Neural Networks

History of Neural Networks

第一阶段

- 1943年, McCulloch和Pitts 提出第一个神经元数学模型, 即M-P模型, 并从原理上证明了人工神经网络能够计算任何算数和逻辑函数

- 1958年, Rosenblatt 提出感知机网络（Perceptron）模型和其学习规则

- 1969年, Minsky和Papert 发表《Perceptrons》一书, 指出单层神经网路不能解决非线性问题, 多层网络的训练算法尚无希望. 这个论断导致神经网络进入低谷

History of Neural Networks

第二阶段

- 1986年, Rumelhart 等编辑的著作《Parallel Distributed Processing: Explorations in the Microstructures of Cognition》报告了反向传播算法

- 1987年, IEEE 在美国加州圣地亚哥召开第一届神经网络国际会议（ICNN）

- 90年代初, 伴随统计学习理论和SVM的兴起, 神经网络由于理论不够清楚, 试错性强, 难以训练, 再次进入低谷

History of Neural Networks

第三阶段

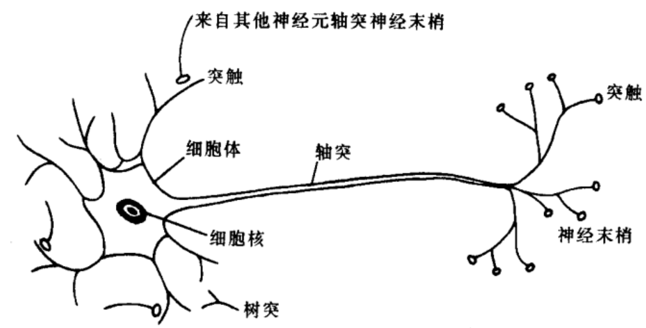
- 2006年, Hinton提出了深度信念网络(DBN), 通过“预训练+微调”使得深度模型的最优化变得相对容易
- 2012年, Hinton 组参加ImageNet 竞赛, 使用 CNN 模型以超过第二名10个百分点的成绩夺得当年竞赛的冠军
- 伴随云计算、大数据时代的到来，计算能力的大幅提升，使得深度学习模型在计算机视觉、自然语言处理、语音识别等众多领域都取得了较大的成功
- 2018年图灵奖-Hinton, Bengio, LeCun

Neural Network Intro

“神经网络是由具有适应性的简单单元组成的广泛并行互联的网络, 它的组织能够模拟生物神经系统对真实世界物体所作出的反应”

[Kohonen, 1988]

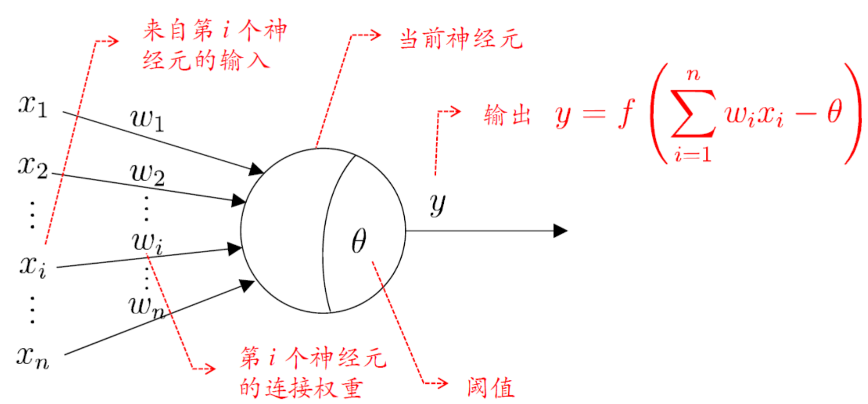
- 机器学习中的神经网络通常是指“神经网络学习”或者机器学习与神经网络两个学科的交叉部分
- 神经元模型即上述定义中的“简单单元”是神经网络的基本成分
- 生物神经网络：每个神经元与其他神经元相连, 当它“兴奋”时, 就会向相连的神经云发送化学物质, 从而改变这些神经元内的电位；如果某神经元的电位超过一个“阈值”, 那么它就会被激活, 即“兴奋”起来, 向其它神经元发送化学物质



Neural Network Intro

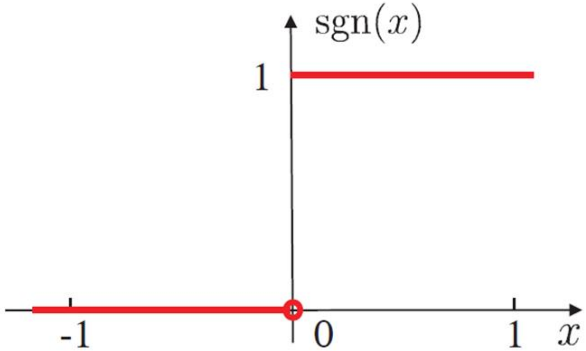
M-P 神经元模型 [McCulloch and Pitts, 1943]

- 输入：来自其它n个神经元传递过来的输入信号
- 处理：输入信号通过带权重的连接进行传递, 神经元接受到总输入值将与神经元的阈值进行比较
- 输出：通过激活函数的处理以得到输出



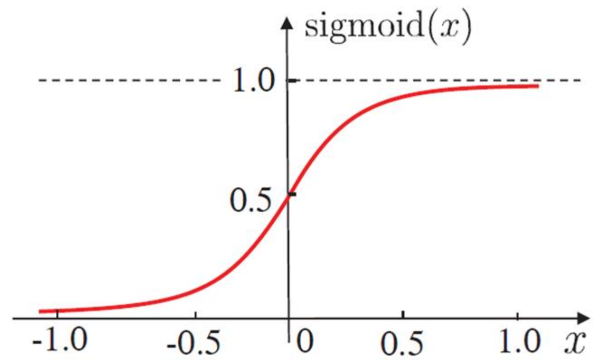
Neural Network Intro

激活函数(Activation function)



$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

(a) 阶跃函数

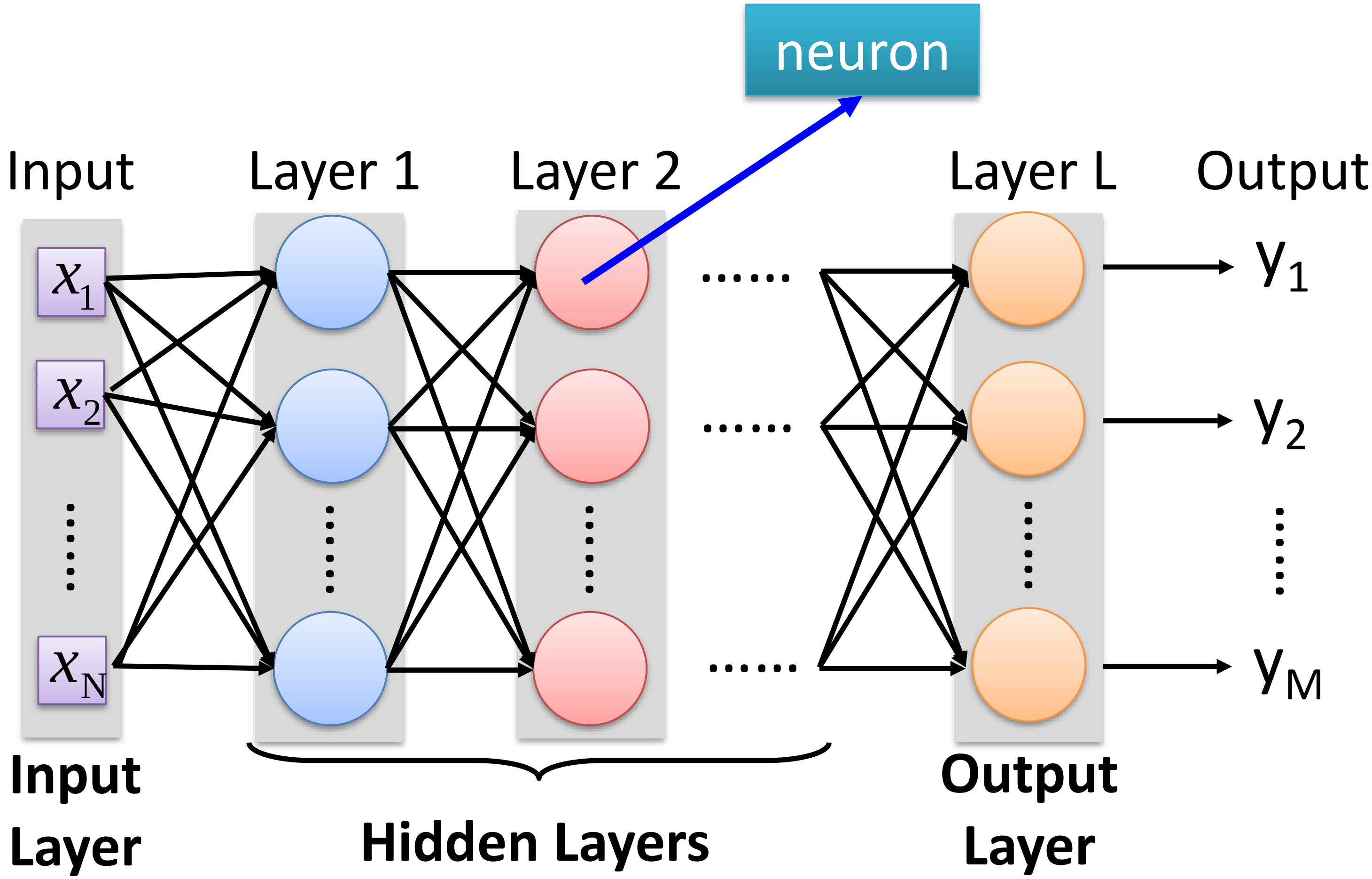


$$\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

(b) Sigmoid 函数

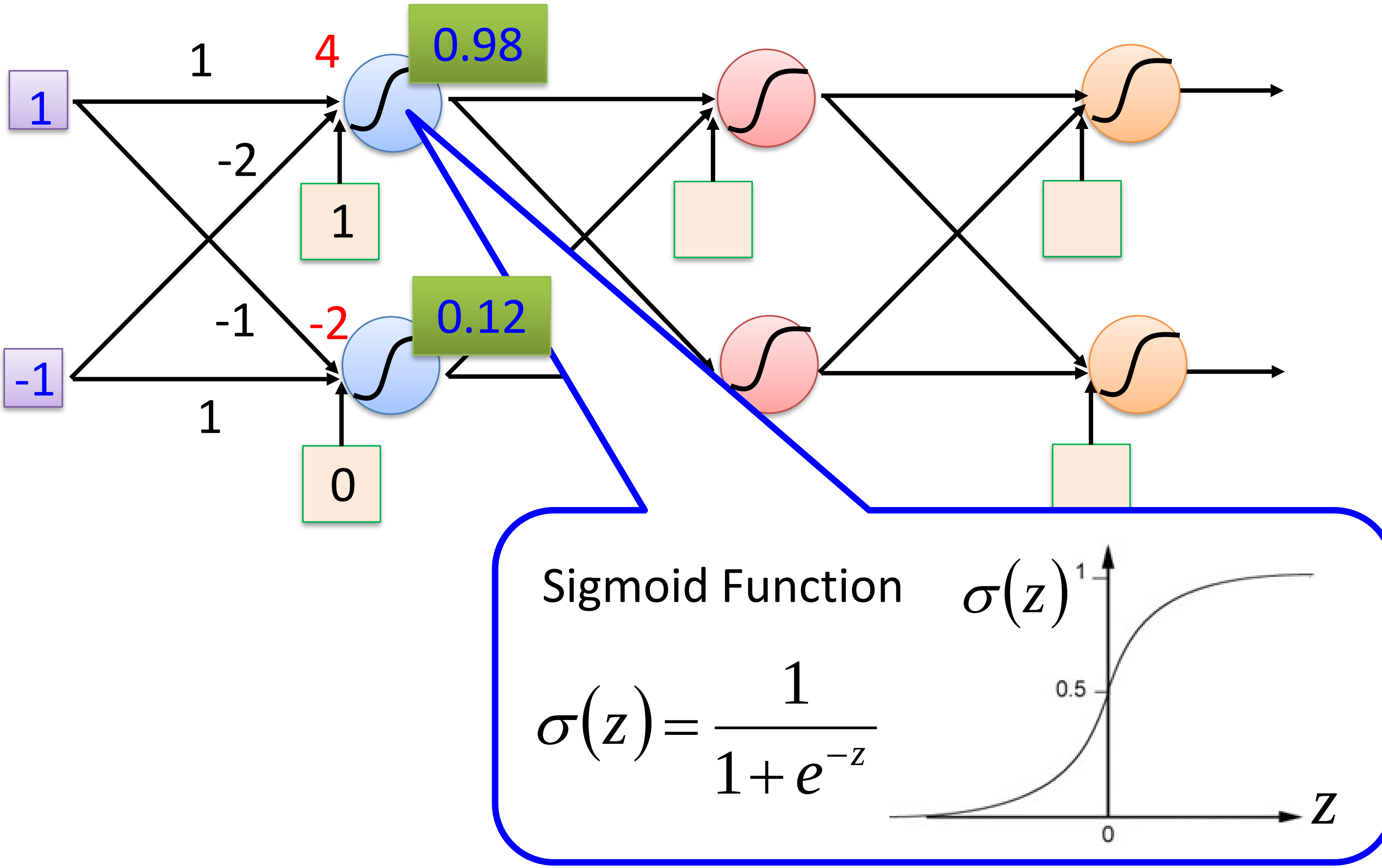
- 理想激活函数是阶跃函数, 0表示抑制神经元而1表示激活神经元
- 阶跃函数具有不连续、不光滑等不好的性质, 常用的是 Sigmoid 函数

