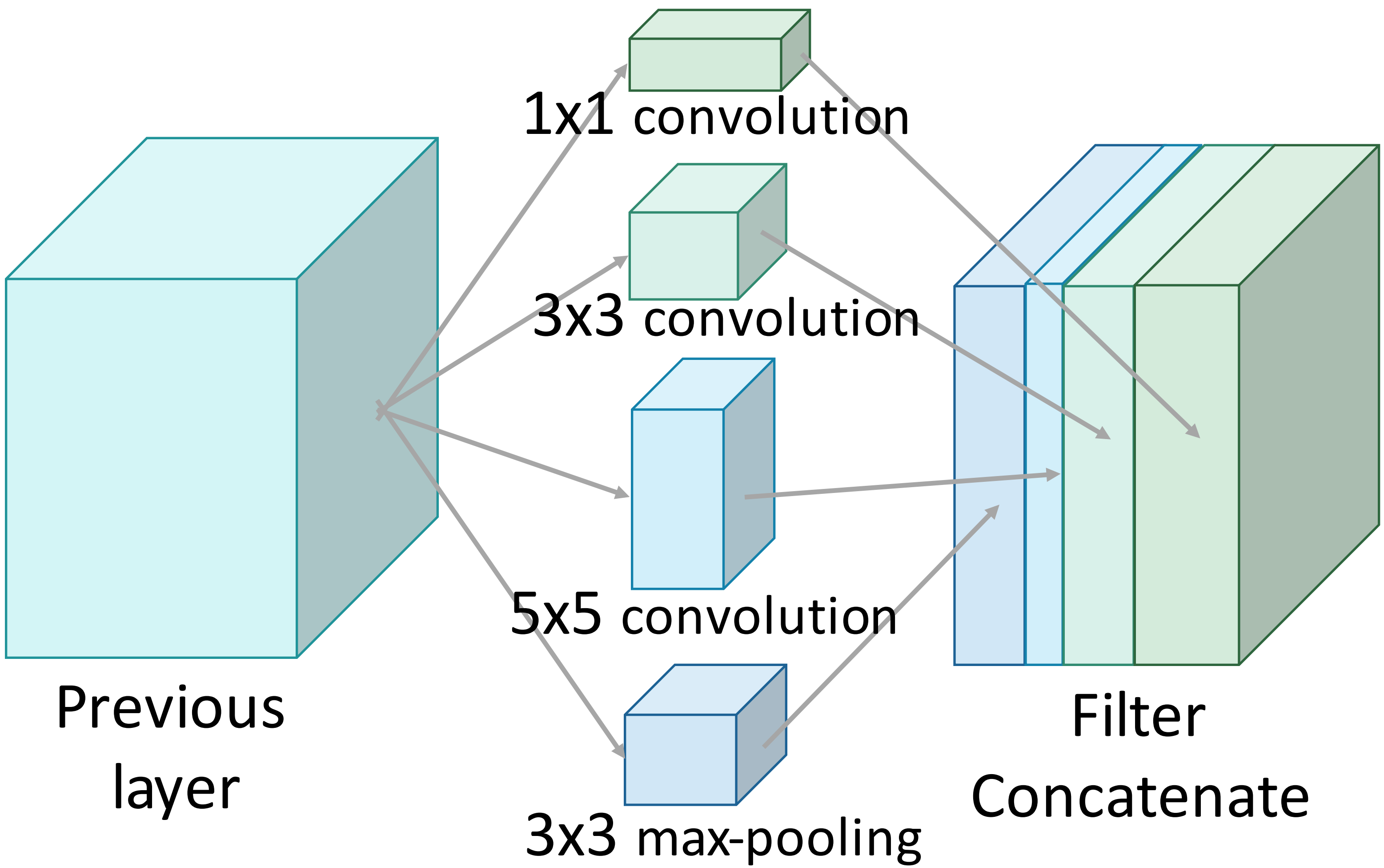


Inception Module

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

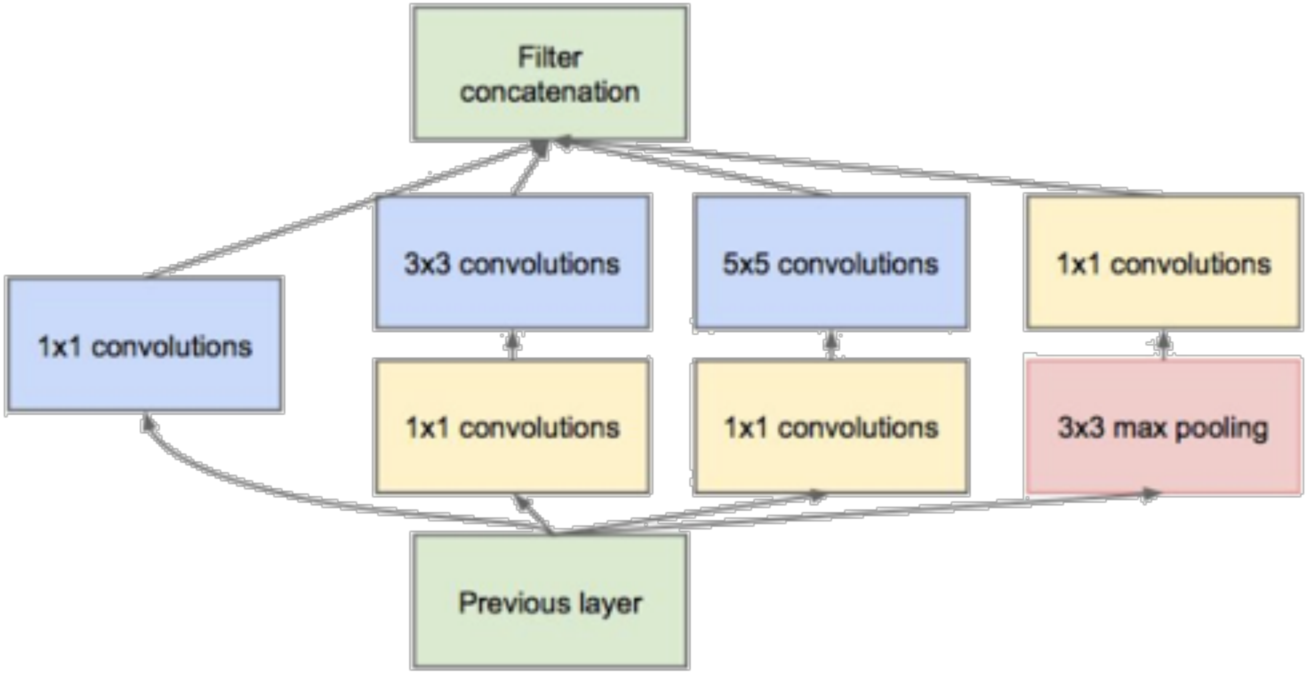


Inception Module

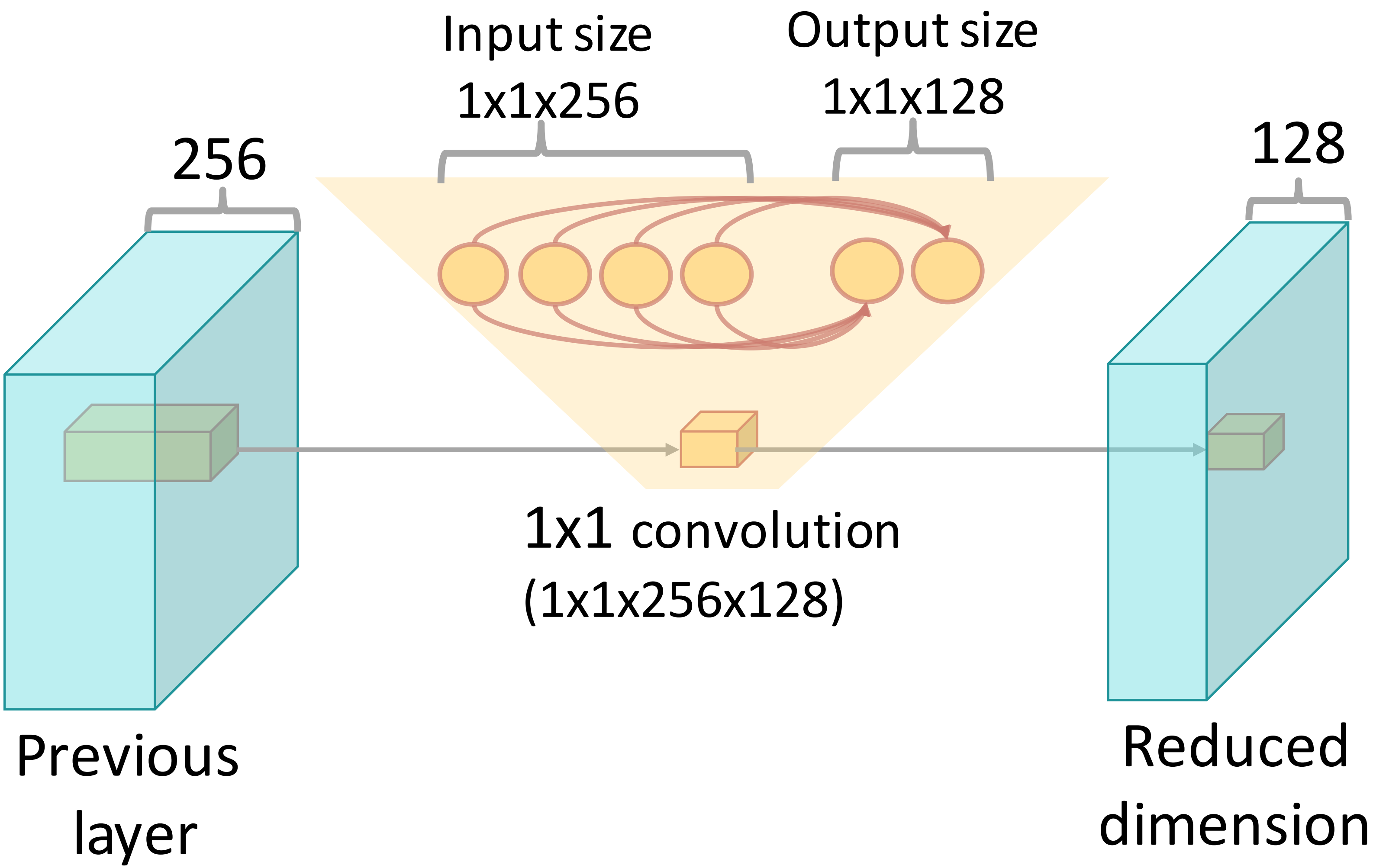


Inception Module with Dimension Reduction

- Use 1x1 filters to reduce dimension

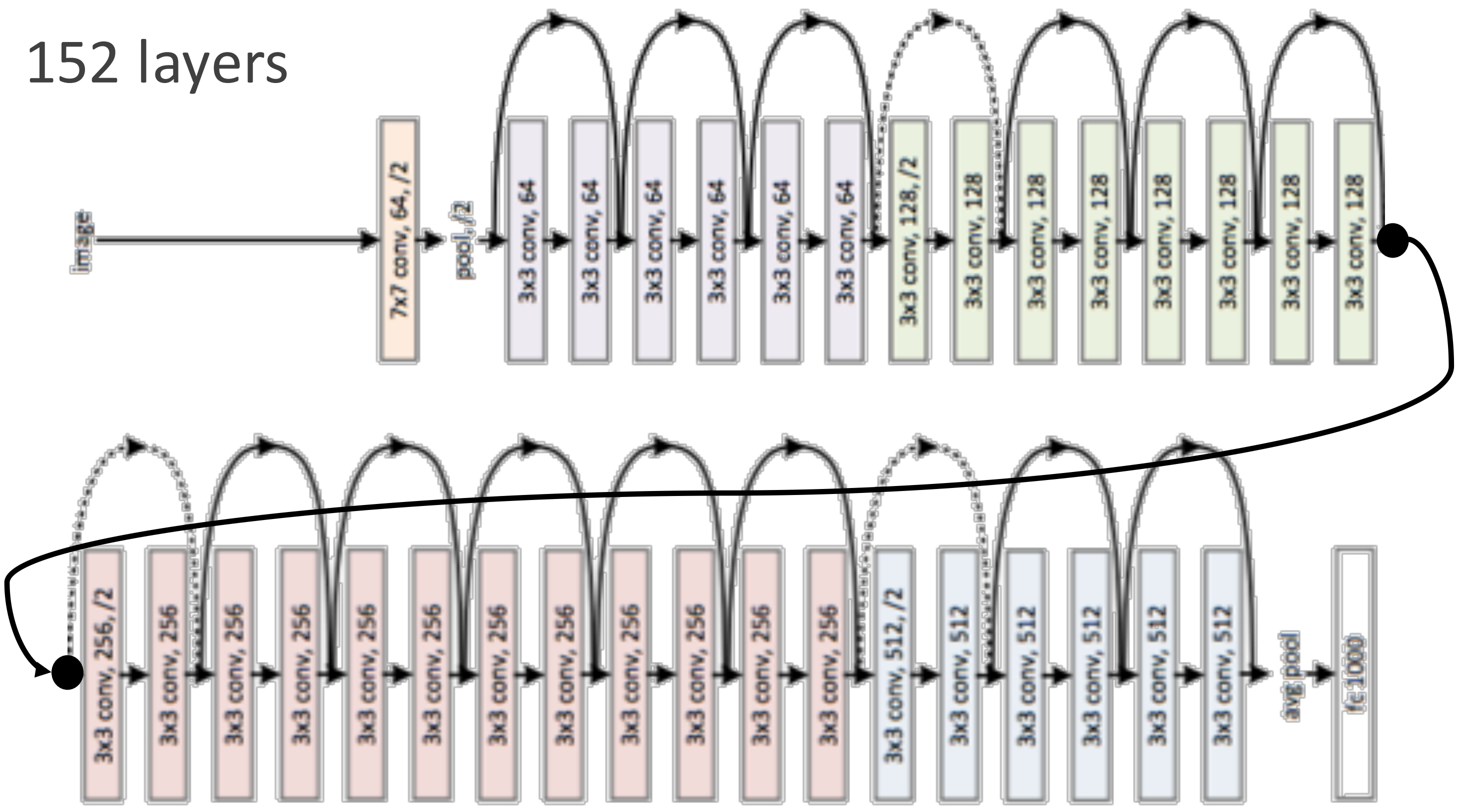


Inception Module with Dimension Reduction



ResNet (2015)

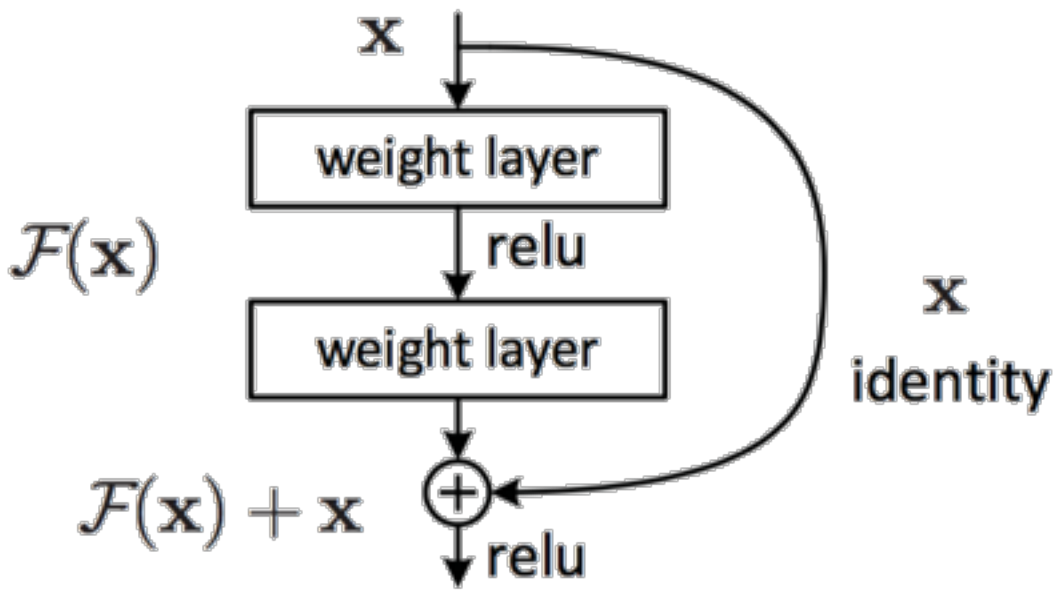
- Paper: https://arxiv.org/abs/1512.03385
- Residual Networks
- 152 layers



ResNet

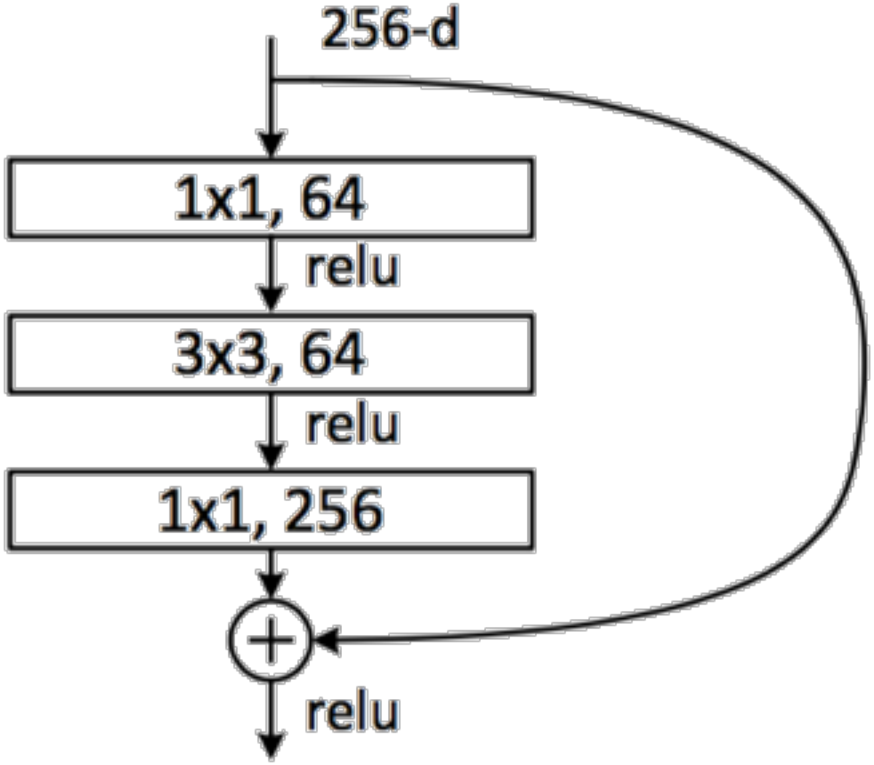
- Residual learning: a building block

Residual function



Residual Learning with Dimension Reduction

- using 1x1 filters

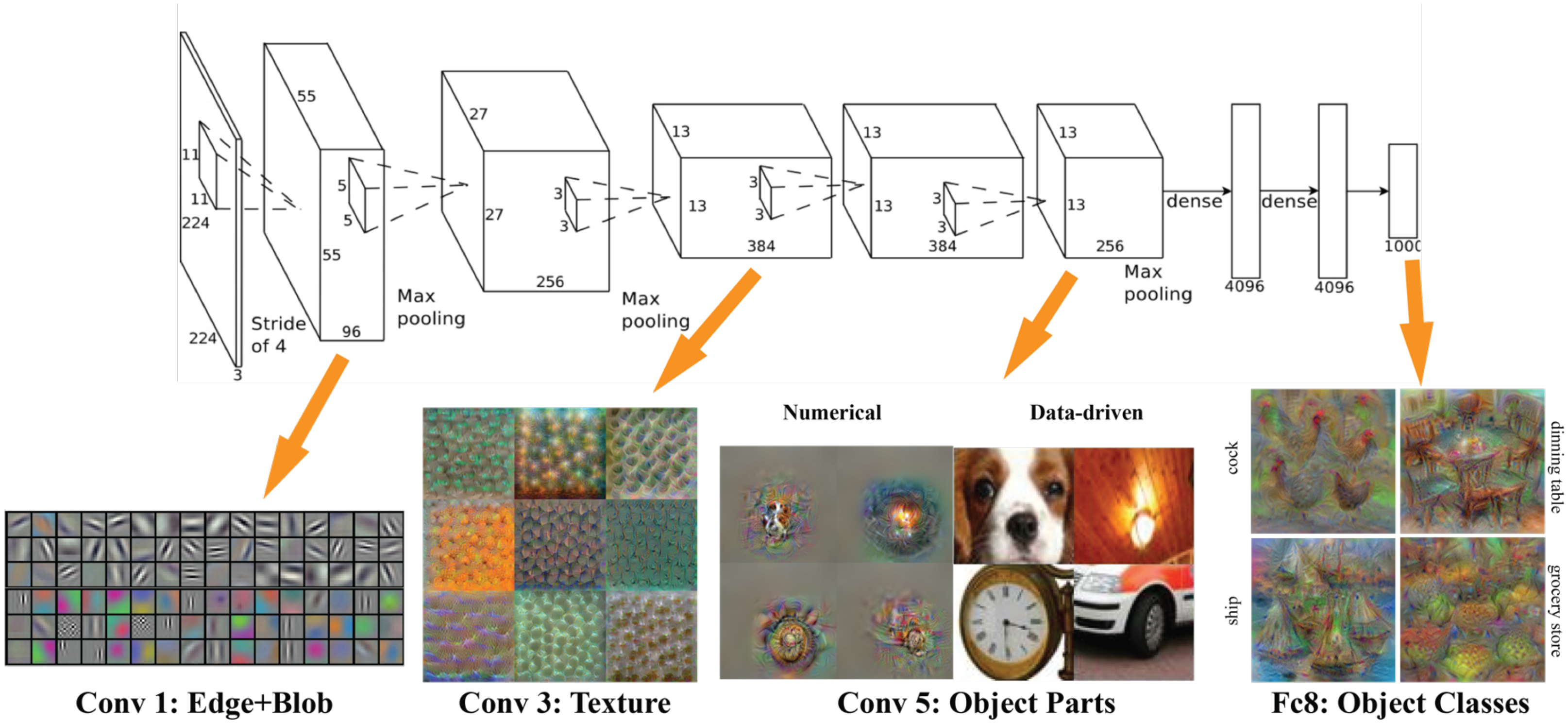


Pretrained Model Download

- 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)

