机器学习

黄雷
人工智能研究院
huangleiAI@buaa.edu.cn

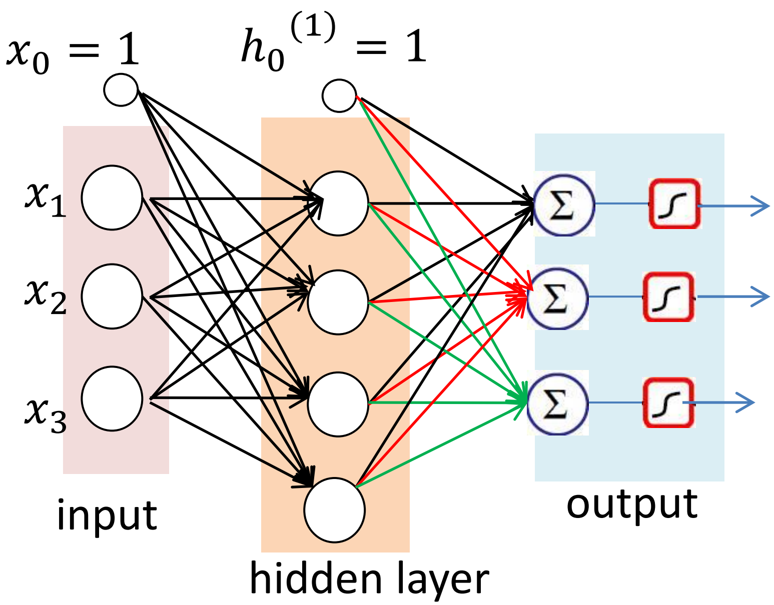
2021年10月25日

主要内容

- 多层感知机(Multi-layer Perceptron, MLP)
  – 反向传播算法(Back-propagation)
- 卷积神经网络 （Convolutional Neural Network, CNN）
  – 卷积操作和卷积层
  – 池化 （Pooling）
- 循环神经网络（Recurrent Neural Network, RNN）
  – 建模和训练
  – LSTM模型

Multi-layer perceptron (多层感知机)

- **Multi-layer perceptron or feed-forward neural network**



x_i: 第i 个输入节点

$h_i^{(k)}$: 第 k 层隐藏层的第i个节点

$w_{ij}^{(k)}$:第 k 层隐藏层, 第 i 个输出神经元，连接第j 个输入神经元

y_i: 第i 个输出节点

预激活值：（Pre-activation）

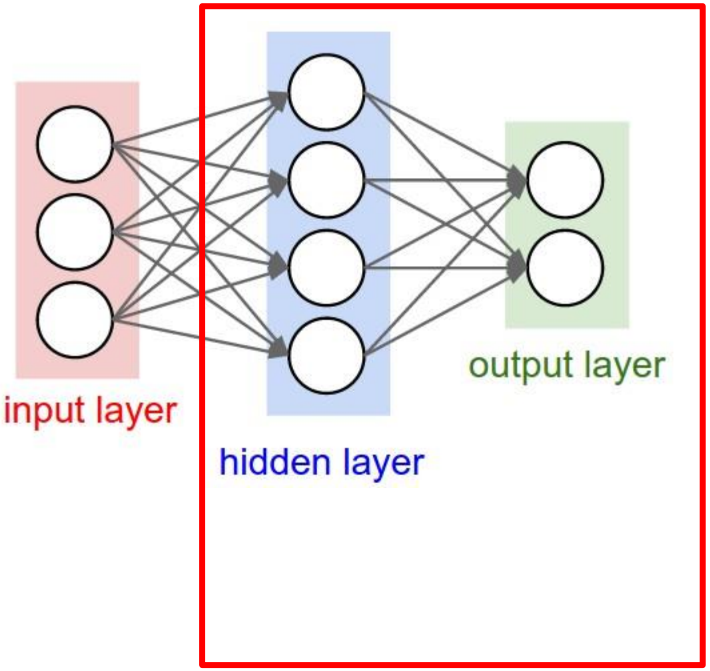
激活值（Activation）

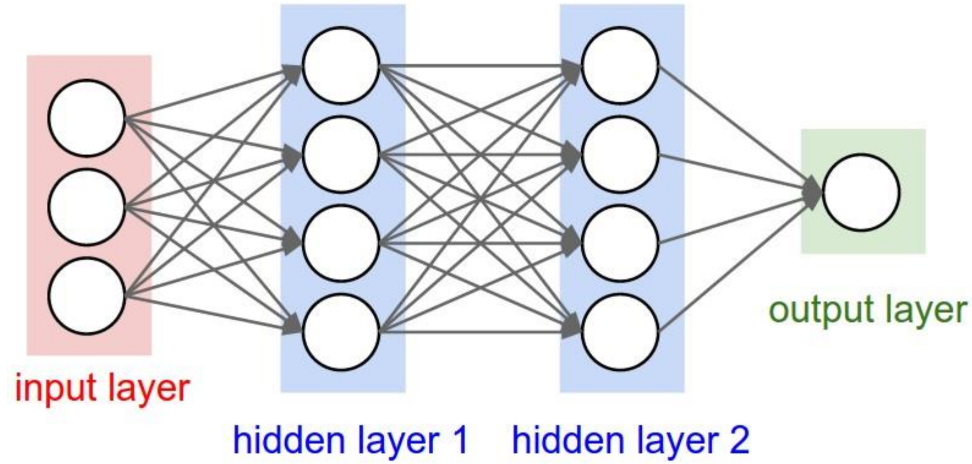
$$\boldsymbol{a}^{(1)} = \boldsymbol{W}^{(1)} \cdot \boldsymbol{x}$$
$$\boldsymbol{h}^{(1)} = \sigma\left(\boldsymbol{a}^{(1)}\right)$$

$$\boldsymbol{a}^{(2)} = \boldsymbol{W}^{(2)} \cdot \boldsymbol{h}^{(1)}$$
$$\boldsymbol{y} = \sigma\left(\boldsymbol{a}^{(2)}\right)$$

$$\boldsymbol{a}^{(i)} = \boldsymbol{W}^{(i)} \cdot \boldsymbol{h}^{(i-1)}$$
$$\boldsymbol{h}^{(i)} = \sigma(\boldsymbol{a})$$
$$(\boldsymbol{h}^{(0)} = \boldsymbol{x}, \boldsymbol{h}^{(L)} = \boldsymbol{y})$$

Neural Networks: Architectures



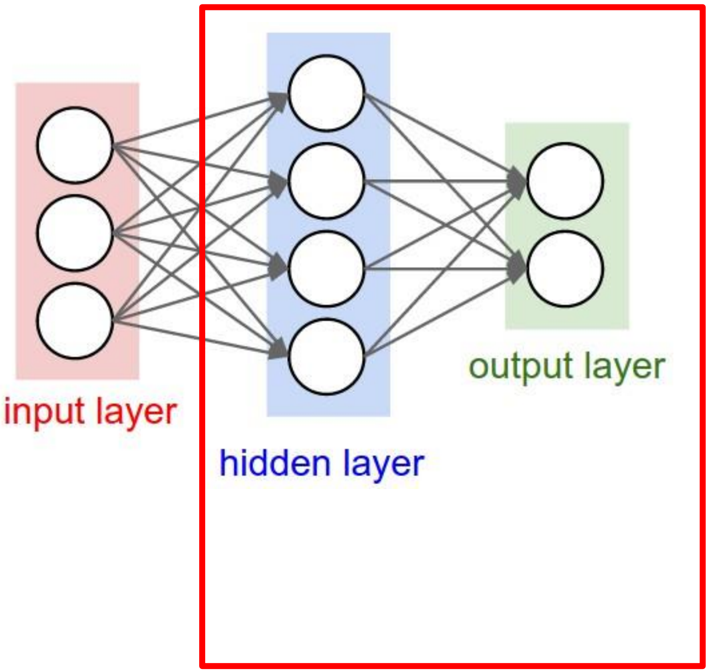


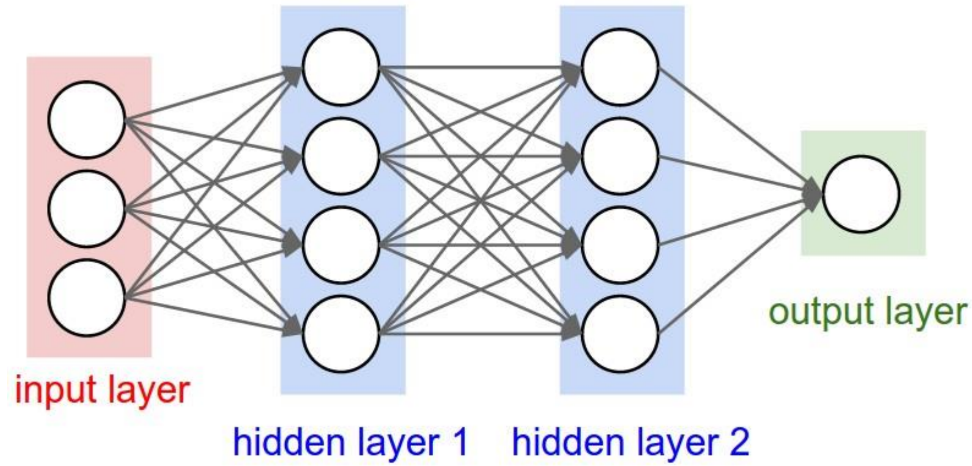
“2-layer Neural Net”, or
“1-hidden-layer Neural Net”

“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

Source: Andrej Karpathy & Fei-Fei Li

Neural Networks: Architectures

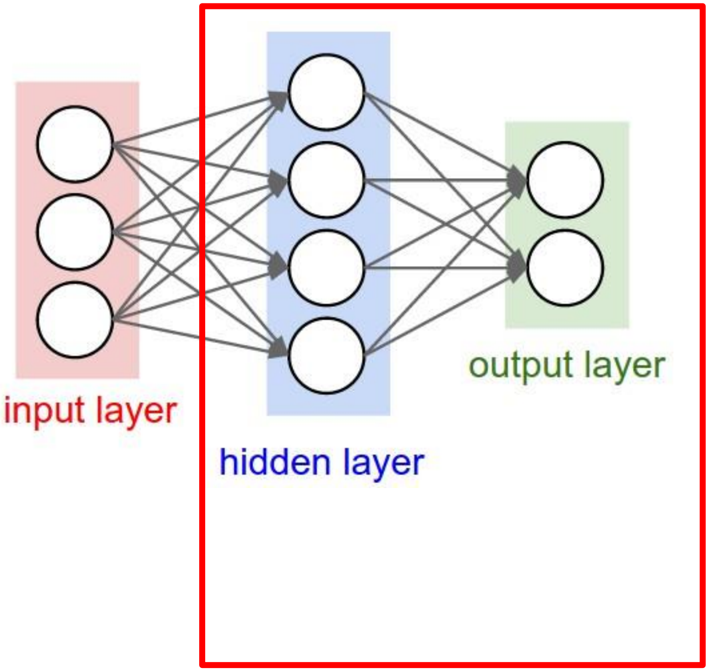


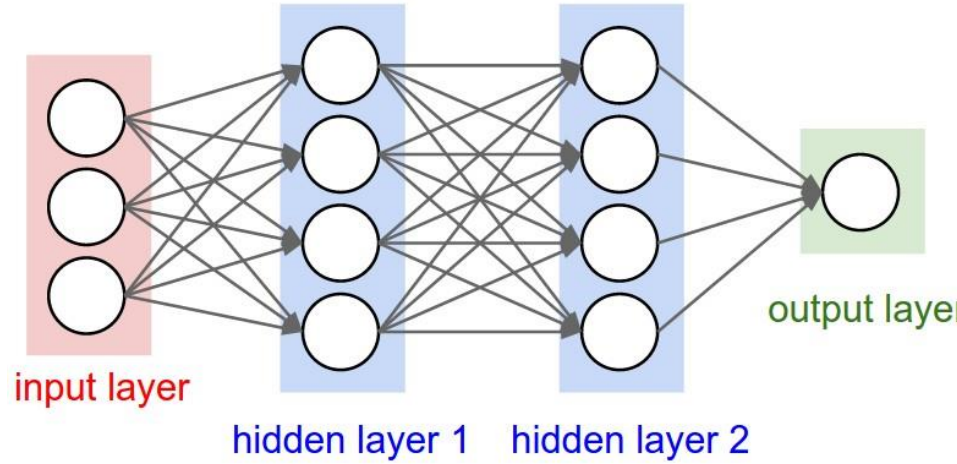


Number of Neurons: ?
Number of Weights: ?
Number of Parameters: ?

Source: Andrej Karpathy & Fei-Fei Li

Neural Networks: Architectures



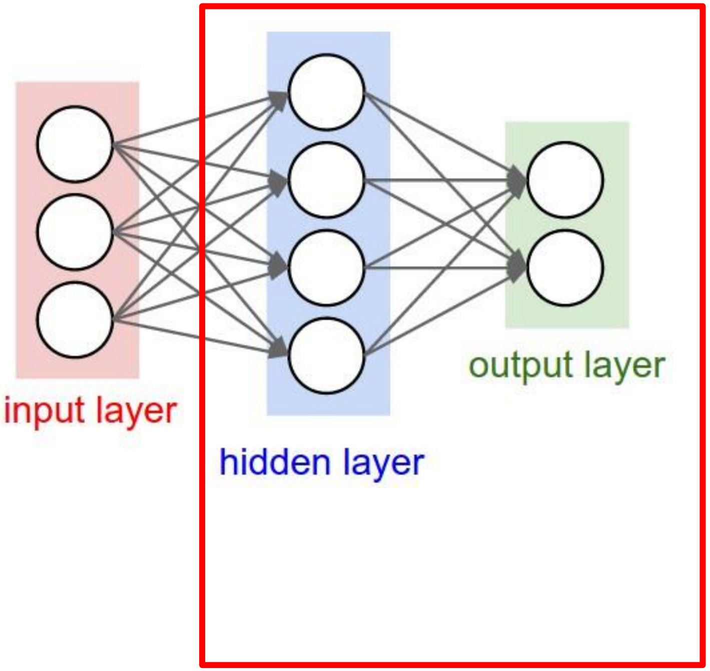


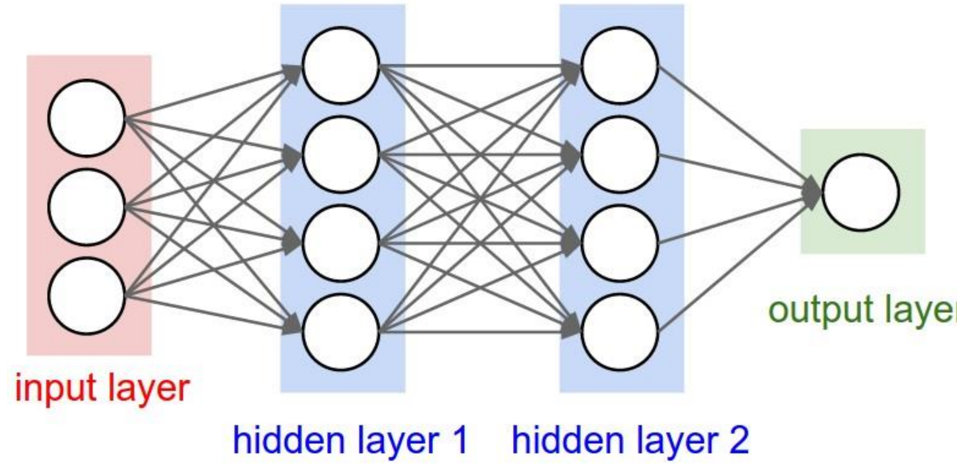
Number of Neurons: 4+2 = 6
Number of Weights: [4x3 + 2x4] = 20
Number of Parameters: 20 + 6 = 26 (biases!)

Number of Neurons: ?
Number of Weights: ?
Number of Parameters: ?