Neural Network



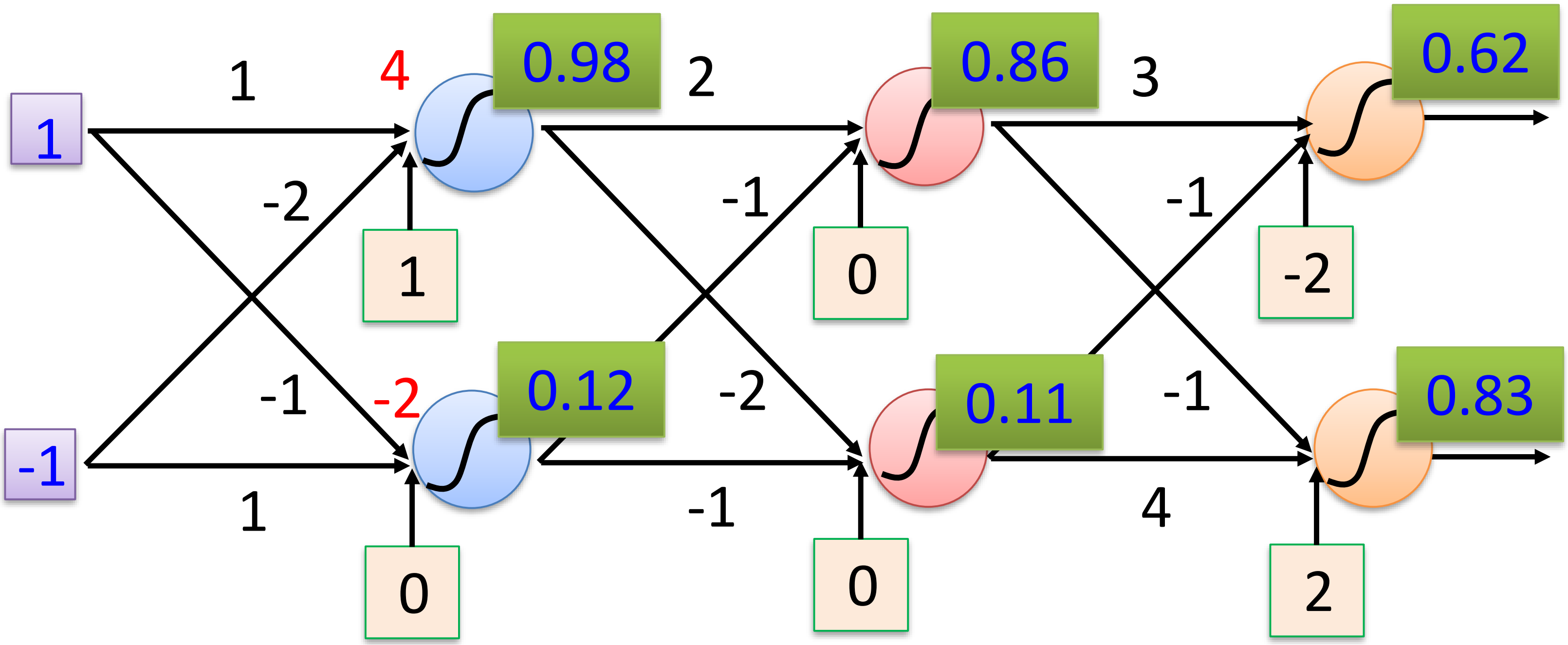
Deep means many hidden layers

Example of Neural Network

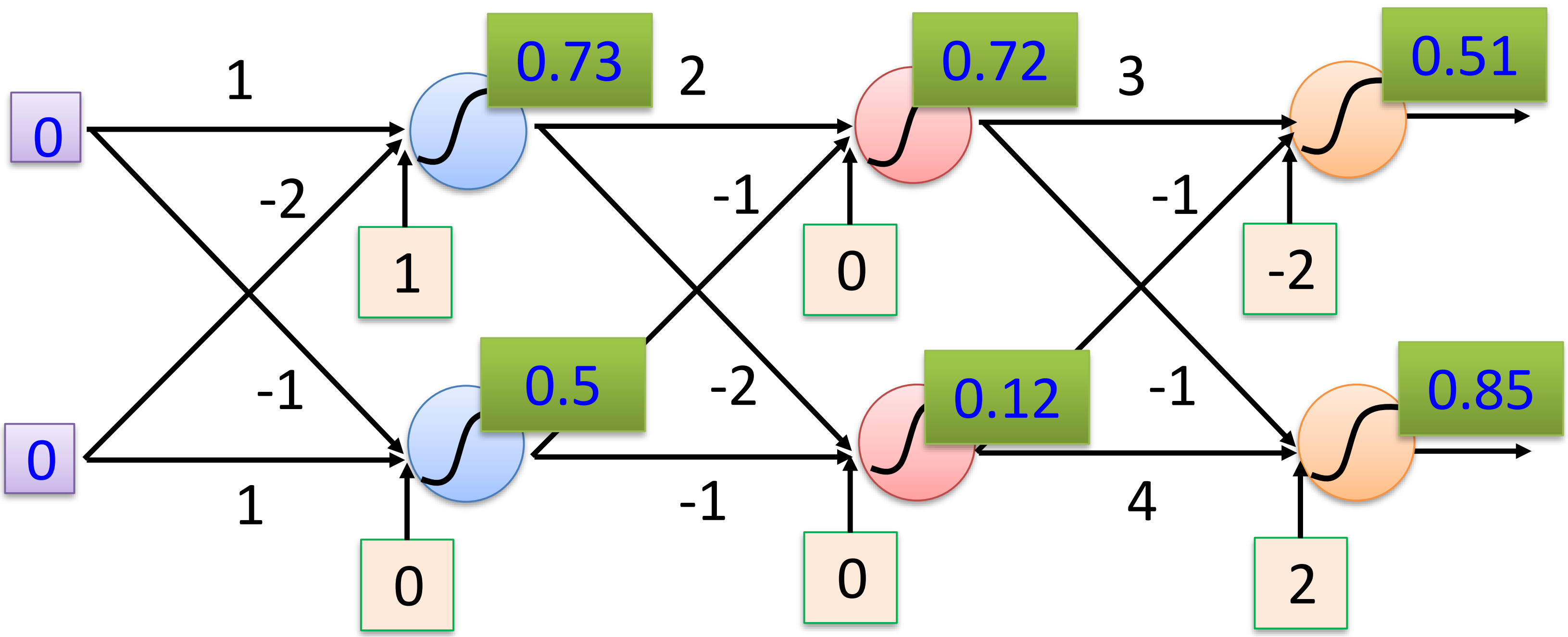


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Example of Neural Network



Example of Neural Network

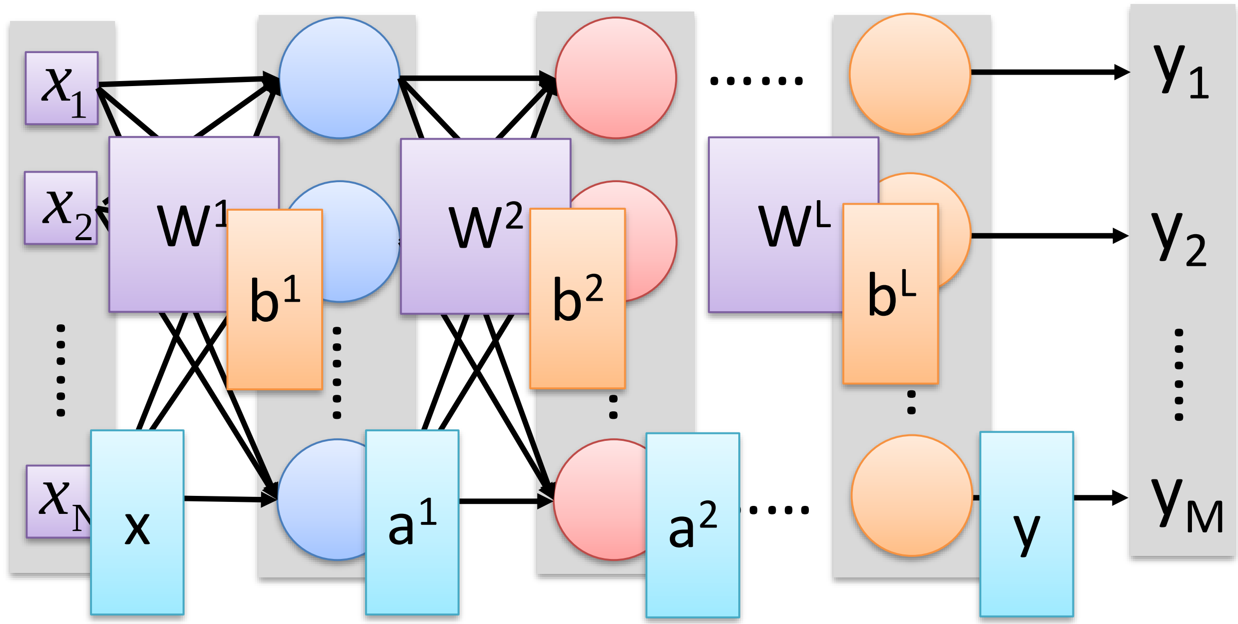


$$f: R^2 \to R^2$$

$$f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

Different parameters define different function

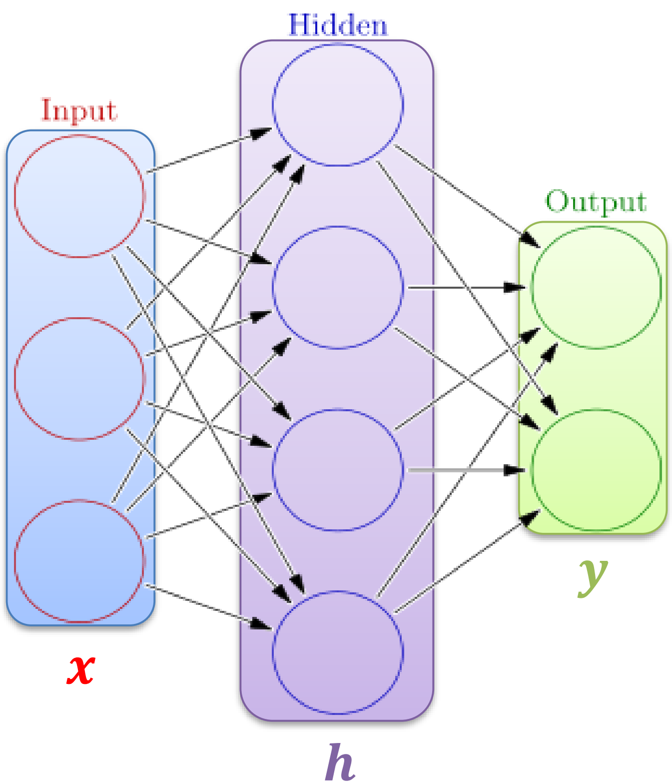
Neural Network



$$y = f(x)$$

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

Neural Network Intro



Weights

$$\boldsymbol{h} = \boldsymbol{\sigma}(\mathbf{W}_1 \boldsymbol{x} + \boldsymbol{b}_1)$$

$$\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{W}_2 \boldsymbol{h} + \boldsymbol{b}_2)$$

Activation functions

How do we train?