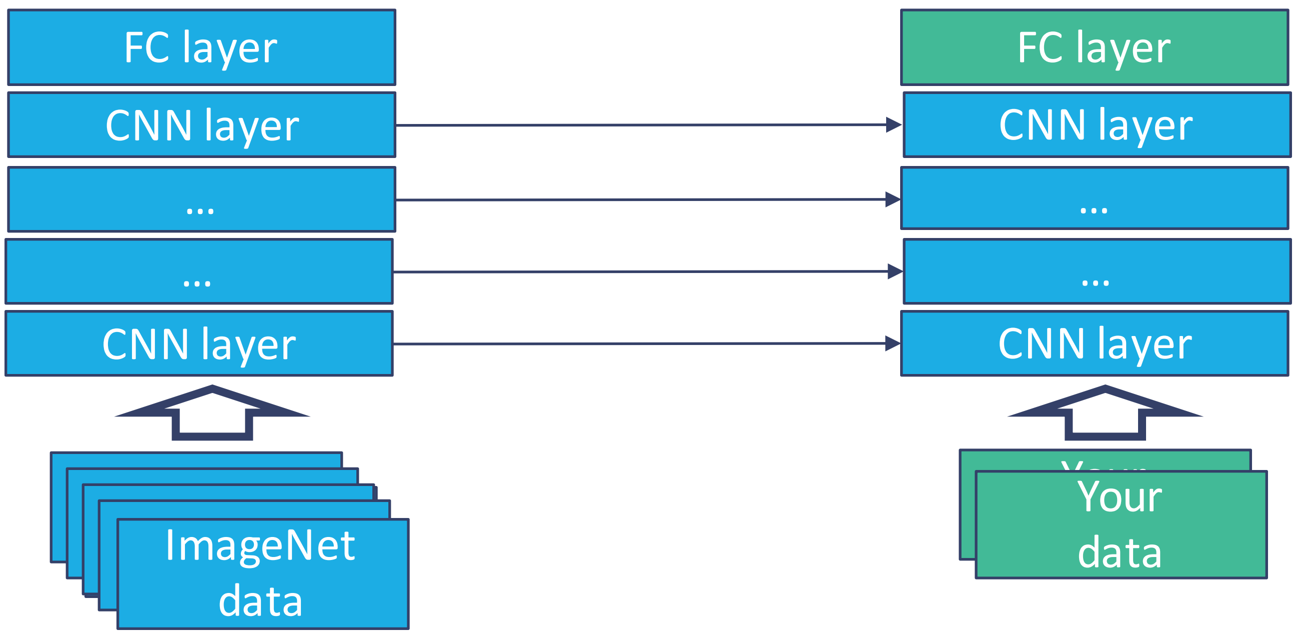


http://vision03.csail.mit.edu/cnn_art/data/single_layer.png

Transfer Learning

- The Pretrained Model is trained on ImageNet dataset

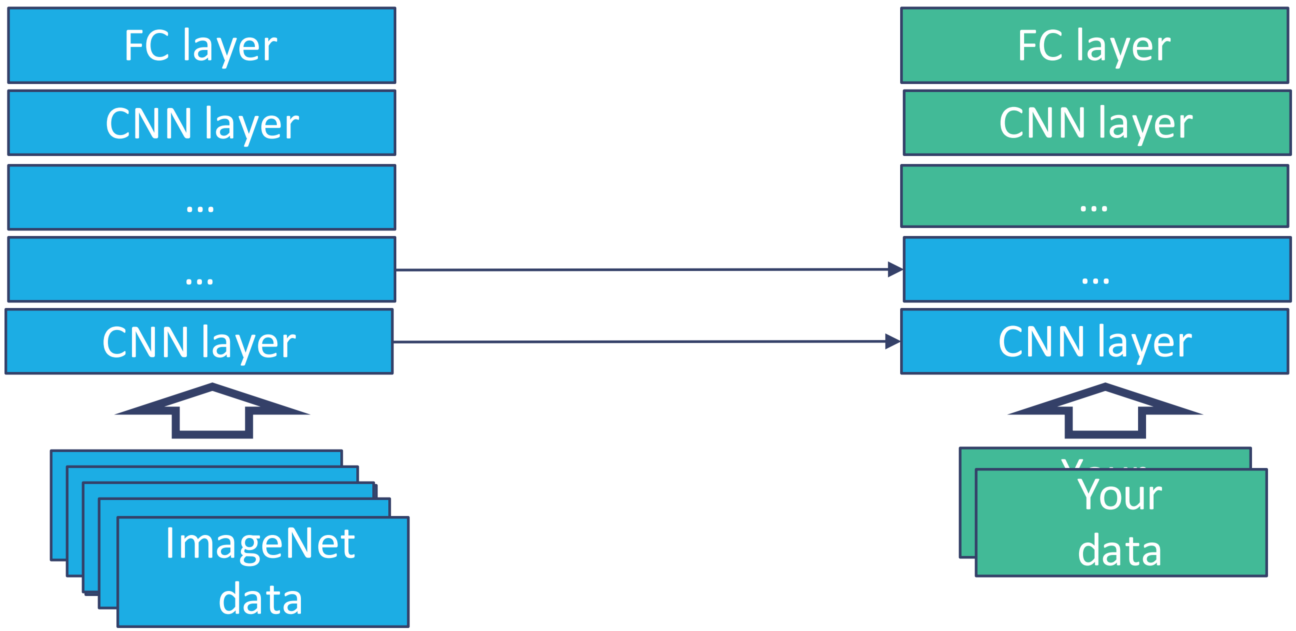
- If your data is similar to the ImageNet data
  - Fix all CNN Layers
  - Train FC layer



Transfer Learning

- The Pretrained Model is trained on ImageNet dataset

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



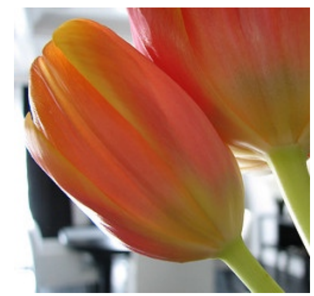
Tensorflow Transfer Learning Example

- https://www.tensorflow.org/versions/r0.11/how_tos/style_guide.html

daisy
634
photos

dandelion
899
photos

roses
642
photos

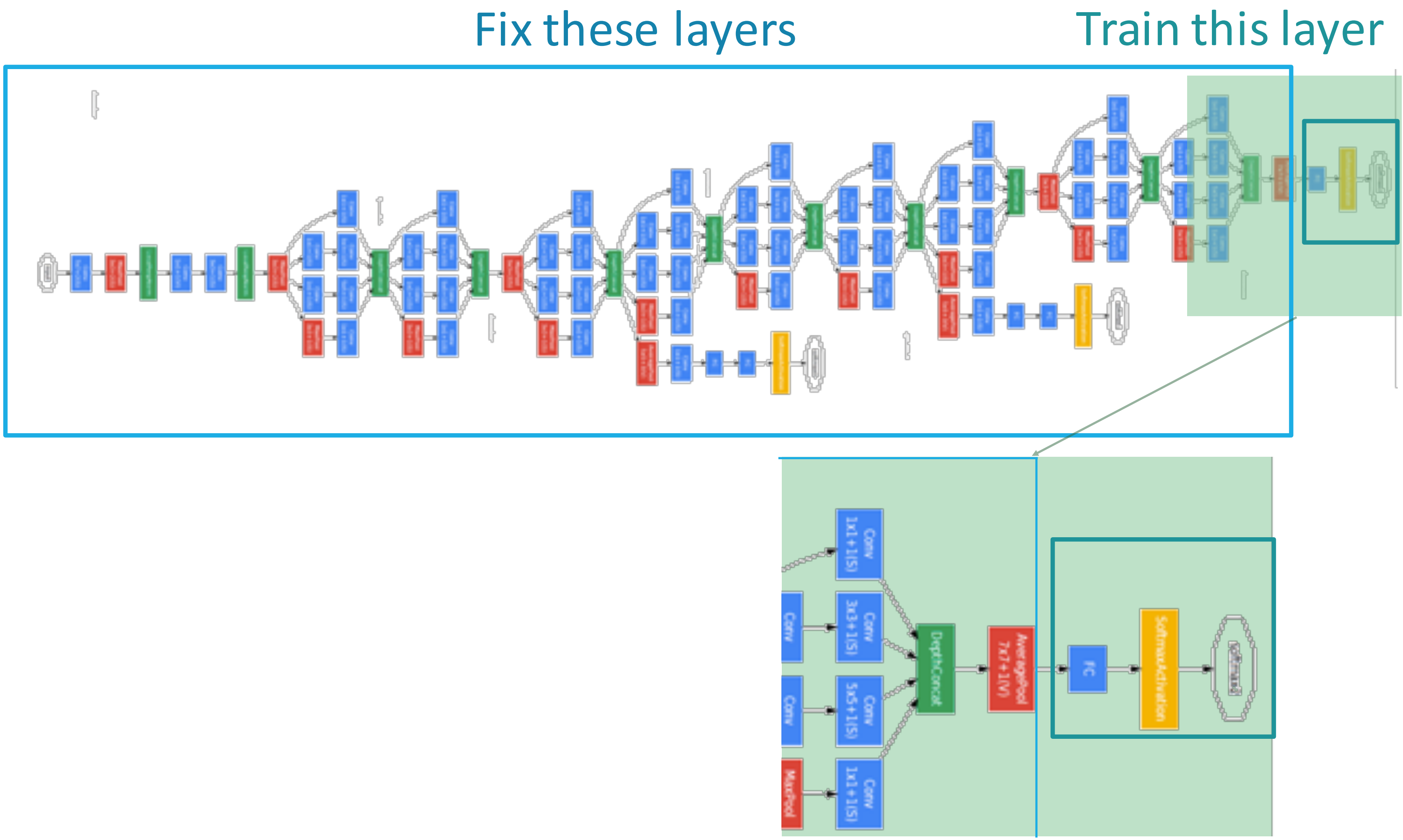


tulips
800
photos

sunflowers
700
photos

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

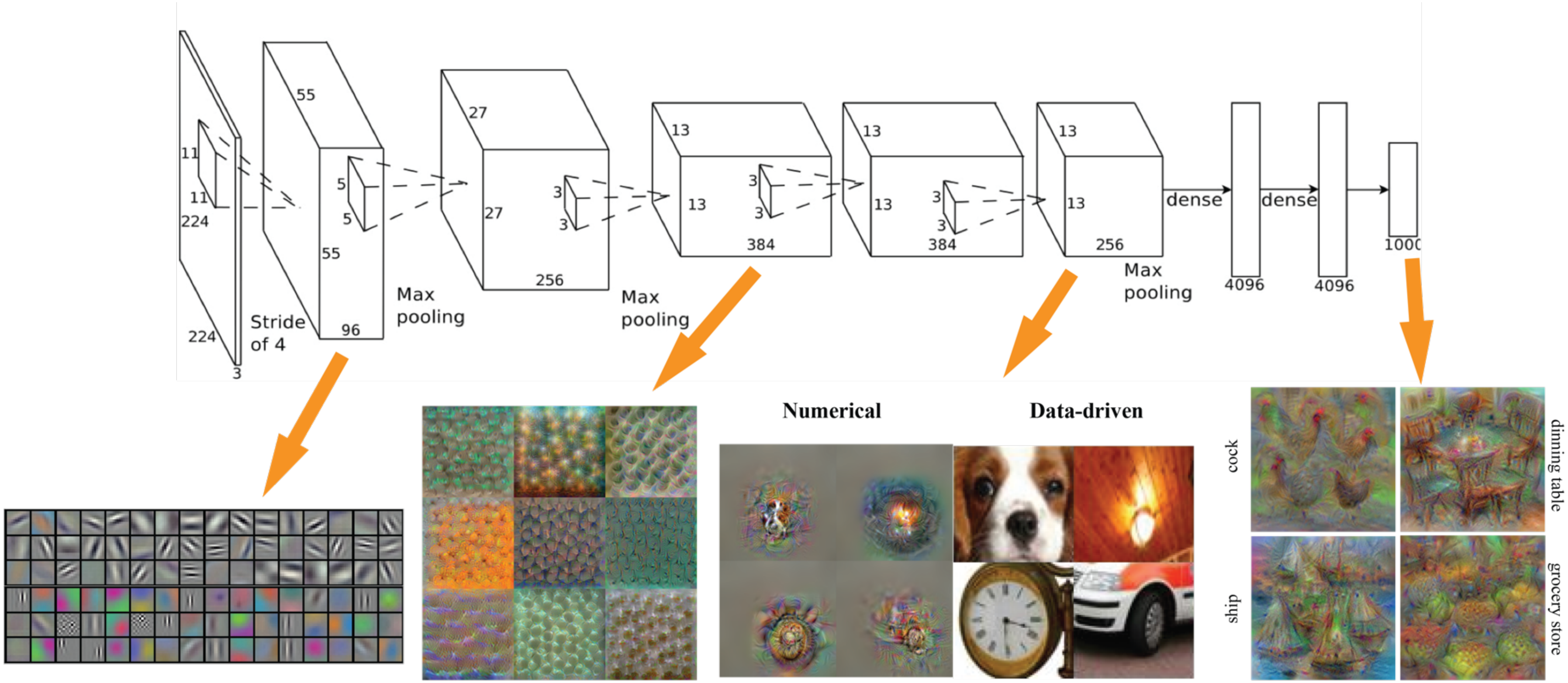
Tensorflow Transfer Learning Example



Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- Sentiment Analysis by CNN

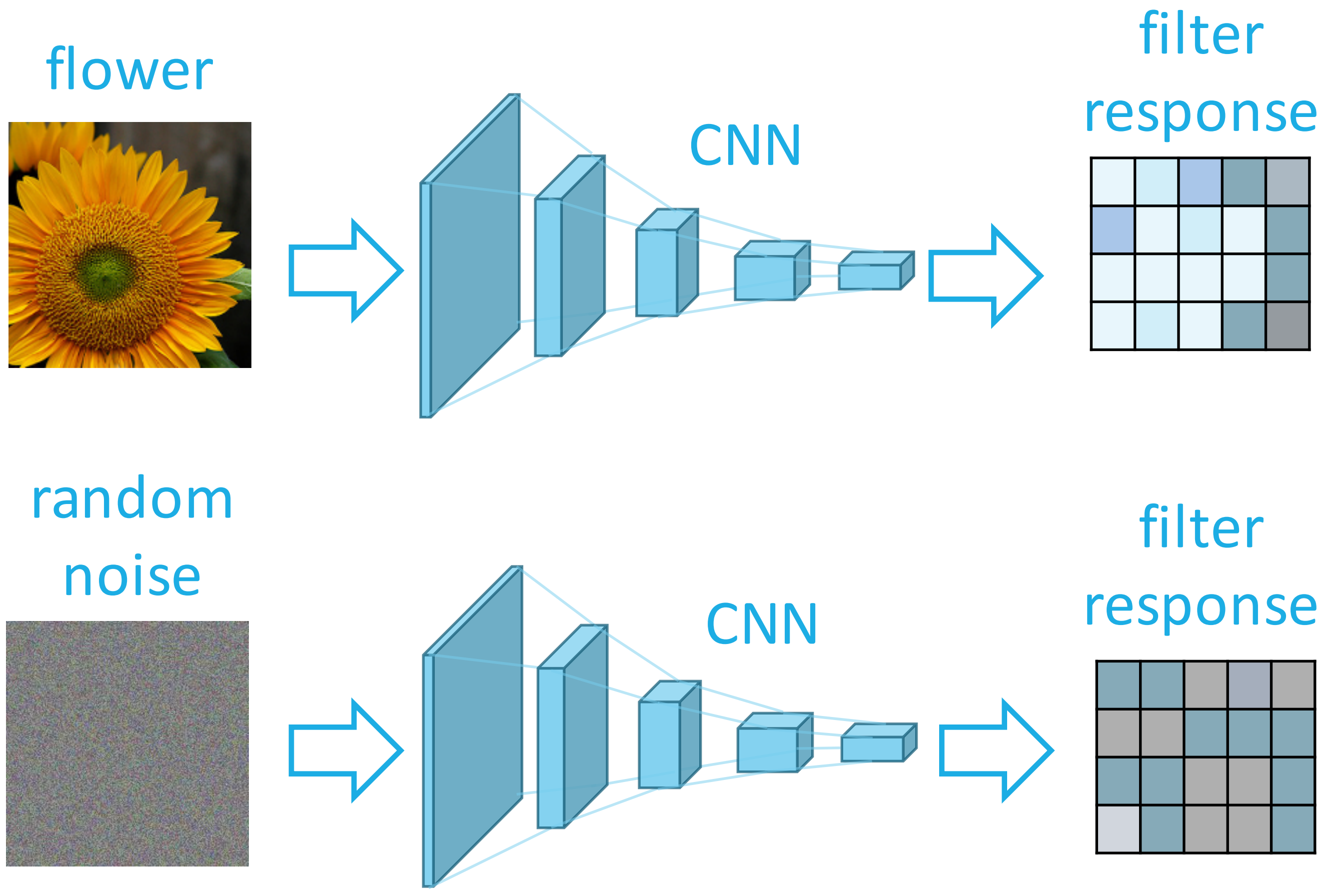
Visualizing CNN



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

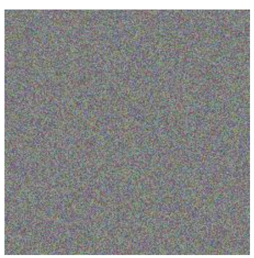
Visualizing CNN



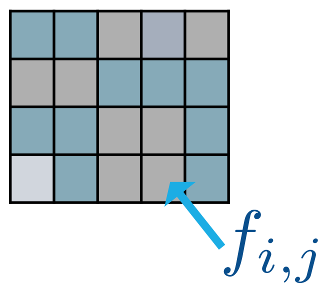
Gradient Ascent

- Magnify the filter response

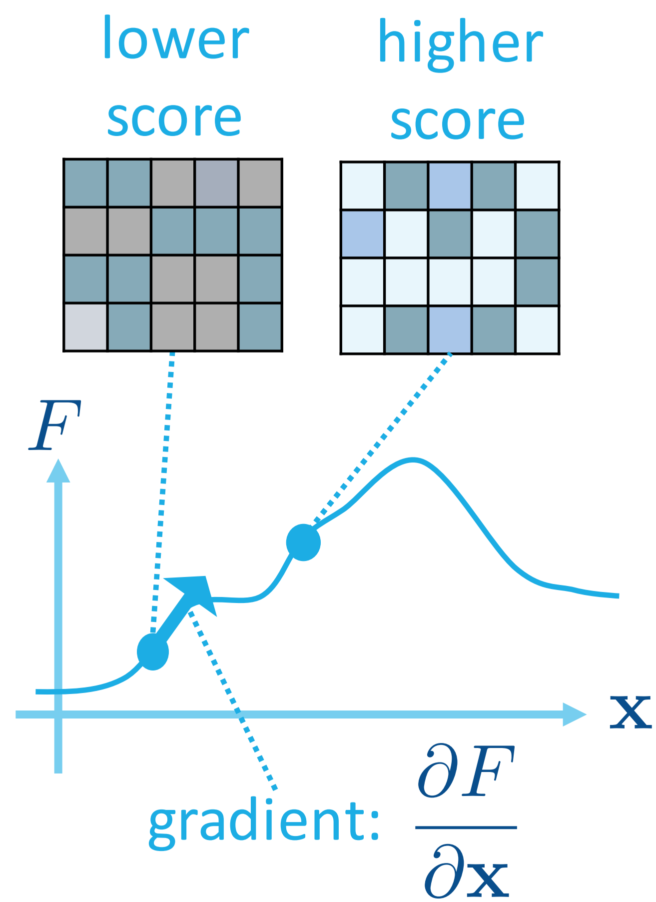
random noise: $\mathbf{x}$



filter response: $\mathbf{f}$



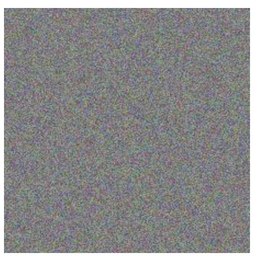
score: $F = \sum_{i,j} f_{i,j}$



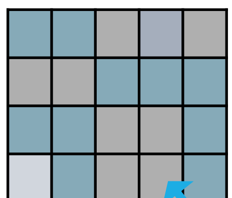
Gradient Ascent

- Magnify the filter response

random noise: $\mathbf{x}$



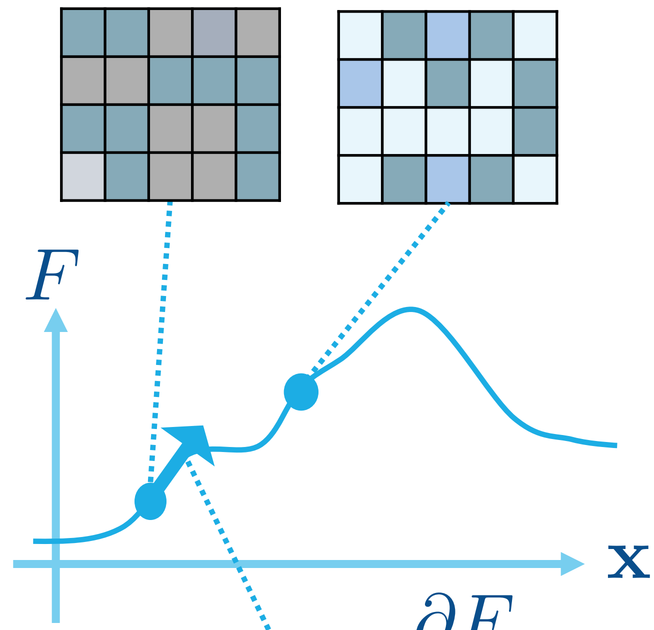
filter response: $\mathbf{f}$



$f_{i,j}$

lower score

higher score



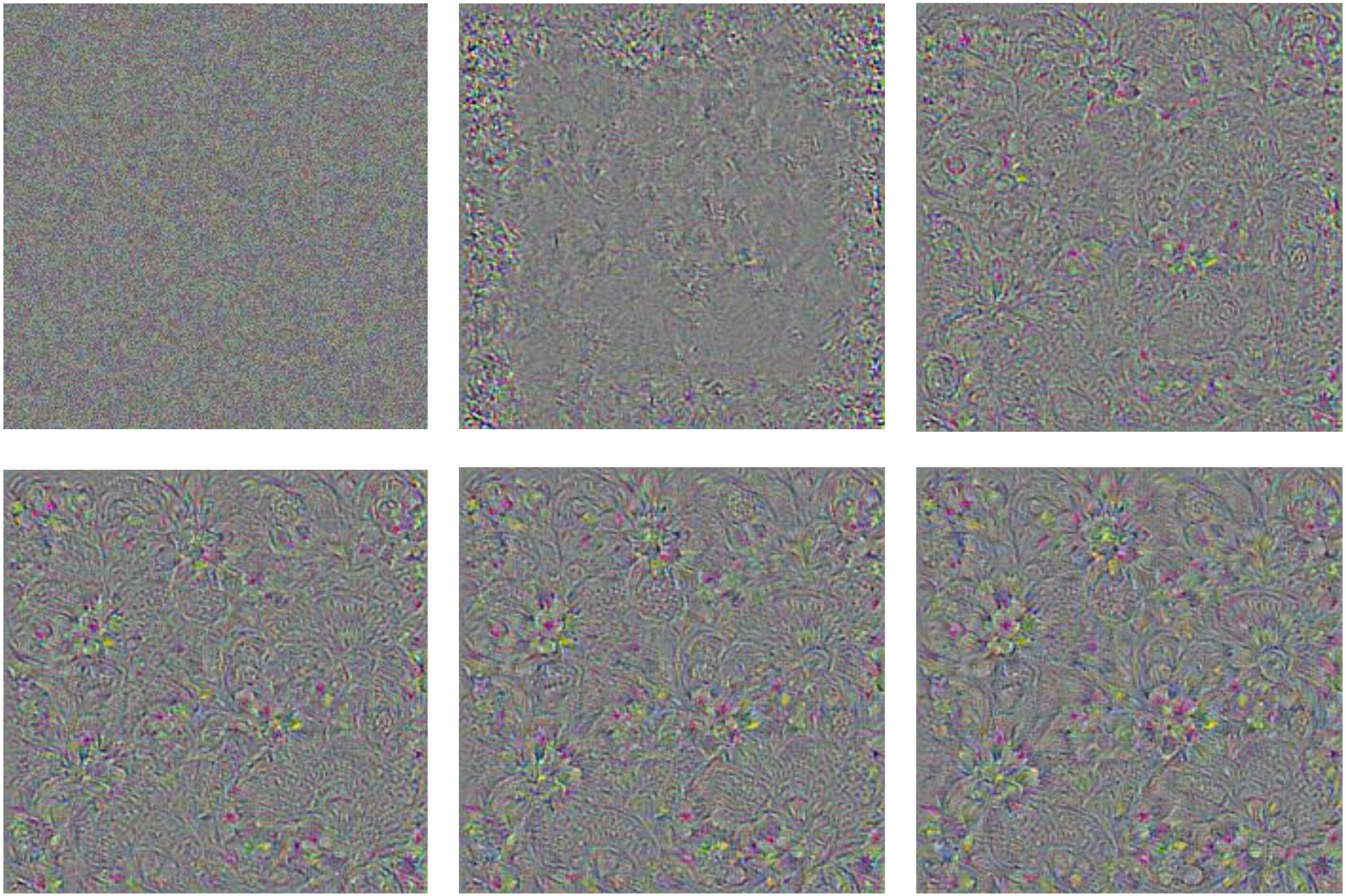
gradient: $\dfrac{\partial F}{\partial \mathbf{x}}$