Source: Andrej Karpathy & Fei-Fei Li

Neural Networks: Architectures



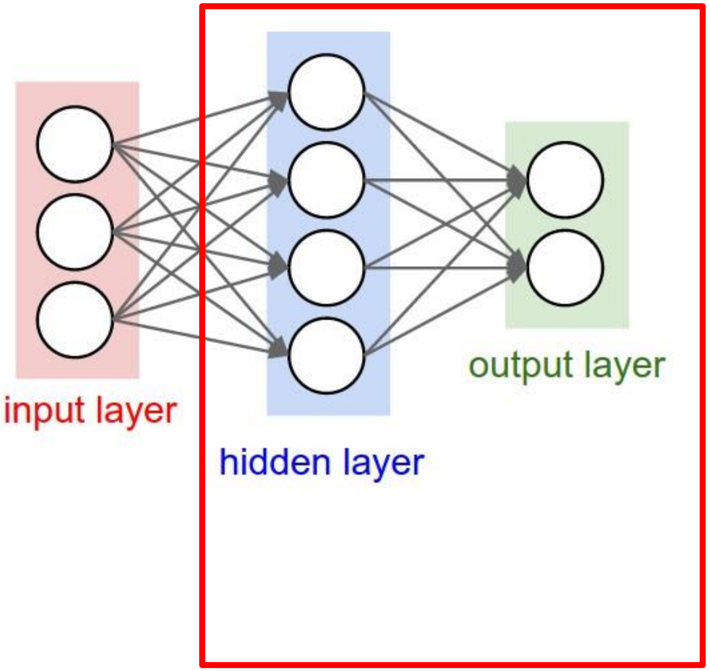


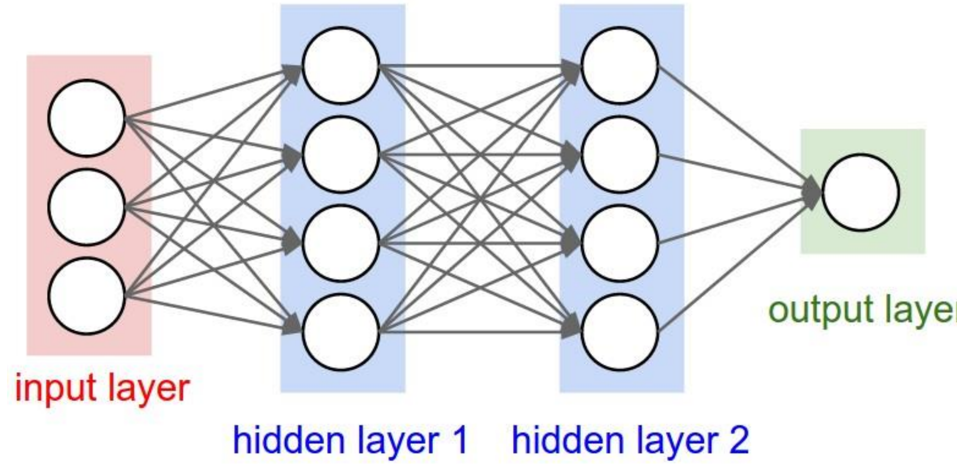
Number of Neurons: 4+2 = 6
Number of Weights: [4x3 + 2x4] = 20
Number of Parameters: 20 + 6 = 26 (biases!)

Number of Neurons: 4 + 4 + 1 = 9
Number of Weights: [4x3+4x4+1x4]=32
Number of Parameters: 32+9 = 41

Source: Andrej Karpathy & Fei-Fei Li

Neural Networks: Architectures





Modern CNNs: ~10 million neurons

Human visual cortex: ~5 billion neurons

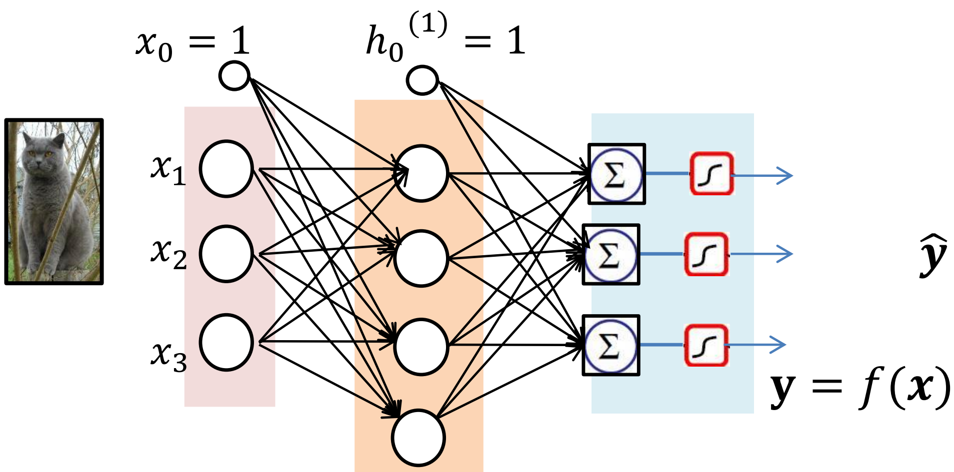
Source: Andrej Karpathy & Fei-Fei Li

outline

- Multi-layer perceptron (多层感知机)
  - Model representation (模型表示)
  - Loss function: the goal for learning
  - Training
    - Gradient based optimization
    - Back-propagation

Target of learning: Loss function

- 损失函数（Loss function）



Cat	dog	car
$(1, 0, 0)^T$	$(0, 1, 0)^T$	$(0, 0, 1)^T$

$$L = (\mathbf{y} - \hat{\boldsymbol{y}})^2$$
$$= (1 - y_1)^2 + y_2{}^2 + y_3{}^2$$

均方误差（Mean Squared Error）： L=$(\mathbf{y} - \hat{\boldsymbol{y}})^2$, 其中 $\mathbf{y} = f(\boldsymbol{x})$

- 优化目标函数: min L=$(\mathbf{y} - \hat{\boldsymbol{y}})^2$

outline

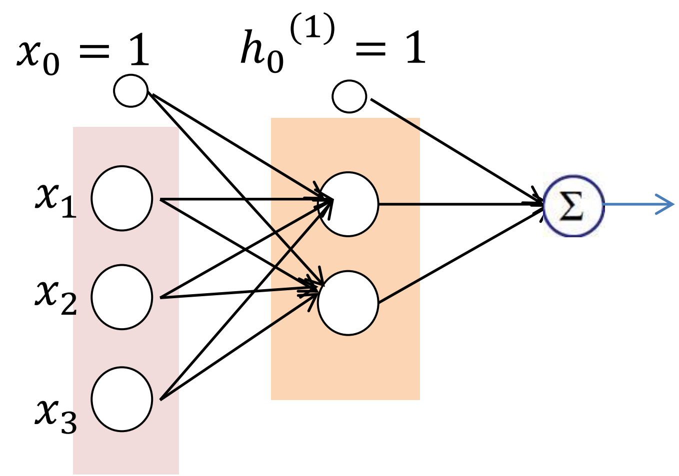
- Multi-layer perceptron (多层感知机)
  - Model representation (模型表示)
  - Loss function: the goal for learning
  - Training
    - Gradient based optimization
    - Back-propagation

目标函数

- 求解目标函数

$$\min_{\mathrm{W}} E_{(X,Y)\sim D}\ \mathrm{L}(\mathrm{F}(\mathrm{X};\mathrm{W}),\hat{Y}))$$

- 方案一：令 $\frac{\partial \mathrm{L}}{\partial W} = 0$，求解方程组



$$\mathrm{L}= \left(\frac{w_{21}^{(2)}}{1+\mathrm{e}^{-(x_1 w_{11}^{(1)}+x_2 w_{12}^{(1)}+x_3 w_{13}^{(1)}+w_{10}^{(1)})}} + \frac{w_{22}^{(2)}}{1+\mathrm{e}^{-(x_1 w_{11}^{(1)}+x_2 w_{12}^{(1)}+x_3 w_{13}^{(1)}+w_{10}^{(1)})}} + w_{20}^{(2)} - \hat{y}\right)^2$$

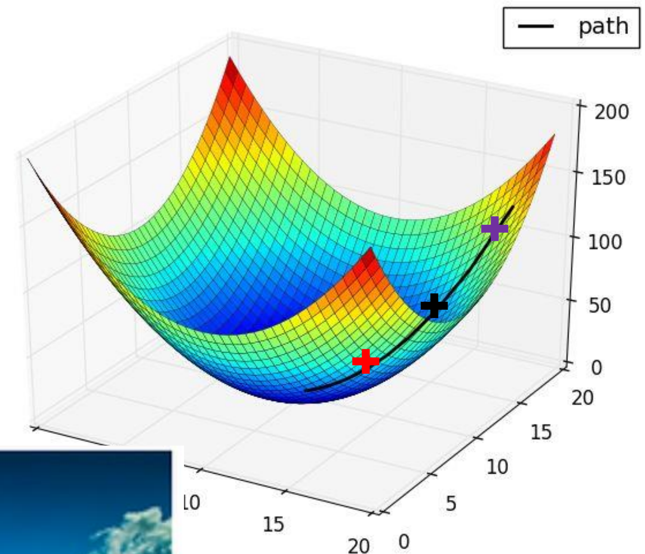
目标函数

- 求解目标函数

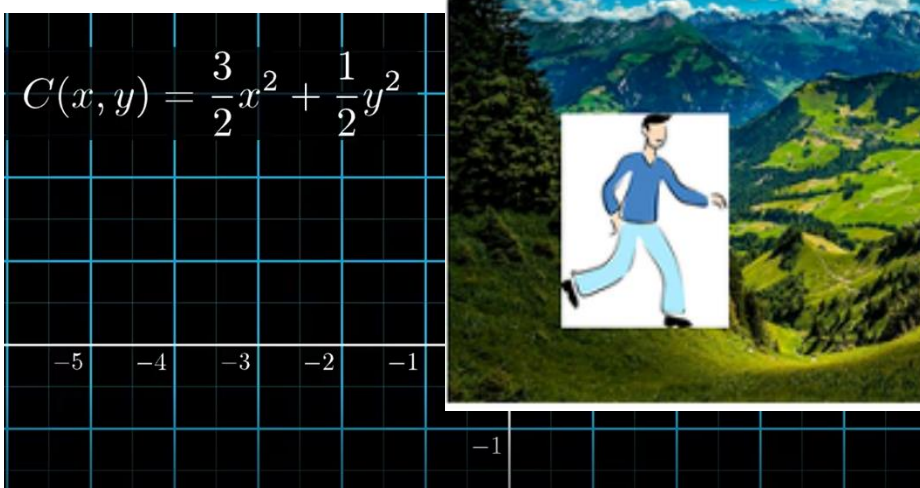
$$\min_{W} E_{(X,Y)\sim D} \ L(F(X;W),\hat{Y}))$$

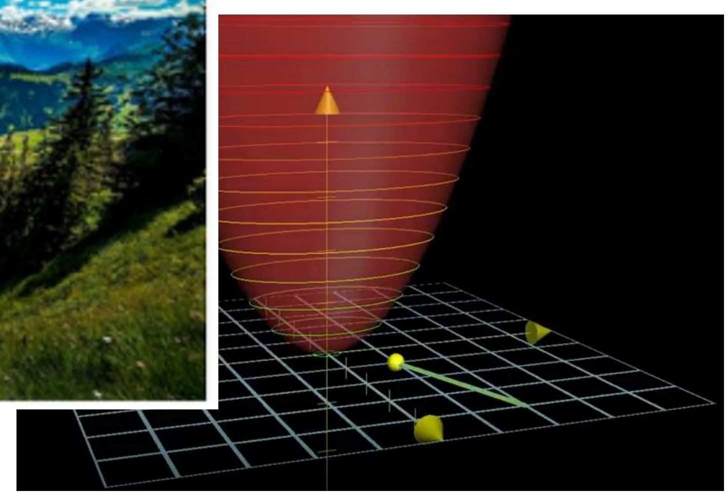
基于迭代的训练方式

$$\min_{\mathbf{W}} E_{(X,Y)\sim D}\ \mathrm{L}(\mathrm{F}(\mathrm{X};\mathbf{W}),\hat{Y}))$$



- 局部下降搜索
  - 基于目前的参数W^{t}，给其多个扰动ΔW，确保存在某个ΔW，使得$L(W^t+\Delta W)<L(W^t)$,
  - 更新 $W^{t+1}=W^t+\Delta W$
- 更高效的下降搜
  - 基于梯度的下降

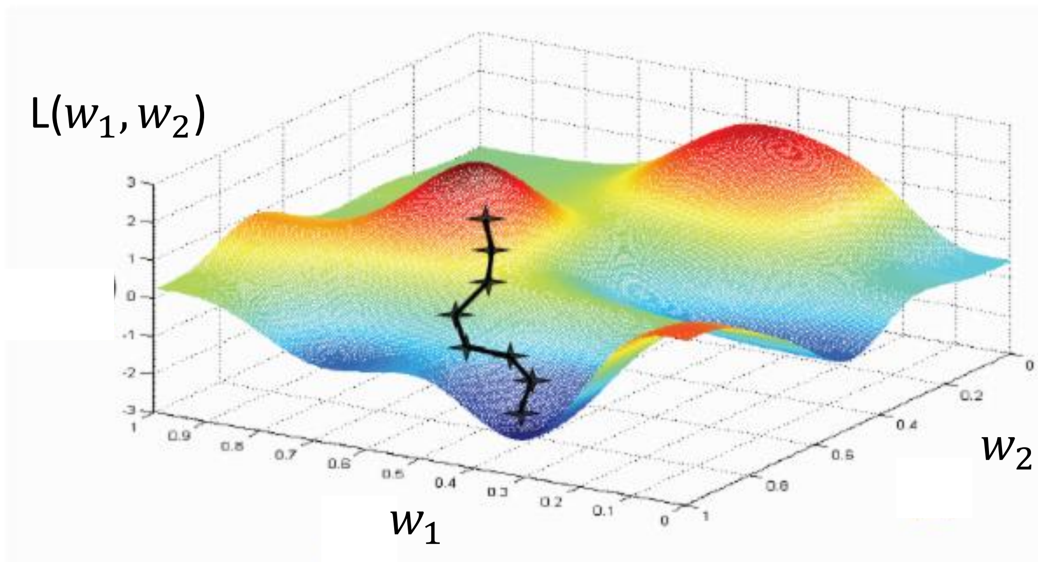




梯度下降算法

- 0.初始化权重 $\boldsymbol{W}^{(0)}$
- 1. 前向过程：
  - 1.1根据输入，计算输出值 $\boldsymbol{y}$
  - 1.2.计算损失函数值L($\mathbf{y}, \widehat{\boldsymbol{y}}$)。
- 2.计算梯度$\frac{dL}{\boldsymbol{dW}}$
- 3.更新梯度

$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta \frac{d\,\mathrm{L}}{\boldsymbol{d\,W}^{(t)}}$$



gradient：$\left(\frac{\mathrm{dL}(w_1,w_2)}{w_1}, \frac{\mathrm{dL}(w_1,w_2)}{w_2}\right)$

outline

- Multi-layer perceptron (多层感知机)
  - Model representation (模型表示)
  - Loss function: the goal for learning
  - Training
    - Gradient based optimization
    - Back-propagation

计算梯度：反向传播

- 求导基础知识回顾

  ➢实值函数对一维实值变量的导数:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

  ➢实值函数对多维向量变量的梯度为向量（偏导数）：

$$\frac{\partial \mathrm{f}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \left(\frac{\partial f(\theta_0, \theta_1)}{\partial \theta_0}, \frac{\partial \mathrm{f}(\theta_0, \theta_1)}{\partial \theta_1}\right), \quad \boldsymbol{\theta} = (\theta_0, \theta_1)$$

计算梯度：反向传播

- 神经网络中的基本操作

加法： $f(x,y)=x+y$ $\frac{\partial f}{\partial \boldsymbol{x}}=1$ $\frac{\partial f}{\partial \boldsymbol{y}}=1$

乘法： $f(x,y)=xy$ $\frac{\partial f}{\partial \boldsymbol{x}}=y$ $\frac{\partial f}{\partial \boldsymbol{y}}=x$

非线性变换： $\sigma(x)=\frac{1}{1+e^{-x}}$

$$\frac{d\sigma(x)}{dx}=\frac{e^{-x}}{(1+e^{-x})^2}=\left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)=(1-\sigma(x))\sigma(x)$$

反向传播（Back-Propagation）

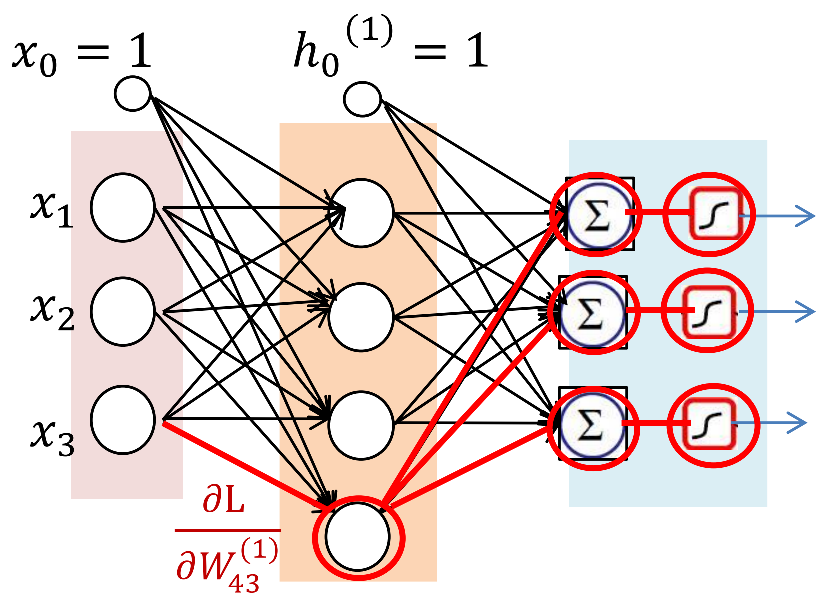
- 链式法则（Chain rule）

$f=q(x)$ $\qquad \dfrac{\partial f}{\partial x} = \dfrac{\partial f}{\partial q}\dfrac{\partial q}{\partial x}$

复合表达式: $f(x,y,z) = (x+y)z$

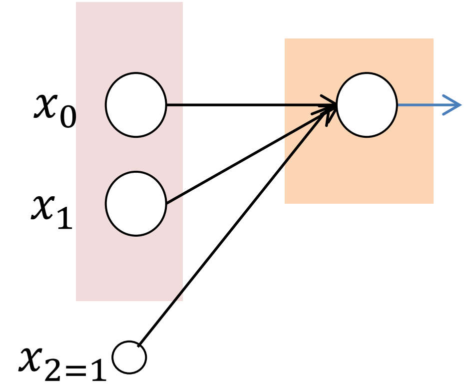
$q=x+y \qquad \dfrac{\partial q}{\partial x}=1 \qquad \dfrac{\partial q}{\partial y}=1$