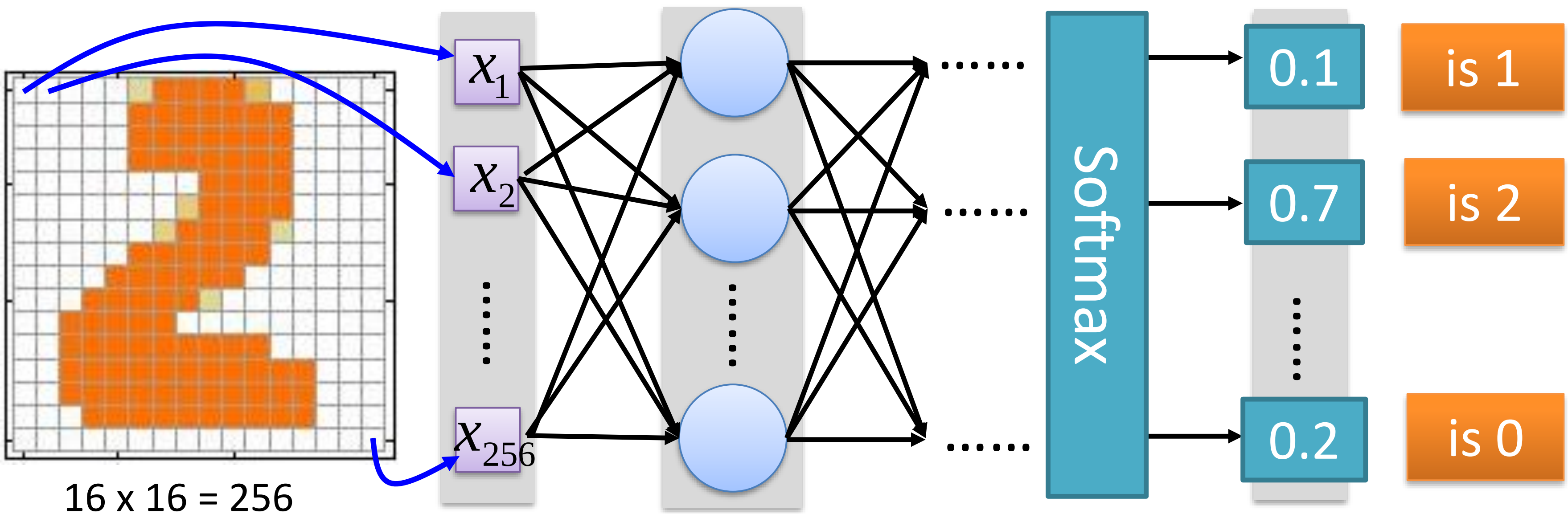
4 + 2 = 6 neurons (not counting inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases

---

26 learnable **parameters**

How to learn network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



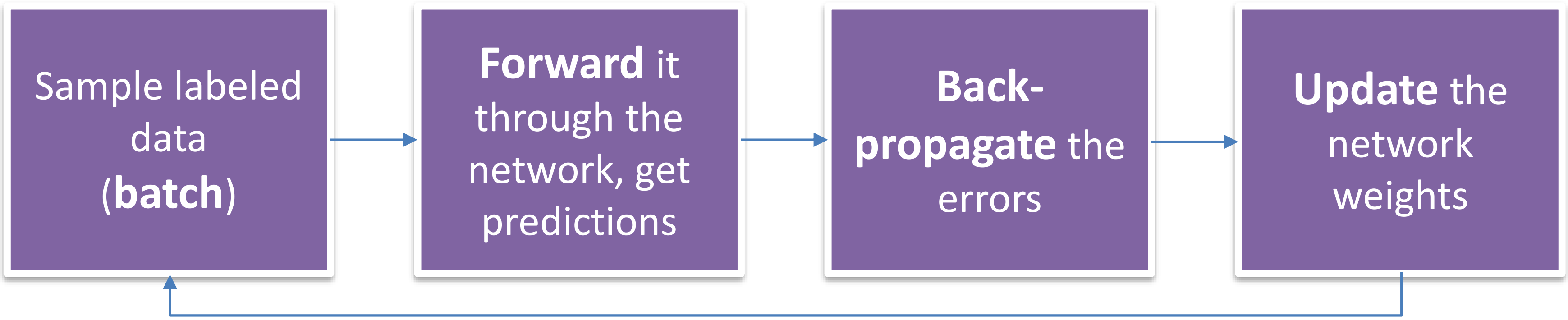
Set the network parameters θ such that ......

Input m value

Input:  y_2 has the maximum value

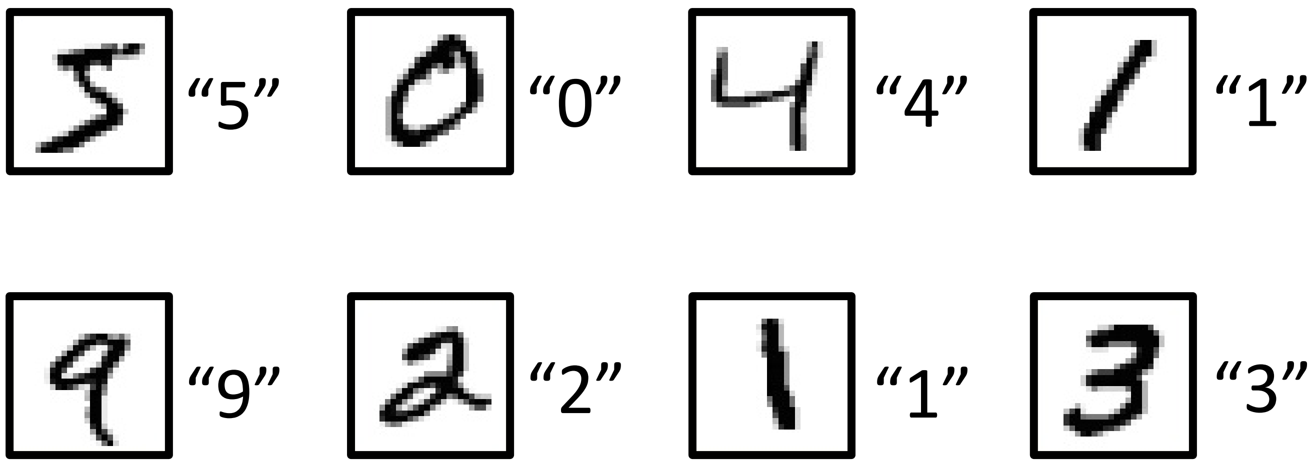
How to let the neural network achieve this

How to learn network parameters



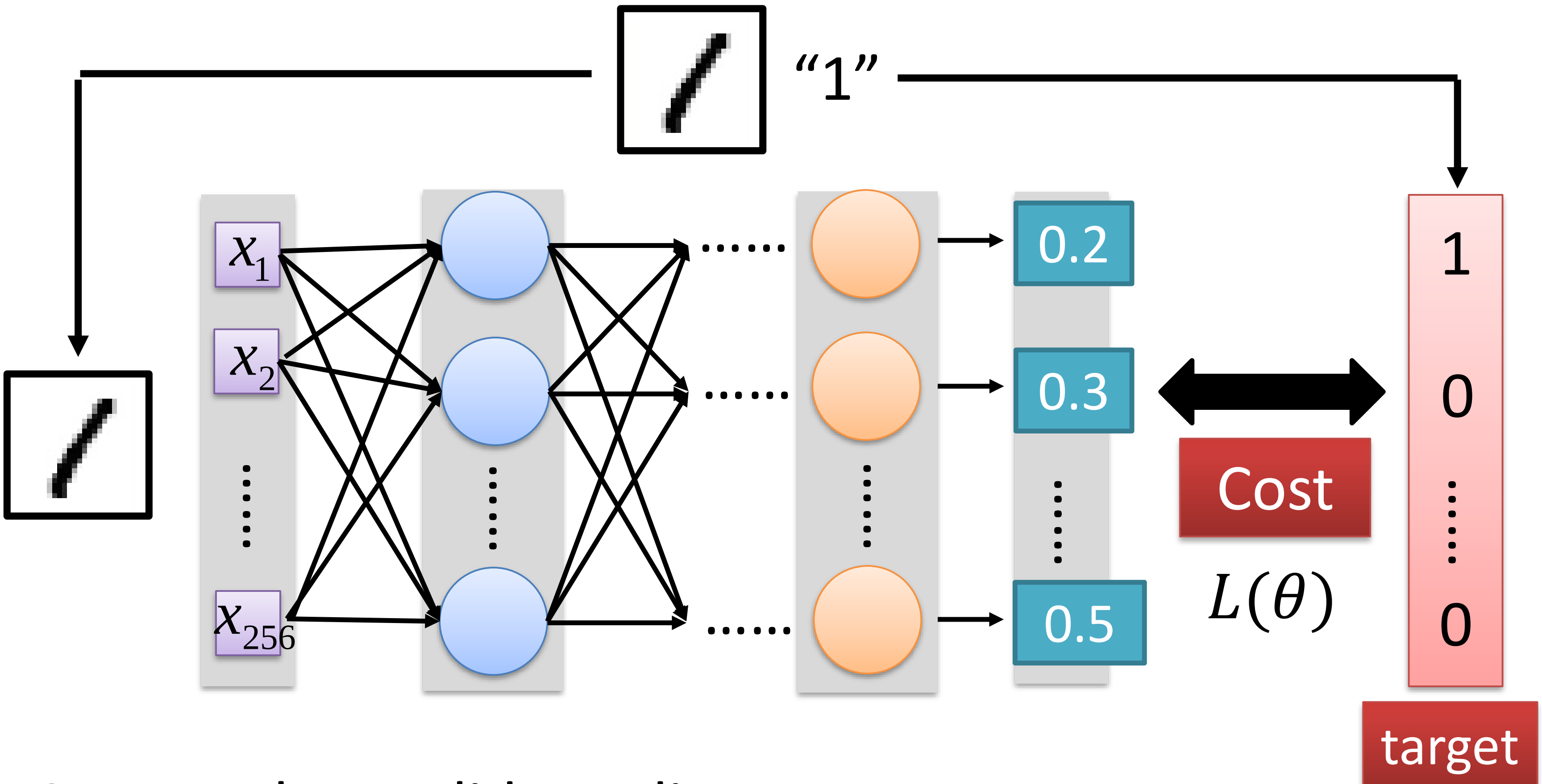
Training Data

- Preparing training data: images and their labels



Using the training data to find the network parameters.

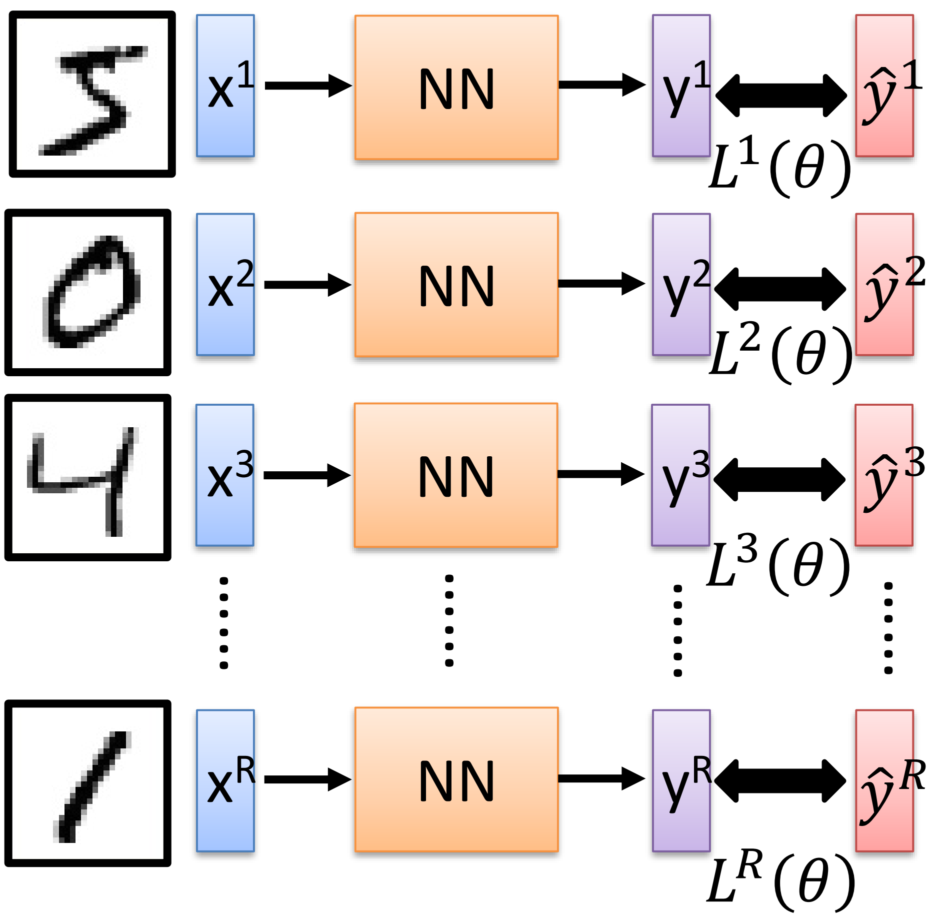
Cost



Cost can be Euclidean distance or cross entropy of the network output and target

Total Cost

For all training data …



Total Cost:

$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

How bad the network parameters θ is on this task

Find the network parameters θ^* that minimize this value

Optimization

- Back Propagation
  - Chain rule of computing gradient
  - Forward

$$x^{(1)} = f(w^{(0)} input)$$

$$x^{(l+1)} = f\left(w^{(l)} x^{(l)}\right)$$

$$\ldots\ldots$$

$$loss = g(x^{(L)})$$

Optimization

- Backward

$$\frac{\partial loss}{\partial x^{(L)}} = g'(x^{(L)})$$

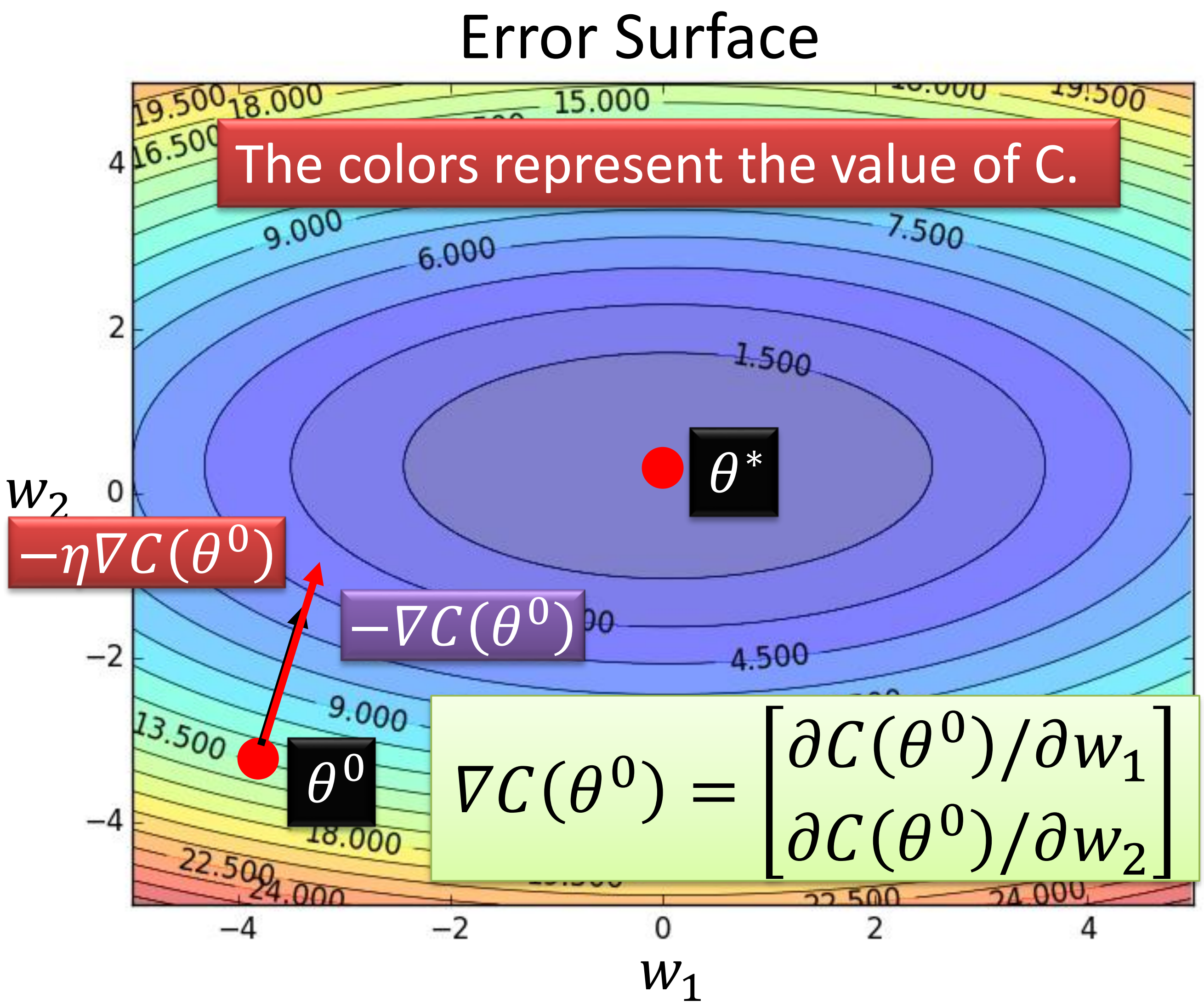
$$x^{(L+1)} = f\left(w^{(L)}x^{(L)}\right)$$

$$\frac{\partial loss}{\partial x^{(L-1)}} = W^{(L-1)} \cdot \frac{\partial loss}{\partial x^{(L)}} \odot f'\left(W^{(L-1)}x^{(L-1)}\right)$$

$$\frac{\partial loss}{\partial w^{(L-1)}} = \frac{\partial loss}{\partial x^{(L)}} \odot f'\left(W^{(L-1)}x^{(L-1)}\right) \cdot \left(x^{(L-1)}\right)^T$$

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.



$\theta = \{w_1, w_2\}$

Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

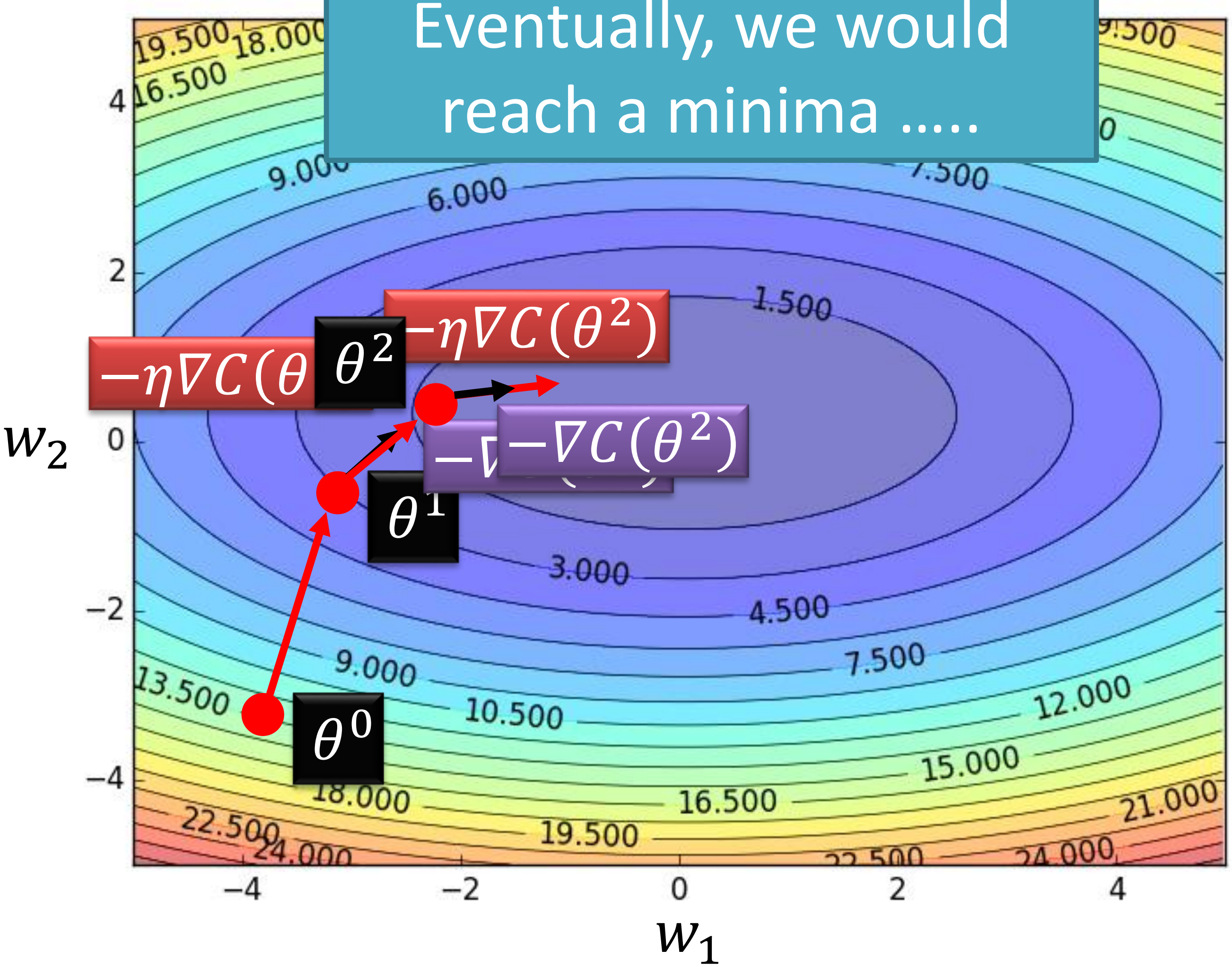
$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\Rightarrow -\eta \nabla C(\theta^0)$

Gradient Descent

Eventually, we would reach a minima …..



Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

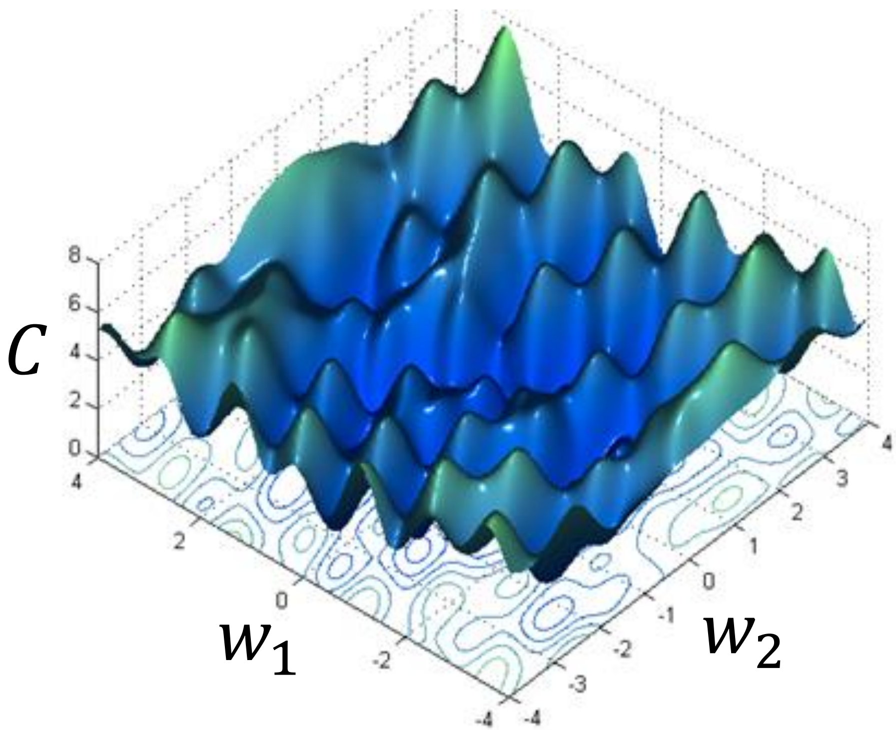
$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\Rightarrow -\eta\nabla C(\theta^0)$

Local Minima

- Gradient descent never guarantee global minima



Different initial point θ^0

Reach different minima, so different results

Who is Afraid of Non-Convex Loss Functions?

多层前馈网络表示能力

只需要一个包含足够多神经元的隐层, 多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数

[Hornik et al.，1989]

多层前馈网络局限

- 神经网络由于强大的表示能力，经常遭遇过拟合．表现为：训练误差持续降低，但测试误差却可能上升
- 如何设置隐层神经元的个数仍然是个未决问题．实际应用中通常使用“试错法”调整

缓解过拟合的策略

- 早停：在训练过程中，若训练误差降低，但验证误差升高，则停止训练
- 正则化：在误差目标函数中增加一项描述网络复杂程度的部分，例如连接权值与阈值的平方和

DEEP LEARNING

Training of Deep Neural Networks

Pre-training + Fine-tuning

- Pre-training(预训练):监督逐层训练是多隐层网络训练的有效手段, 每次训练一层隐层结点, 训练时将上一层隐层结点的输出作为输入, 而本层隐结点的输出作为下一层隐结点的输入, 这称为"预训练".

- Fine-tuning(微调):在预训练全部完成后, 再对整个网络进行微调训练. 微调一般使用BP算法.

Comments:预训练+微调 的做法可以视为将大量参数分组, 对每组先找到局部看起来比较好的设置, 然后再基于这些局部较优的结果联合起来进行全局寻优.