update $\mathbf{x}$

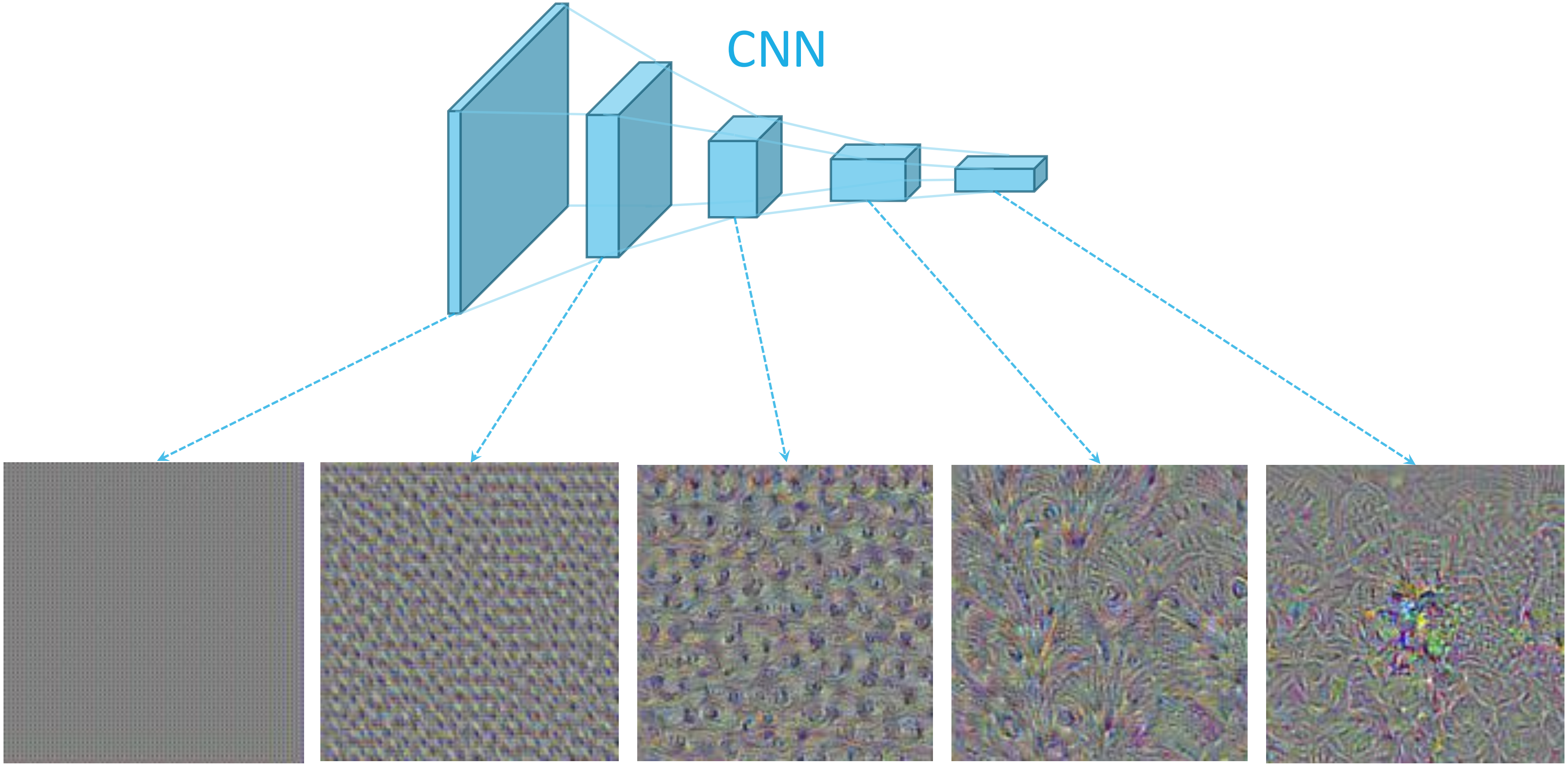
$$\mathbf{x} \leftarrow \mathbf{x} + \eta \frac{\partial F}{\partial \mathbf{x}}$$

learning rate

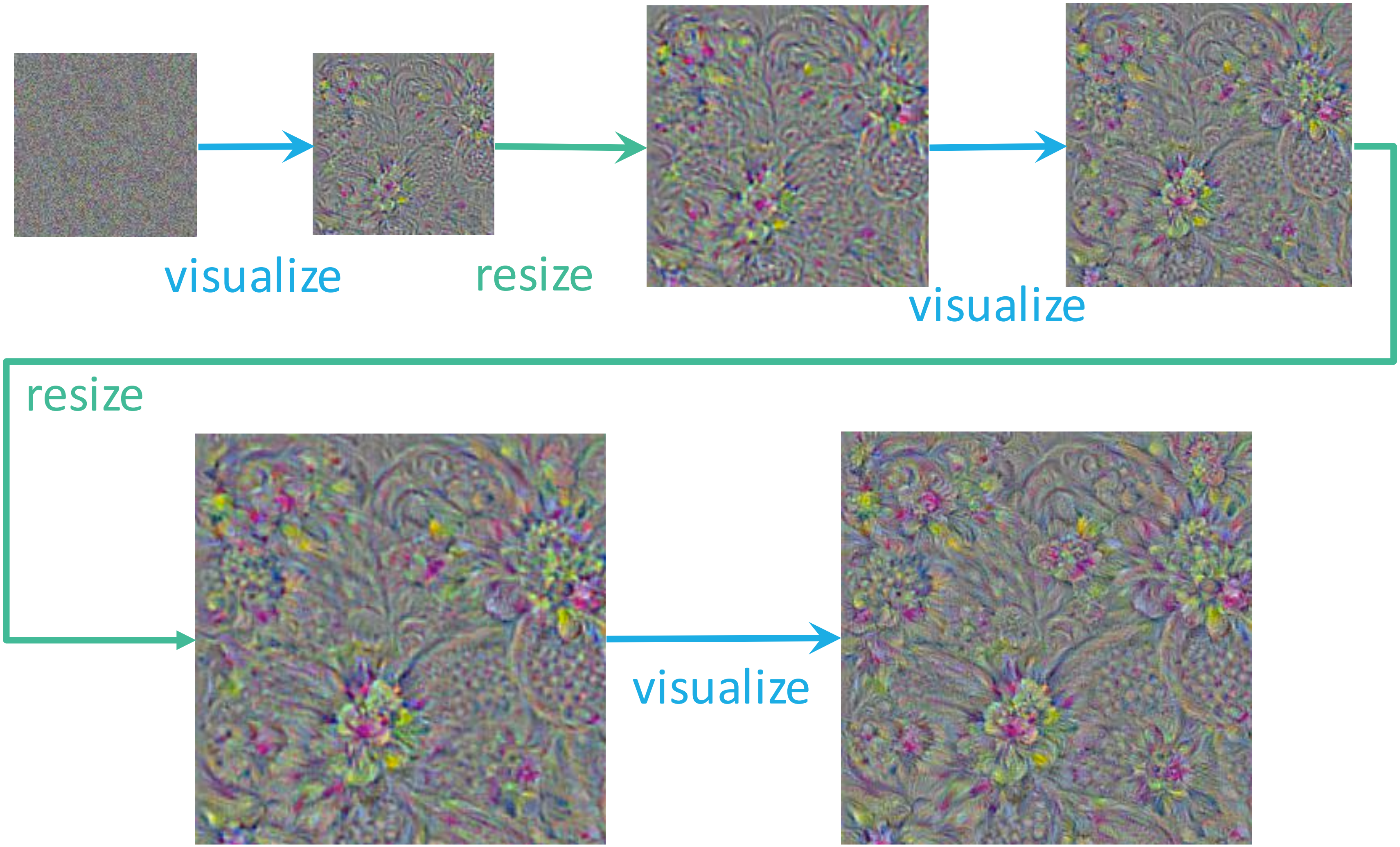
Gradient Ascent



Different Layers of Visualization



Multiscale Image Generation



Multiscale Image Generation

Deep Dream

- https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
- Source code:

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb

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

Deep Dream

Deep Dream

Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- Sentiment Analysis by CNN

Neural Art

- Paper: https://arxiv.org/abs/1508.06576
- Source code : https://github.com/ckmarkoh/neuralart_tensorflow

content

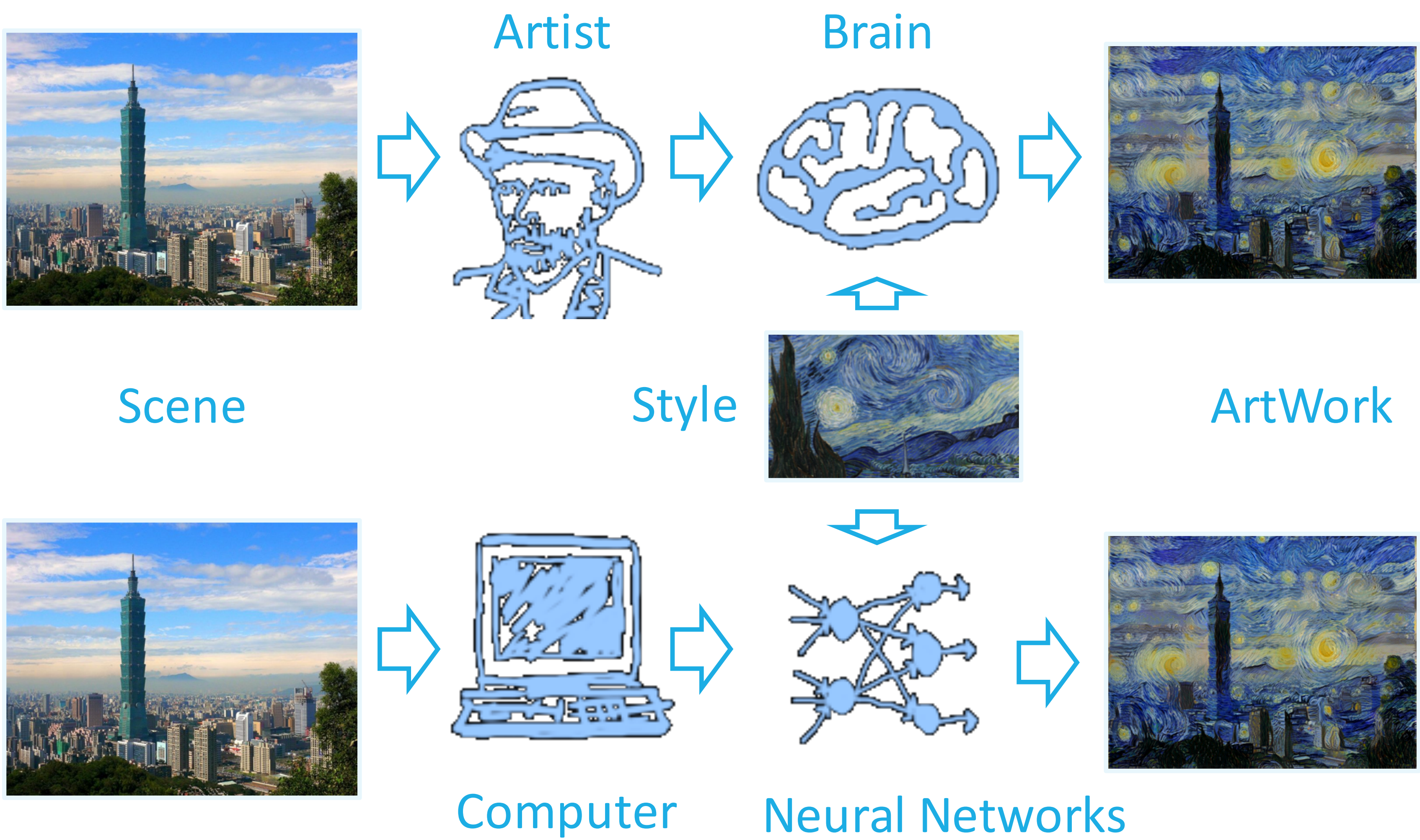
http://www.taipei-101.com.tw/upload/news/201502/2015021711505431705145.JPG

style

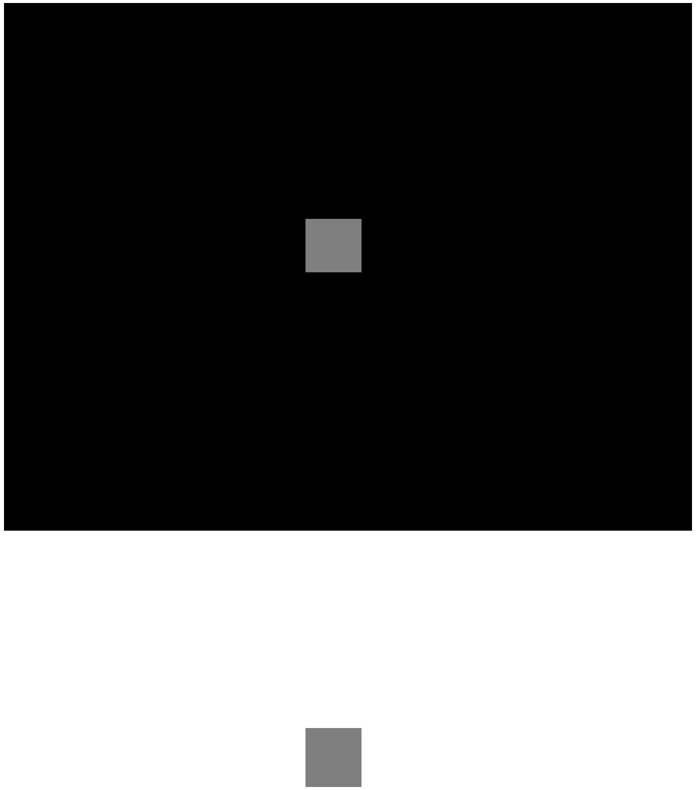
https://github.com/andersbll/neural_artistic_style/blob/master/images/starry_night.jpg?raw=true

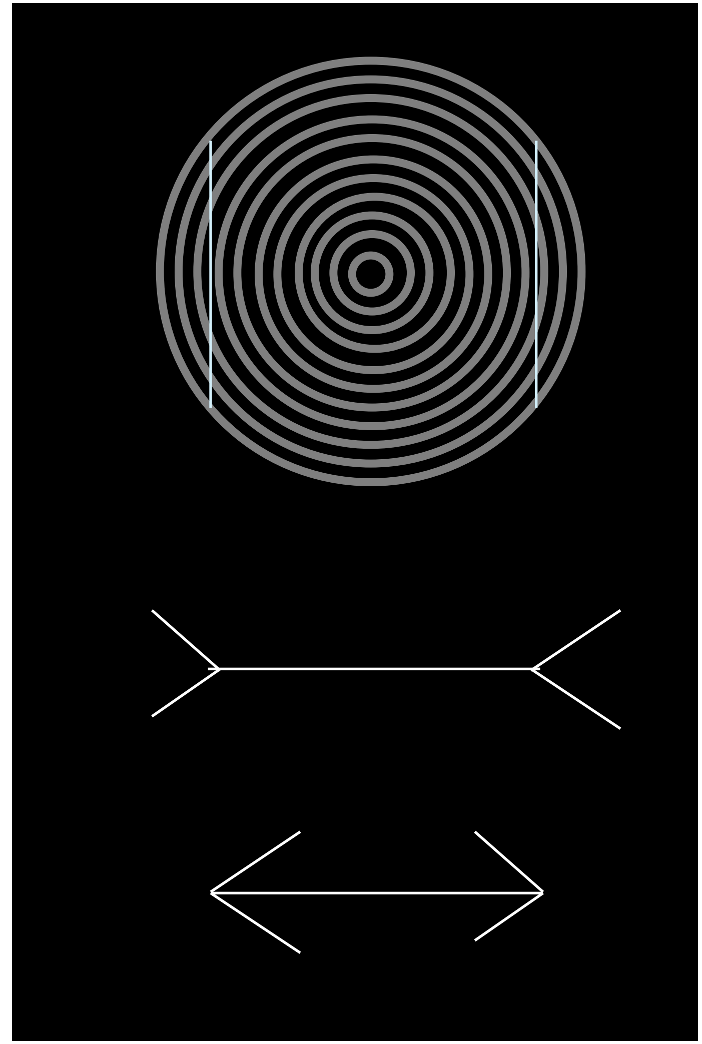
artwork

The Mechanism of Painting

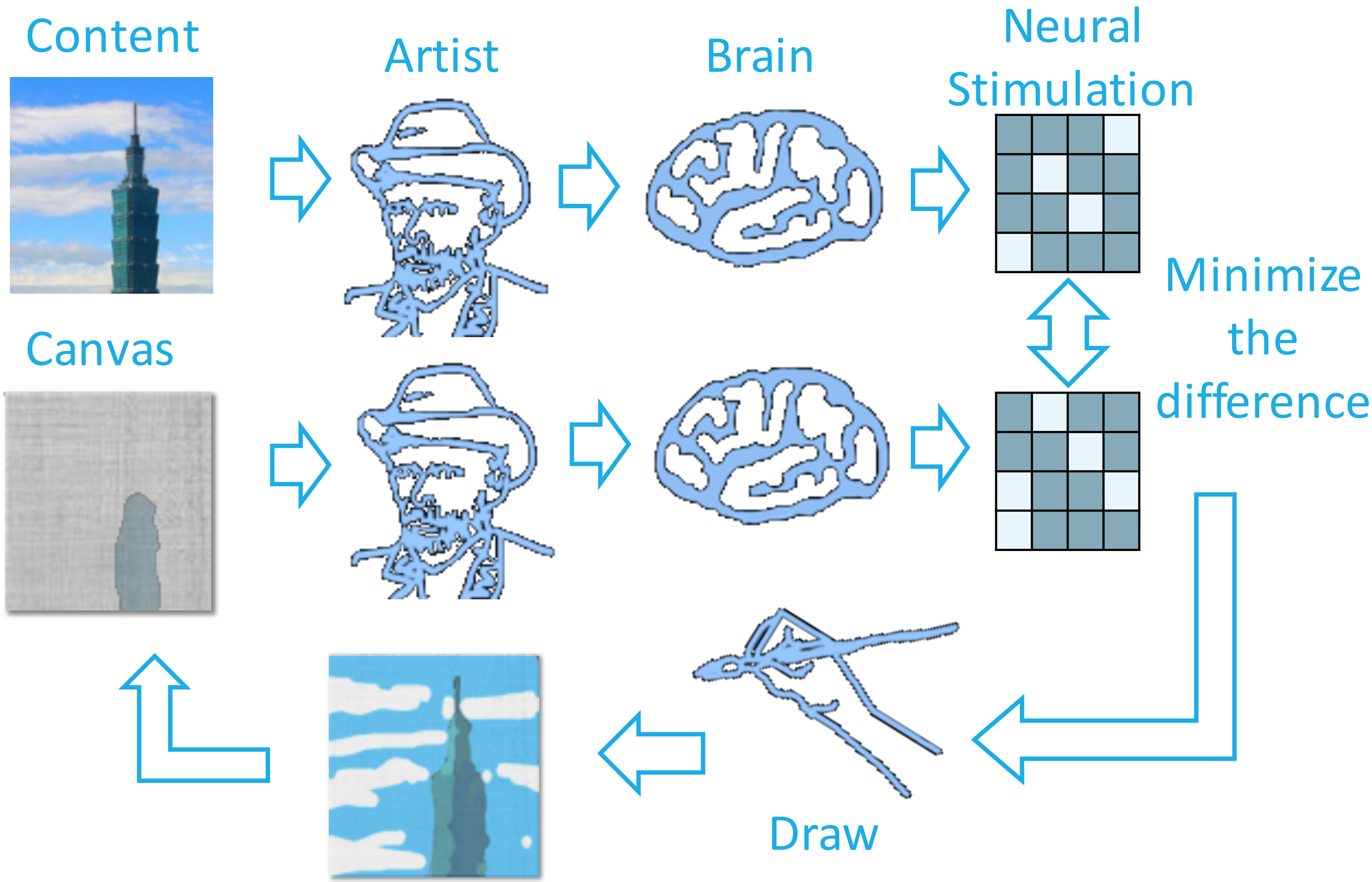


Misconception

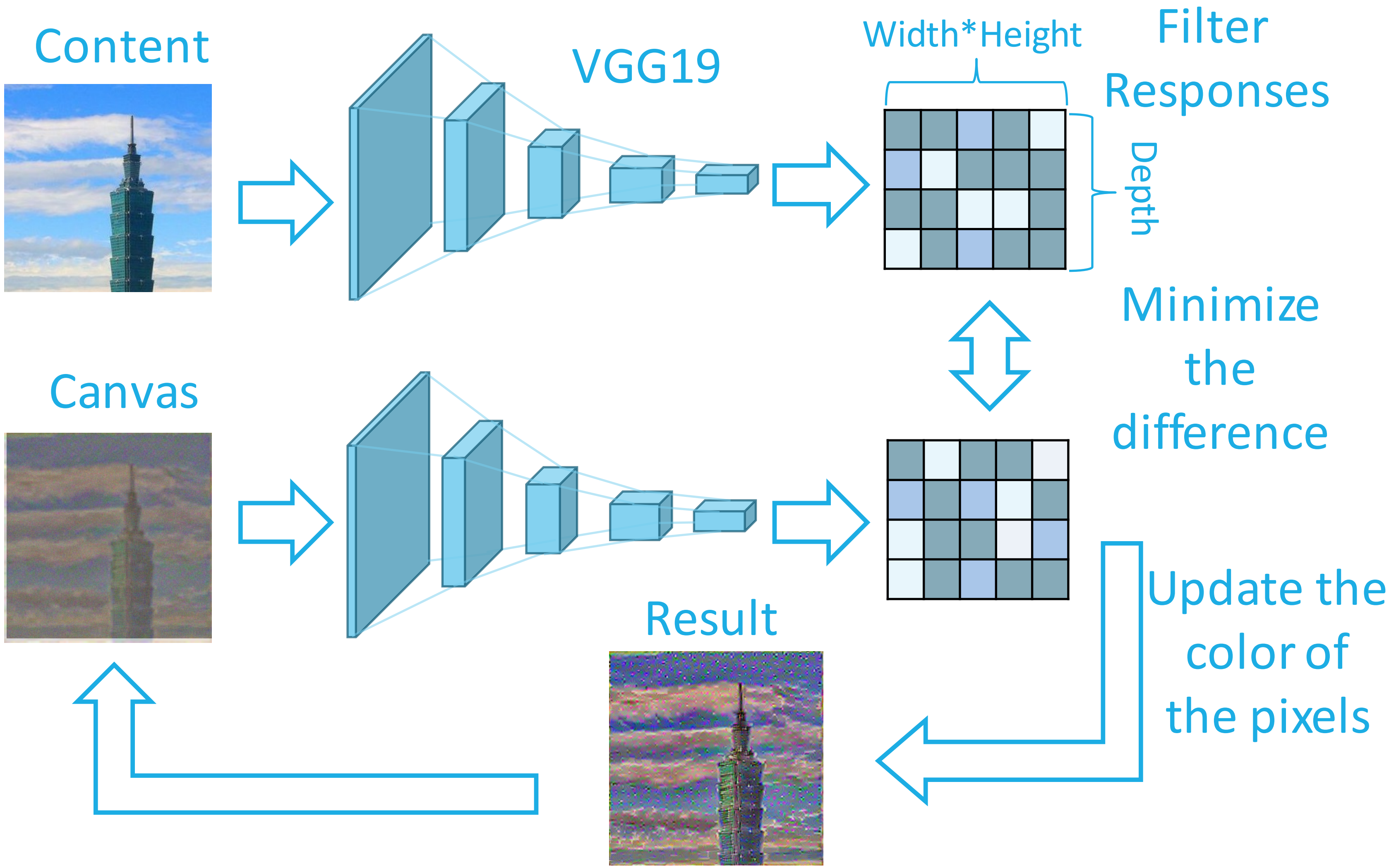




Content Generation



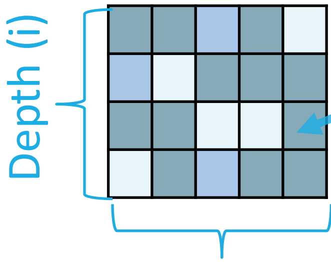
Content Generation



Content Generation

Input Photo: $\mathbf{p}$

Layer l's Filter Responses: P^l

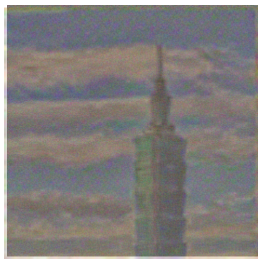


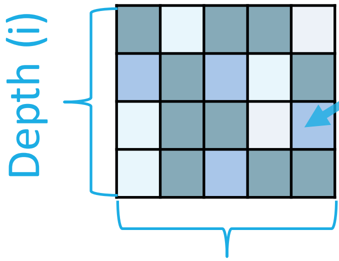
$P^l_{i,j}$

Width*Height (j)