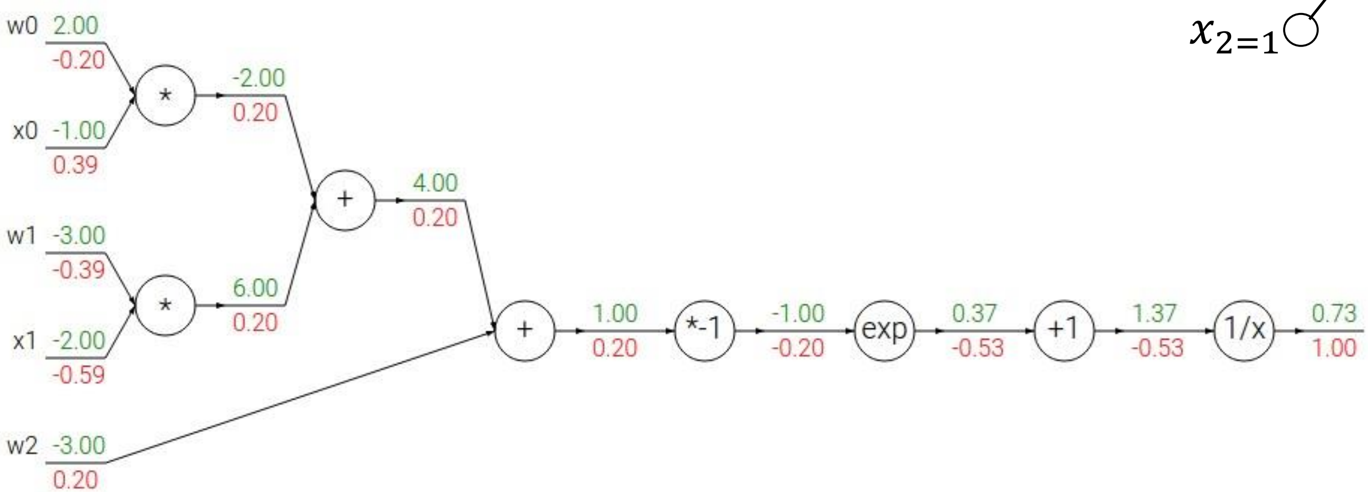
$f=qz \qquad \dfrac{\partial f}{\partial q}=z \qquad \dfrac{\partial f}{\partial z}=q$



One example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$





$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

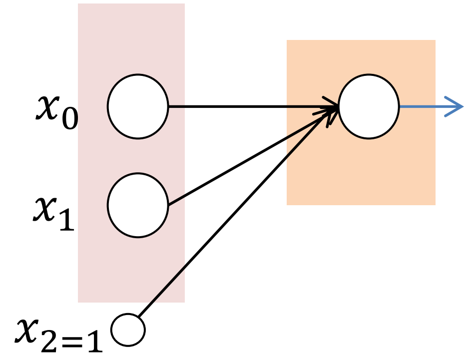
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

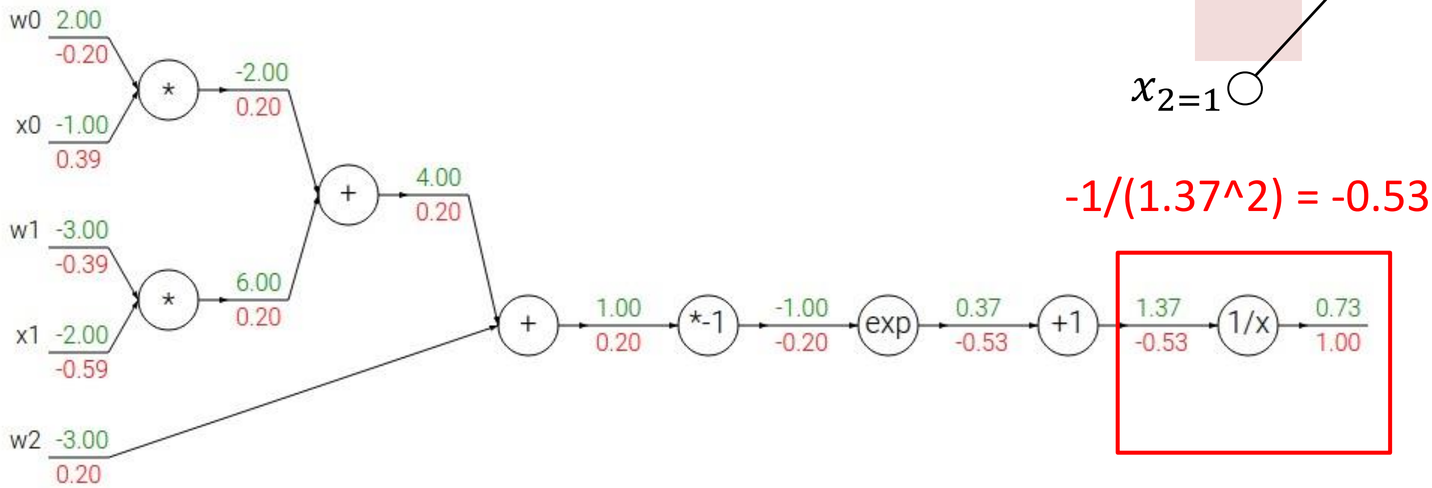
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Source: Andrej Karpathy & Fei-Fei Li

One example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$





$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

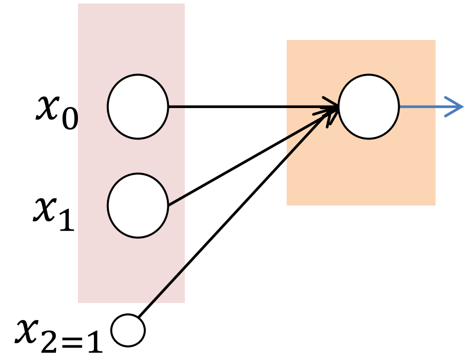
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Source: Andrej Karpathy & Fei-Fei Li

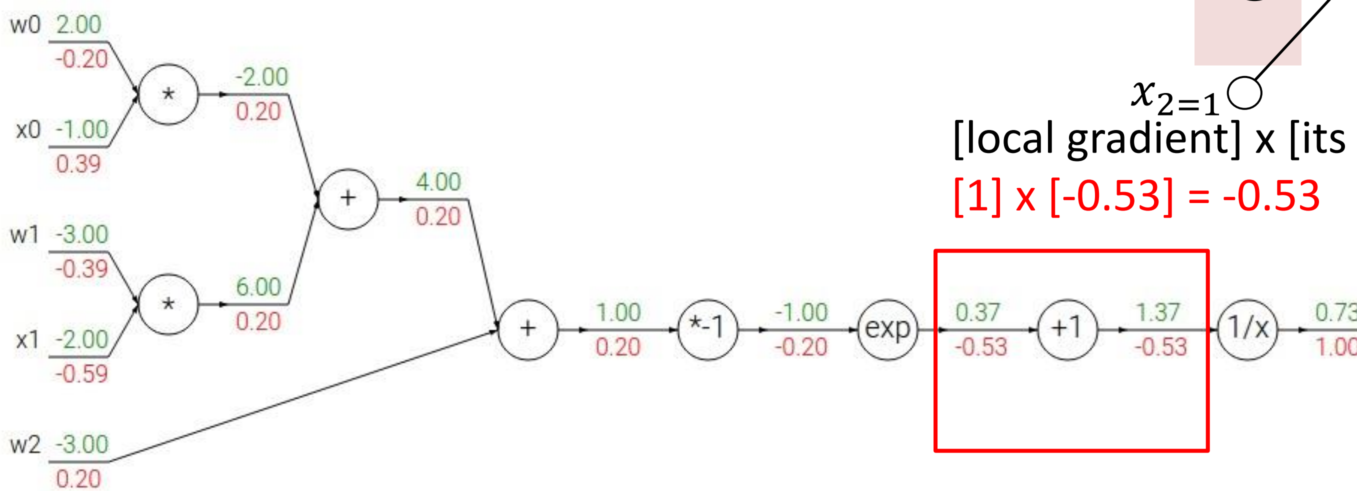
Another example:

$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



[local gradient] x [its gradient]

[1] x [-0.53] = -0.53



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

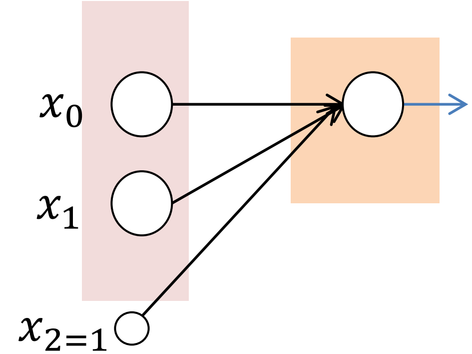
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Source: Andrej Karpathy & Fei-Fei Li

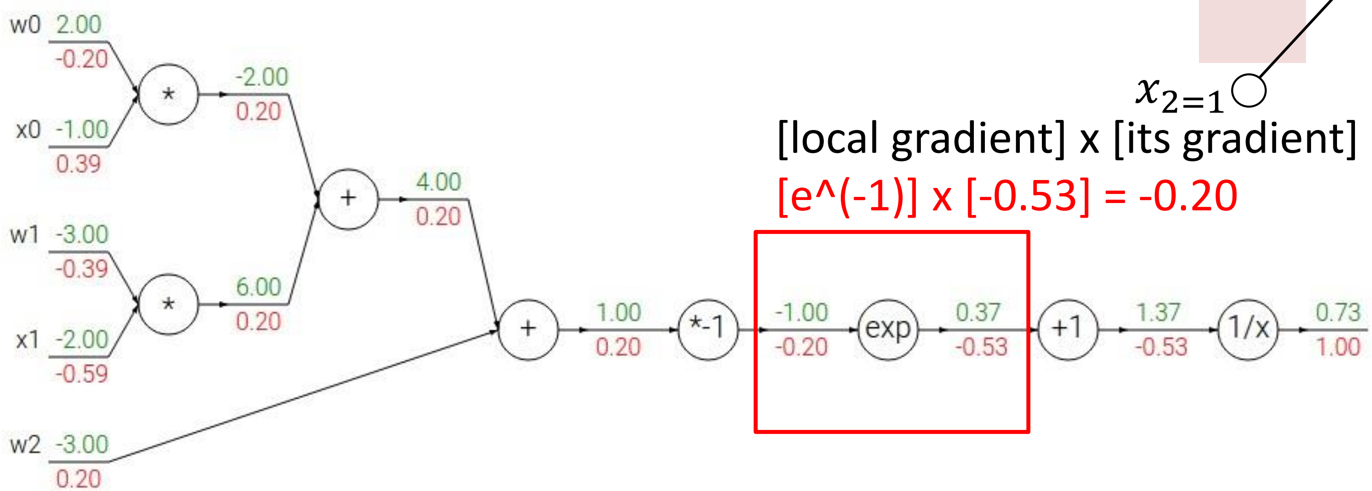
Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



[local gradient] x [its gradient]

[e^(-1)] x [-0.53] = -0.20



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

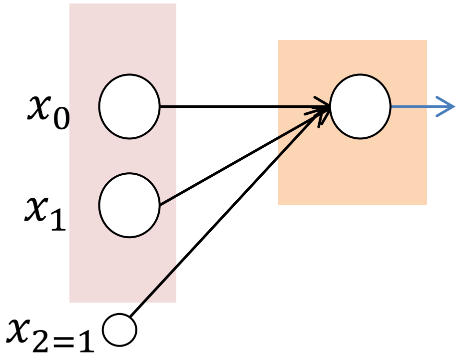
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

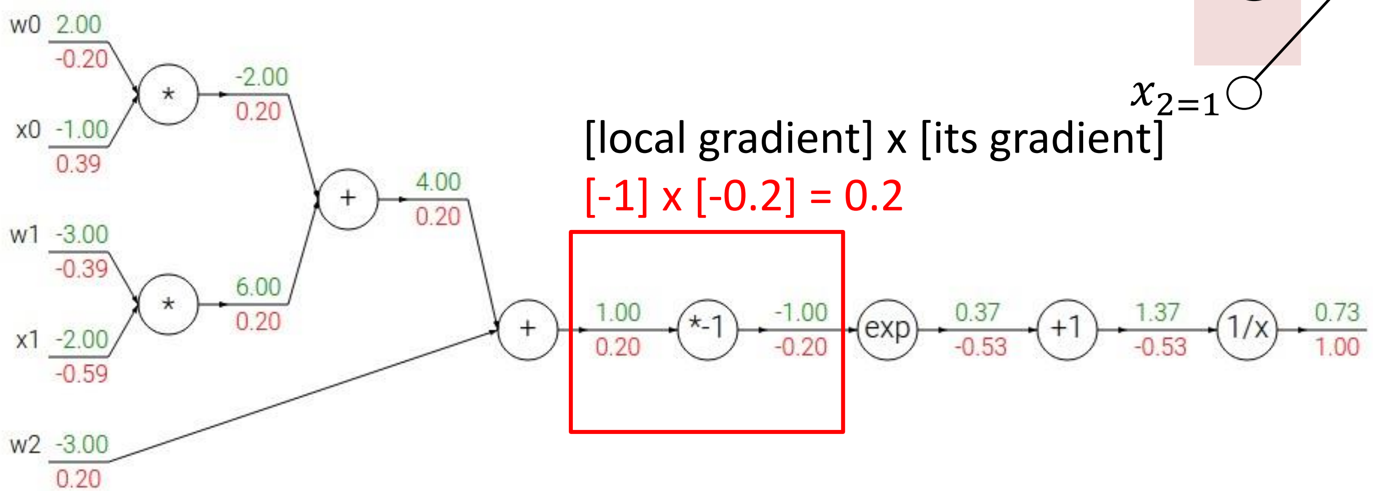
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Source: Andrej Karpathy & Fei-Fei Li

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$





$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

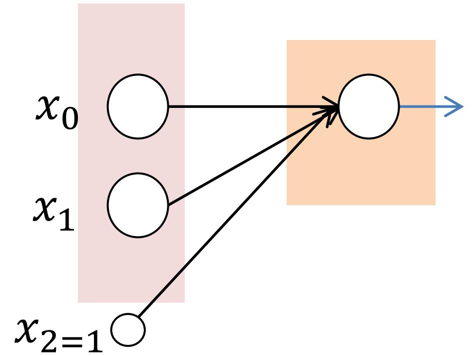
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Source: Andrej Karpathy & Fei-Fei Li

Another example:

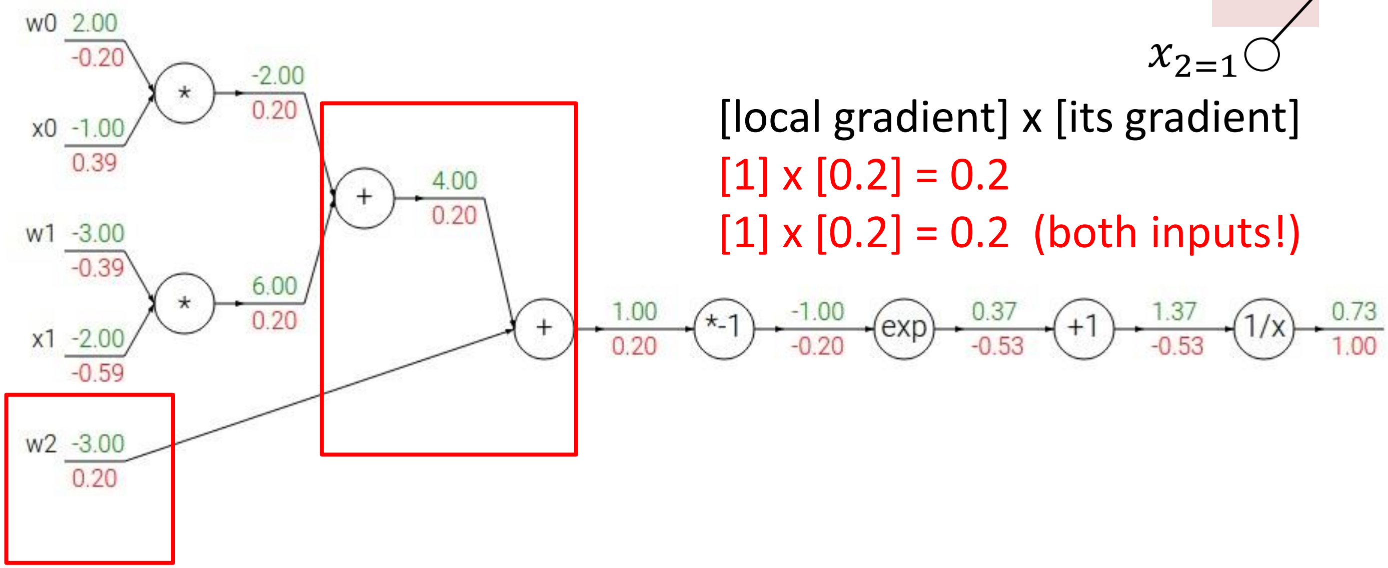
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



[local gradient] x [its gradient]

[1] x [0.2] = 0.2

[1] x [0.2] = 0.2 (both inputs!)