Training of Deep Neural Networks

Parameter Sharing

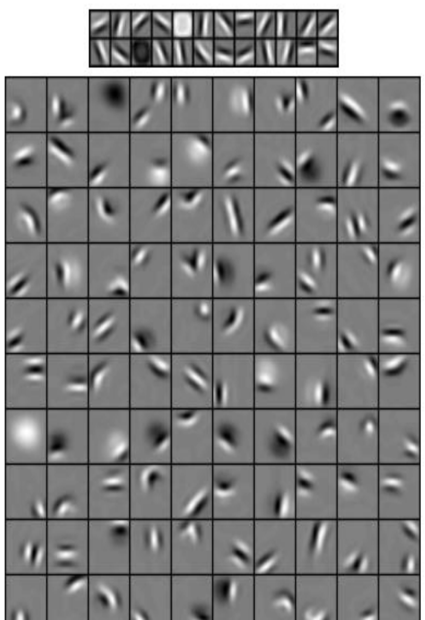
- Parameter shared by a group of neurons
- Convolutional Neural Networks (CNN)

CONVOLUTIONAL NEURAL NETWORKS

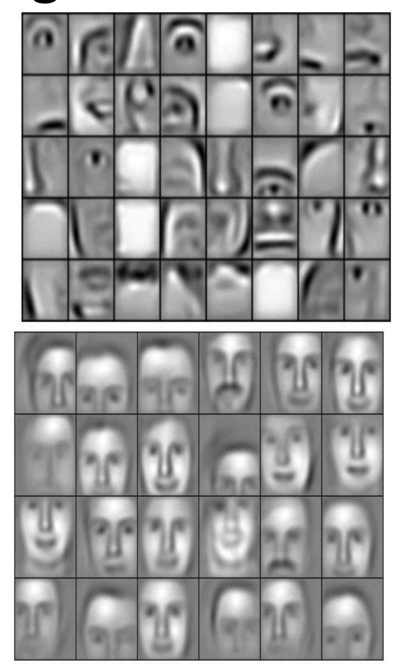
CNN

- CNN is a **hierarchical feature extractor**, which extract higher and higher level feature. Because the receptive fields of features get bigger and bigger, the feature change from local to global.

low-level feature

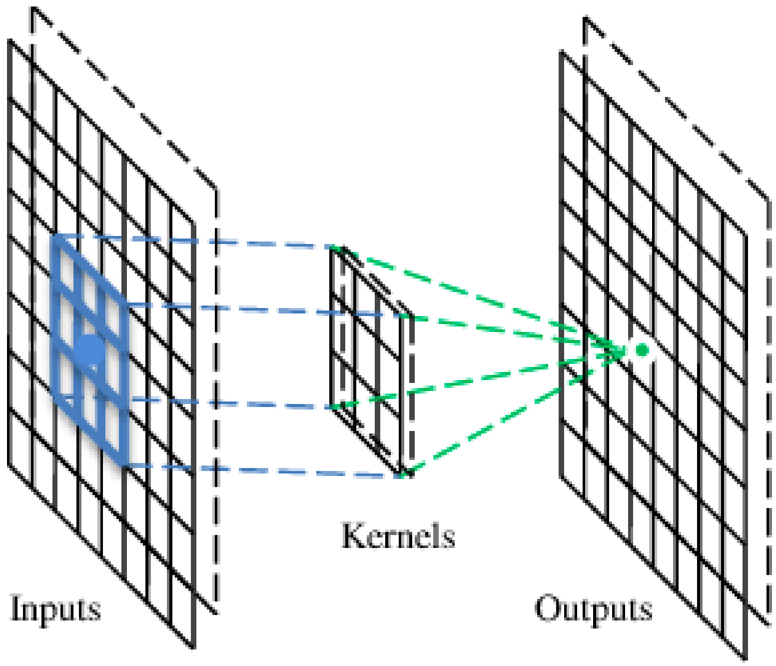


high-level feature



Convolution

- Convolution (weighted sum)
  - Input: data to be convolved
  - Filter: convolution kernel, also the weight
  - Feature map: the output after the data being convolved.



Convolution

- Convolution (weighted sum)
  - ➢ Input: data to be convolved
  - ➢ Filter: convolution kernel, also the weight
  - ➢ Feature map: the output after the data being convolved.

Example

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

*

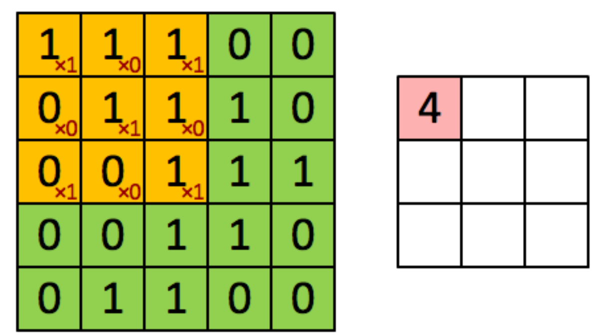
1	0	1
0	1	0
1	0	1

Filter

=

4	3	4
2	4	3
2	3	4

Feature Map



Image

Convolved Feature

Convolution

- Usually we use more than one filters to extract different features in a layer.

- Why we need more than one features in a layer?
  - If there is only one feature extracted, lots of information of the input data is lost.
  - High-level feature is combination of low-level features.

Convolution

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

These are the network parameters to be learned.

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Dot product

3 -1

Convolution

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

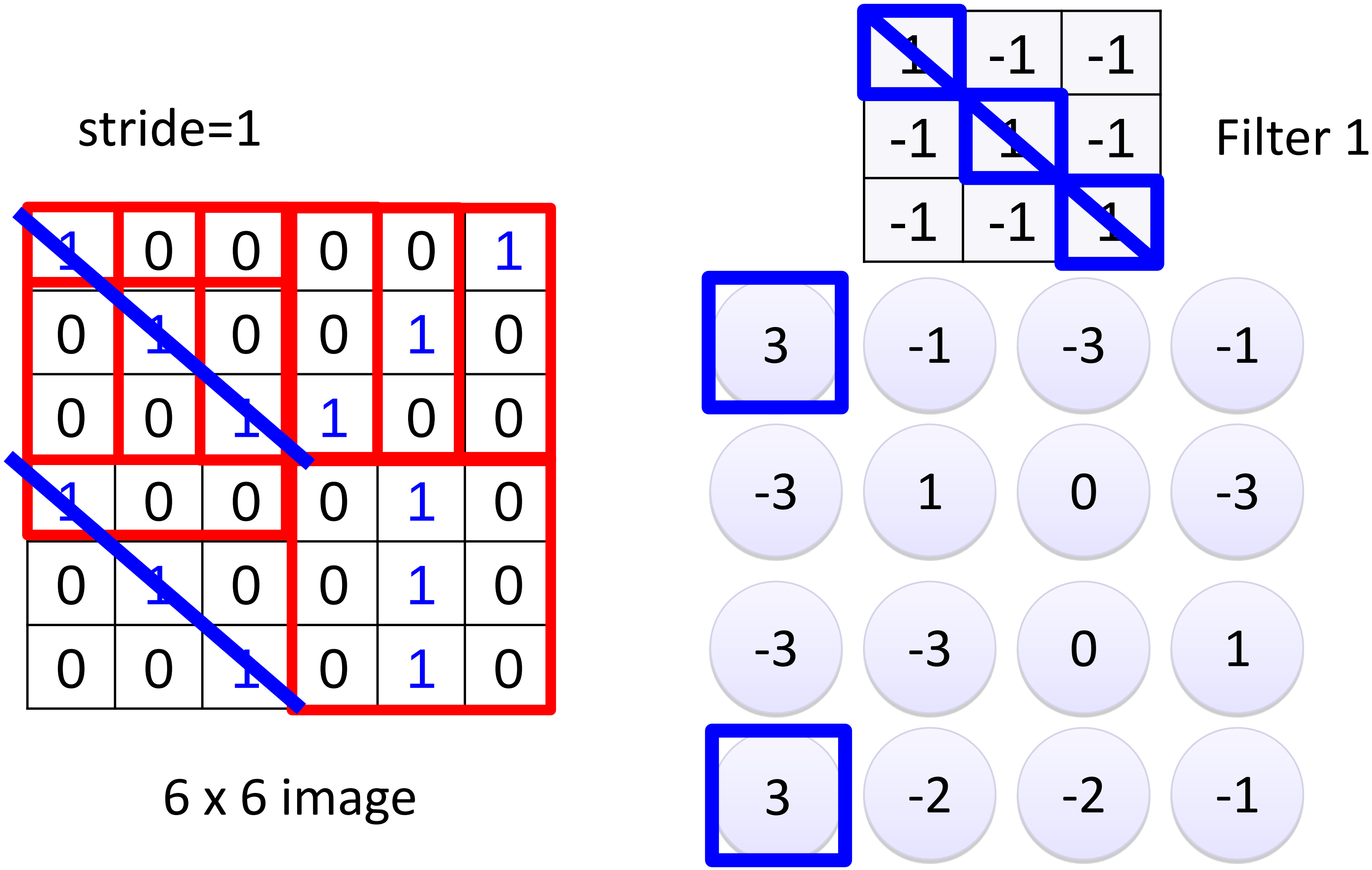
6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

3 -3

Convolution



Convolution

-1	1	-1
-1	1	-1
-1	1	-1

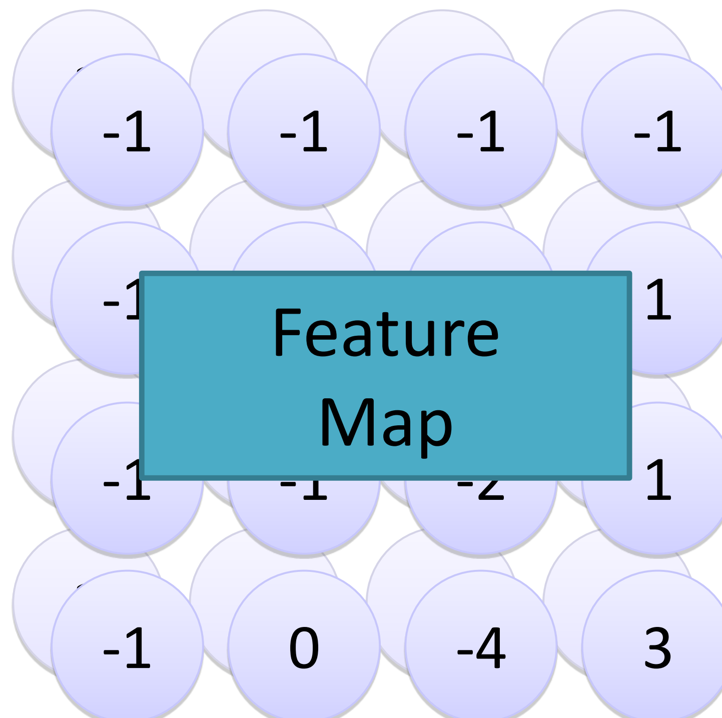
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