Input Canvas: $\mathbf{x}$

Layer l's Filter l Responses: X^l





$X^l_{i,j}$

Width*Height (j)

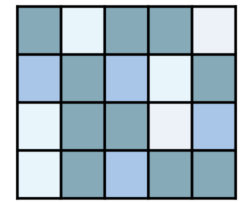
$$L_{content}(\mathbf{p}, \mathbf{x}, l) = \frac{1}{2} \sum_{i,j} (X^l_{i,j} - P^l_{i,j})^2$$

$$\frac{\partial L_{content}(\mathbf{p}, \mathbf{x}, l)}{\partial X^l_{i,j}} = X^l_{i,j} - P^l_{i,j}$$

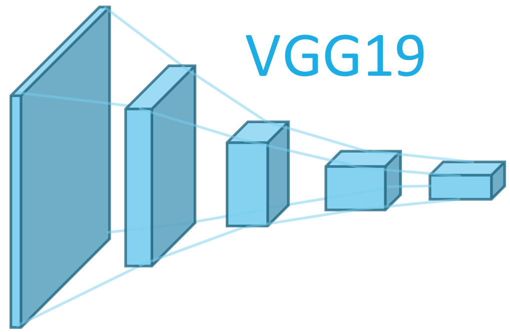
Content Generation

- Backward Propagation

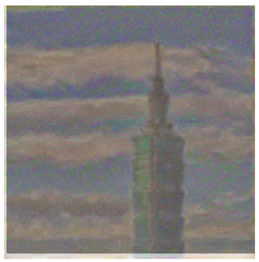
Layer l's Filter l Responses: X^l



VGG19

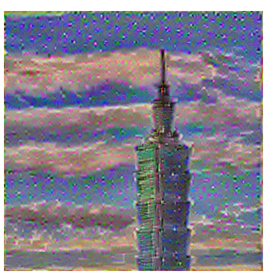


Input Canvas: $\mathbf{x}$



$$\frac{\partial L_{content}}{\partial \mathbf{x}} = \frac{\partial L_{content}}{\partial X^l} \frac{\partial X^l}{\mathbf{x}}$$

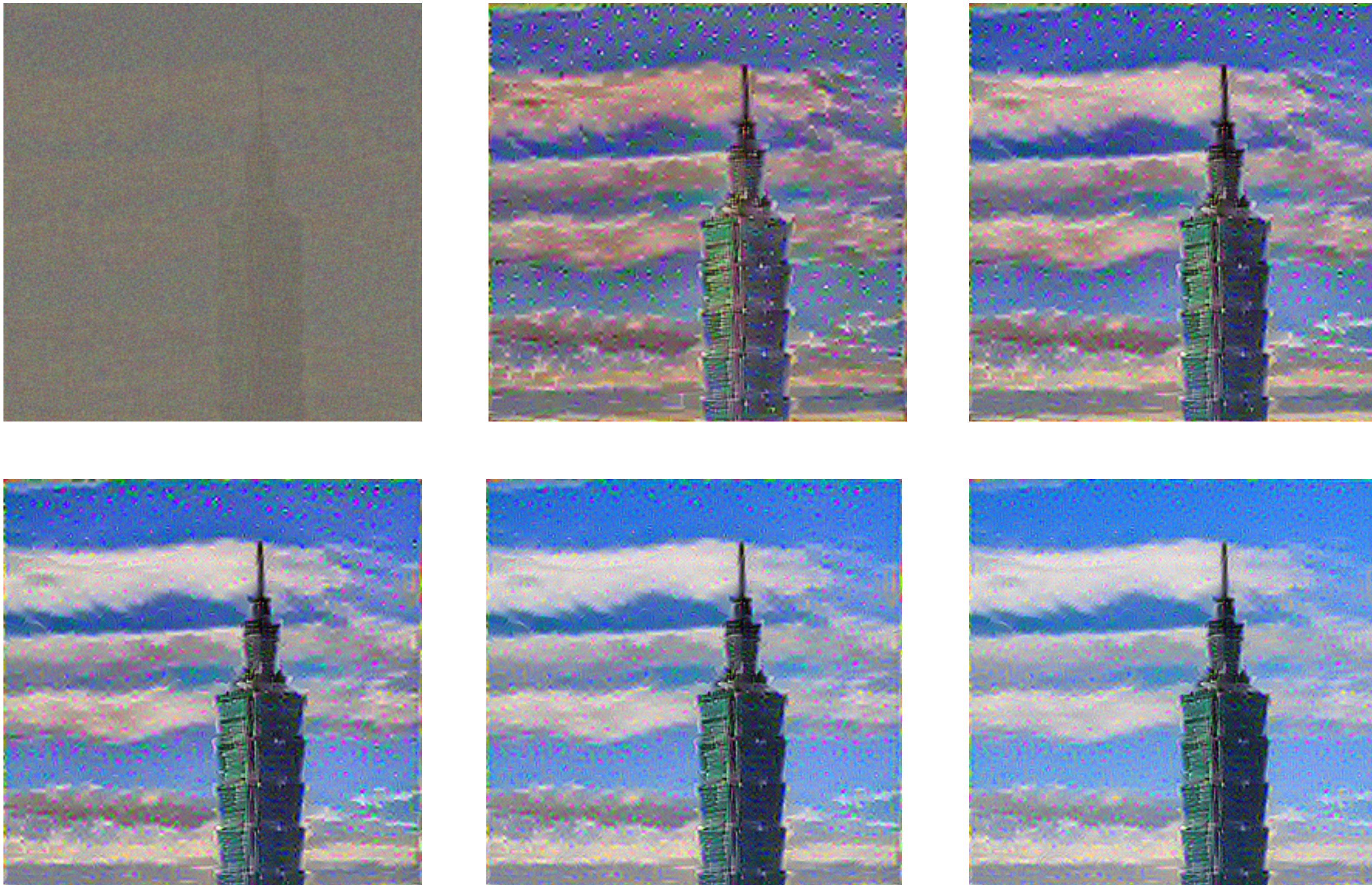
Update Canvas



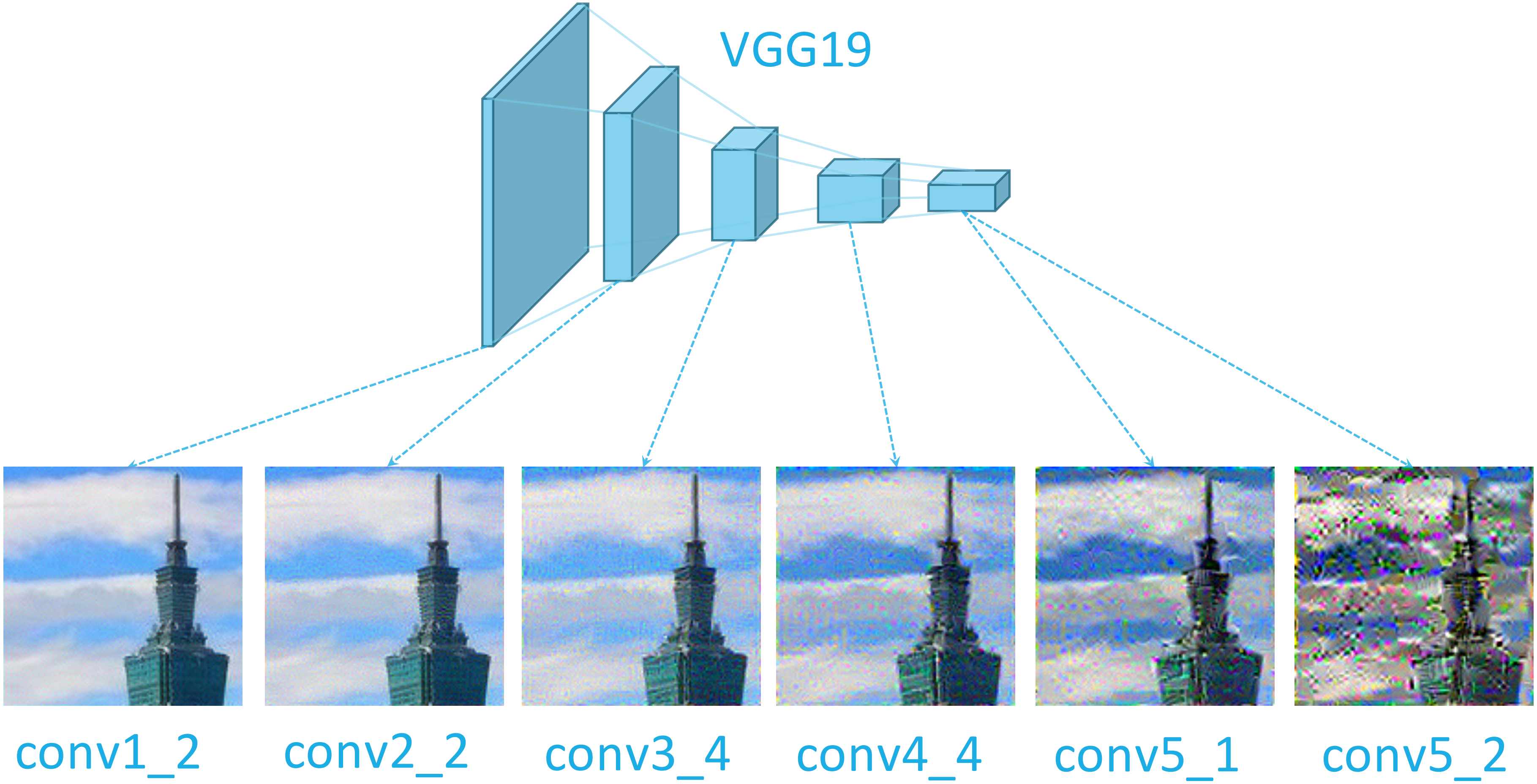
$$\mathbf{x} \leftarrow \mathbf{x} - \eta \frac{\partial L_{content}}{\partial \mathbf{x}}$$

Learning Rate

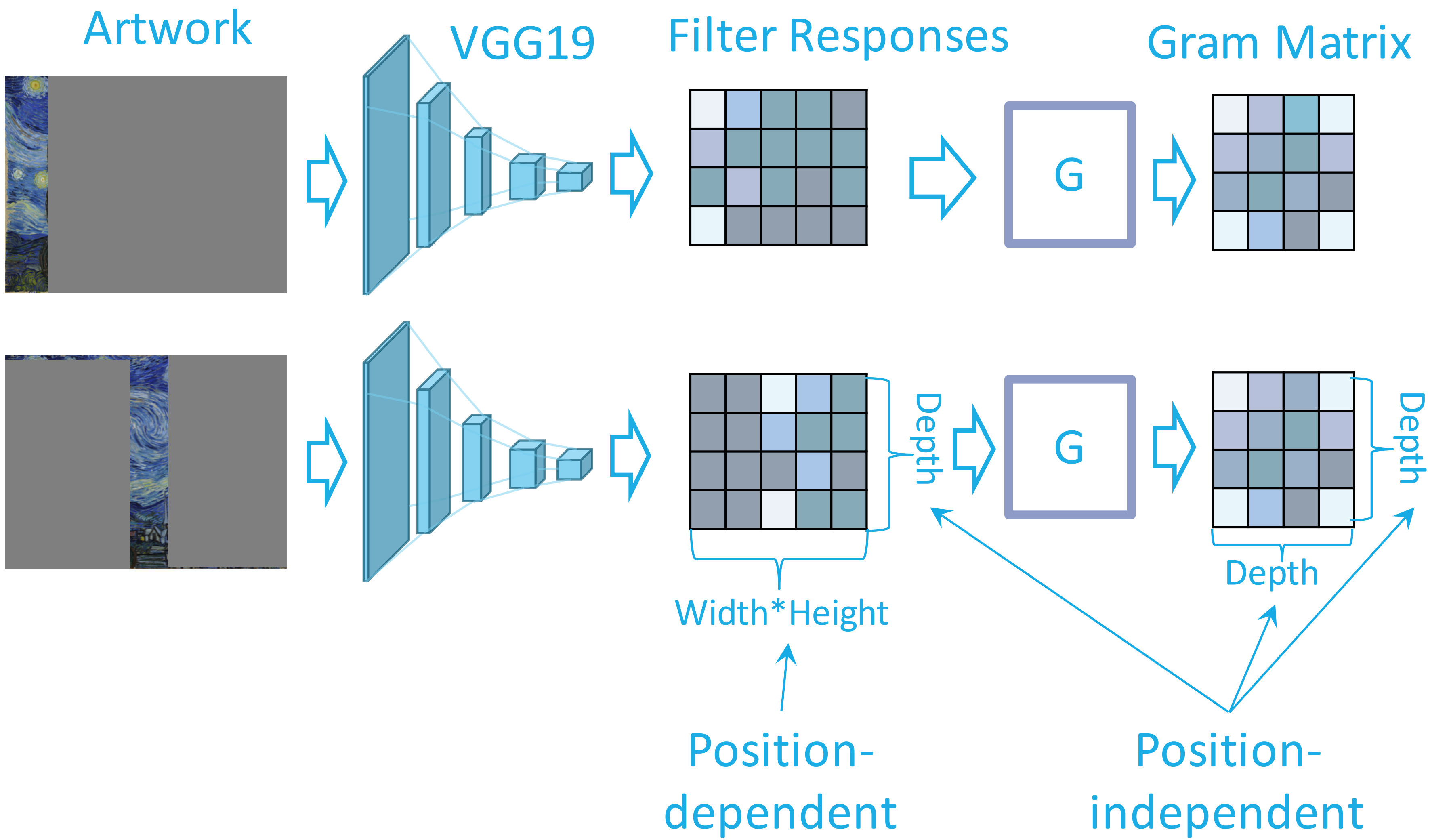
Content Generation



Content Generation

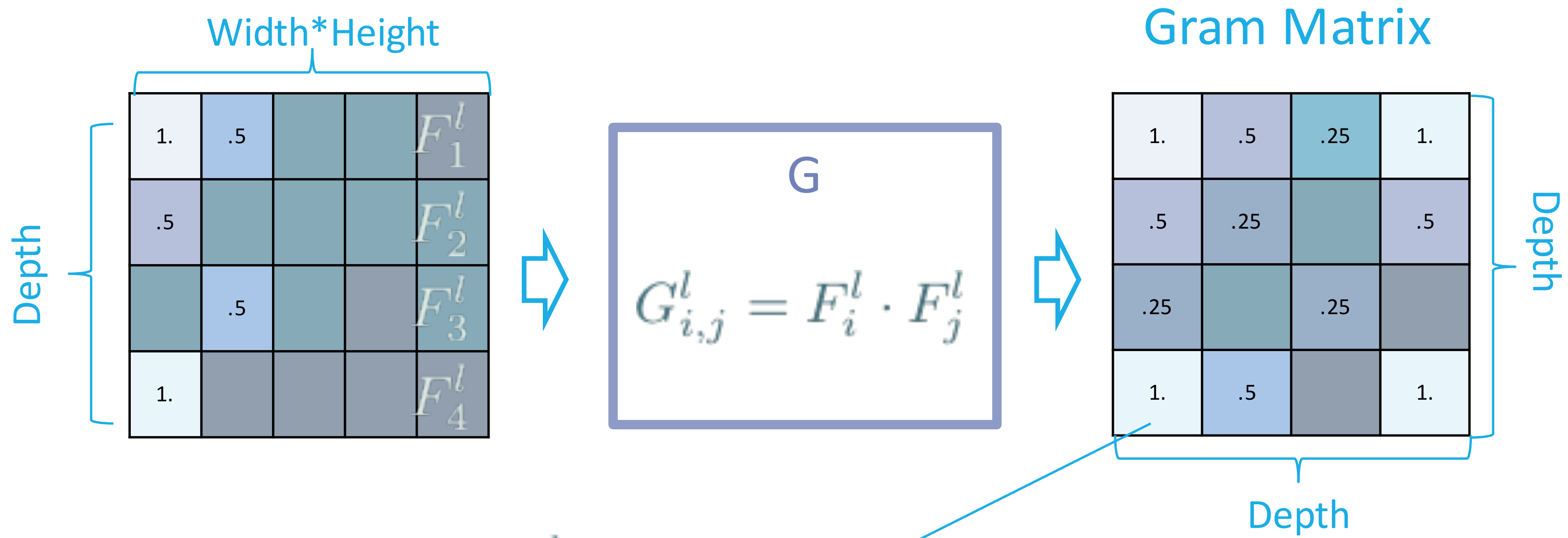


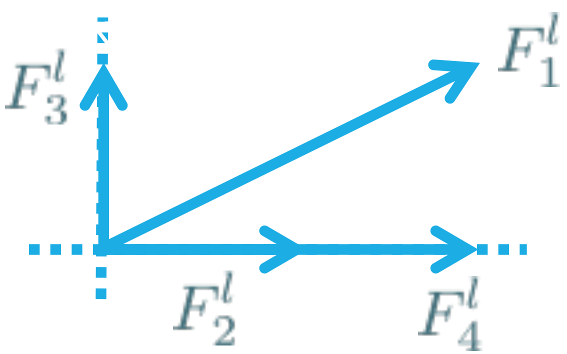
Style Generation



Style Generation

Layer l's Filter Responses



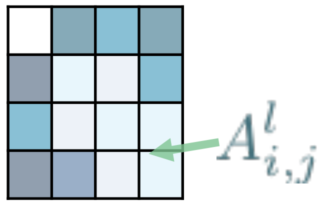


$$G_{4,1}^l = F_4^l \cdot F_1^l$$
$$= 1 \times 1 + 0 \times 0.5 + 0 \times 0 + \ldots$$
$$= 1$$