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

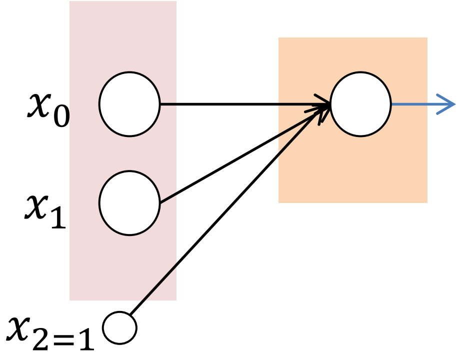
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

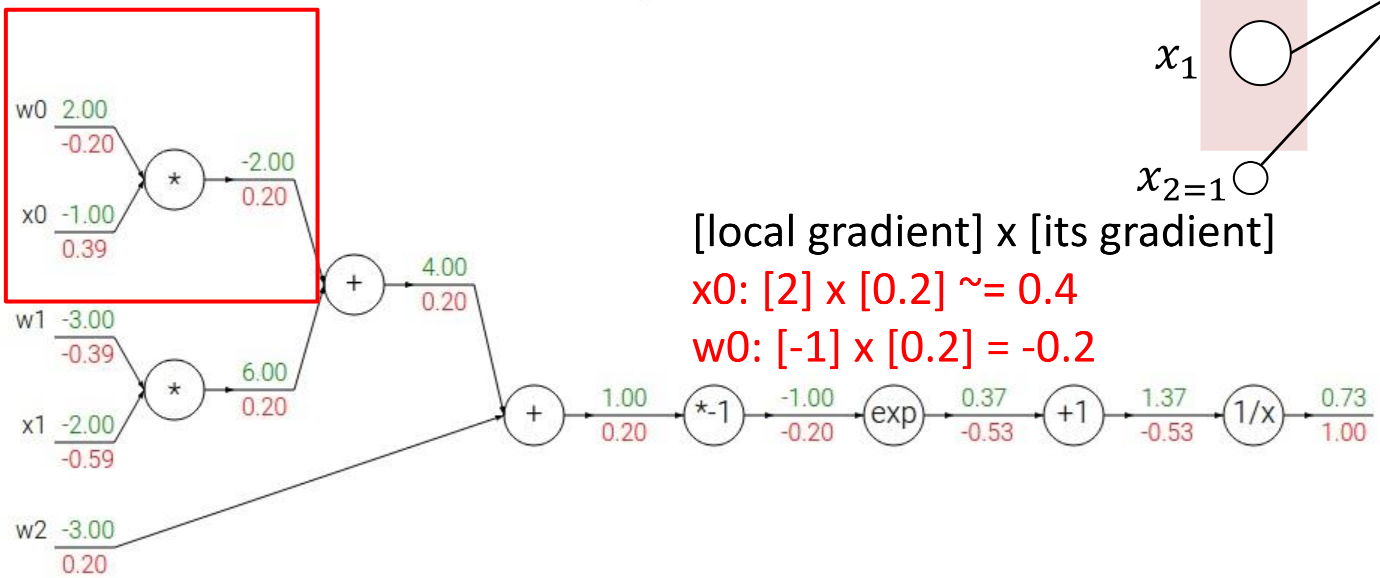
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Source: Andrej Karpathy & Fei-Fei Li

Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$





[local gradient] x [its gradient]

x0: [2] x [0.2] ~= 0.4

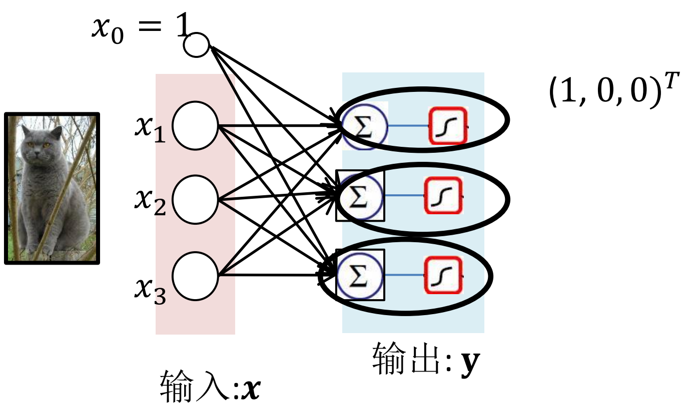
w0: [-1] x [0.2] = -0.2

$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

Source: Andrej Karpathy & Fei-Fei Li

反向传播

- 一层神经网络（线性模型）



➢ 1.给定输入，计算输出值：

$$a_i = \sum_{j=0}^{3} w_{ij} x_i = \boldsymbol{w}_{i.} \cdot \boldsymbol{x},$$

$$i = (1,2,3)$$

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

MSE Loss: L=$(\mathbf{y} - \hat{\mathbf{y}})^2$ =$(1 - y_1)^2$+$y_2{}^2 + y_3{}^2$

➢ 2.根据链规则，计算梯度$\frac{\partial L}{\partial \boldsymbol{w}}$：

$$\frac{\partial \mathrm{L}}{\partial y_1} = 2(y_1\text{-}1)$$

$$\frac{\partial \mathrm{L}}{\partial y_i} = 2y_i, \text{(i=2,3)}$$

$$\frac{\partial \mathrm{L}}{\partial a_i} = \frac{\partial \mathrm{L}}{\partial y_i}\frac{\partial y_i}{\partial a_i} = \frac{\partial \mathrm{L}}{\partial y_i}\sigma(a_i)(1\text{-}\sigma(\mathrm{a_i}))$$

$$\frac{\partial \mathrm{L}}{\partial w_{ij}} = \frac{\partial \mathrm{L}}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}} = \frac{\partial \mathrm{L}}{\partial a_i} x_{ij}$$

$$= \frac{\partial \mathrm{L}}{\partial y_i}\sigma(a_i)(1\text{-}\sigma(a_i))$$

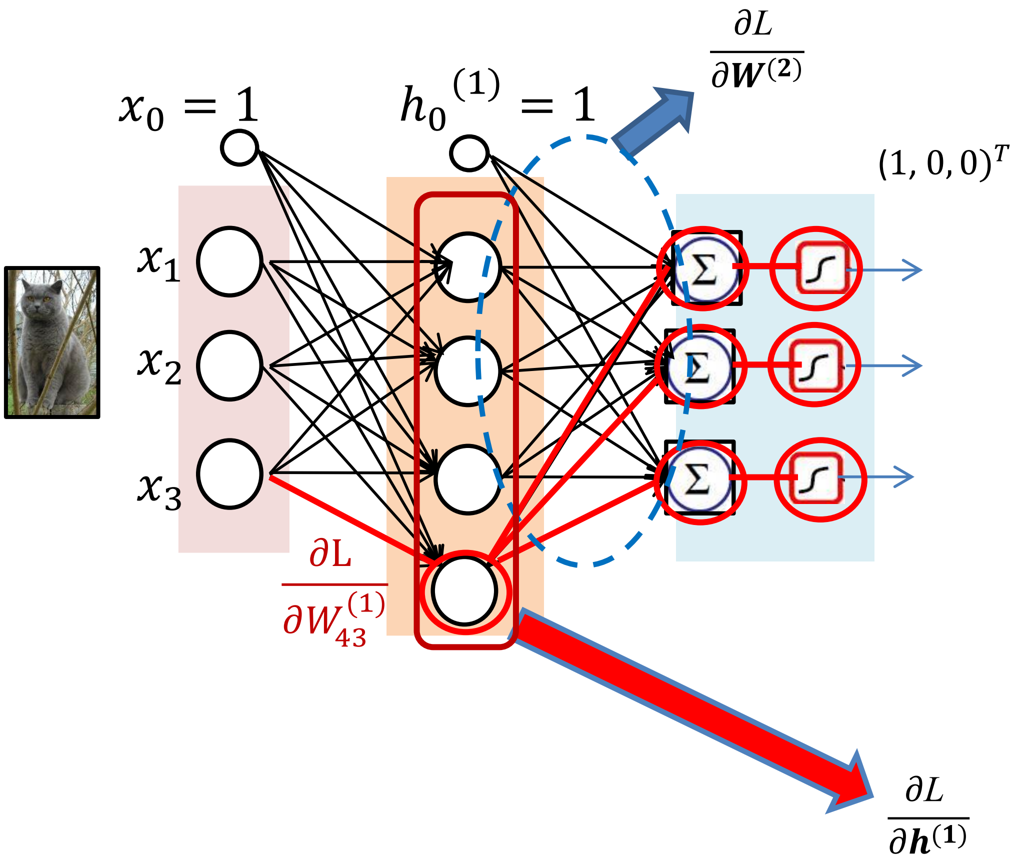
$$\frac{\partial \mathrm{L}}{\partial \mathbf{y}} = 2(\mathbf{y} - \hat{\mathbf{y}})$$

$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{a}} = 2[(\mathbf{y} - \hat{\mathbf{y}}) \cdot \sigma(\boldsymbol{a}) \cdot (1 - \sigma(\boldsymbol{a}))]^T$$

$$\frac{\partial L}{\partial \boldsymbol{W}} = 2[(\mathbf{y} - \hat{\mathbf{y}}) \cdot \sigma(\boldsymbol{a}) \cdot (1 - \sigma(\boldsymbol{a}))]\, \boldsymbol{x}^{\boldsymbol{T}}$$

反向传播

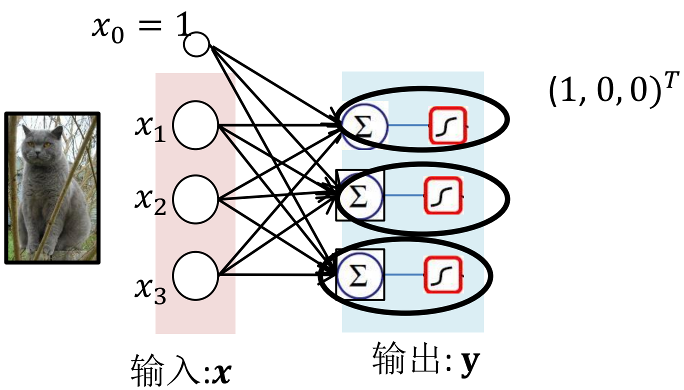
- 两层的网络



MSE Loss: $L=(\mathbf{y}-\hat{\boldsymbol{y}})^2$

反向传播

- 一层神经网络（线性模型）



➢ 1.给定输入，计算输出值:

$$a_i = \sum_{j=0}^{3} w_{ij} x_i = \boldsymbol{w}_{i.} \cdot \boldsymbol{x}, \qquad i = (1,2,3)$$

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

MSE Loss: L=$(\mathbf{y} - \hat{\mathbf{y}})^2$ =$(1 - y_1)^2$+$y_2{}^2 + y_3{}^2$

➢ 2.根据链规则，计算梯度$\frac{\partial L}{\partial \boldsymbol{x}}$：

$$\frac{\partial \mathrm{L}}{\partial y_1} = 2(1 - y_1)$$

$$\frac{\partial \mathrm{L}}{\partial y_i} = 2y_i, \text{ (i=2,3)}$$

$$\frac{\partial \mathrm{L}}{\partial a_i} = \frac{\partial \mathrm{L}}{\partial y_i}\frac{\partial y_i}{\partial a_i} = \frac{\partial \mathrm{L}}{\partial y_i}\sigma(a_i)(1\text{-}\sigma(a_i))$$

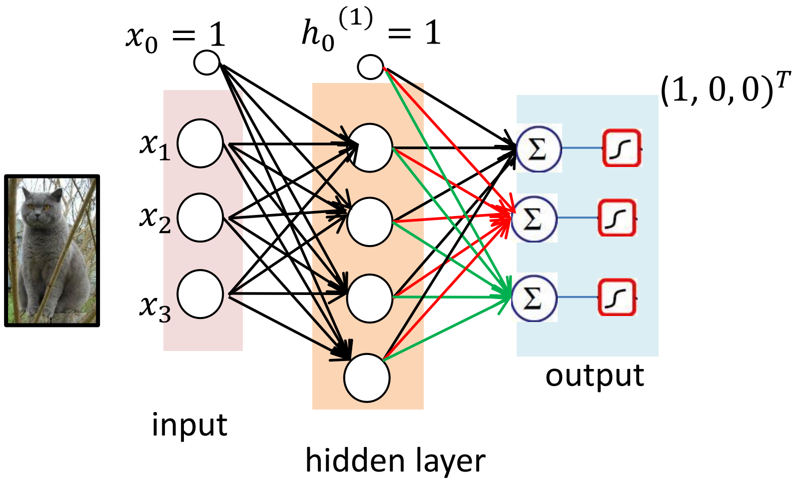
$$\frac{\partial \mathrm{L}}{\partial x_i} = \frac{\partial \mathrm{L}}{\partial a_i}\frac{\partial a_i}{\partial x_i} = \frac{\partial \mathrm{L}}{\partial a_i}\sum_{j=0}^{3} w_{ij}$$

$$\Downarrow$$

$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{x}} = \frac{\partial \mathrm{L}}{\partial \boldsymbol{a}} \boldsymbol{W}$$

反向传播

- 两层的网络



➢1给定输入，计算输出值：

$$\boldsymbol{a}^{(1)} = \boldsymbol{W}^{(1)} \cdot \boldsymbol{x}$$
$$\boldsymbol{h}^{(1)} = \sigma\left(\boldsymbol{a}^{(1)}\right)$$
$$\boldsymbol{a}^{(2)} = \boldsymbol{W}^{(2)} \cdot \boldsymbol{h}^{(1)}$$
$$\boldsymbol{y} = \sigma\left(\boldsymbol{a}^{(2)}\right)$$

➢MSE Loss: L=$(\mathbf{y} - \hat{\mathbf{y}})^2$

➢2根据链规则，计算梯度$\frac{\partial L}{\partial \boldsymbol{a}^{(i)}}$，$\frac{\partial L}{\partial \boldsymbol{x}}$：

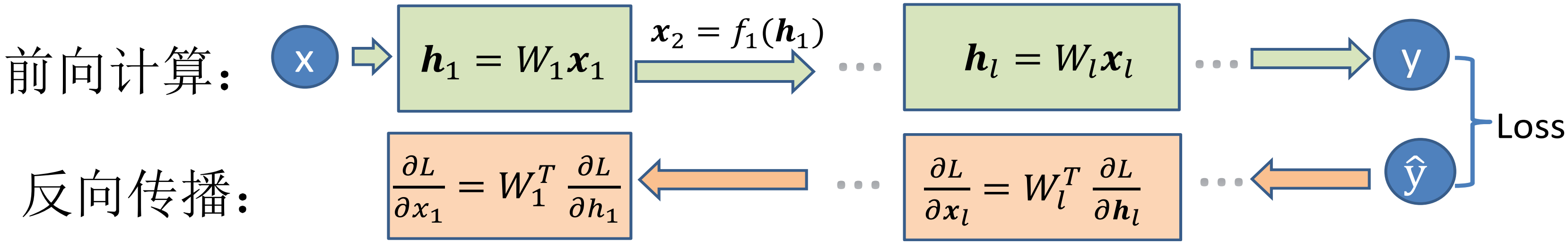
$$\frac{\partial \mathrm{L}}{\partial \mathbf{y}} = 2(\mathbf{y} - \hat{\mathbf{y}})$$
$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{a}^{(2)}} = \frac{\partial \mathrm{L}}{\partial \mathbf{y}} \cdot \sigma\left(\boldsymbol{a}^{(2)}\right) \cdot \left(1 - \sigma\left(\boldsymbol{a}^{(2)}\right)\right)$$
$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \mathrm{L}}{\partial \boldsymbol{a}^{(2)}} \boldsymbol{W}^{(2)}$$
$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{a}^{(1)}} = \frac{\partial \mathrm{L}}{\partial \boldsymbol{h}^{(1)}} \cdot \sigma\left(\boldsymbol{a}^{(1)}\right) \cdot \left(1 - \sigma\left(\boldsymbol{a}^{(1)}\right)\right)$$
$$\frac{\partial L}{\partial \boldsymbol{x}} = \frac{\partial \mathrm{L}}{\partial \boldsymbol{a}^{(1)}} \boldsymbol{W}^{(1)}$$

➢3.根据链规则，计算梯度$\frac{\partial L}{\partial \boldsymbol{W}^{(i)}}$：

$$\frac{\partial L}{\partial \boldsymbol{W}^{(2)}} = \frac{\partial \mathrm{L}}{\partial \boldsymbol{a}^{(2)}} \boldsymbol{h}^{(1)}$$
$$\frac{\partial L}{\partial \boldsymbol{W}^{(1)}} = \frac{\partial \mathrm{L}}{\partial \boldsymbol{a}^{(1)}} \boldsymbol{x}$$

反向传播

- 利用链式法则计算梯度
- 利用了动态规划的思想

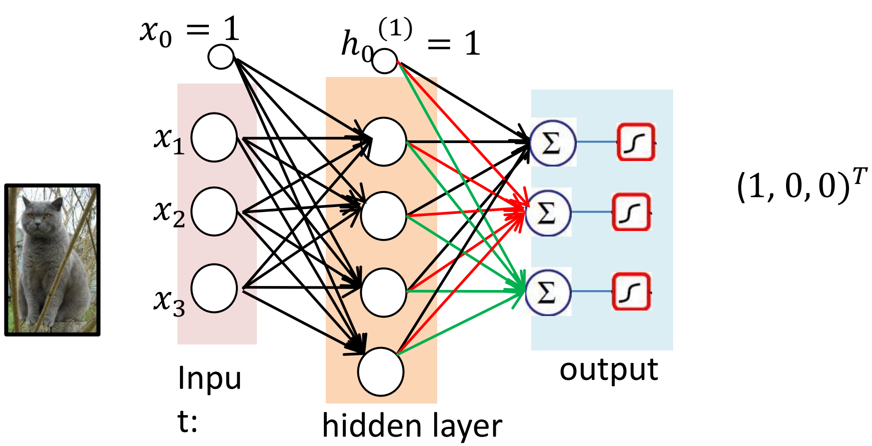


$$\frac{\partial L}{\partial W_l} = \sum (x_l \frac{\partial L}{\partial h_l})^T$$