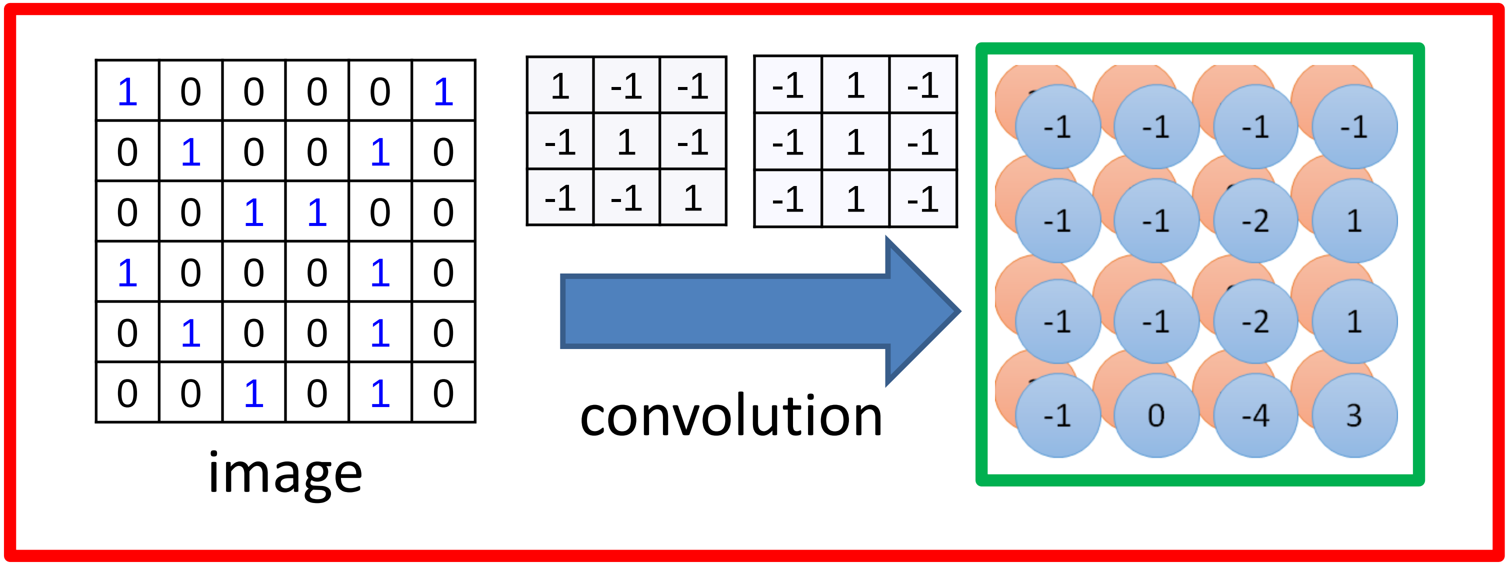
6 x 6 image

Repeat this for each filter

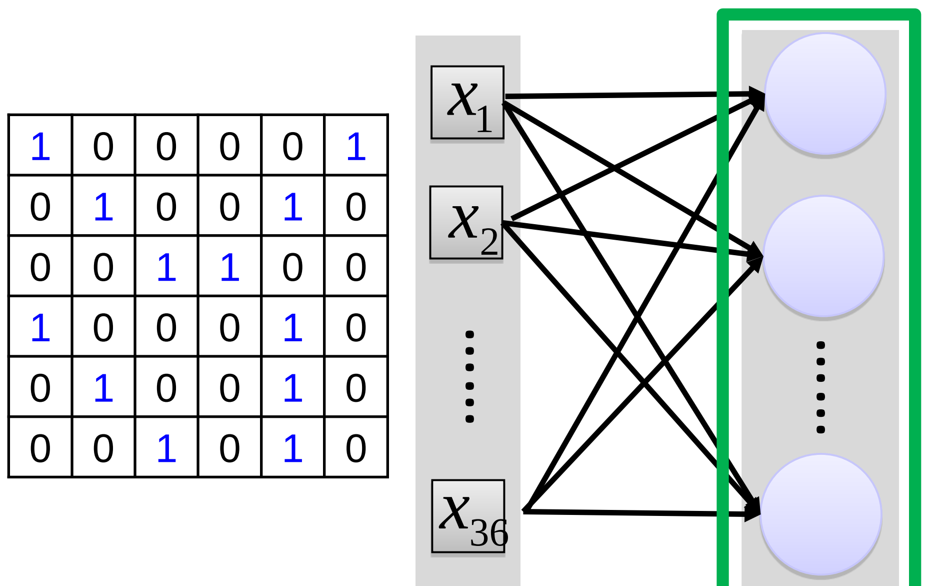


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

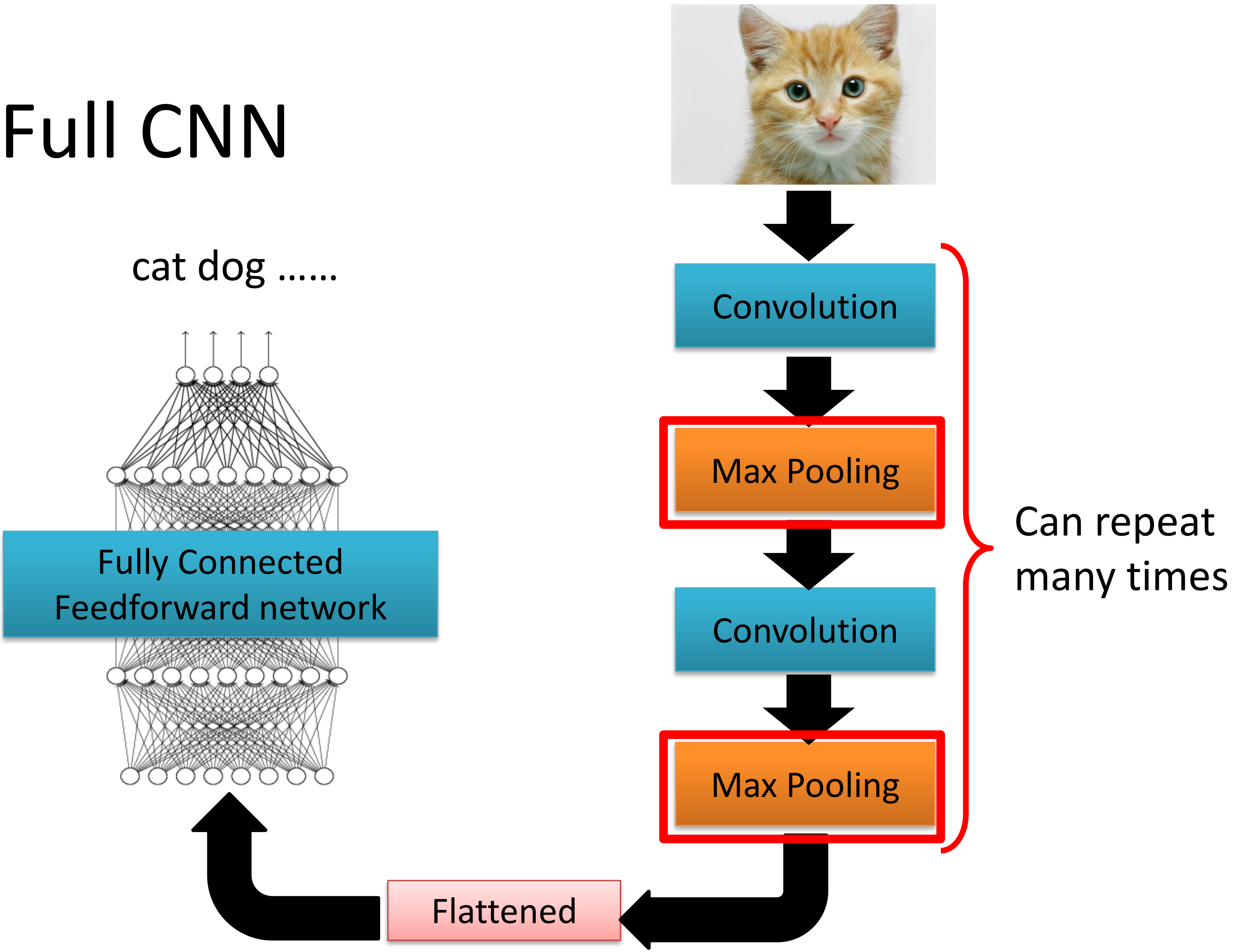
Convolution v.s. Fully Connected



Fully-connected



Full CNN


cat dog ......
Fully Connected
Feedforward network
Convolution
Max Pooling
Convolution
Max Pooling
Can repeat
many times
Flattened

Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

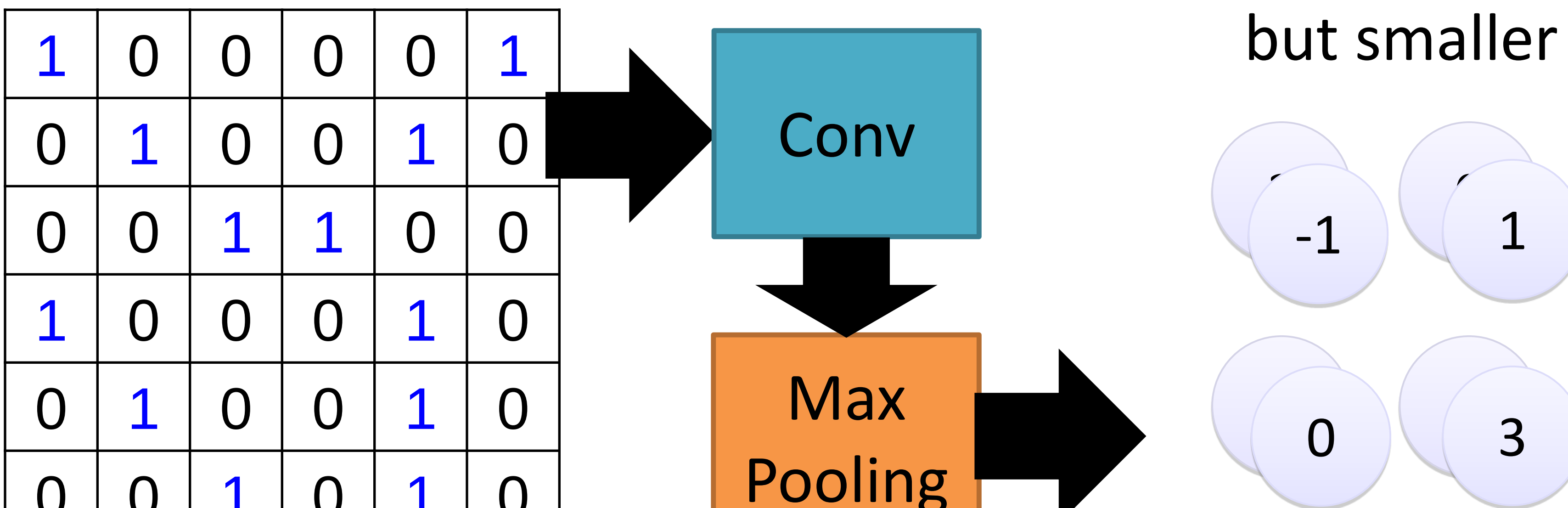
fewer parameters to characterize the image

A CNN compresses a fully connected network in two ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling

New image
but smaller

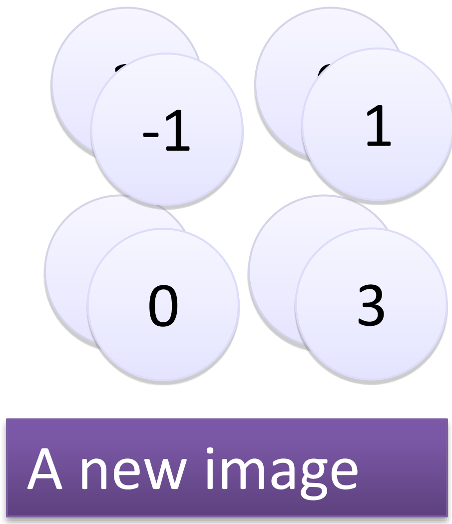


6 x 6 image

2 x 2 image

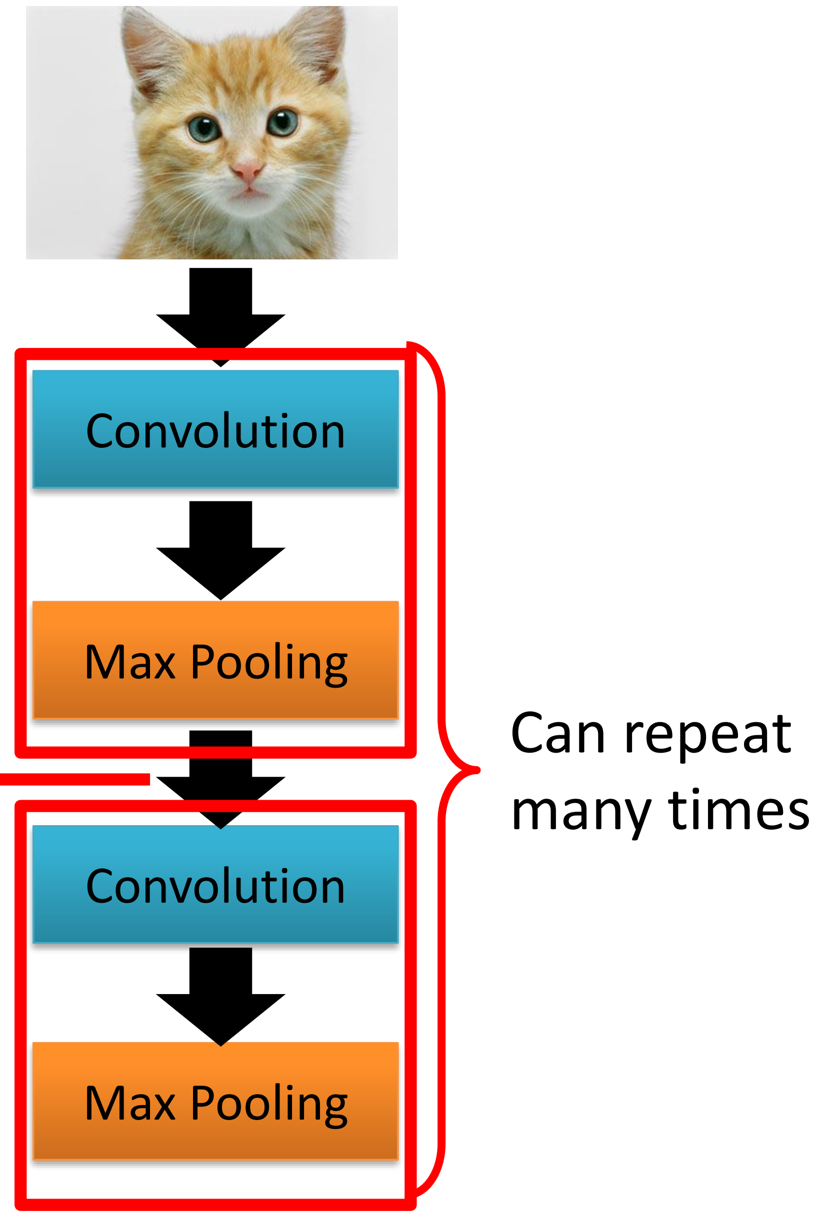
Each filter
is a channel

Full CNN

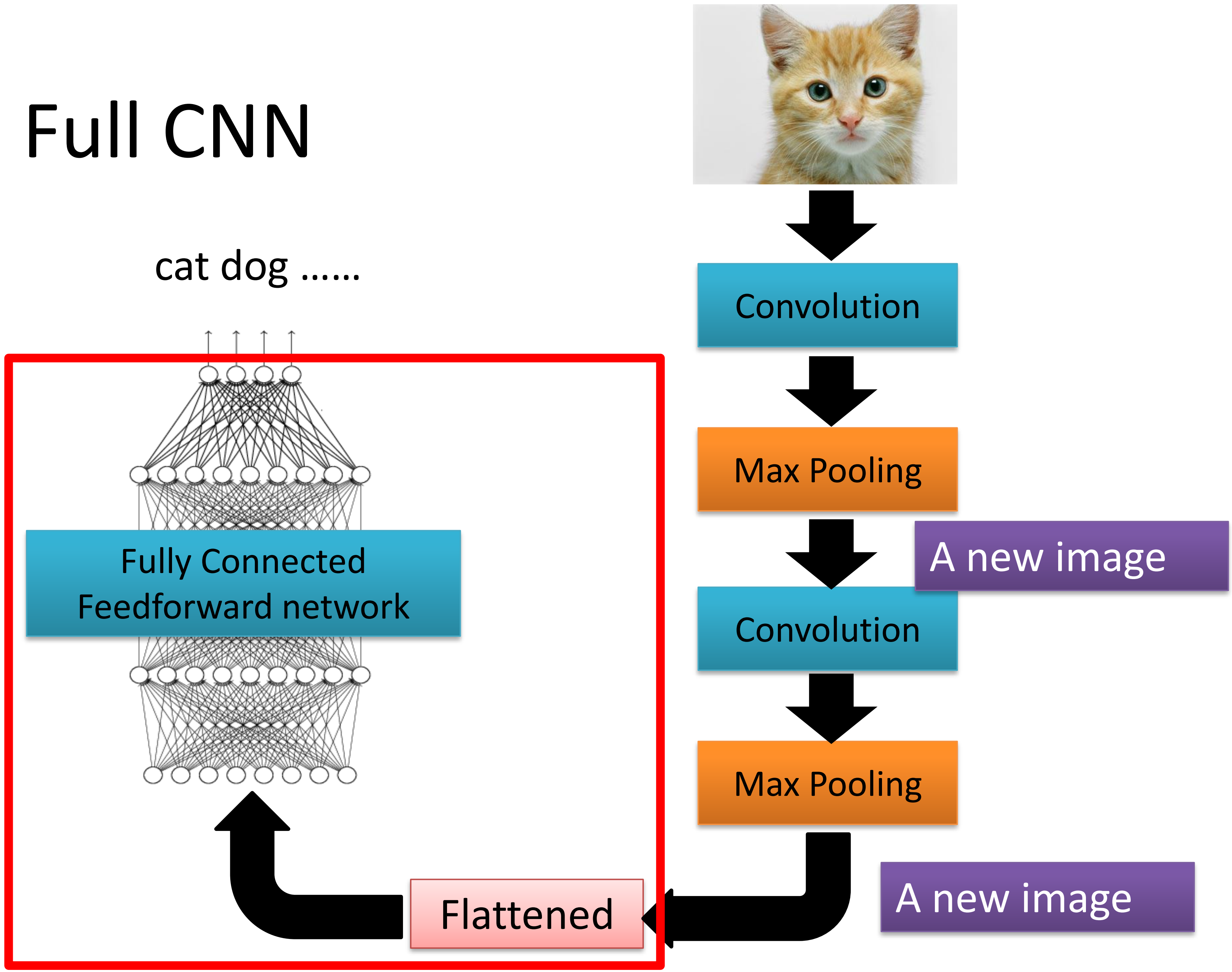


Smaller than the original image

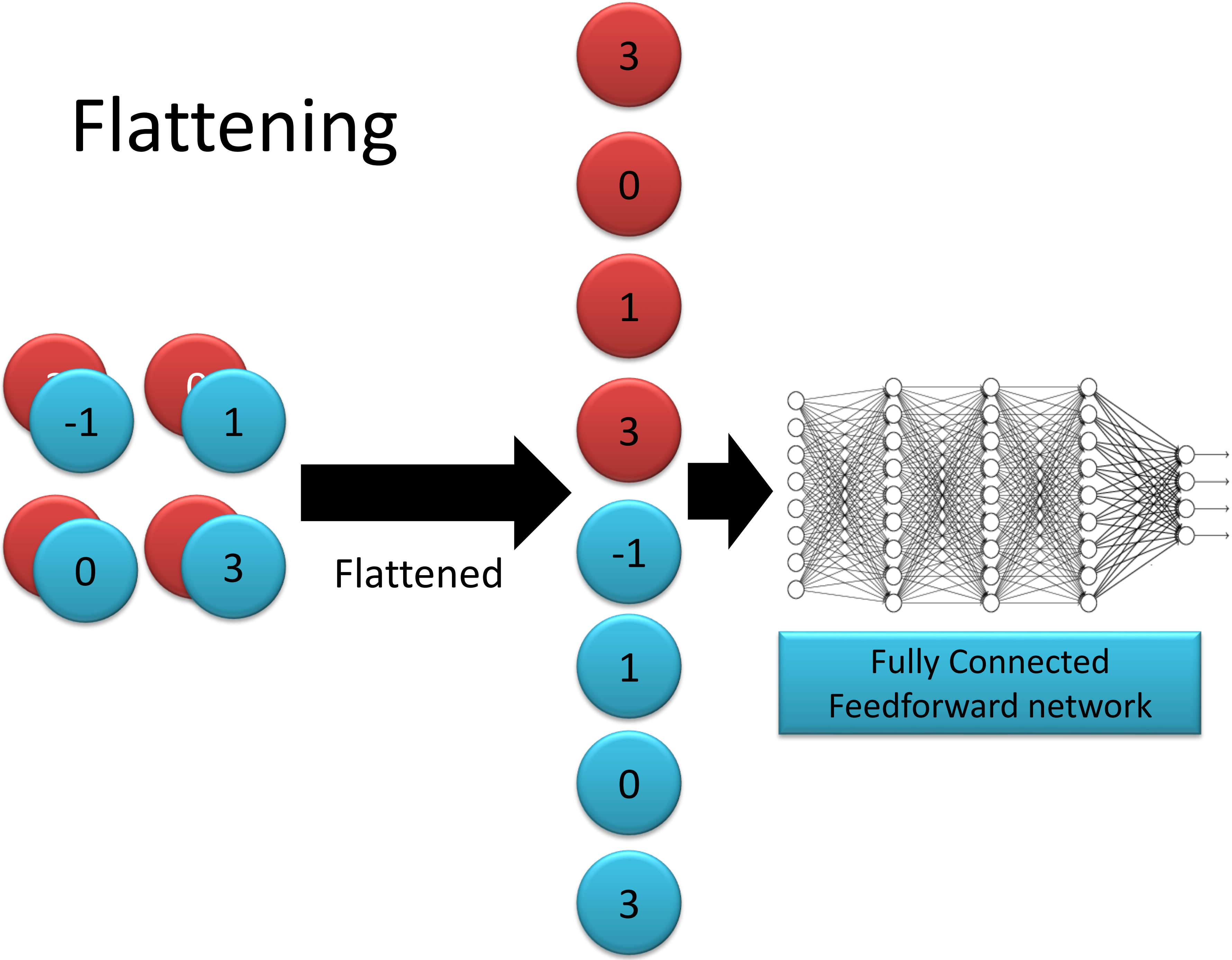
The number of channels is the number of filters



Full CNN

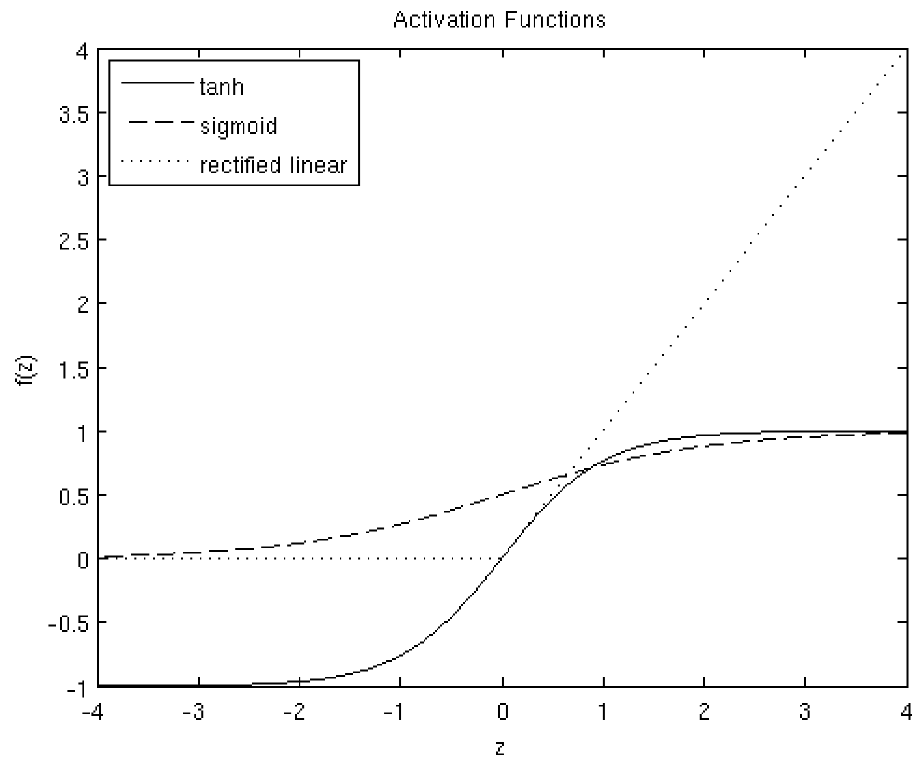

cat dog ......
Convolution
Max Pooling
A new image
Convolution
Max Pooling
A new image
Fully Connected
Feedforward network
Flattened

Flattening


-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

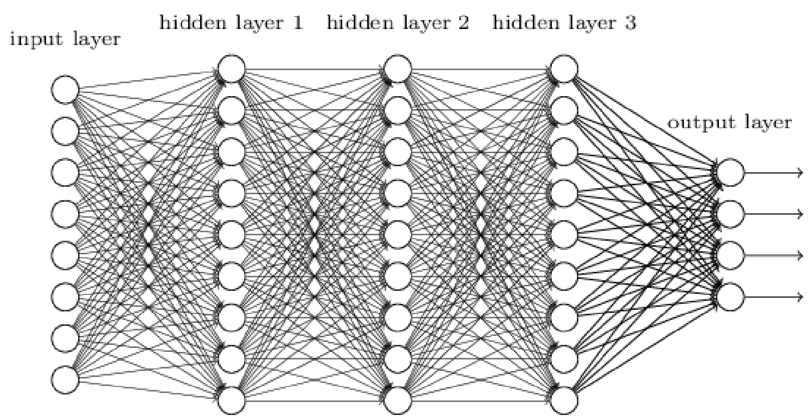
Activations

- Sigmoid
  - $f(x) = \frac{1}{1+\exp(-x)}$
- Tanh
  - $f(x) = \tanh(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- ReLU*
  - $f(x) = \max(0, x)$



Optimization

- When compute the gradient of parameters from output back to input, that is why it is called backpropagation.



- Since convolution is essentially weighted sum, BP for CNN is similar to BP for fully connected networks.

AlphaGo

19 x 19 matrix

Neural Network

Next move

(19 x 19 positions)

Black: 1

white: -1

none: 0

Fully-connected feedforward network can be used

But CNN performs much better

AlphaGo's policy network

The following is quotation from their Nature article:

Note: AlphaGo does not use Max Pooling.