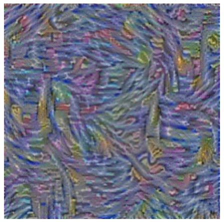
Style Generation

Input Artwork: $\mathbf{a}$

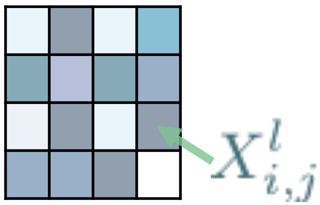
Layer l's Gram Matrix



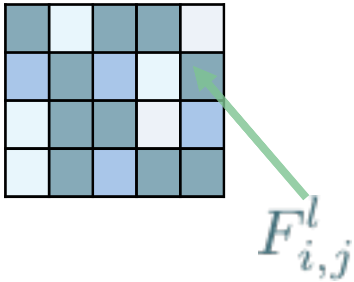
Input Canvas: $\mathbf{x}$



Layer l's Gram Matrix



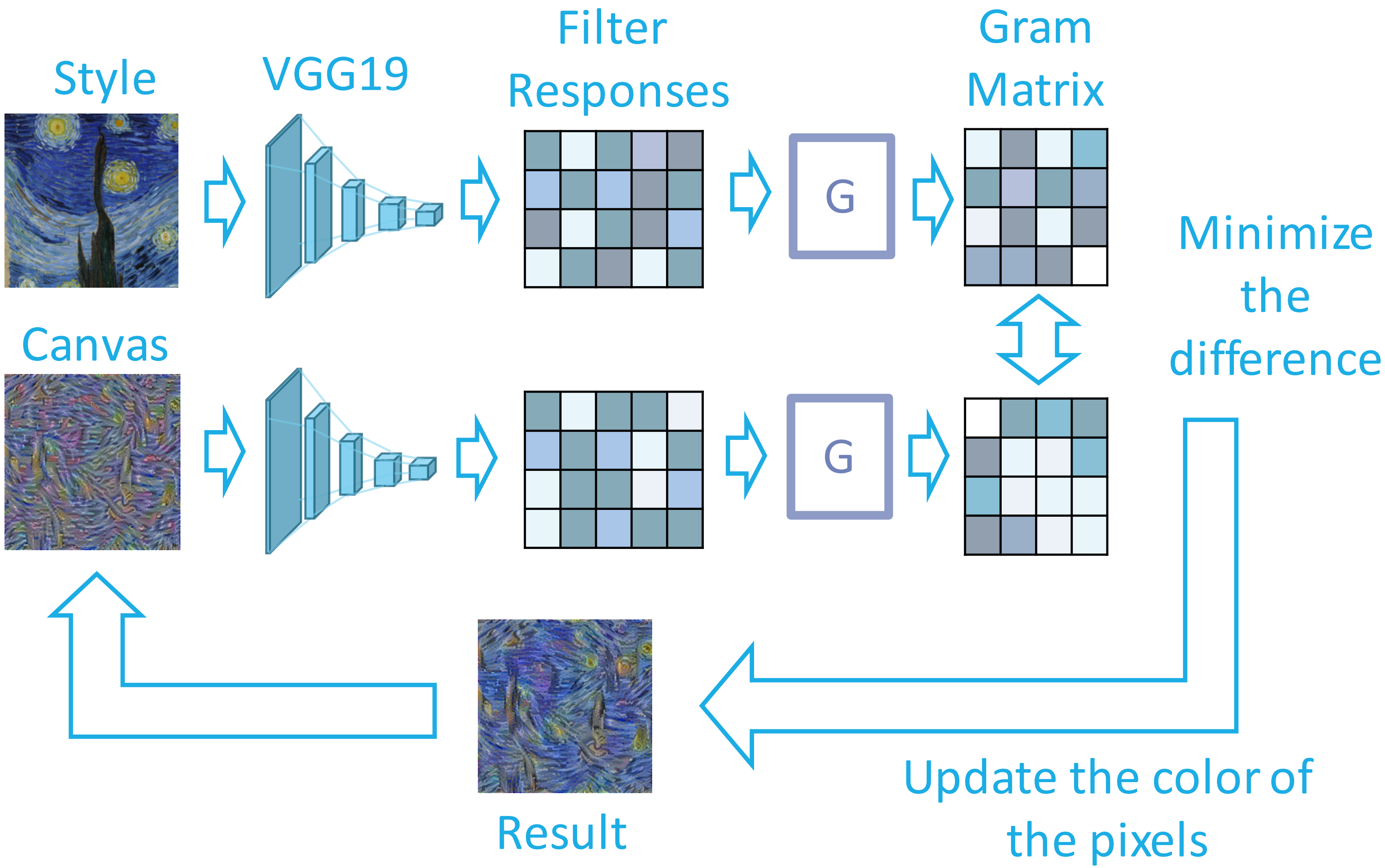
Layer l's Filter Responses



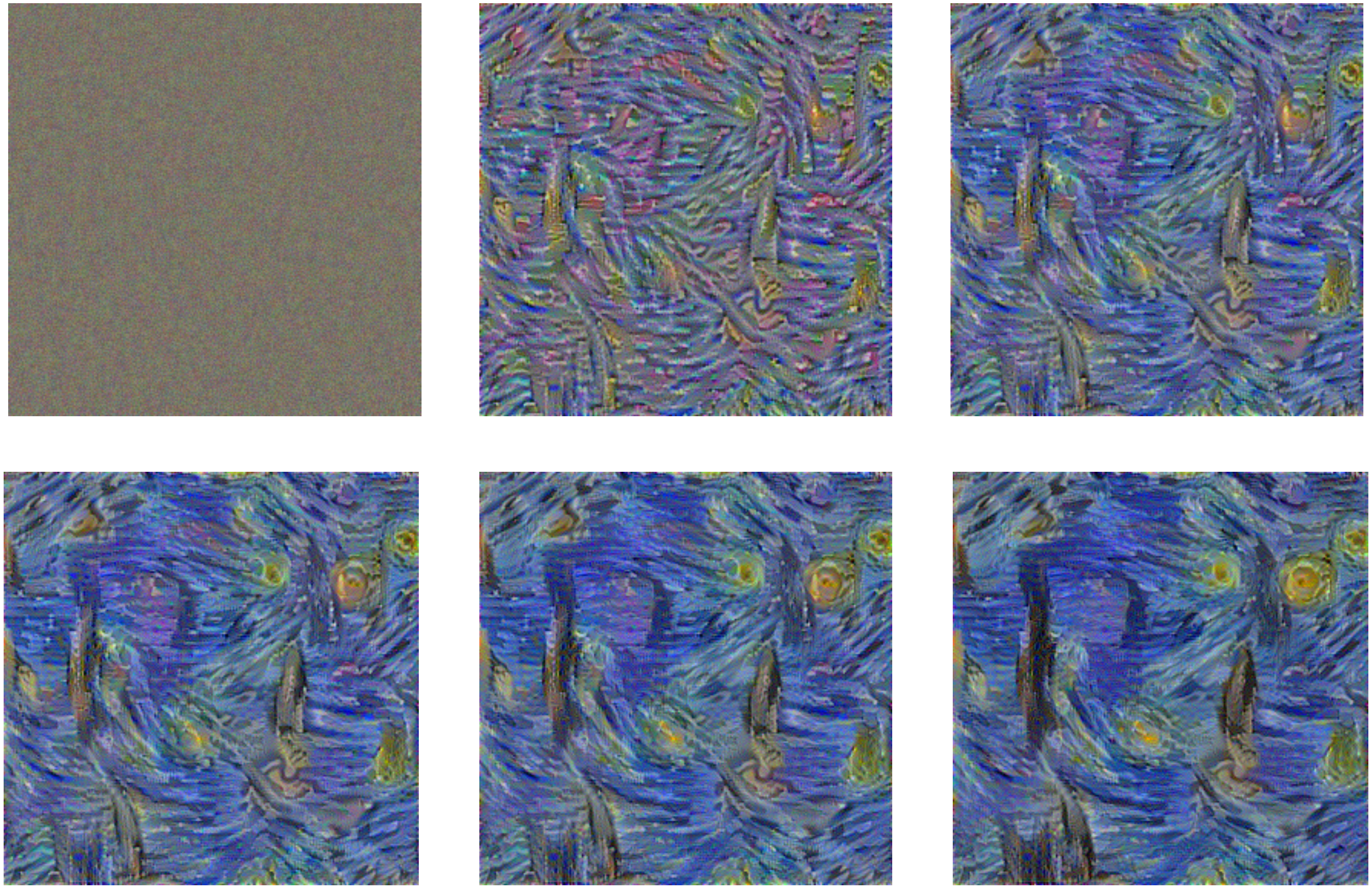
$$L_{style}(\mathbf{a}, \mathbf{x}, l) = \frac{1}{2} \sum_{i,j} (X^l_{i,j} - A^l_{i,j})^2$$

$$\frac{\partial L_{style}(\mathbf{a}, \mathbf{x}, l)}{\partial F^l_{i,j}} = ((F^l)^T (X^l - A^l))_{j,i}$$

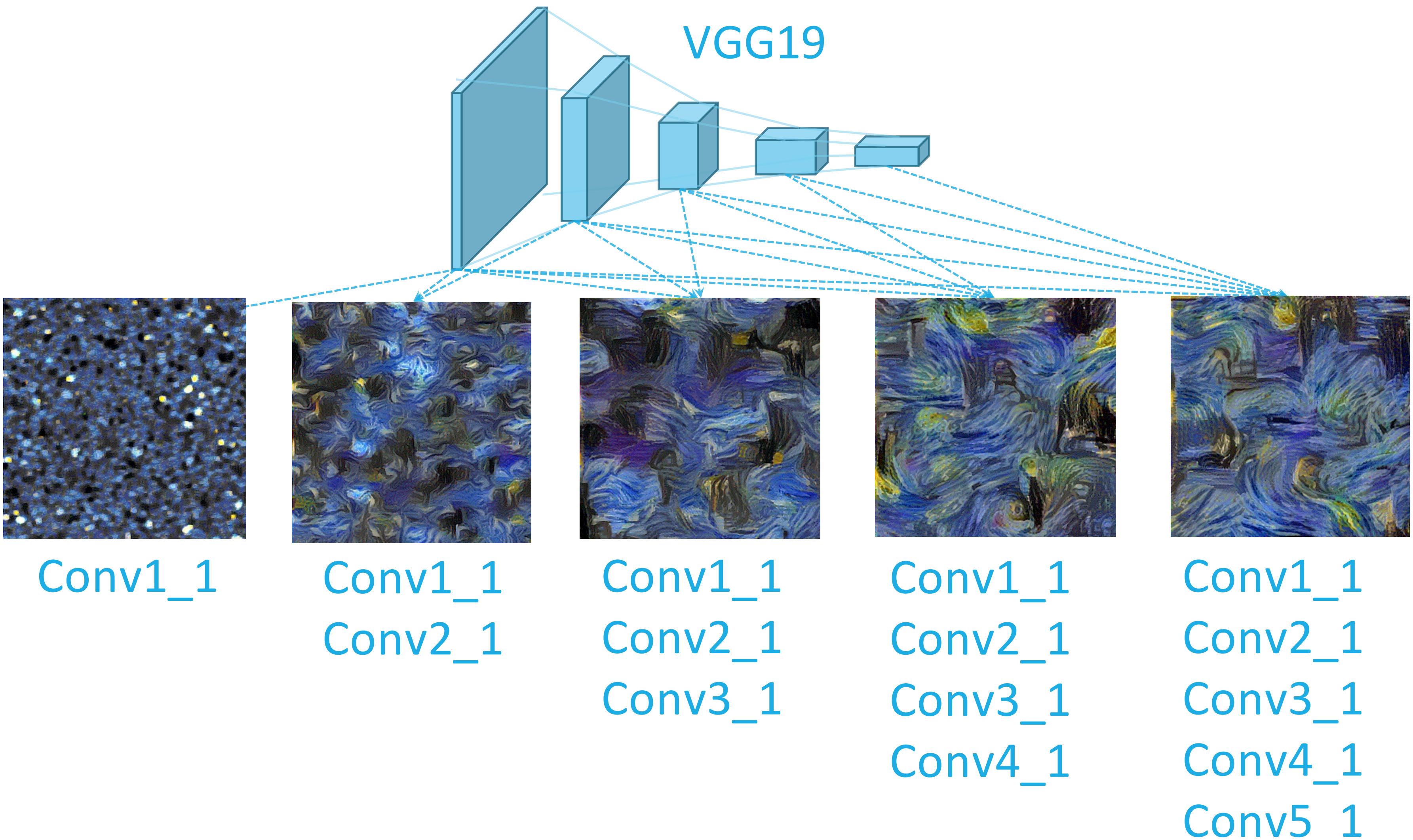
Style Generation



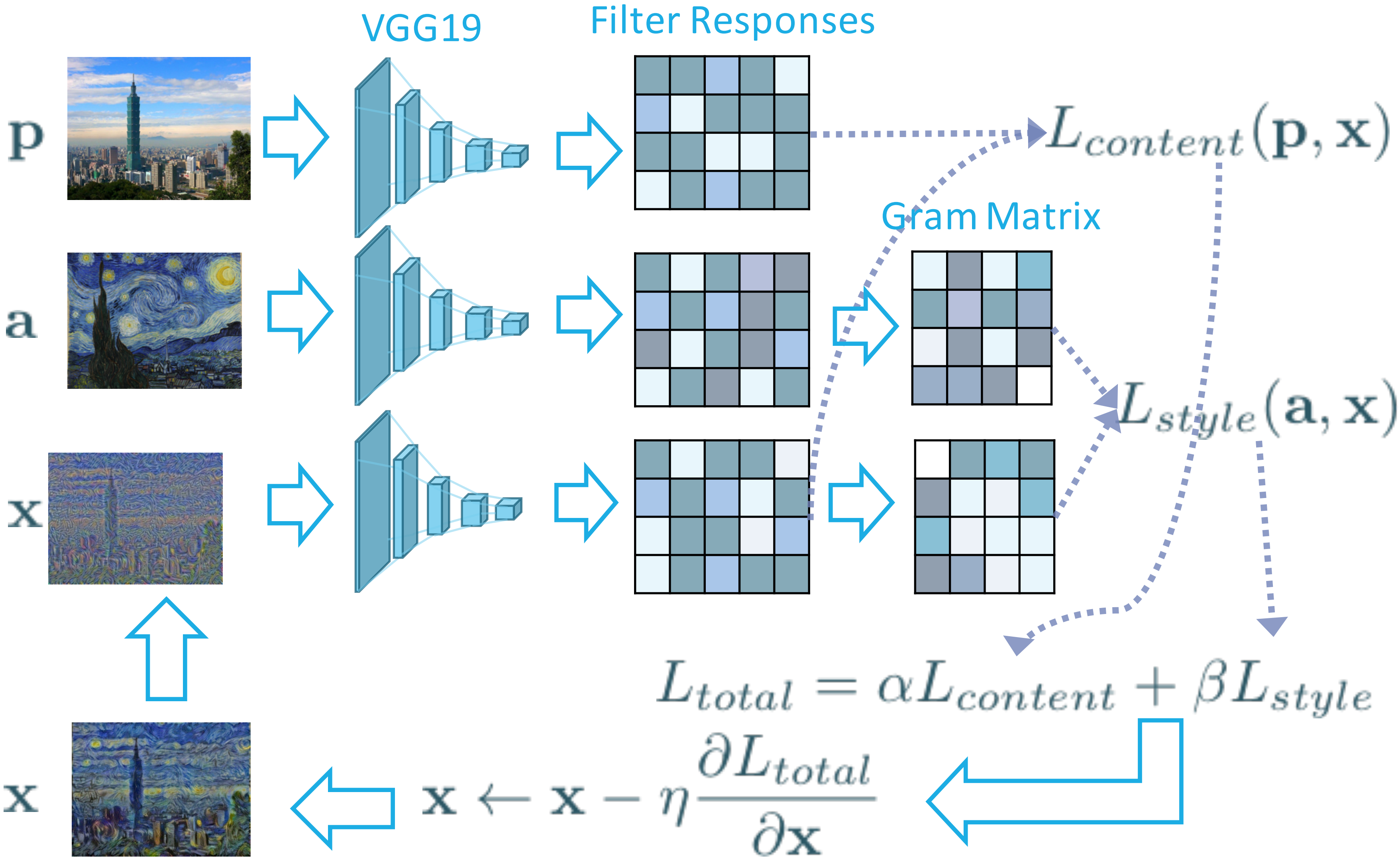
Style Generation



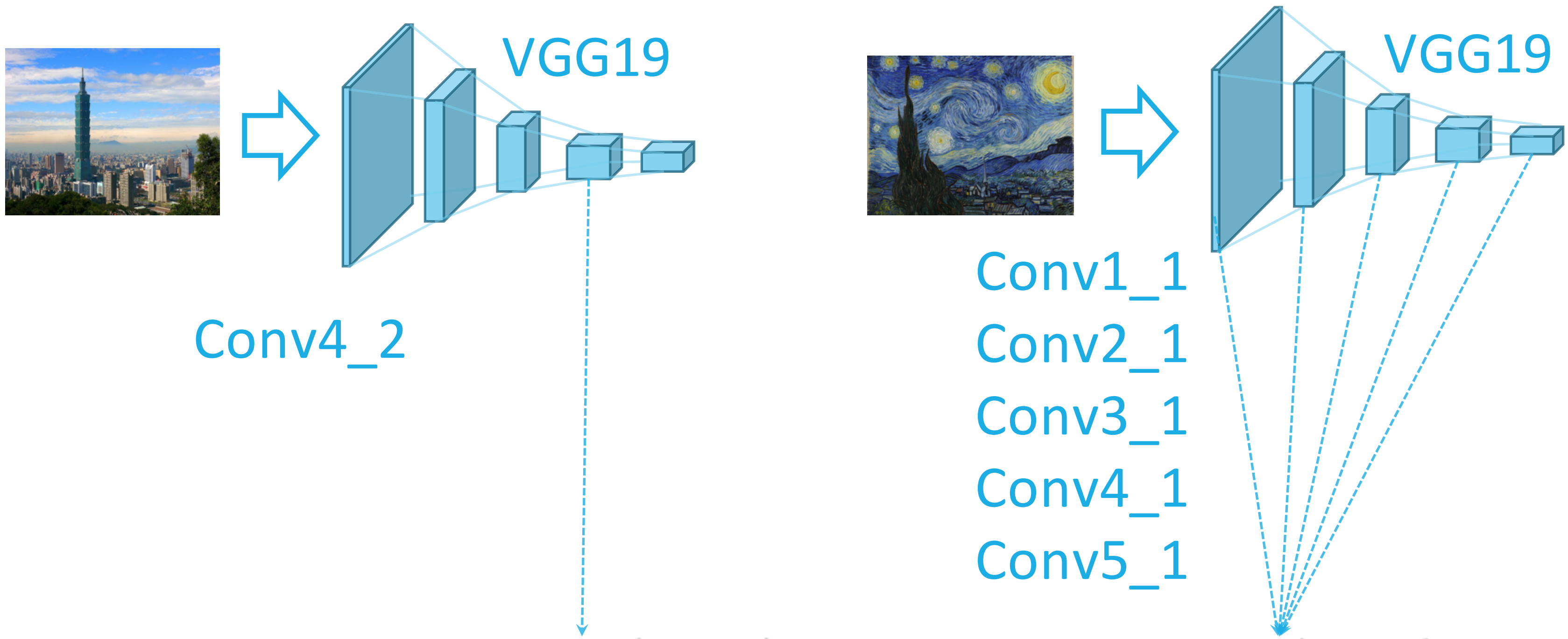
Style Generation



Artwork Generation



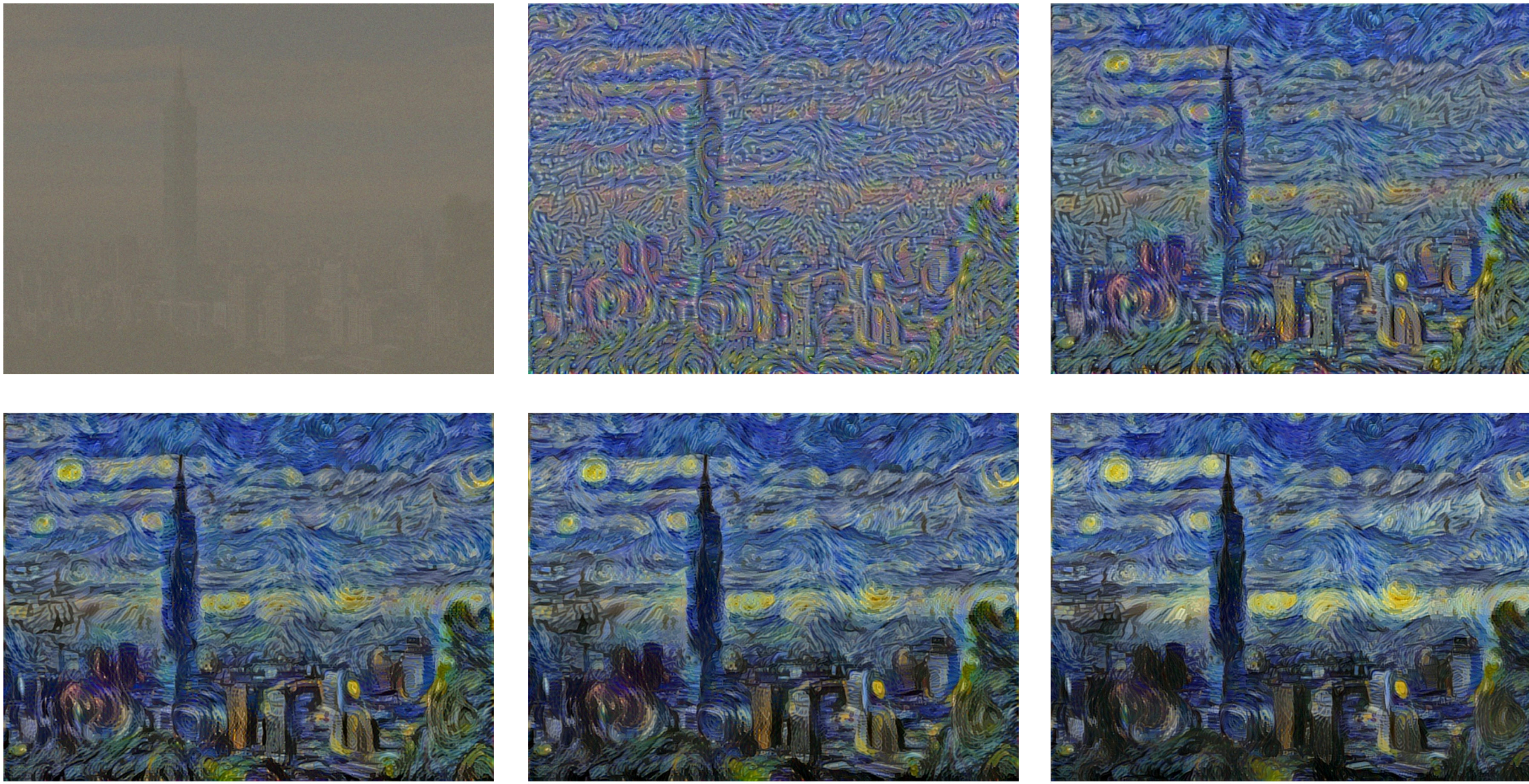
Artwork Generation



$$L_{content}(\mathbf{p}, \mathbf{x}) \qquad L_{style}(\mathbf{a}, \mathbf{x})$$

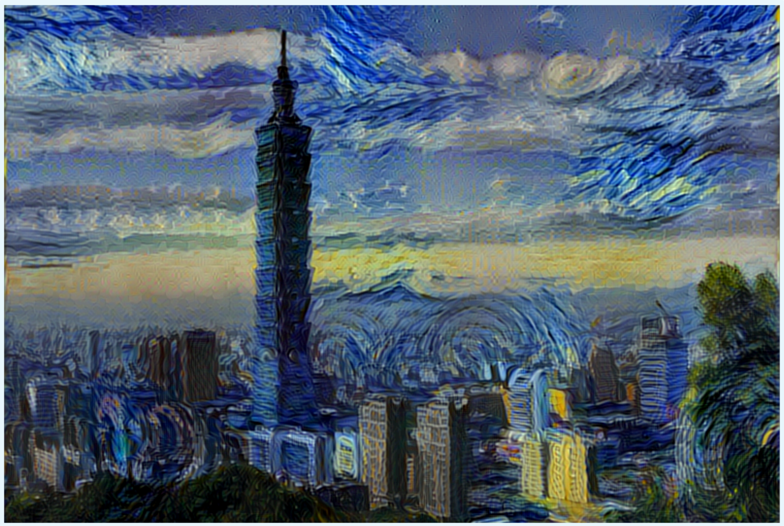
$$L_{total} = \alpha L_{content} + \beta L_{style}$$