前馈神经网络梯度下降训练算法

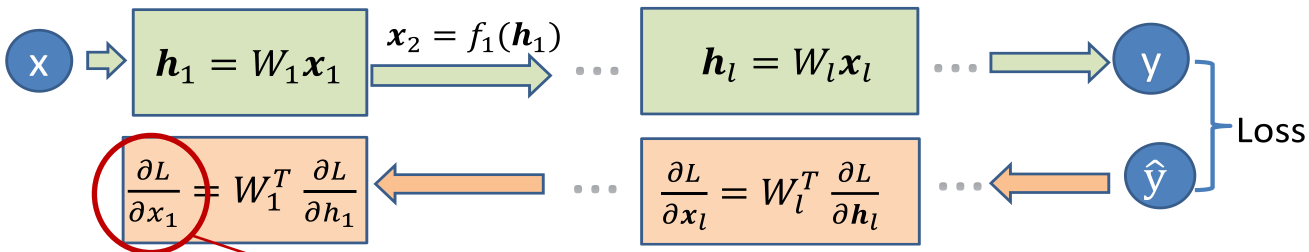
- 0.初始化权重 $\boldsymbol{W}^{(0)}$
- 1. 前向过程：
  - 1.1根据输入，计算输出值 $\boldsymbol{y}$
  - 1.2.计算损失函数值L($\mathbf{y}, \hat{\boldsymbol{y}}$)。
- 2.反向传播
  - 计算$\frac{\partial \mathrm{L}}{\partial \mathbf{y}}$
  - 后向传播直到计算$\frac{\partial \mathrm{L}}{\partial \mathbf{x}}$
- 3.计算梯度$\frac{\partial L}{\partial \boldsymbol{W}}$
- 4.更新梯度

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \frac{\partial \mathrm{L}}{\partial \boldsymbol{W}_t}$$



课外拓展研究

- 对抗样例（Adversarial example ）



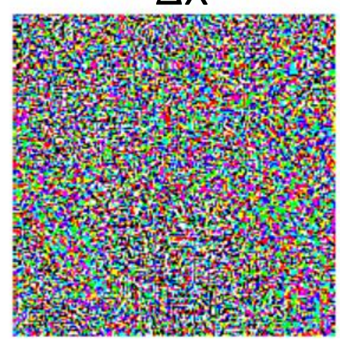
$f(X + \Delta X) \neq f(X)$, ΔX should be imperceptible by human

X

Panda

+

ΔX



=

$X + \Delta X$

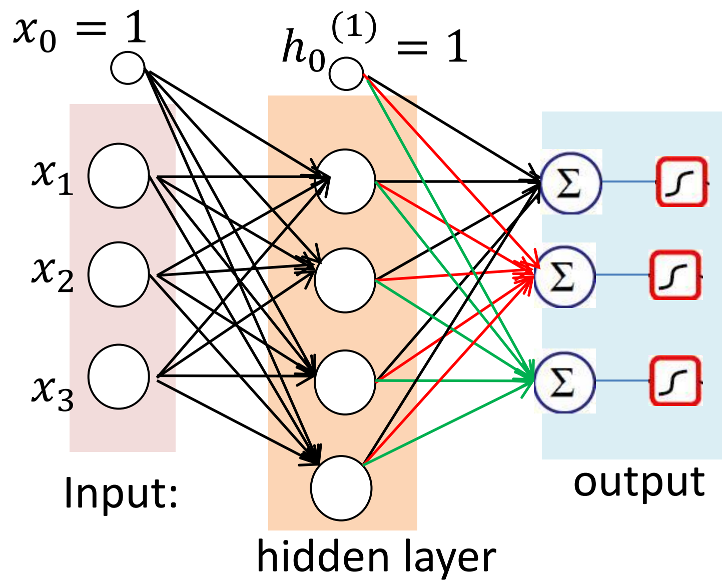
Car

主要内容

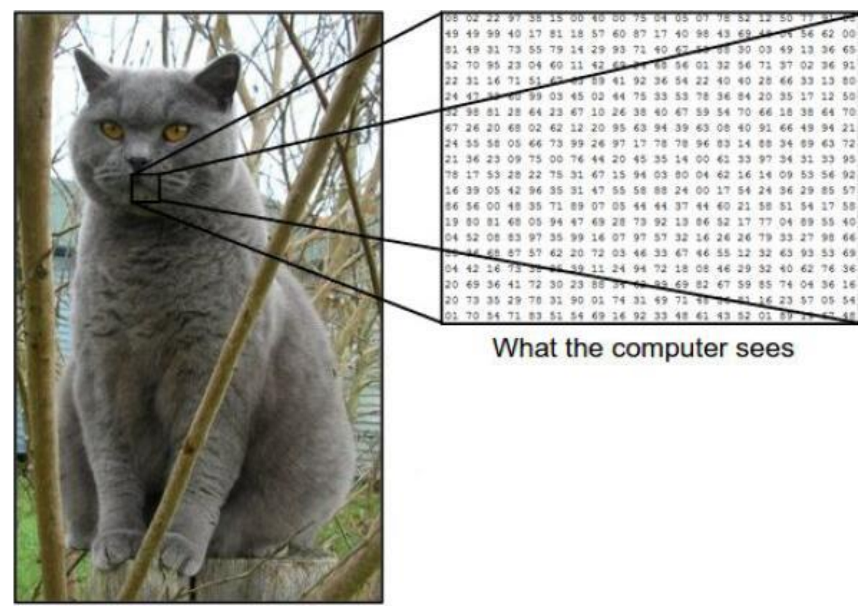
- 多层感知机(Multi-layer Perceptron, MLP)
  - 反向传播算法(Back-propagation)
- 卷积神经网络 （Convolutional Neural Network, CNN）
  - 卷积操作和卷积层
  - 池化 （Pooling）
- 循环神经网络（Recurrent Neural Network, RNN）
  - 建模和训练
  - LSTM模型

Feature extraction

- Feature extraction
  - Pixel-wise input
  - Correlation between features

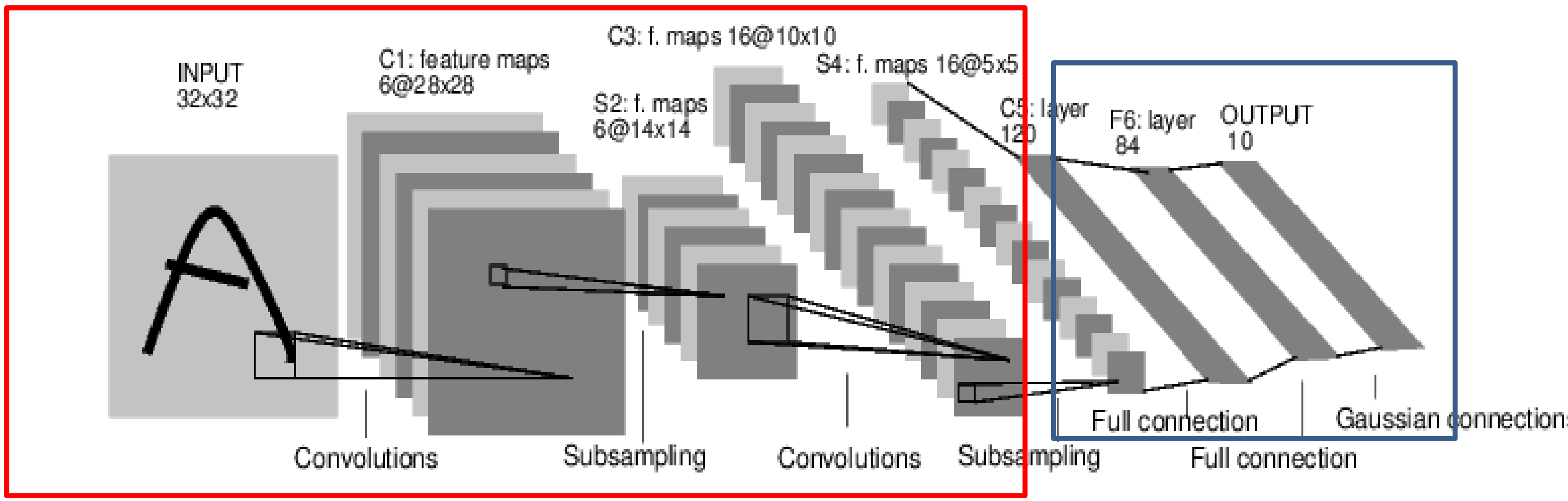


Convolutional Neural Network(CNN),卷积神经网络



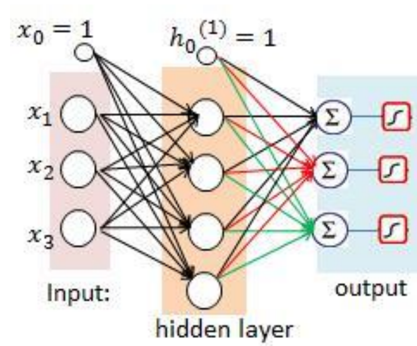
Convolution Neural Network

- Lenet-5



全连接层

Convolution related layers

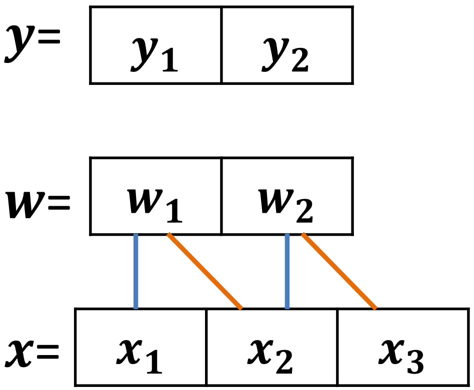


outline

- Convolutional layer (module)
  - Convolution operation
  - Filters
  - Convolution module in a network
- Pooling layer (module)

Convolution (卷积)

- 一维相关操作（correlation）例子



$$y_1 = w_1x_1 + w_2x_2$$

$$y_2 = w_1x_2 + w_2x_3$$

$$y_{i'} = \sum_{i=1}^{M_f=2} w_i x_{i'+i-1}$$

Correlation operator (similarity)

- 一维卷积（correlation）例子

- Flip

$\overline{w}$= | w_2 | w_1 |

$$y_{i'} = \sum_{i=1}^{M_f=2} w_{M_f+1-i} x_{i'+i-1}$$

Convolution

Convolution (卷积)

- 连续空间的卷积:

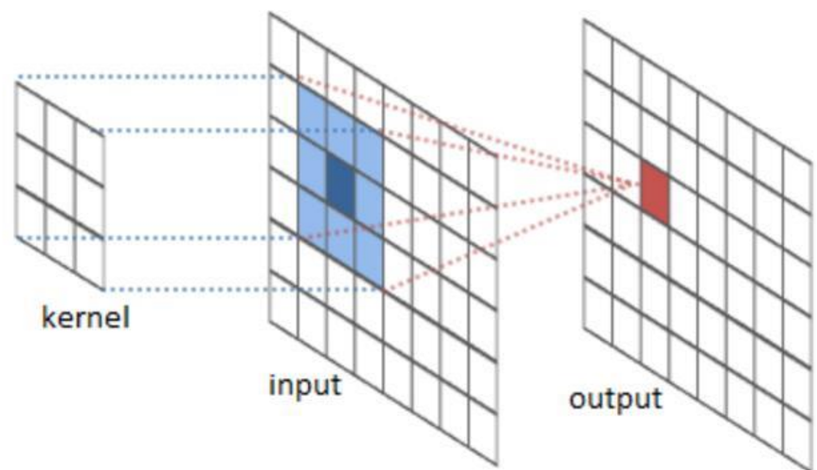
$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(s)h(t-s)\,ds$$

- 离散空间卷积：

$$y(n) = x(n) * w(n) = \sum_{i=-\infty}^{i=+\infty} x(i)w(n-i)$$

- 图像卷积是二维离散卷积

$$g(i,j) = \sum_{k,l} f(k,l)\,w(i-k,j-l)$$



Convolution （卷积）

- **图像卷积，二维, 离散**
  - **Correlation Operator(相关算子)**

定义：$g = f \otimes w$

$$g(i,j) = \sum_{k,l} f(i+k, j+l)\, w(k,l)$$

image

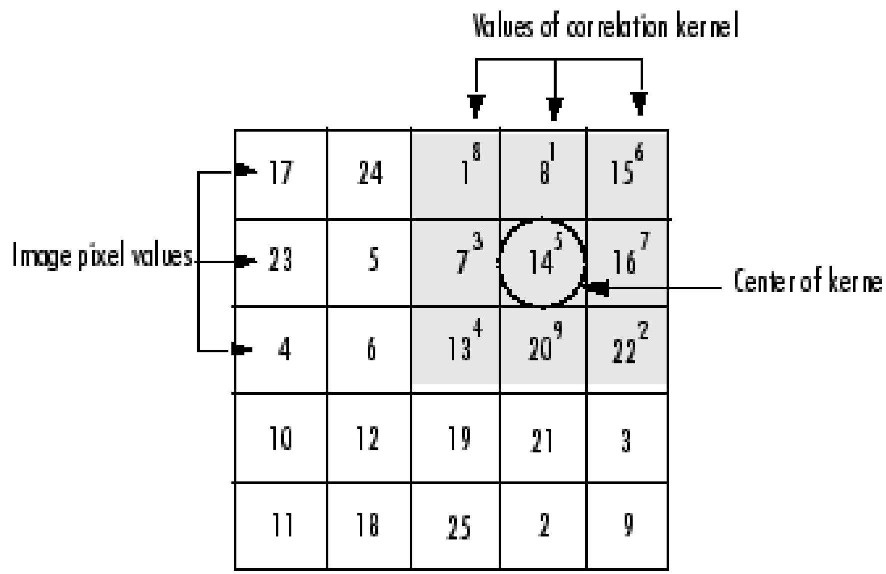
Kernel(filters)

f:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

w:

8	1	6
3	5	7
4	9	2



Convolution （卷积）

- **图像卷积，二维，离散**
  - Convolution operator (卷积算子)

定义：$g = f * w$

$$g(i,j) = \sum_{k,l} f(k,l)\, w(i-k, j-l)$$

image

Kernel(filters)

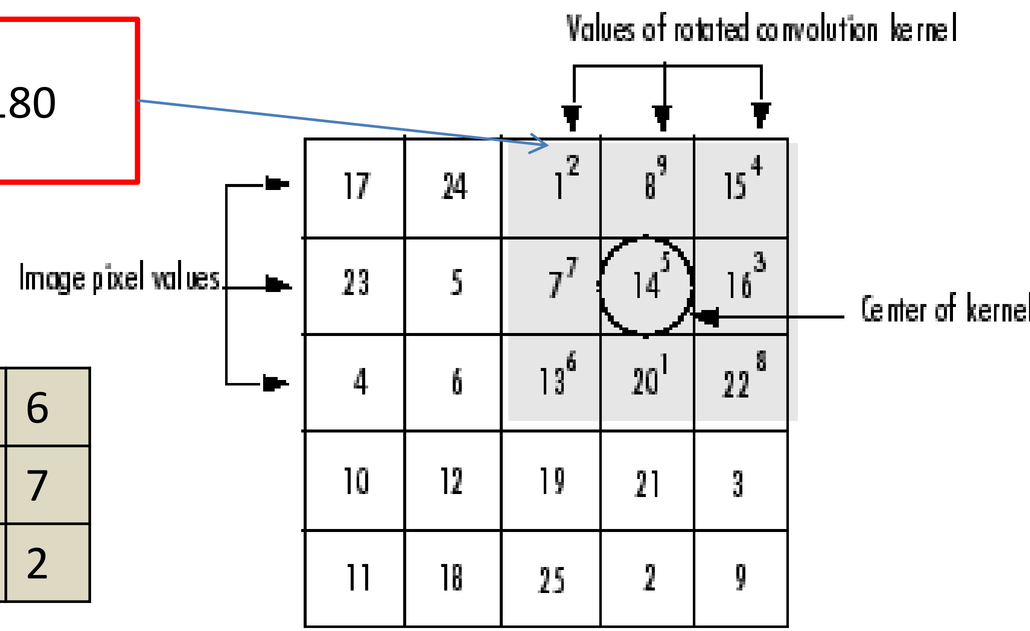
旋转180

f:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

w:

8	1	6
3	5	7
4	9	2



outline

- Convolutional layer (module)
  - Convolution operation
  - Filters
  - Convolution module in a network
- Pooling layer (module)

Practice with linear filters(线性滤波器)

Original

0	0	0
0	1	0
0	0	0

Filter

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

Filter

Shifted *left*
By 1 pixel

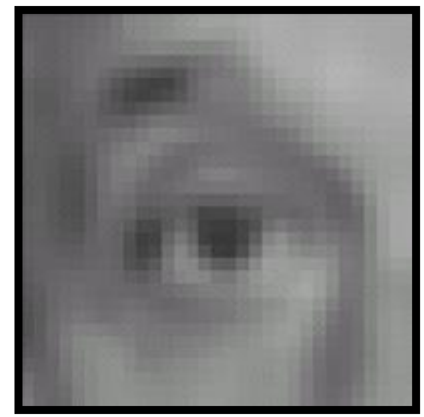
Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Filter