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ 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 match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

Some Classical Architectures

- LeNet-5(1998)

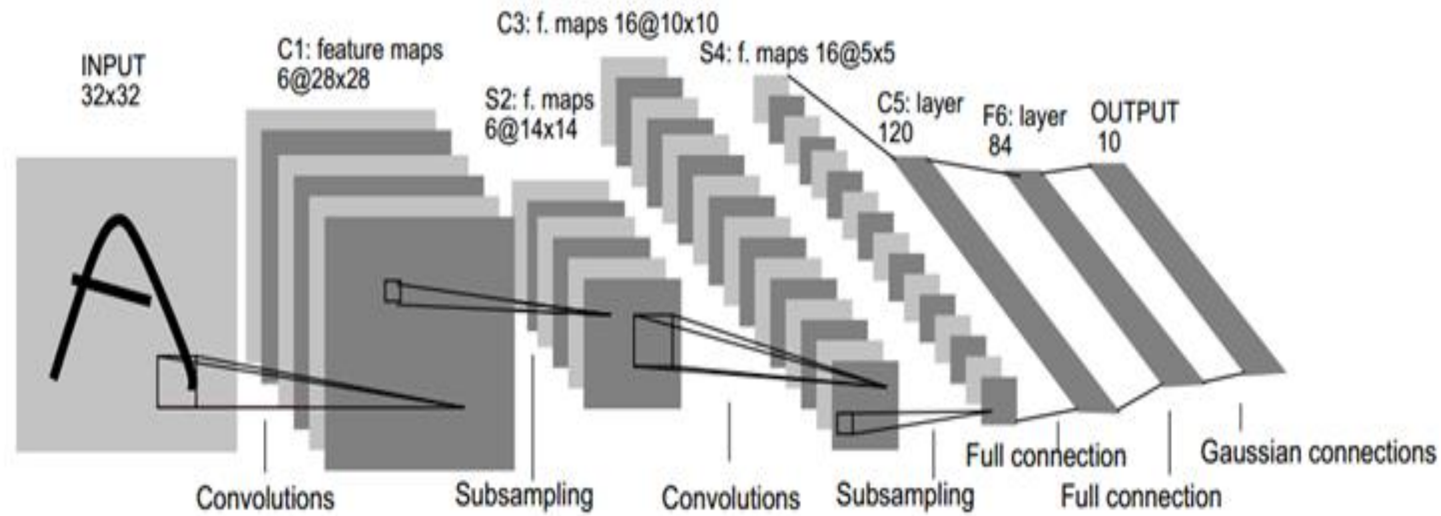
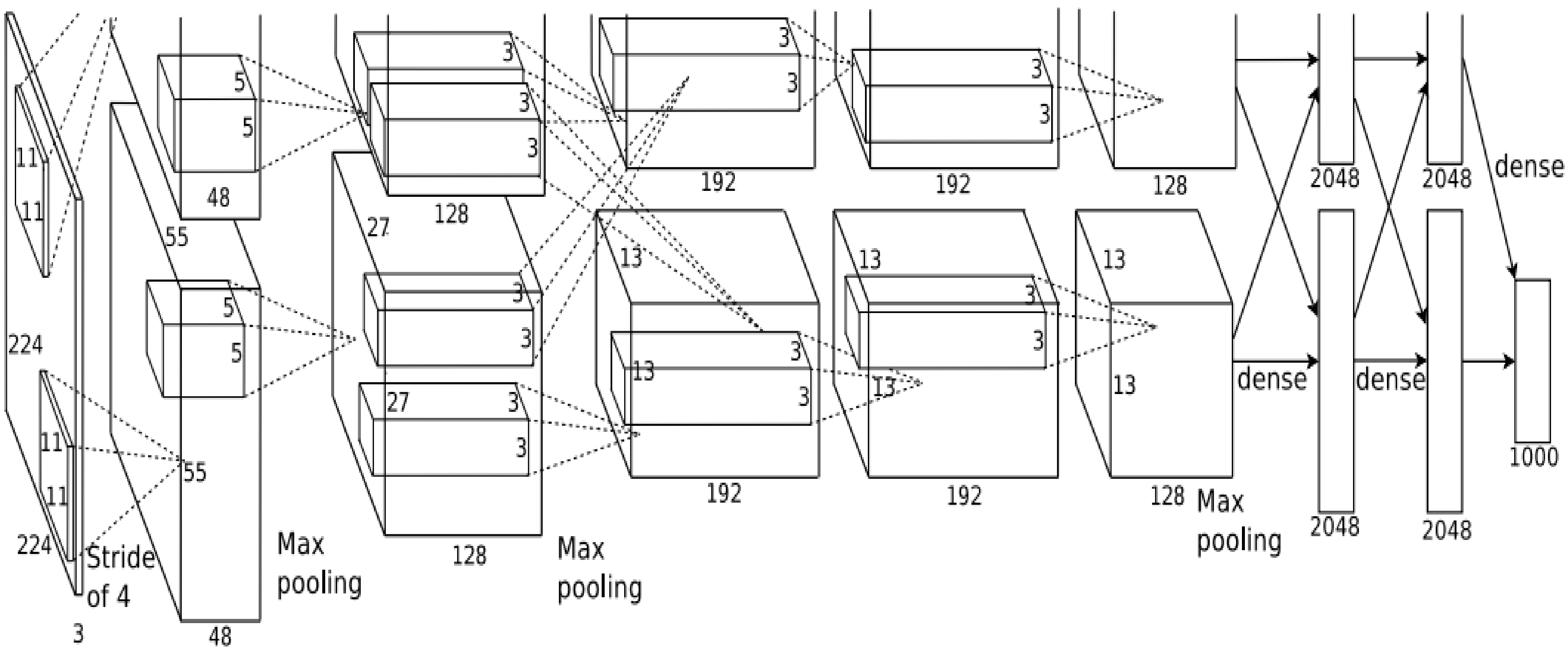
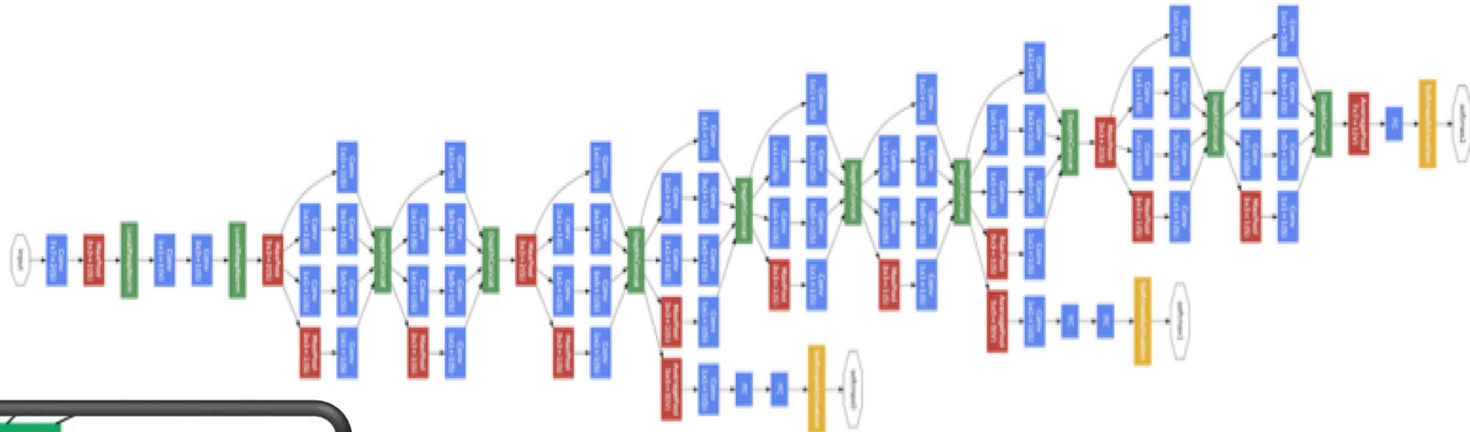


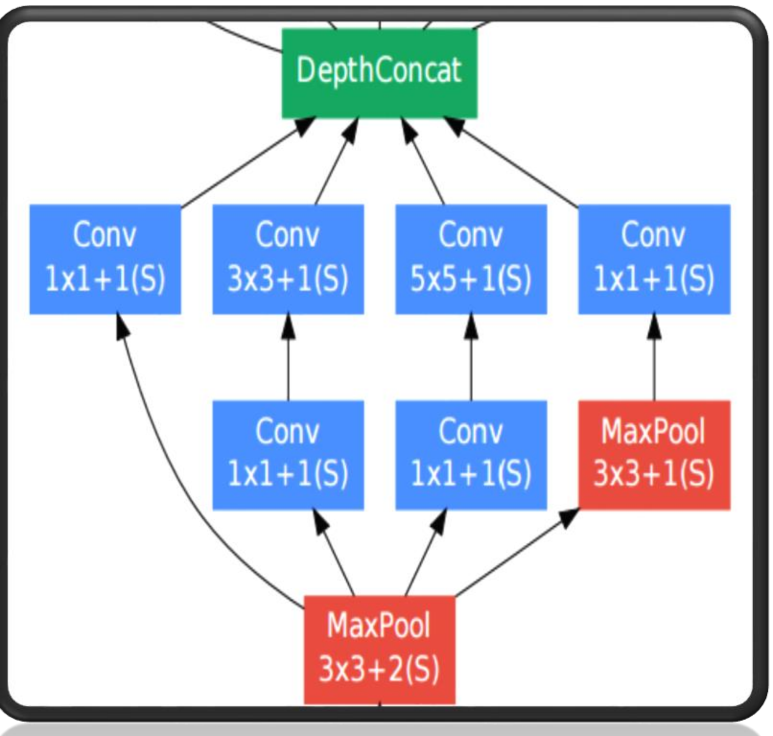
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

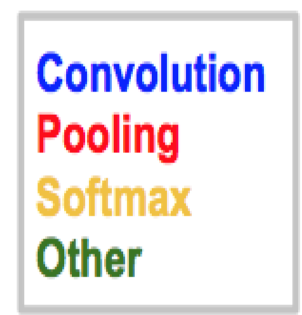
AlexNet(2012)



GoogleNet(2014)







VGG Net(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 imag					
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					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Remarks on CNN

- CNNs are hierarchical feature extractors, higher-layer feature is combination of lower-layer feature.
- Convolution is weighted sum across all channels of input
- The most commonly used activation for CNN is ReLU

Remarks on CNN

- The most commonly used pooling strategy for CNN is max-pooling
- The training strategy is BP
- Finding a patch leading to maximum response in the validation set is a very simple way to visualize features.
- LeNet-5, AlexNet, GoogleNet, VGG-Net, ResNet, BN

Comments on Deep Learning

“特征工程” VS “特征学习”或者 “表示学习”

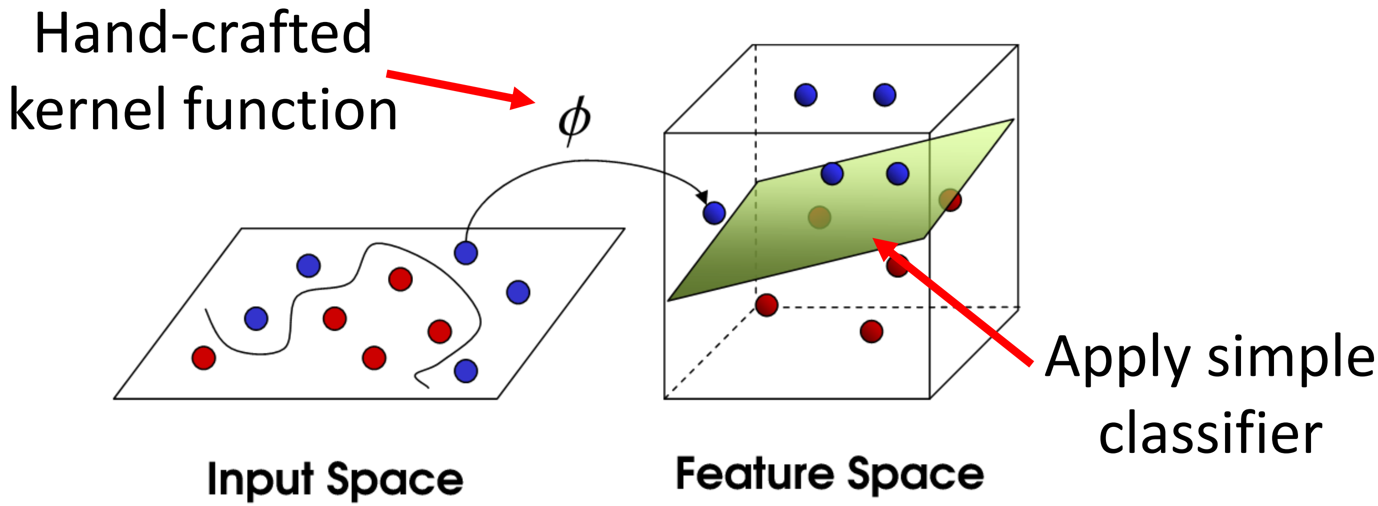
- 特征工程由人类专家根据现实任务来设计，特征提取与识别是单独的两个阶段；

手工设计特征 → 分类识别

- 特征学习通过深度学习技术自动产生有益于分类的特征，是一个端到端的学习框架.

学习生成特征 → 分类识别

SVM



Deep Learning