Artwork Generation

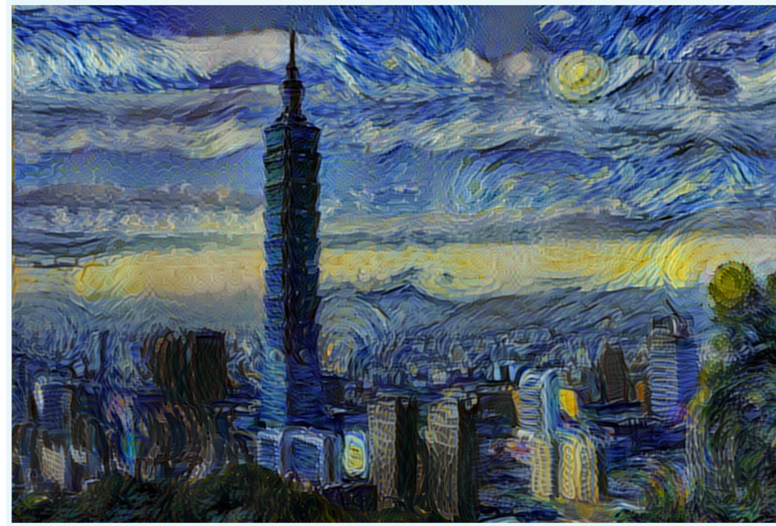


Content v.s. Style

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



0.15



0.05

0.02

0.007

Neural Doodle

- Paper: https://arxiv.org/abs/1603.01768
- Source code: https://github.com/alexjc/neural-doodle

content

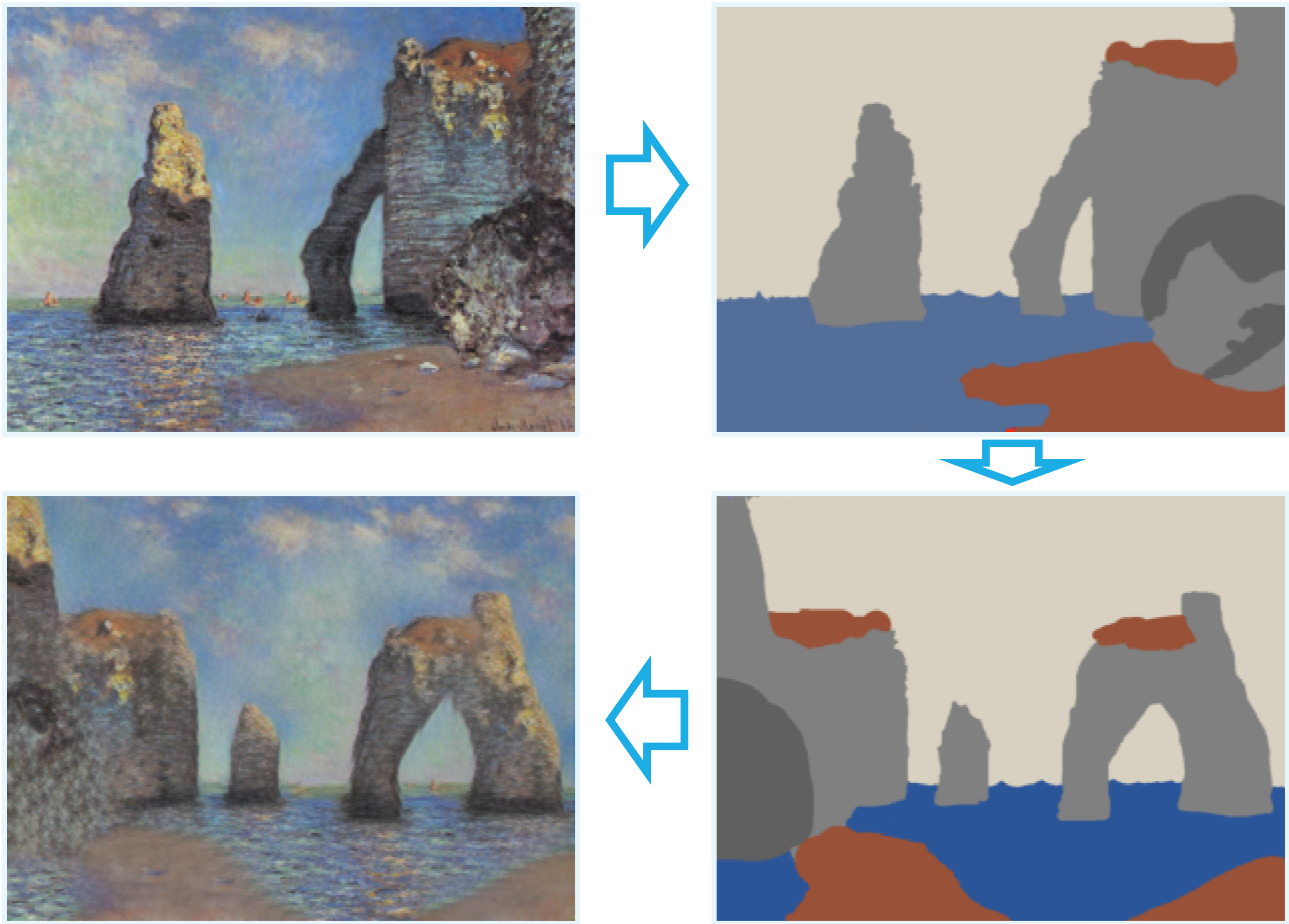
semantic maps

result

style

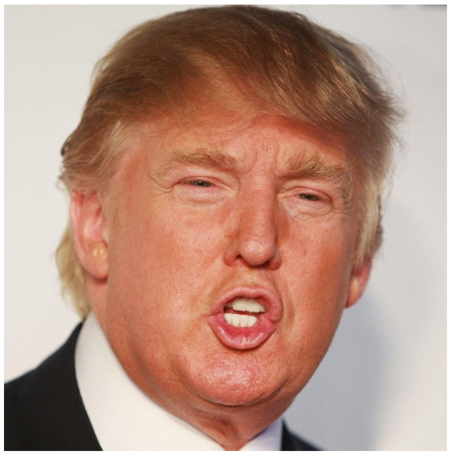
Neural Doodle

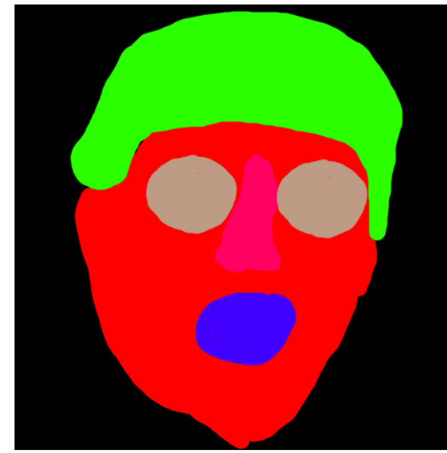
- Image analogy



Neural Doodle

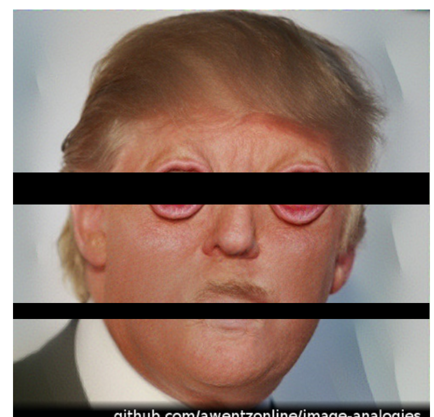
- Image analogy





恐怖連結，慎入！

https://raw.githubusercontent.com/awentzonline/image-analogies/master/examples/images/trump-image-analogy.jpg



Real-time Texture Synthesis

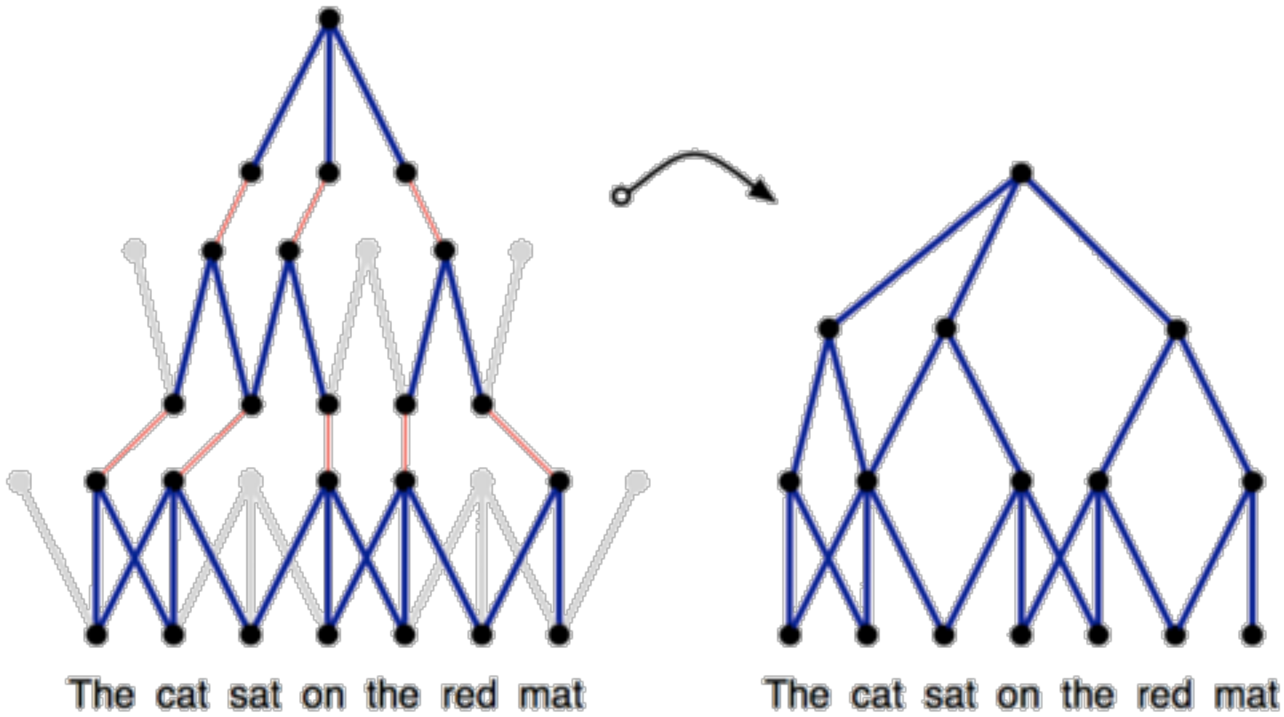
- Paper: https://arxiv.org/pdf/1604.04382v1.pdf
  - GAN: https://arxiv.org/pdf/1406.2661v1.pdf
  - VAE: https://arxiv.org/pdf/1312.6114v10.pdf
- Source Code : https://github.com/chuanli11/MGANs

Outline

- CNN(Convolutional Neural Networks) Introduction
- Evolution of CNN
- Visualizing the Features
- CNN as Artist
- Sentiment Analysis by CNN

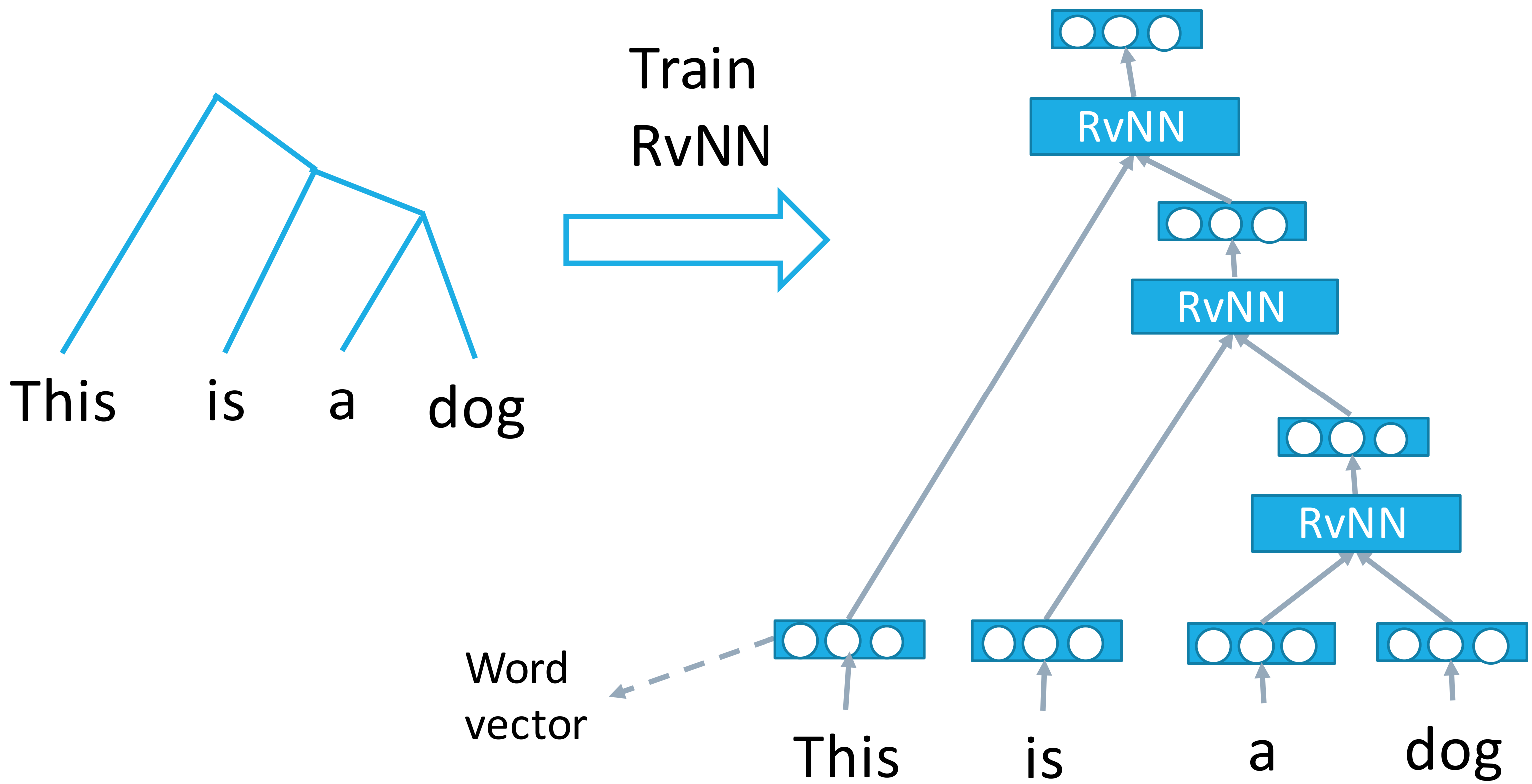
A Convolutional Neural Network for Modelling Sentences

- Paper: https://arxiv.org/abs/1404.2188
- Source code: https://github.com/FredericGodin/DynamicCNN



Drawbacks of Recursive Neural Networks(RvNN)

- Need human-labeled syntax tree during training



Drawbacks of Recursive Neural Networks(RvNN)

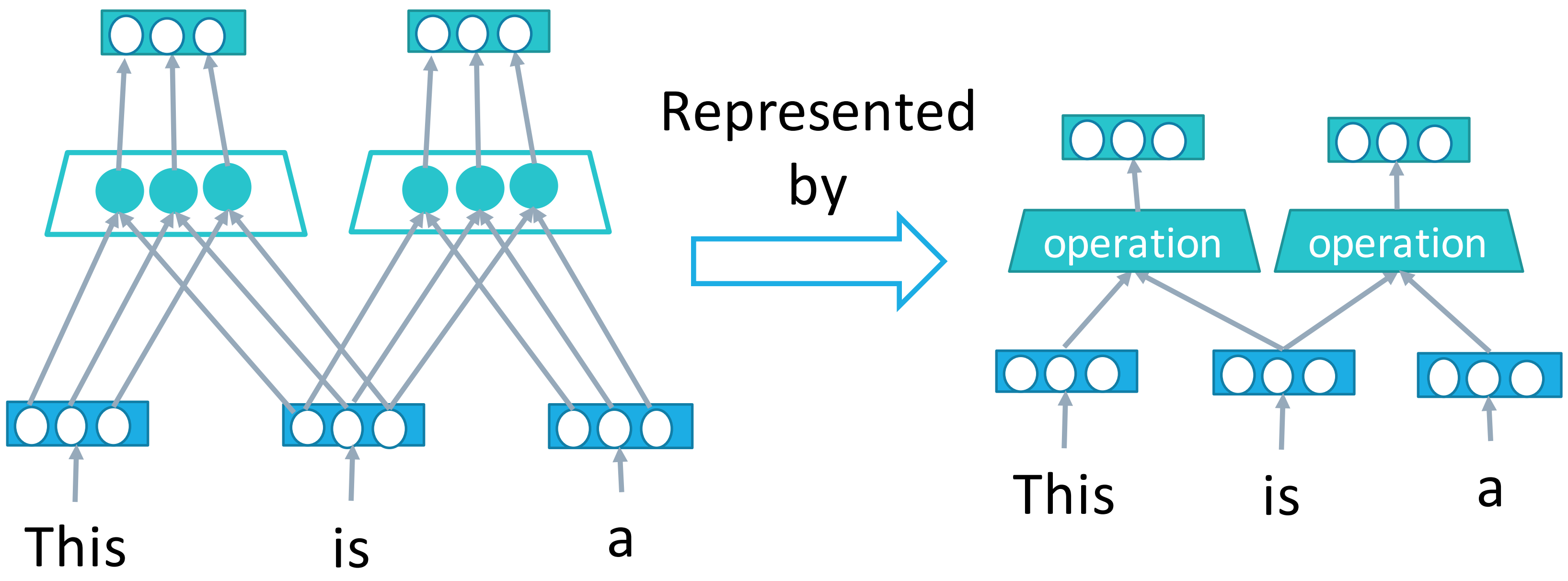
- Ambiguity in natural language

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

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

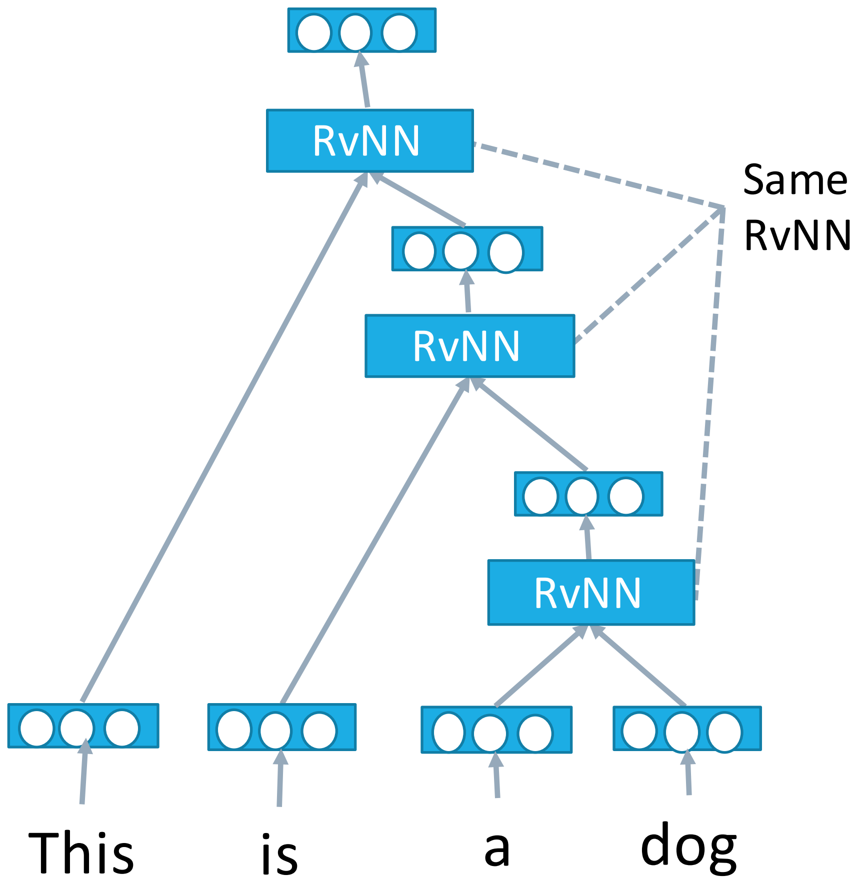
Element-wise 1D operations on word vectors