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

0	1	0
1	-4	1
0	1	0

Filter

Output Image

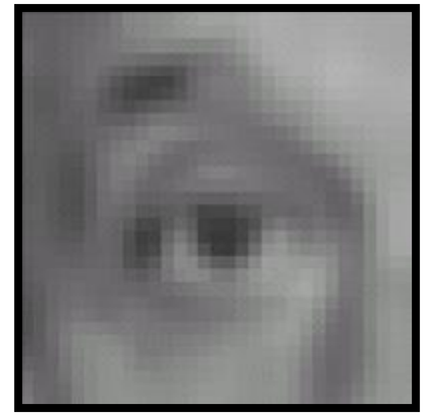
Edge detect （边缘检测）

Filters in practise

Input image

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

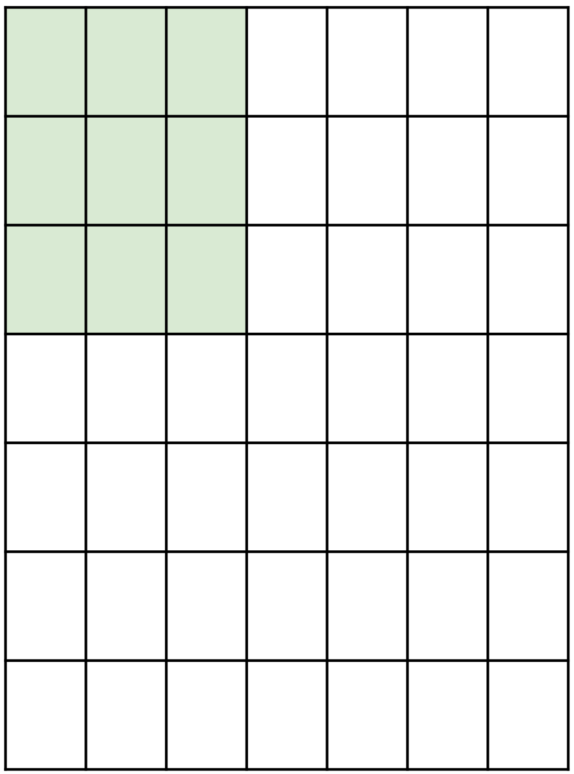
filter



output image

- Size of output image
  - How to move? stride
  - How about the border? padding

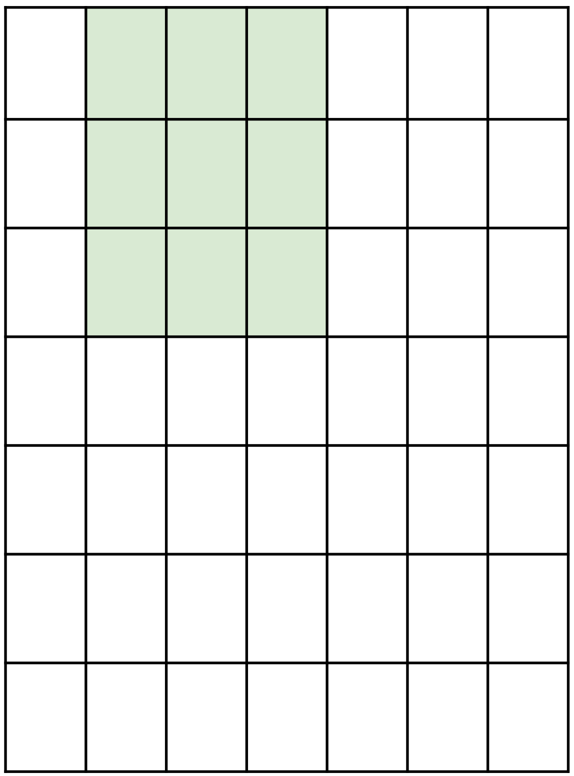
filters: stride（步幅）



7x7 input
assume 3x3 connectivity, stride 1

Source: Andrej Karpathy & Fei-Fei Li

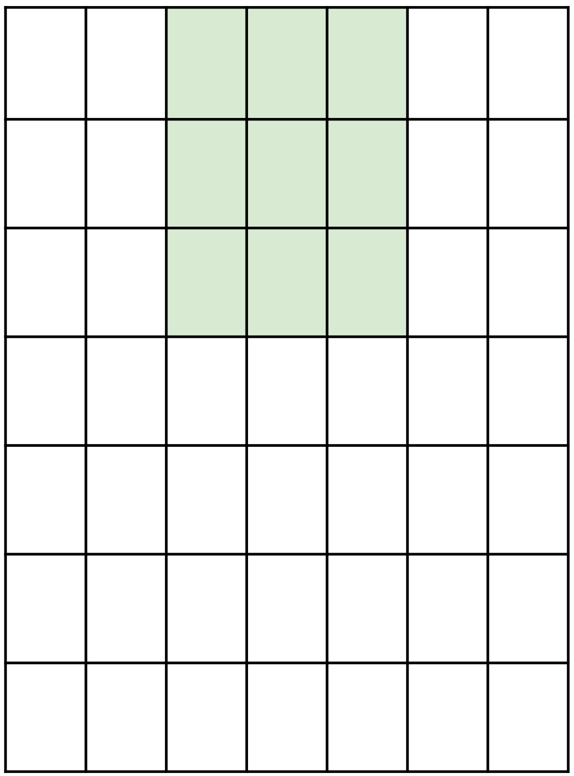
filters: stride （步幅）



7x7 input
assume 3x3 connectivity, stride 1

Source: Andrej Karpathy & Fei-Fei Li

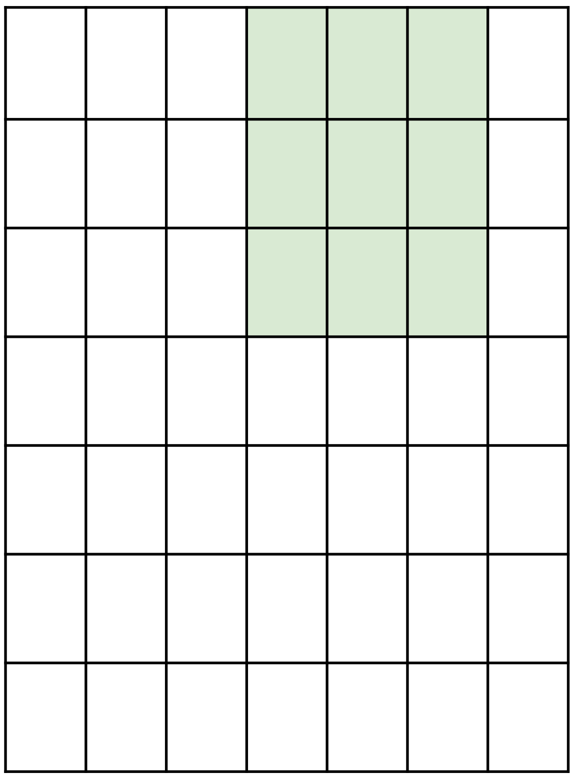
filters: stride （步幅）



7x7 input
assume 3x3 connectivity, stride 1

Source: Andrej Karpathy & Fei-Fei Li

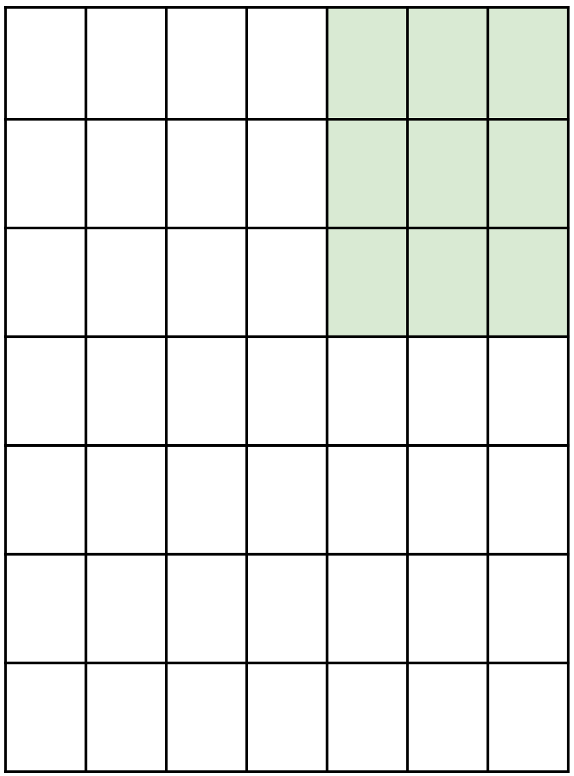
filters: stride （步幅）



7x7 input
assume 3x3 connectivity, stride 1

Source: Andrej Karpathy & Fei-Fei Li

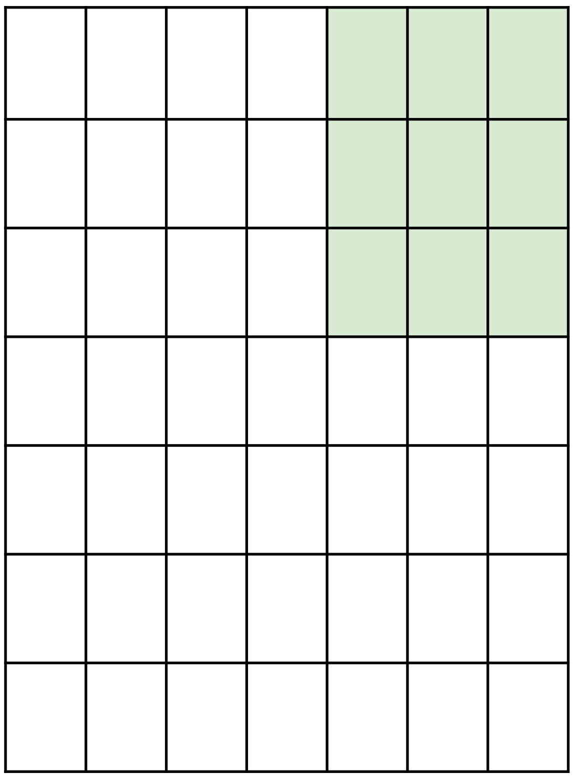
filters: stride （步幅）



7x7 input
assume 3x3 connectivity, stride 1

Source: Andrej Karpathy & Fei-Fei Li

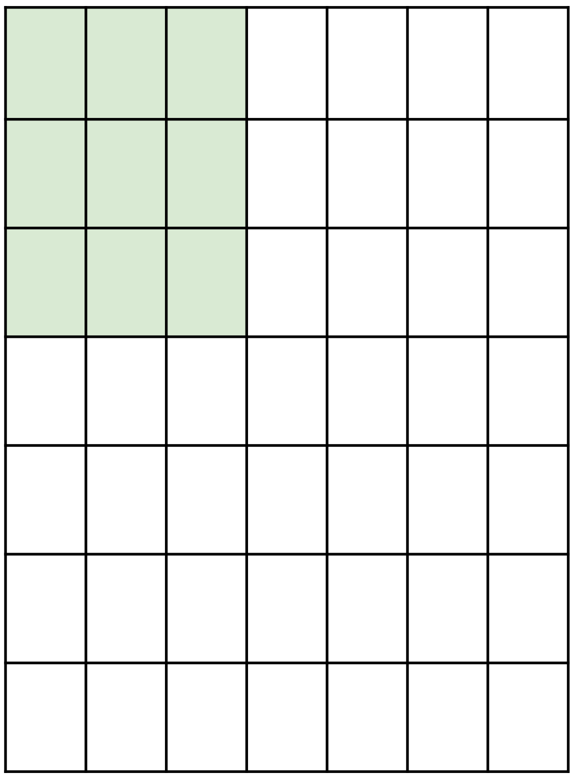
filters: stride （步幅）



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

Source: Andrej Karpathy & Fei-Fei Li

filters: stride （步幅）

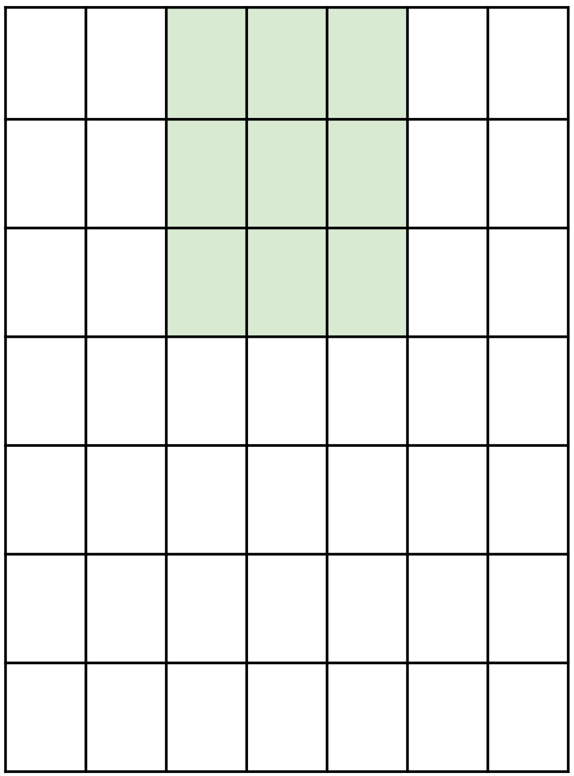


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?

Source: Andrej Karpathy & Fei-Fei Li

filters: stride （步幅）

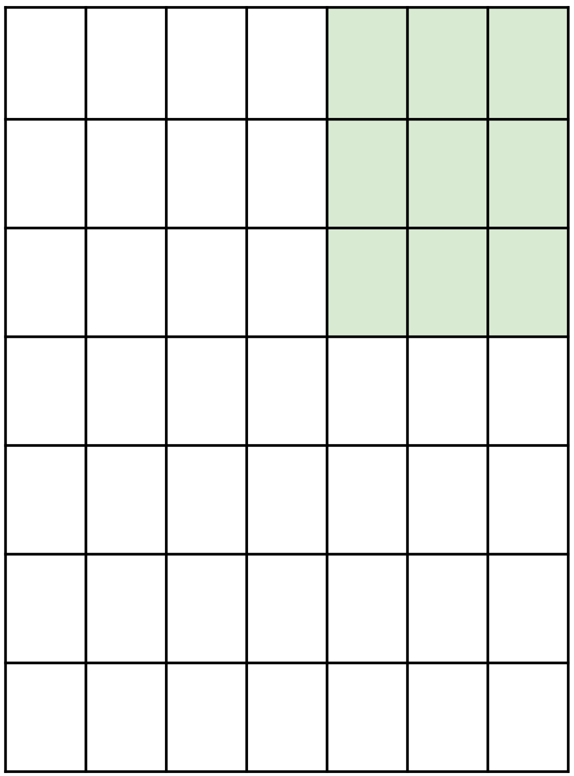


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?

Source: Andrej Karpathy & Fei-Fei Li

filters: stride （步幅）

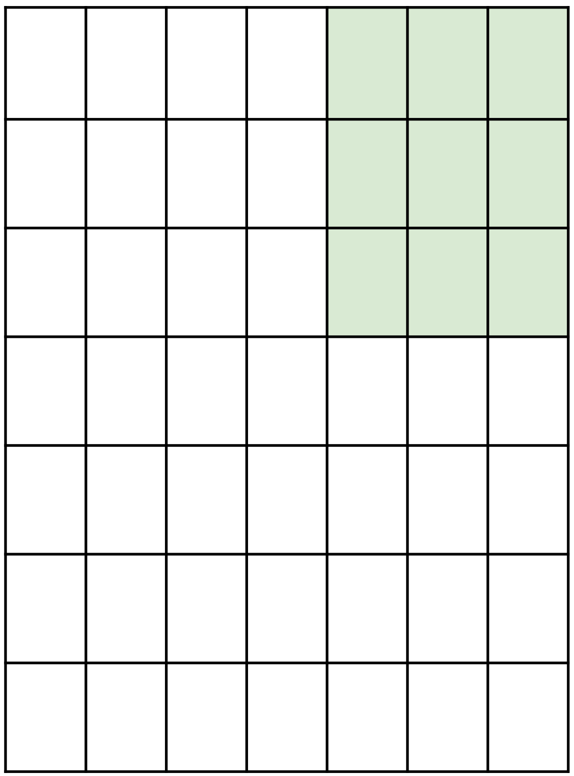


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?