- 1D Convolution or 1D Pooling

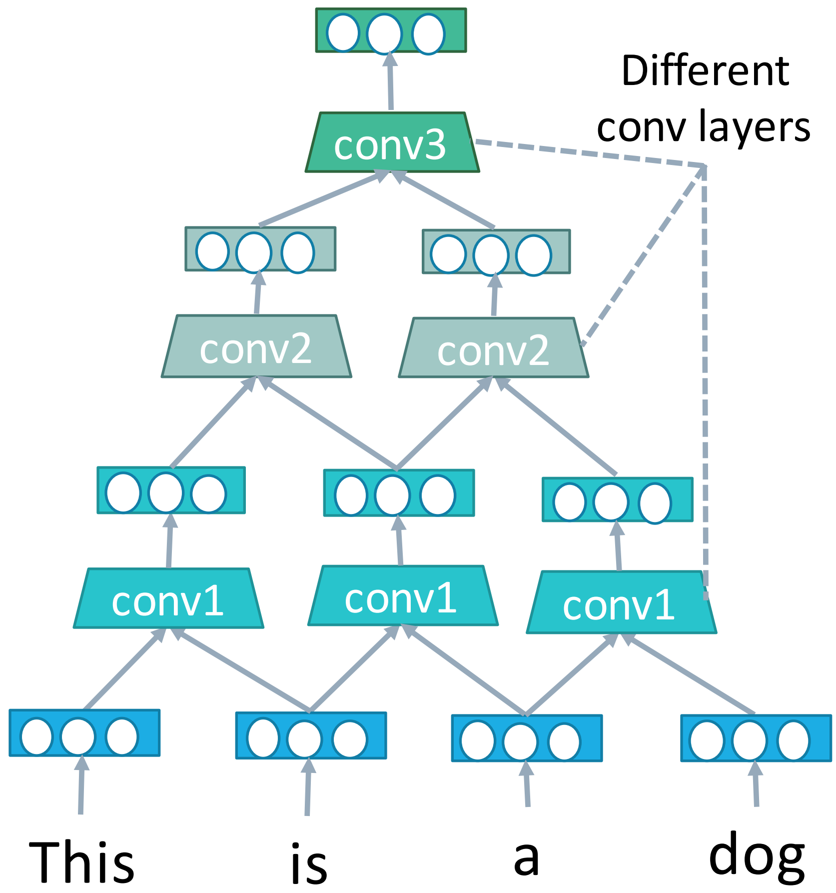


From RvNN to CNN

- RvNN

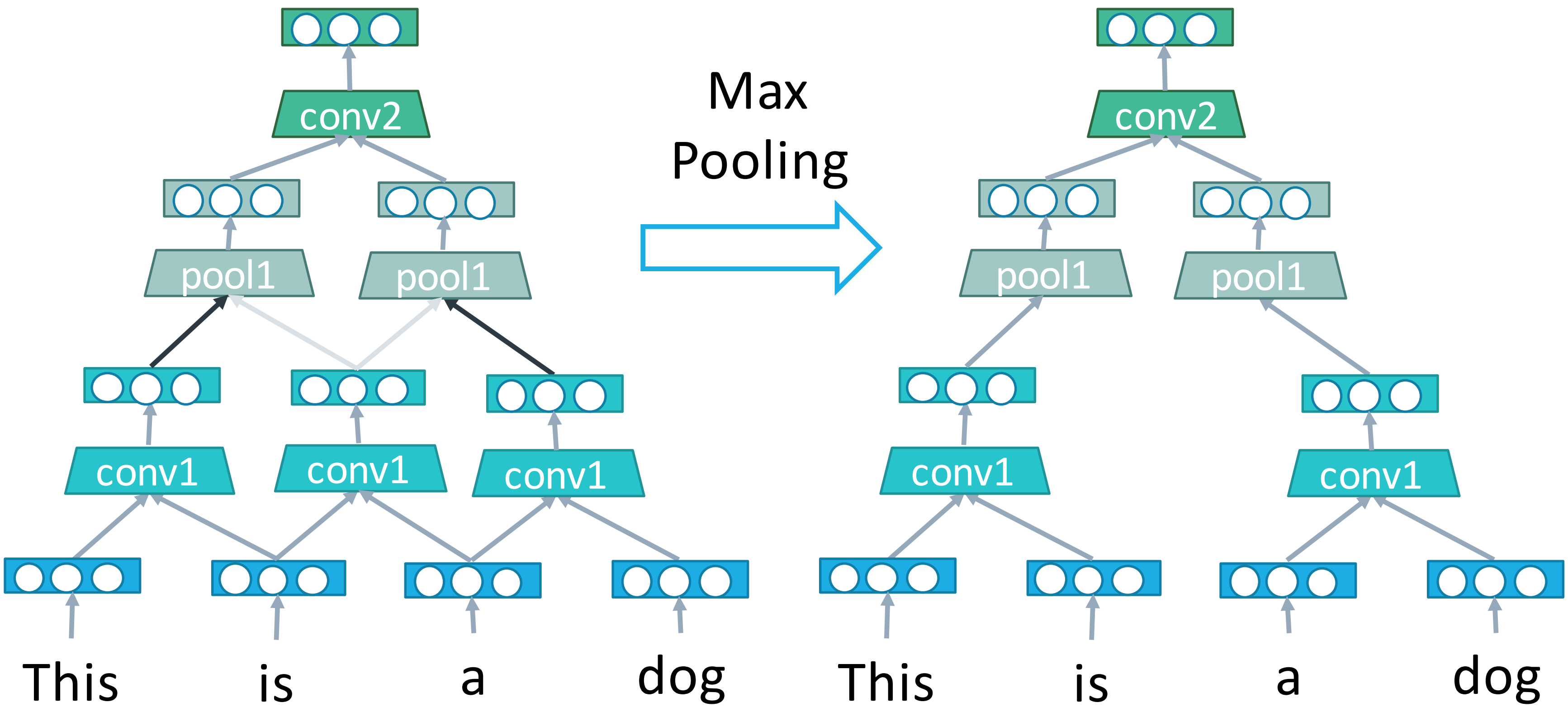


- CNN



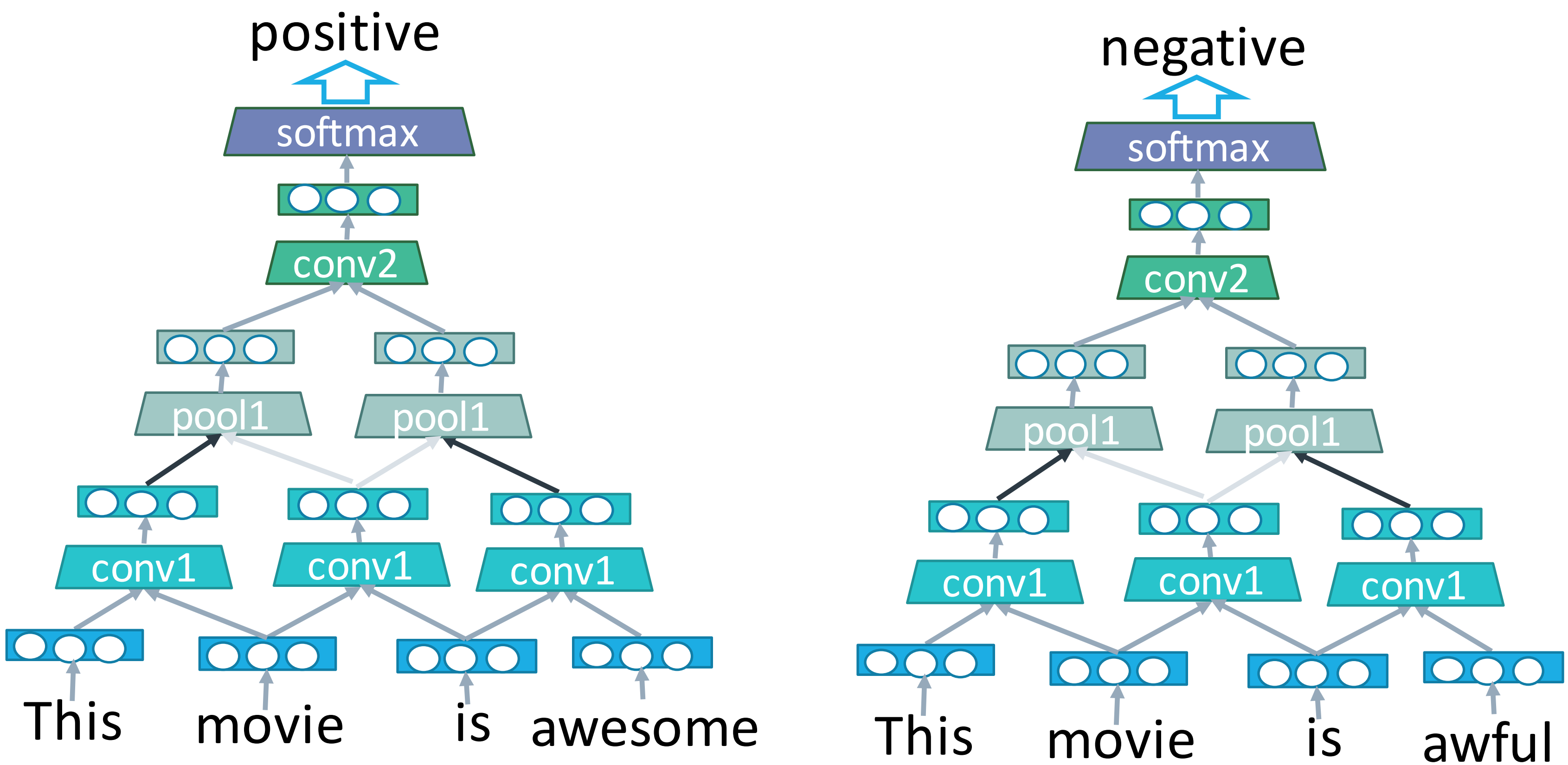
CNN with Max-Pooling Layers

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



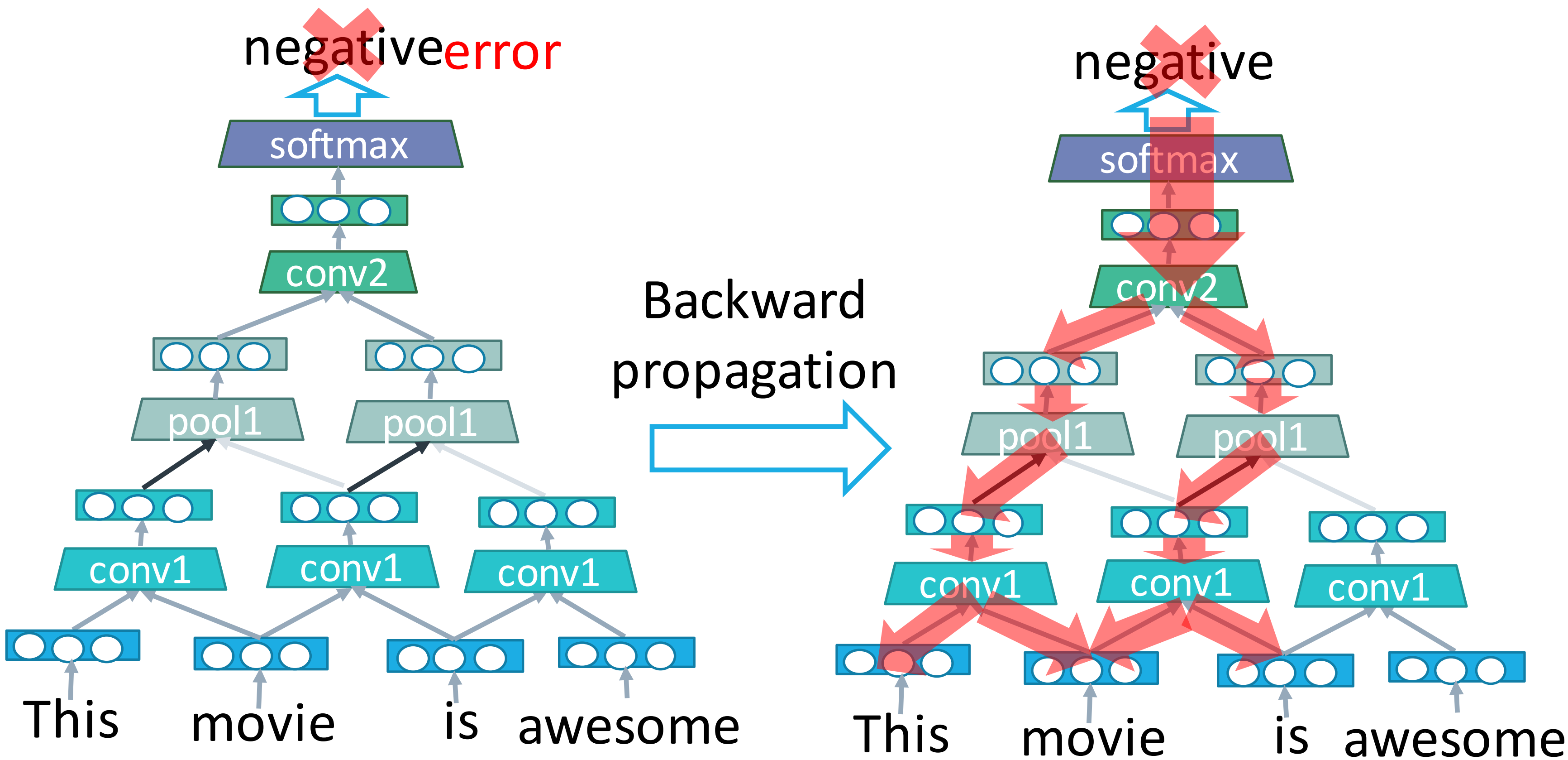
Sentiment Analysis by CNN

- Use softmax layer to classify the sentiments



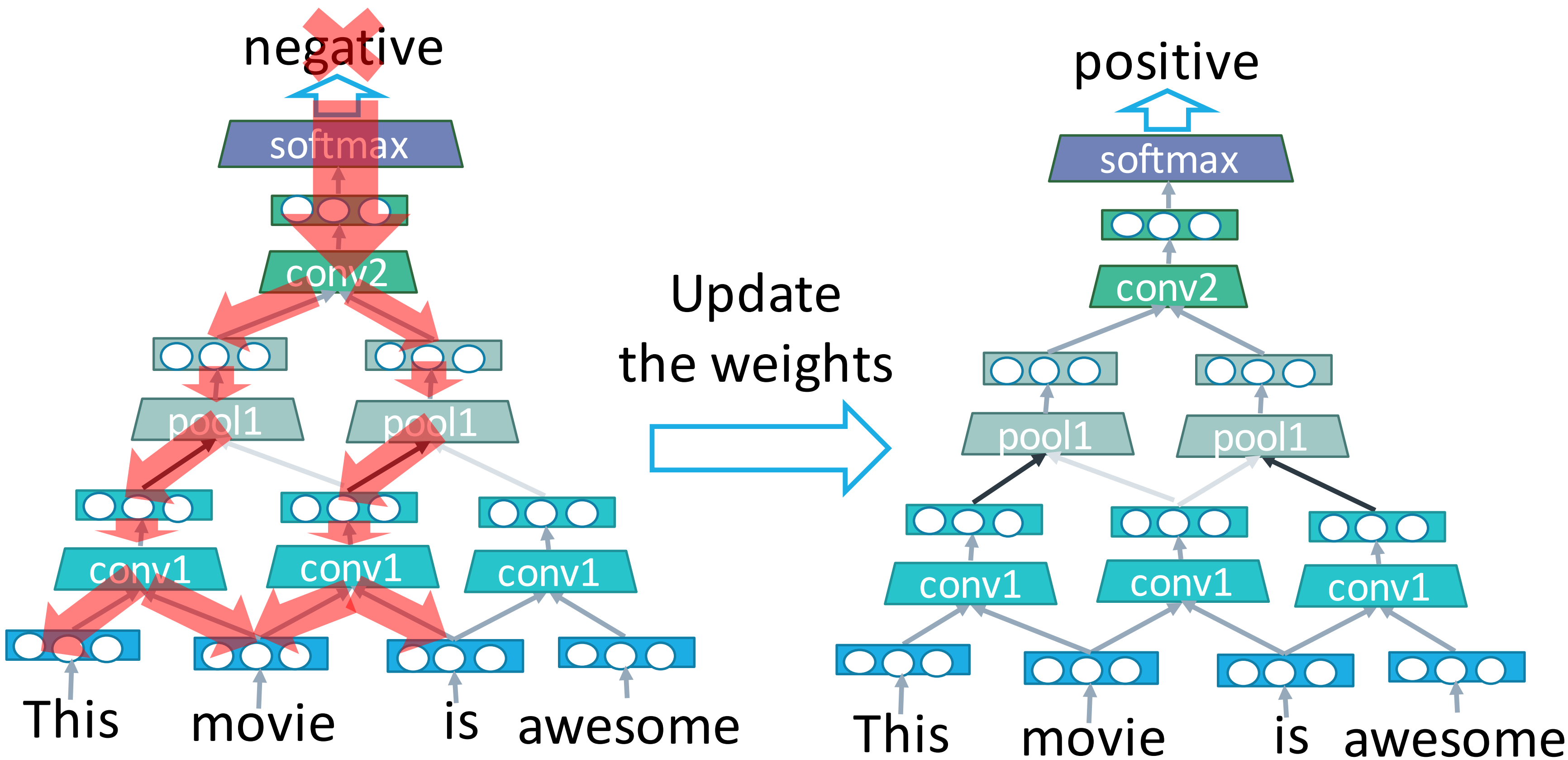
Sentiment Analysis by CNN

- Build the "correct syntax tree" by training



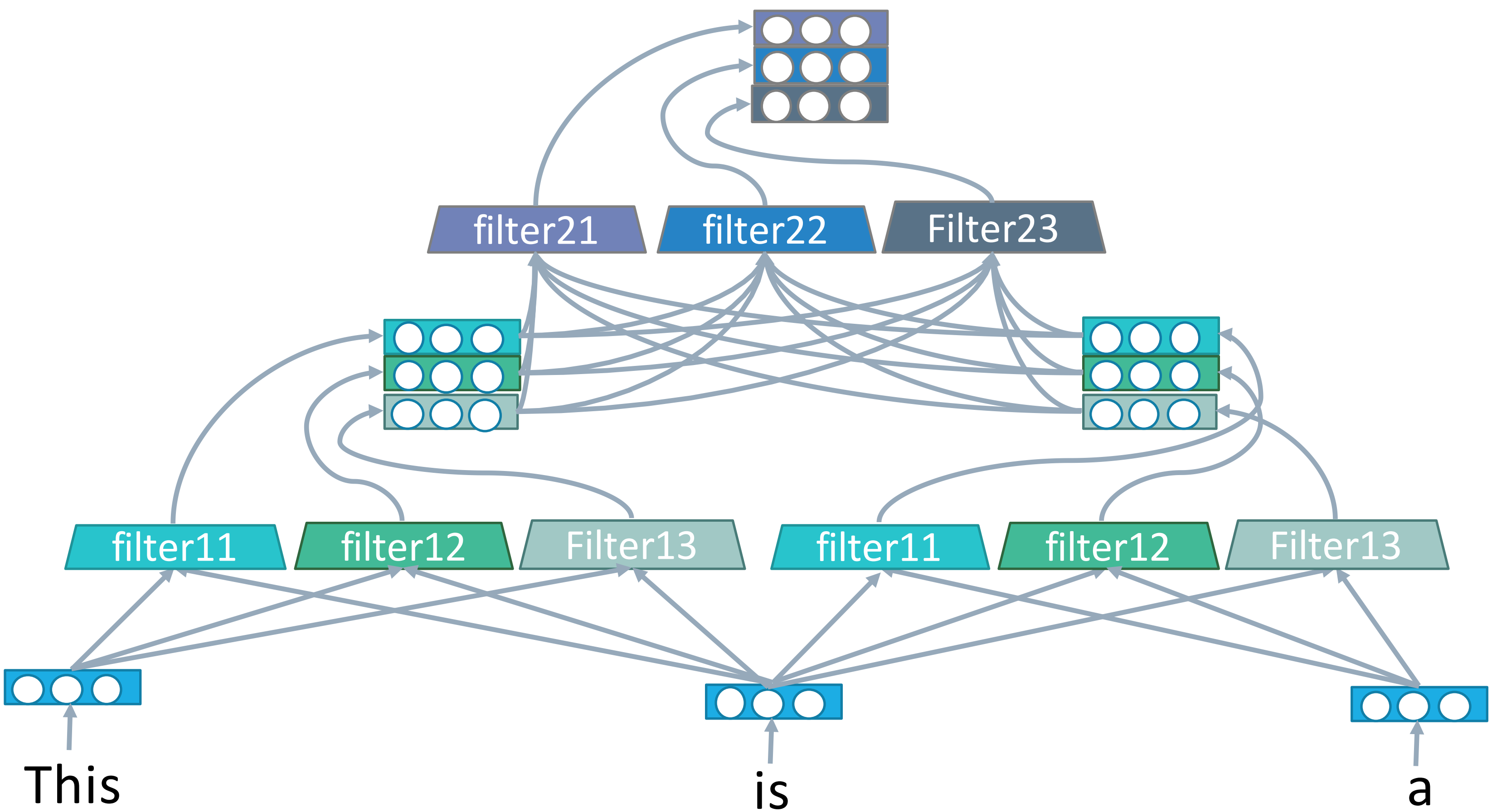
Sentiment Analysis by CNN

- Build the "correct syntax tree" by training



Multiple Filters

- Richer features than RNN



Sentence can’t be easily resized

- Image can be easily resized

resize

- Sentence can’t be easily resized

全台灣最高樓在台北市

resize

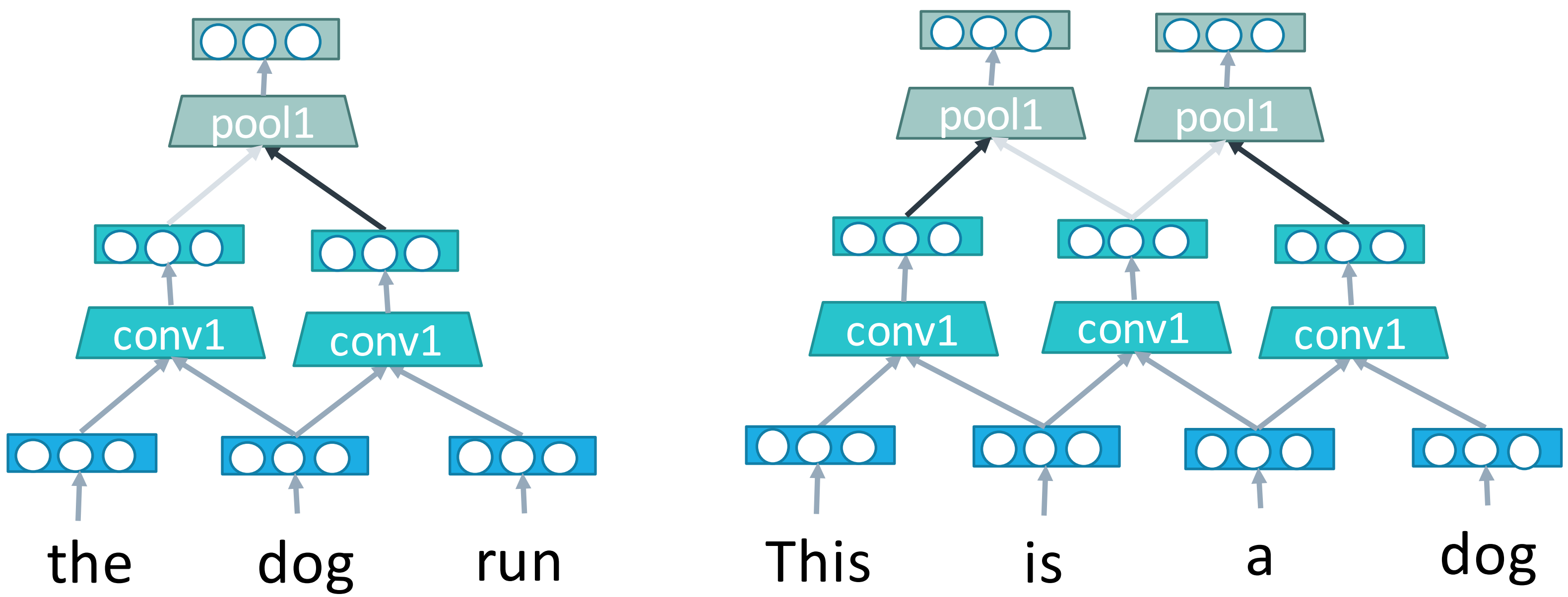
全台灣最高的高樓在台北市

全台灣最高樓在台北

台灣最高樓在台北

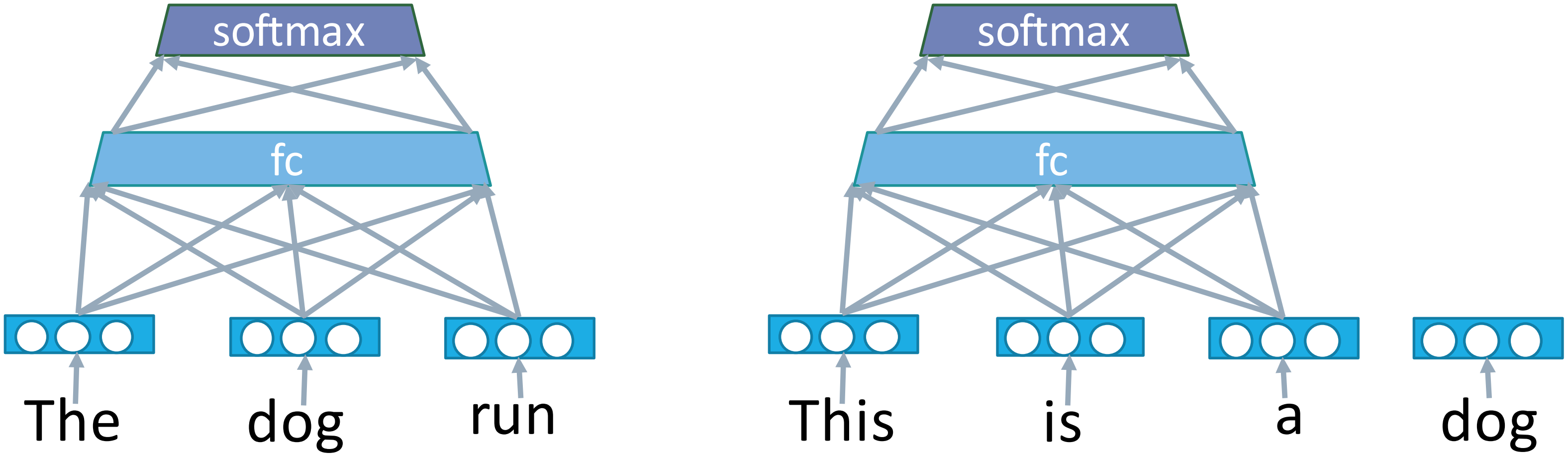
Various Input Size

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



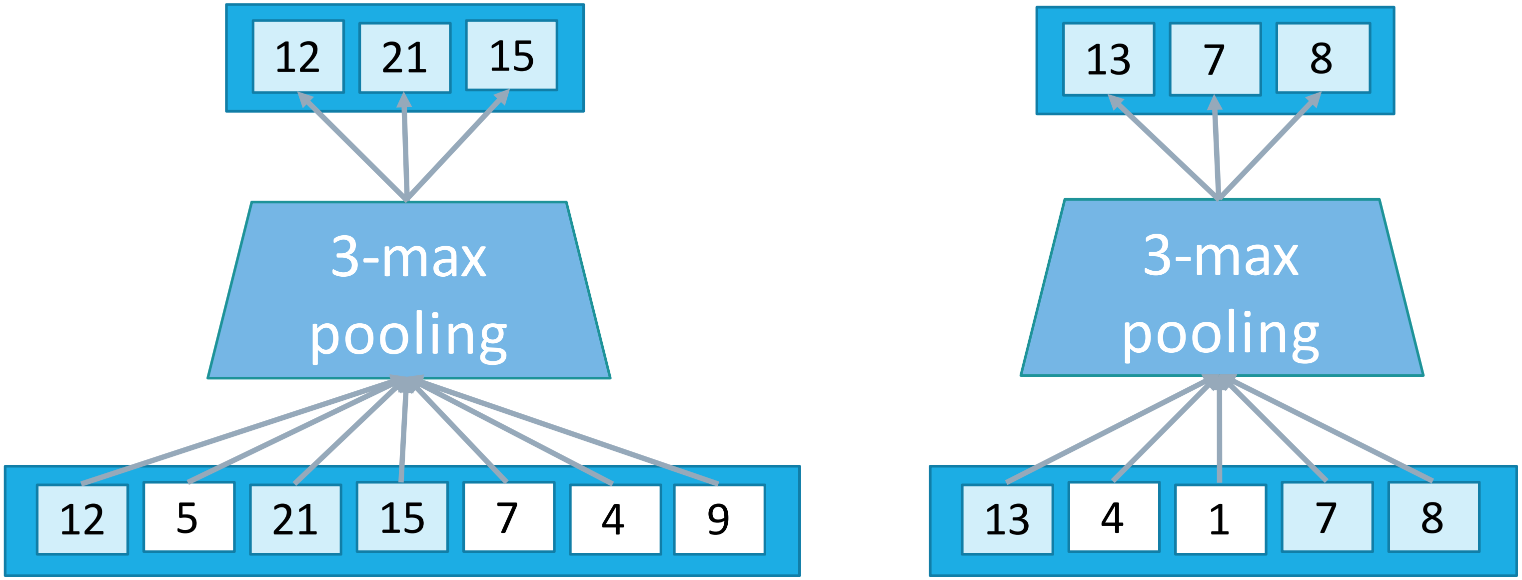
Various Input Size

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



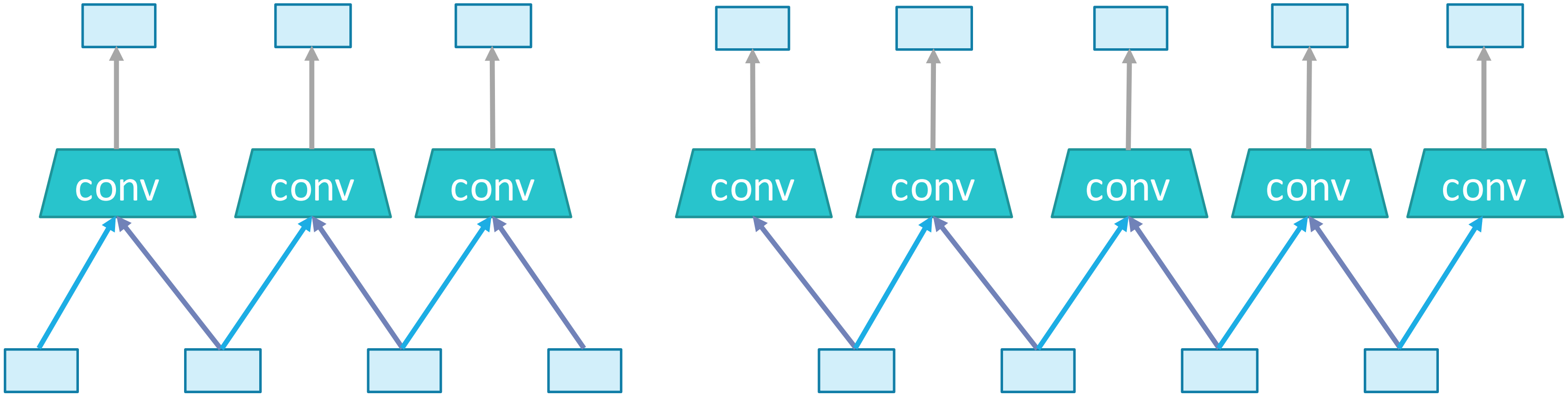
k-max Pooling

- 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

$$k_l = max(k_{top}, \lceil \frac{L-l}{L} s \rceil)$$

l : index of current layer

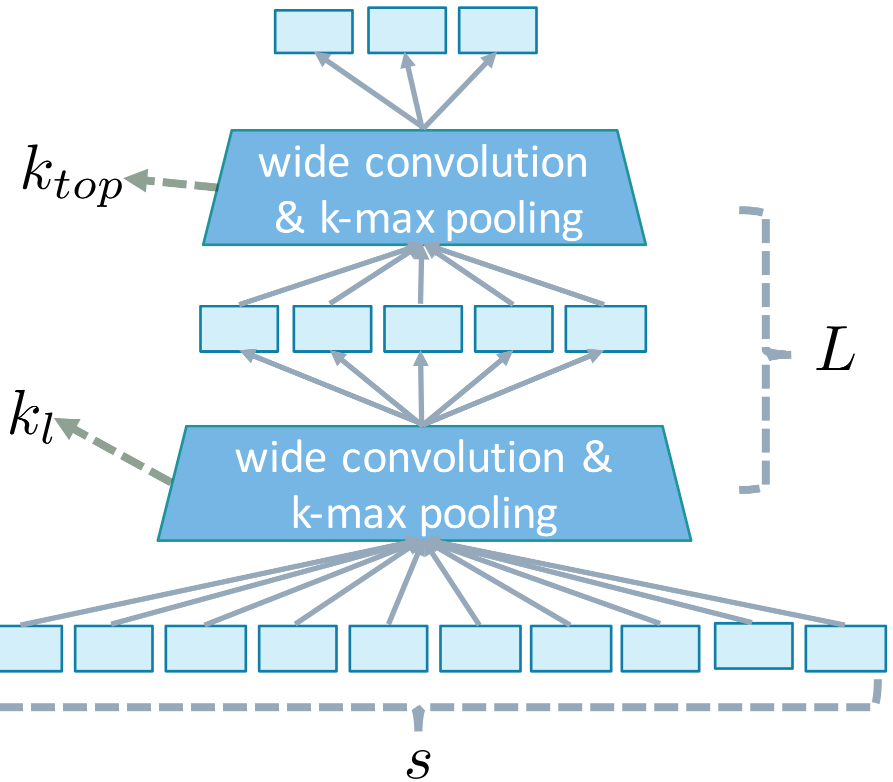
k_l : k of current layer

k_{top} : k of top layer