Source: Andrej Karpathy & Fei-Fei Li

filters: stride （步幅）

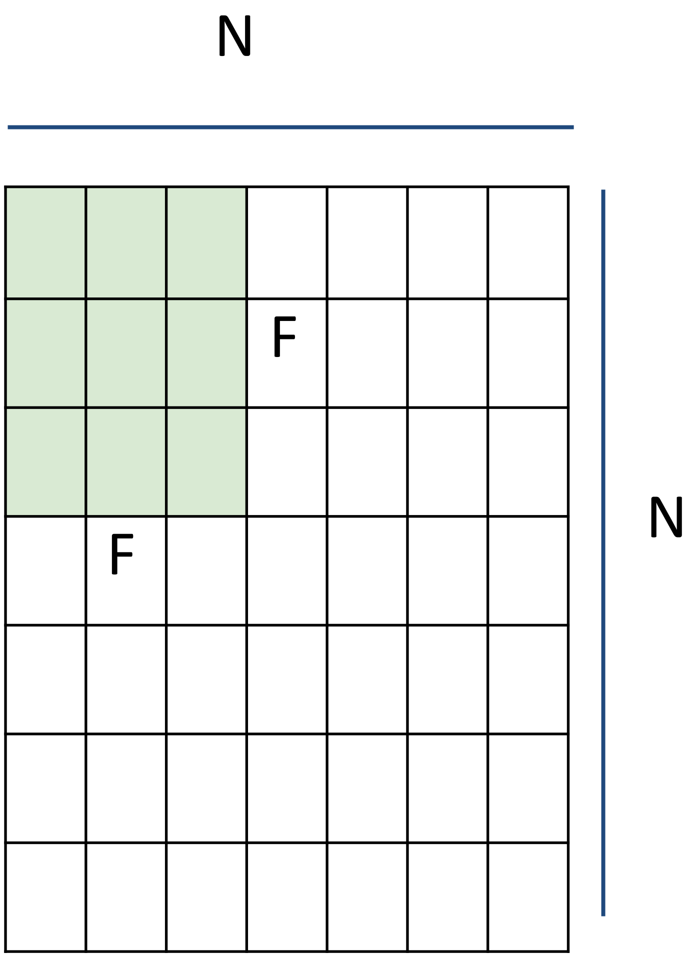


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?
=> **3x3 output**

Source: Andrej Karpathy & Fei-Fei Li

filters: stride （步幅）



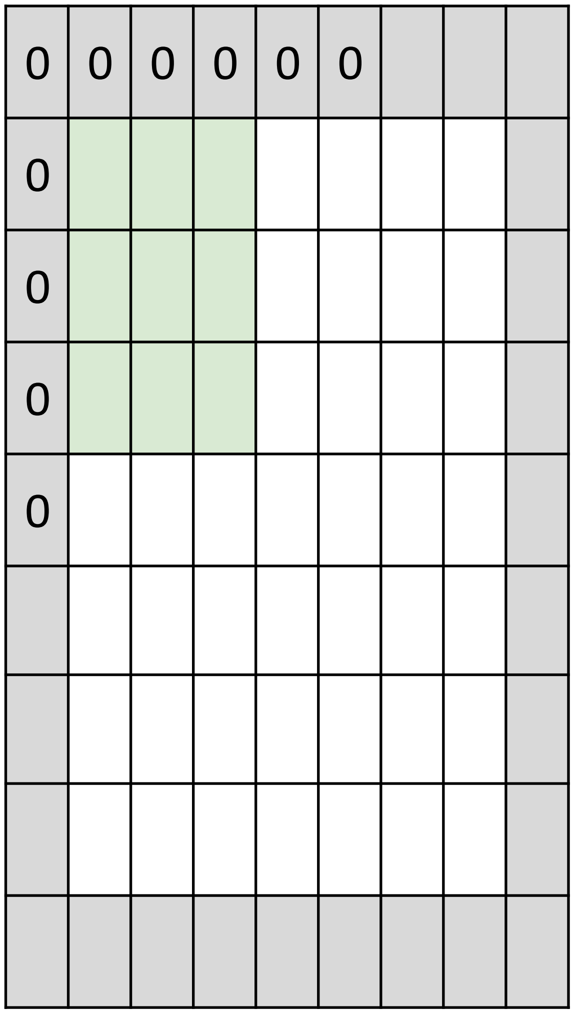
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3

Source: Andrej Karpathy & Fei-Fei Li

filters: padding

- In practice: Common to zero pad the border



e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

7x7 => preserved size!

Source: Andrej Karpathy & Fei-Fei Li

Filters in practise

- **“Same convolution” (preserves size)**

Input [9x9]

3x3 neurons, stride 1, pad **1** => [9x9]

- No headaches when sizing architectures
- Works well

- **“Valid convolution” (shrinks size)**

Input [9x9]

3x3 neurons, stride 1, pad **0** => [7x7]

- **Headaches** with sizing the full architecture
- Works Worse!

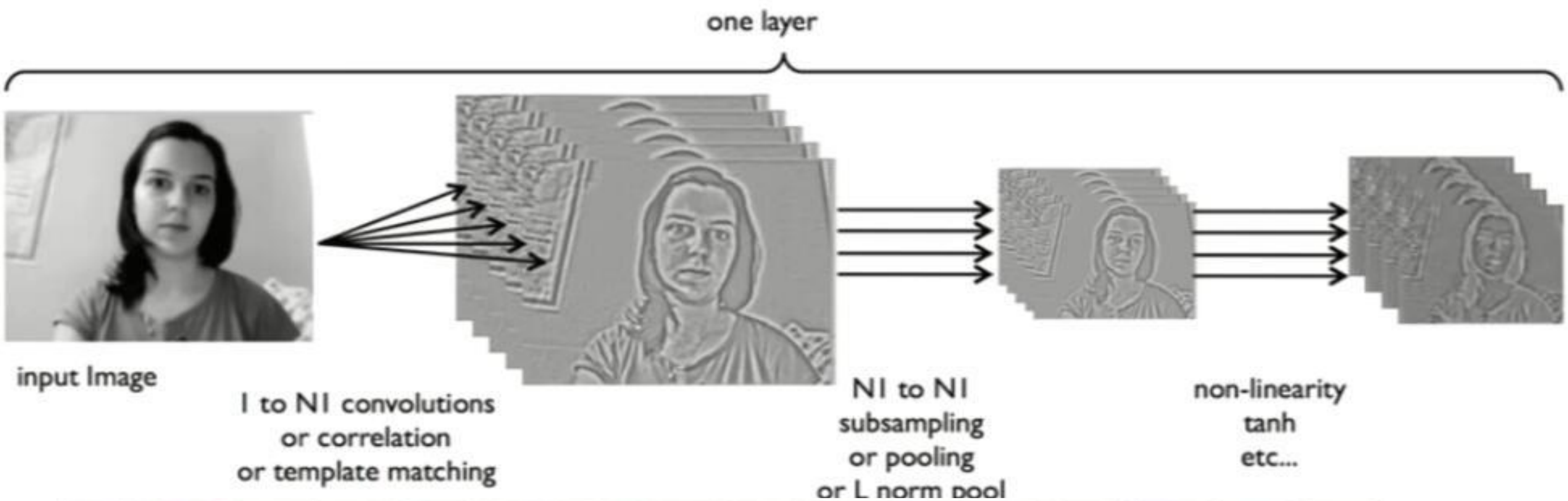
Source: Andrej Karpathy & Fei-Fei Li

outline

- Convolutional layer (module)
  - Convolution operation
  - Filters
  - Convolution module in a network
- Pooling layer (module)

Feature detection (特征检测)

- Learning filters (weights)

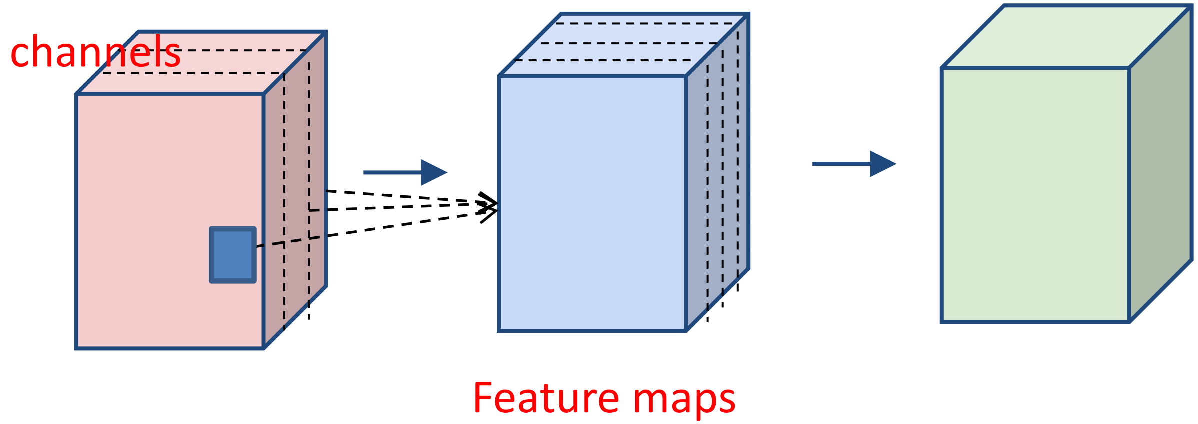


Convolution Layer (卷积层)

Input: X∈ $R^{d_{in} \times h \times w}$

weight: W∈ $R^{d_{out} \times d_{in} \times F_h \times F_w}$

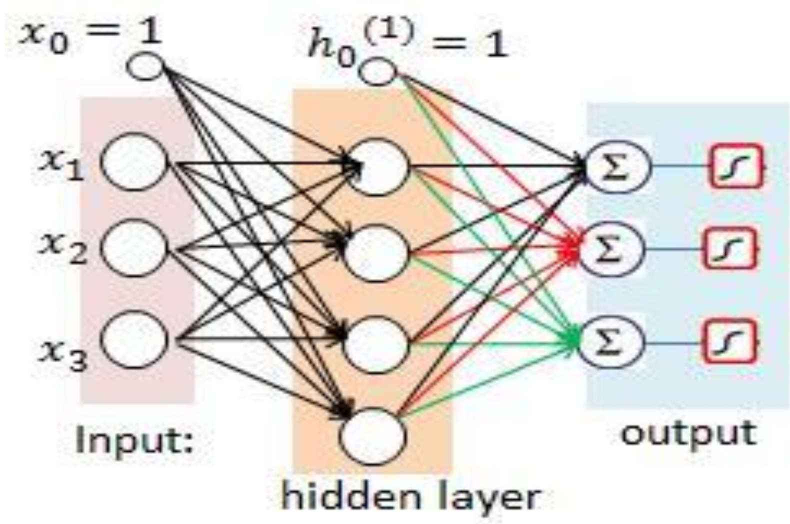
output: Y∈ $R^{d_{out} \times h \times w}$



Input: x∈ $R^{d_{in}}$

weight: W∈ $R^{d_{out} \times d_{in}}$

output: y∈ $R^{d_{out}}$

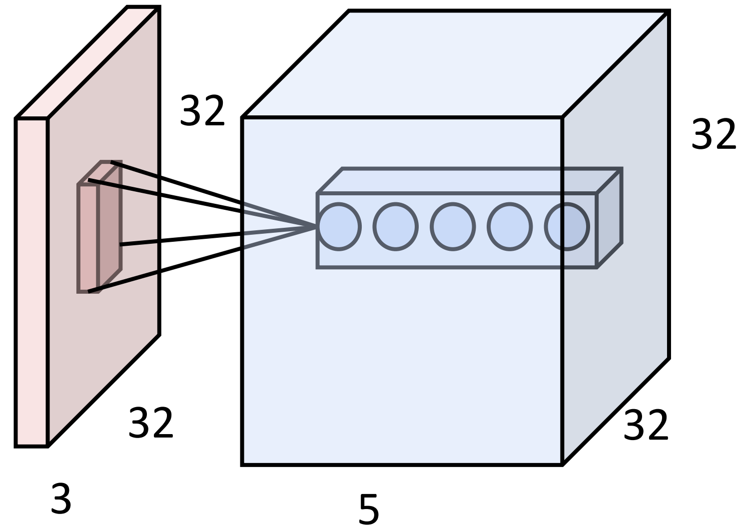


Forward (前向过程)

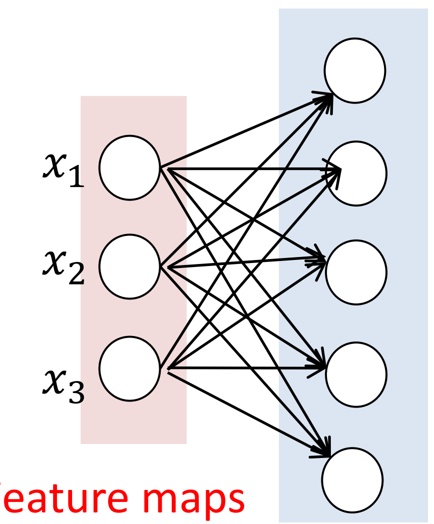
Input: $X \in R^{d_{in} \times h \times w}$

weight: $W \in R^{d_{out} \times d_{in} \times F_h \times F_w}$

output: $Y \in R^{d_{out} \times h \times w}$

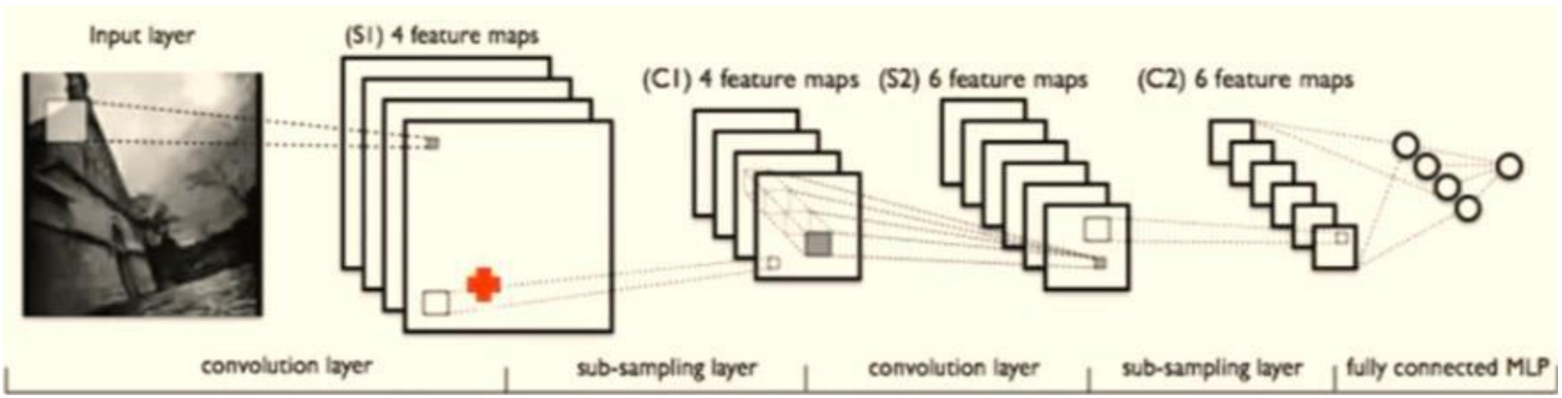


$$y_{f',i',j'} = \sum_{f=1}^{d_{in}} \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{f,i'+i-1,j'+j-1} w_{f',f,i,j} + b_{f'}$$



example

$$y_{4,10,10} = \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{1,10+i-1,10+j-1}\, w_{4,1,i,j} + b_4$$



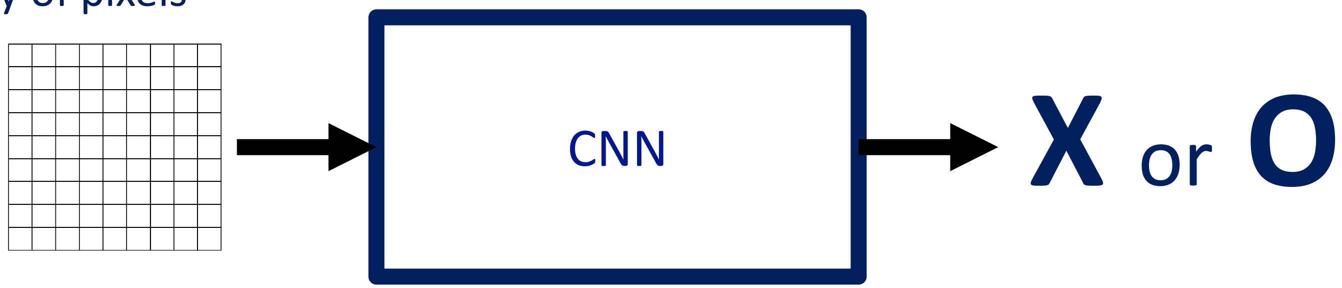
$$y_{4,10,100} = \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{1,10+i-1,100+j-1}\, w_{4,1,i,j} + \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{2,10+i-1,100+j-1}\, w_{4,2,i,j} +$$

$$\sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{3,10+i-1,100+j-1}\, w_{4,3,i,j} + \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{4,10+i-1,100+j-1}\, w_{4,4,i,j} + b_4$$