L : total number of layers

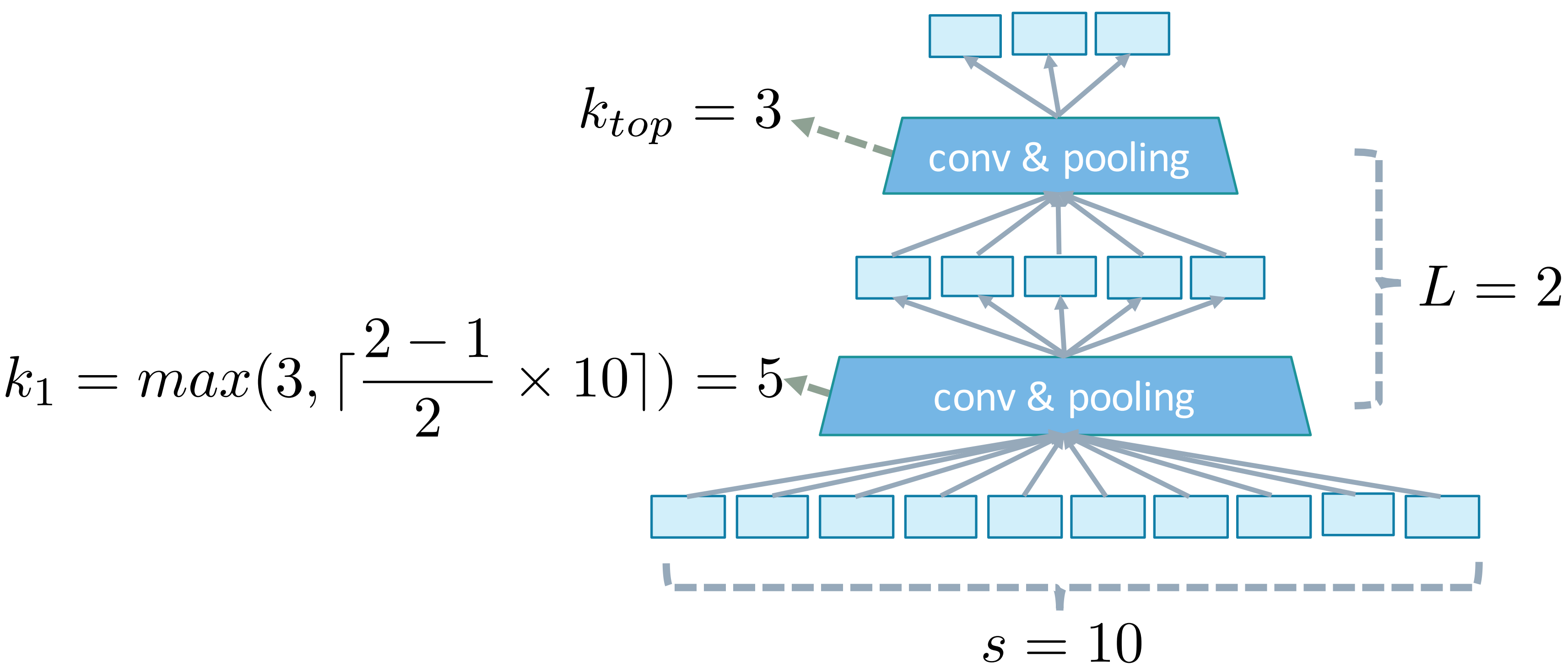
s : length of input sentence

k_{top} and L are constants



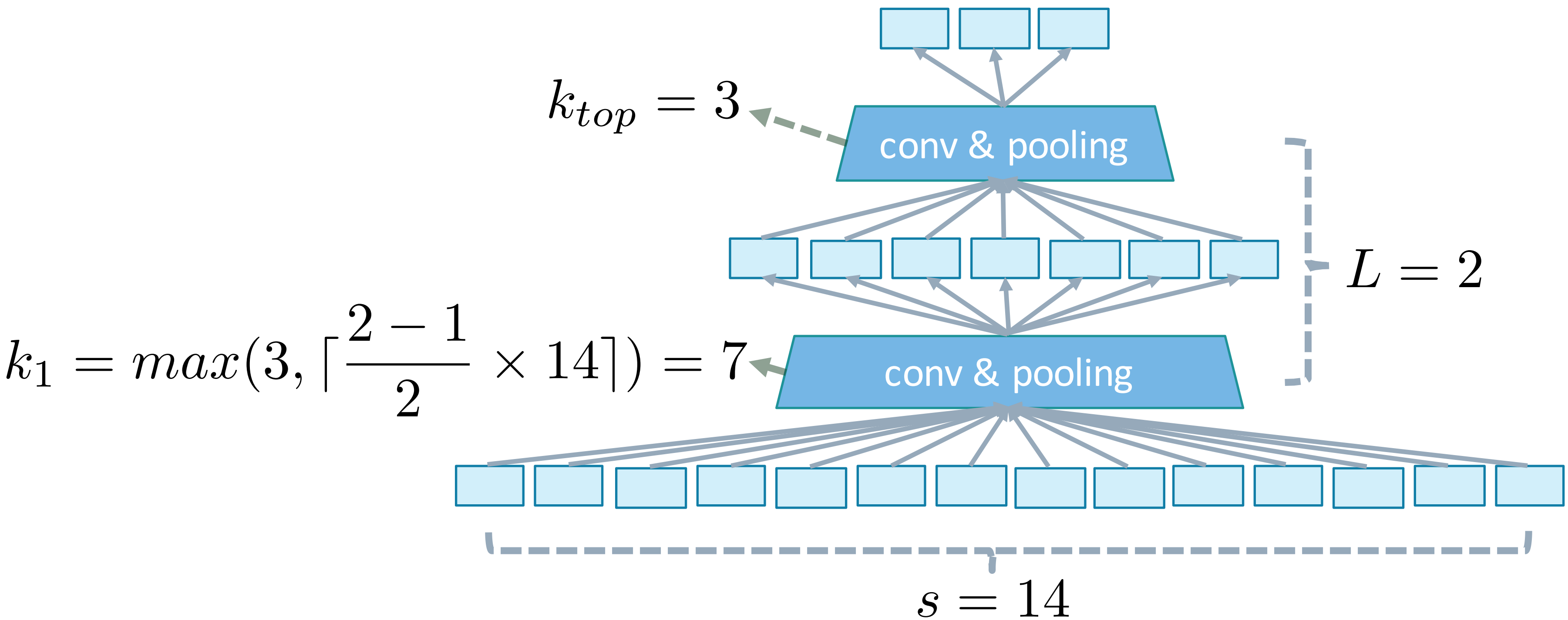
Dynamic k-max Pooling

$$k_l = max(k_{top}, \lceil \frac{L-l}{L} s \rceil)$$



Dynamic k-max Pooling

$$k_l = max(k_{top}, \lceil \frac{L-l}{L} s \rceil)$$

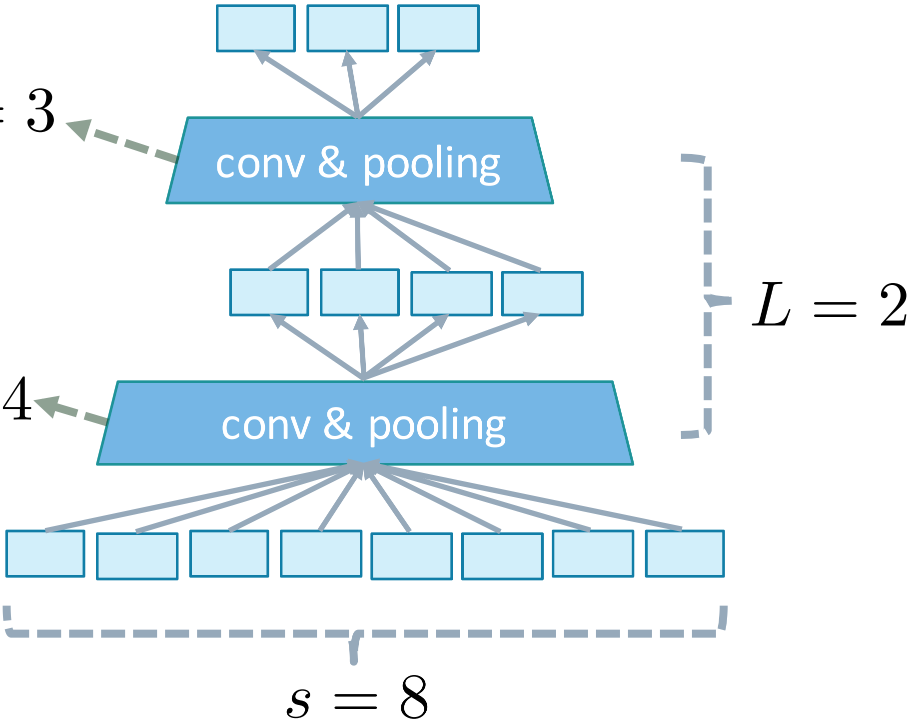


Dynamic k-max Pooling

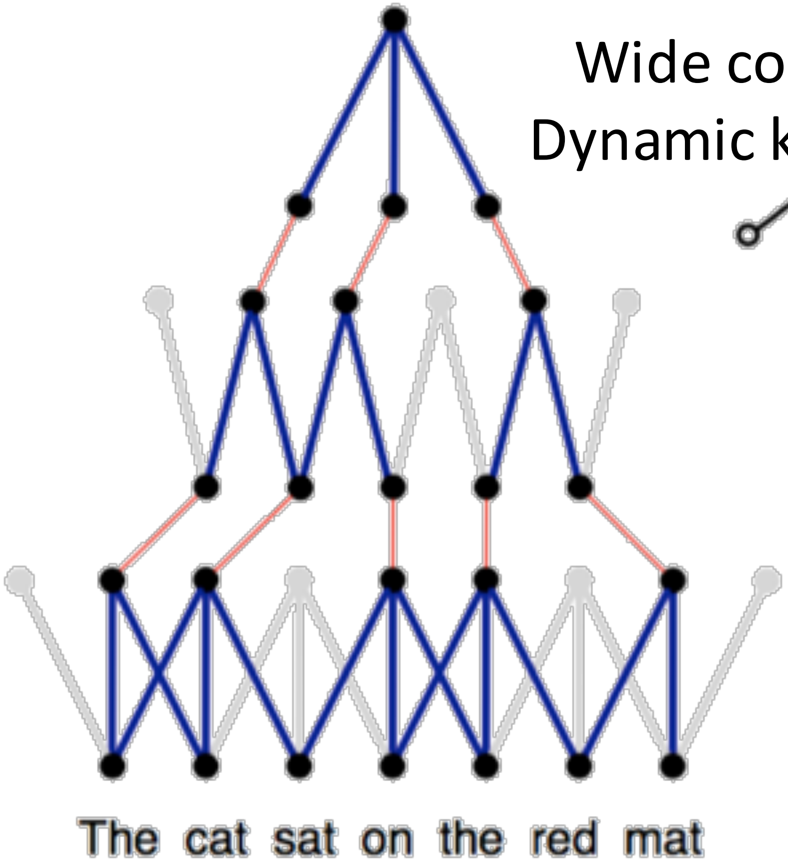
$$k_l = max(k_{top}, \lceil \frac{L-l}{L} s \rceil)$$

$$k_{top} = 3$$

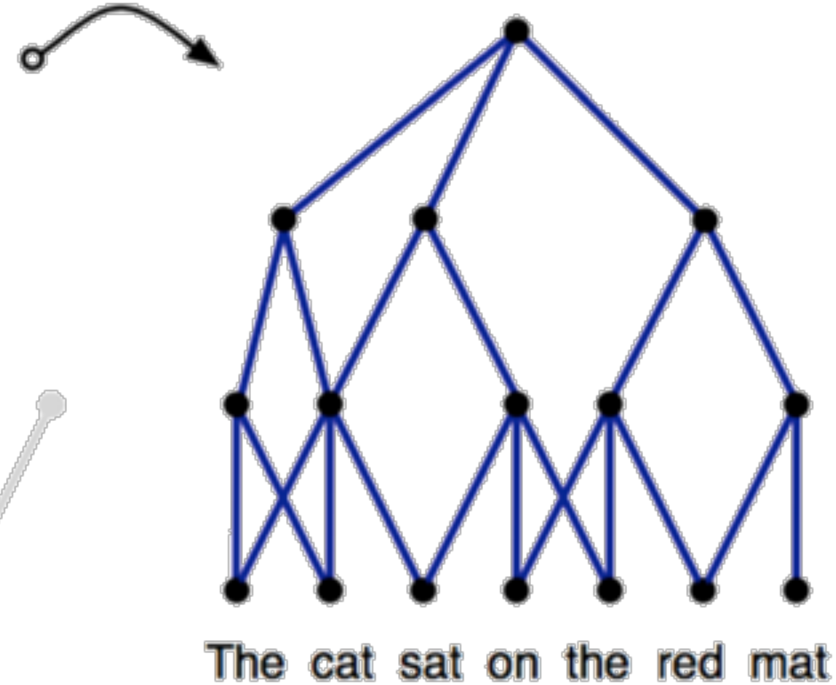
$$k_1 = max(3, \lceil \frac{2-1}{2} \times 8 \rceil) = 4$$



Dynamic k-max Pooling

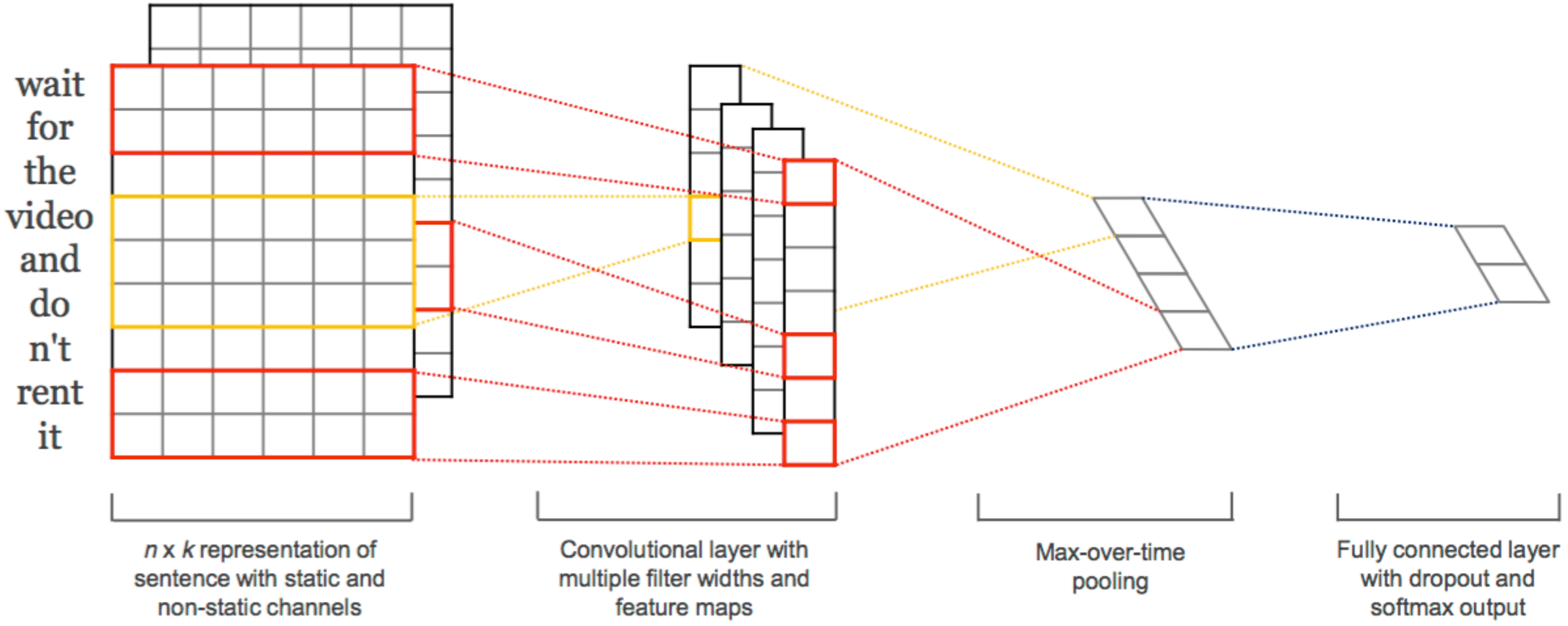


Wide convolution &
Dynamic k-max pooling



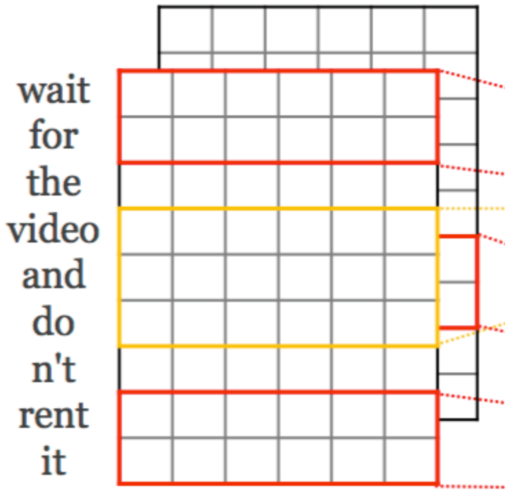
Convolutional Neural Networks for Sentence Classification

- Paper: http://www.aclweb.org/anthology/D14-1181
- Sourcee code: https://github.com/yoonkim/CNN_sentence



Static & Non-Static Channel

- Pretrained by word2vec
- Static: fix the values during training
- Non-Static: update the values during training



About the Lecturer

Mark Chang

HTC Research & Healthcare
Deep Learning Algorithms
Research Engineer

- Email: ckmarkoh at gmail dot com
- Blog: http://cpmarkchang.logdown.com
- Github: https://github.com/ckmarkoh
- Slideshare: http://www.slideshare.net/ckmarkohchang
- Youtube: https://www.youtube.com/channel/UCckNPGDL21aznRhl3EijRQw