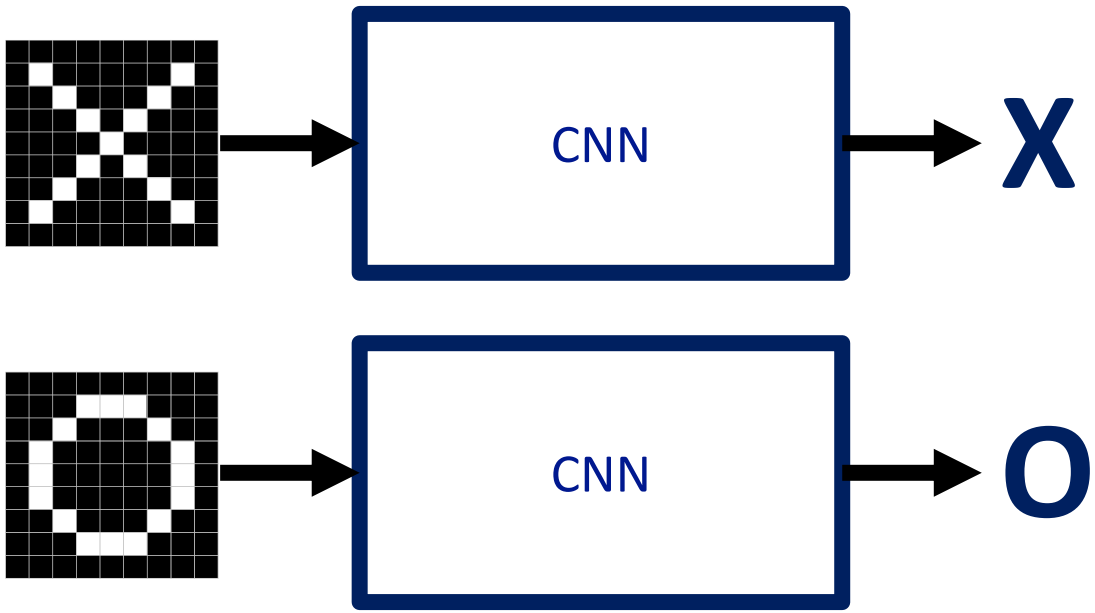
A toy ConvNet: X’s and O’s

Says whether a picture is of an X or an O

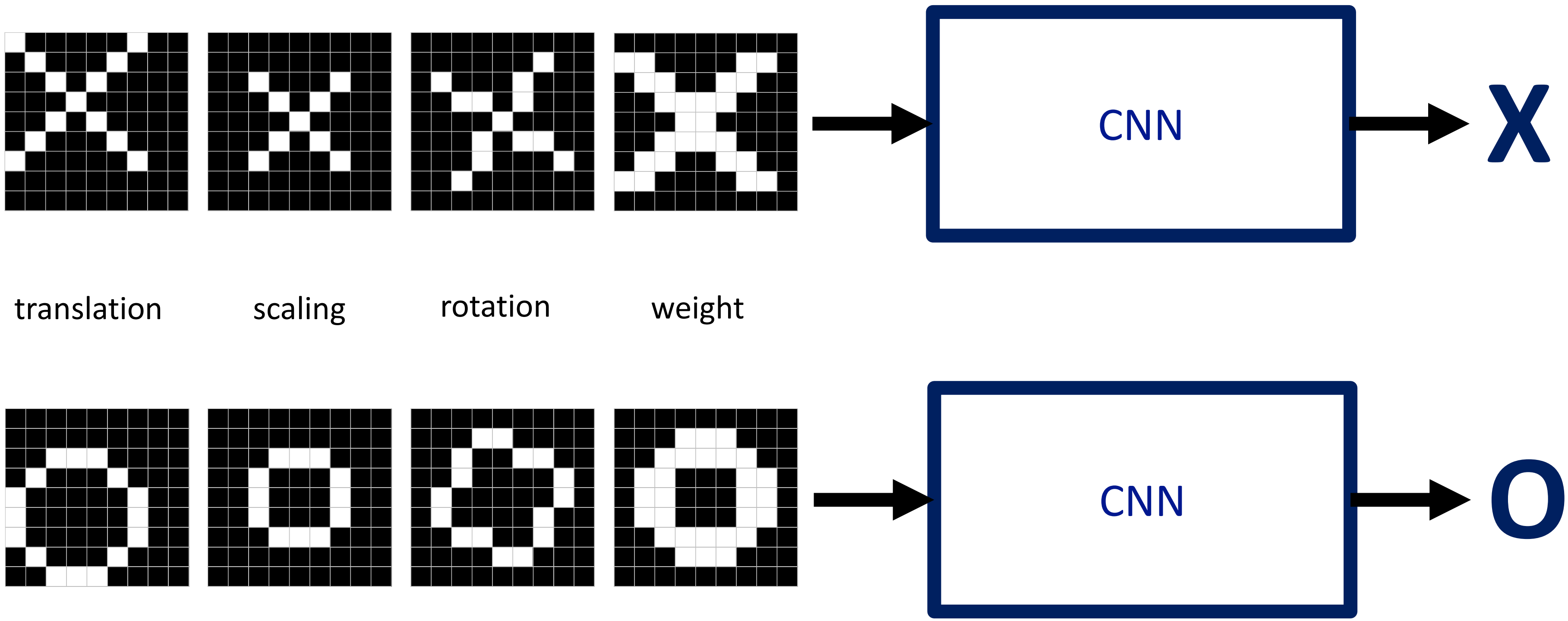
A two-dimensional array of pixels



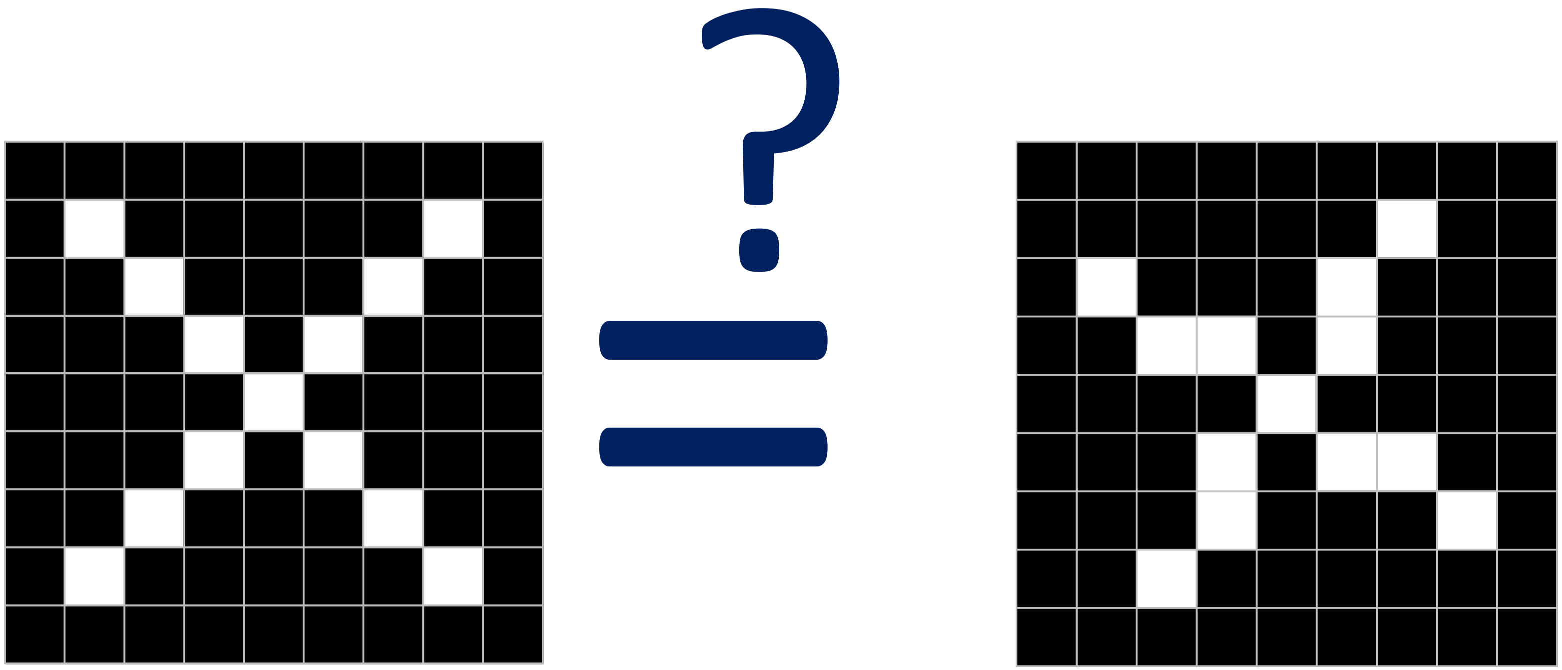
For example



Trickier cases



Deciding is hard



What computers see

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

$\stackrel{?}{=}$

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

What computers see

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

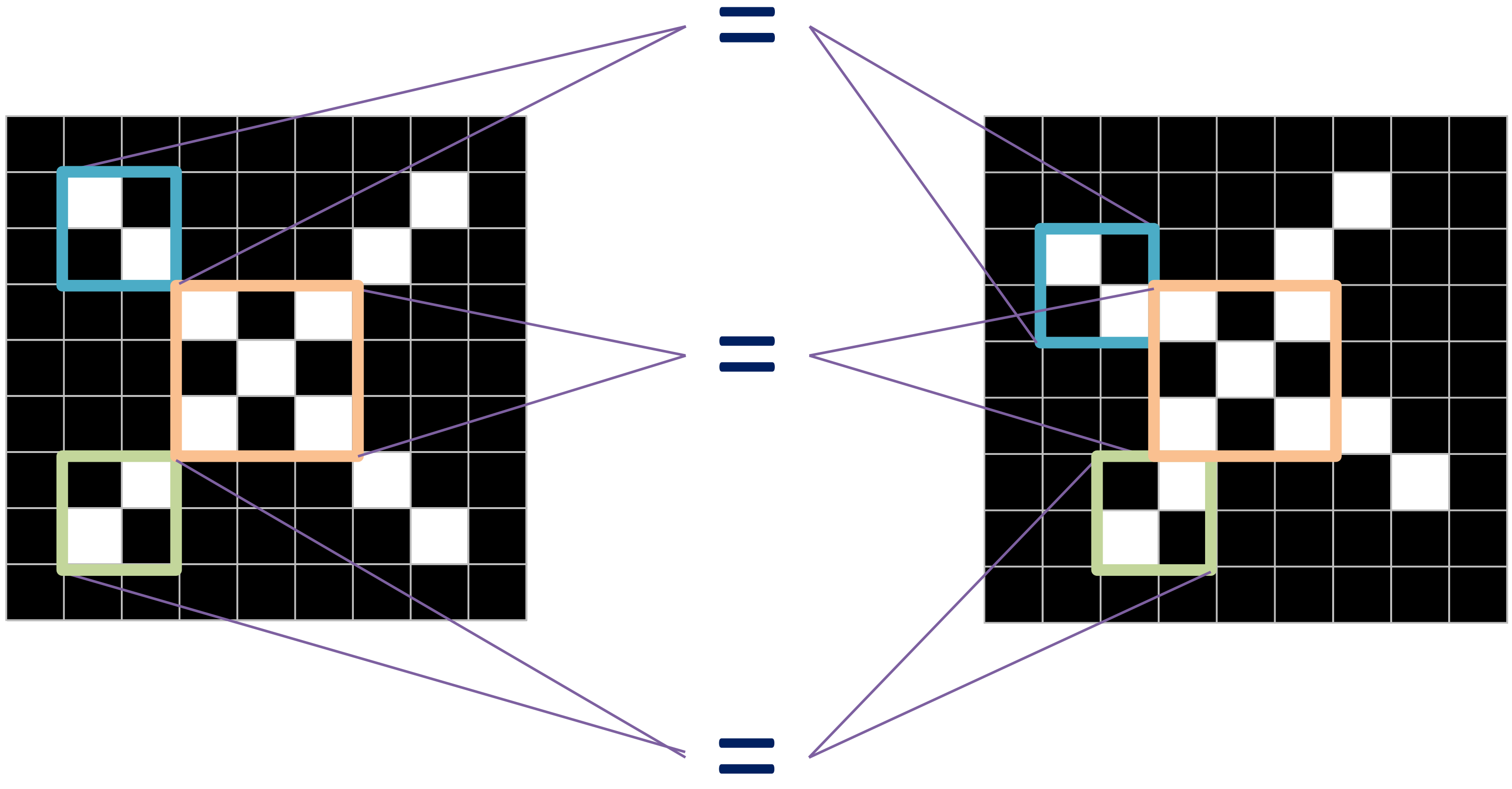
Computers are literal

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

$\neq$

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

ConvNets match pieces of the image

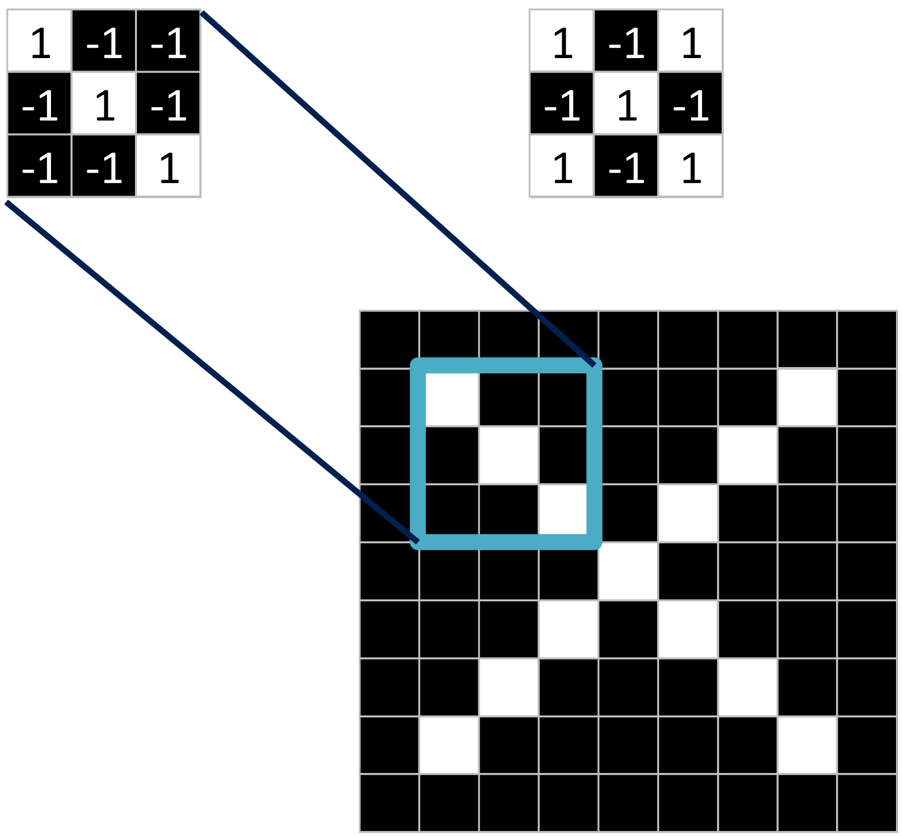


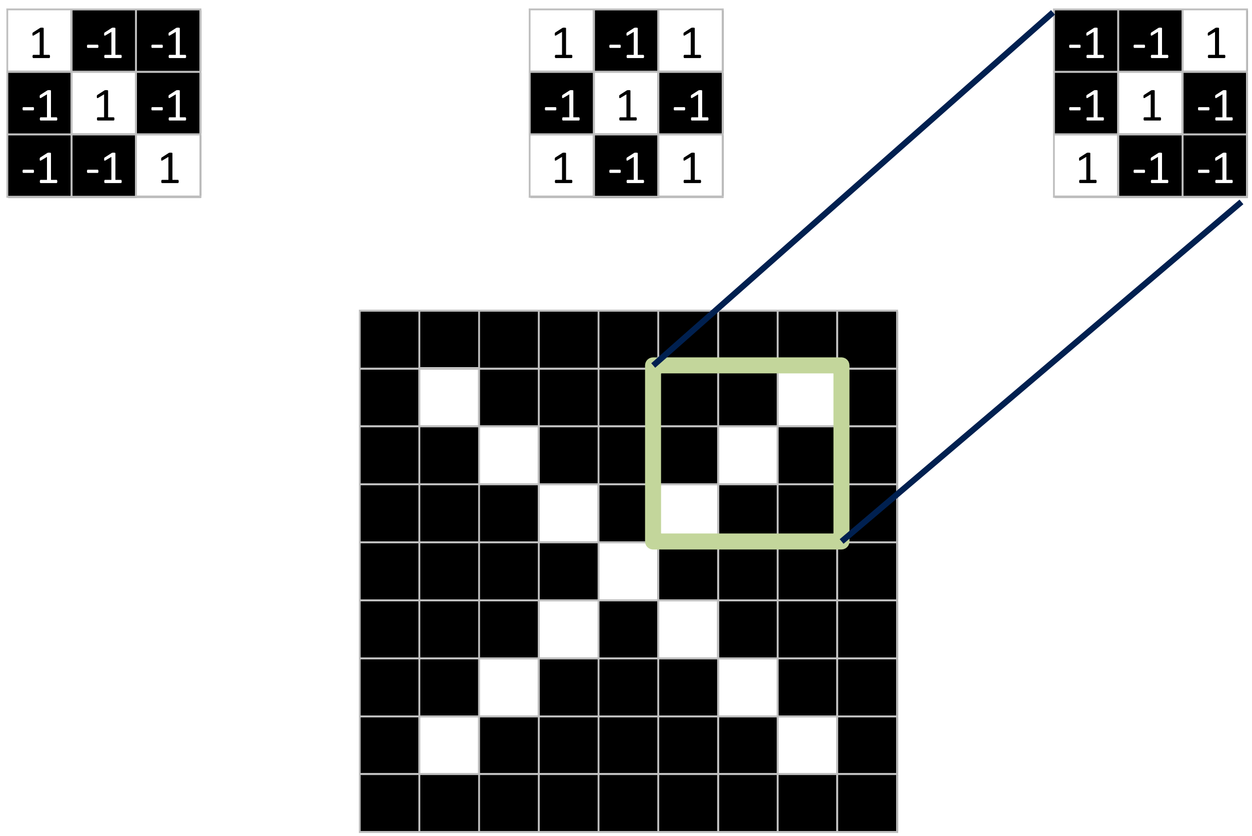
Features match pieces of the image

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

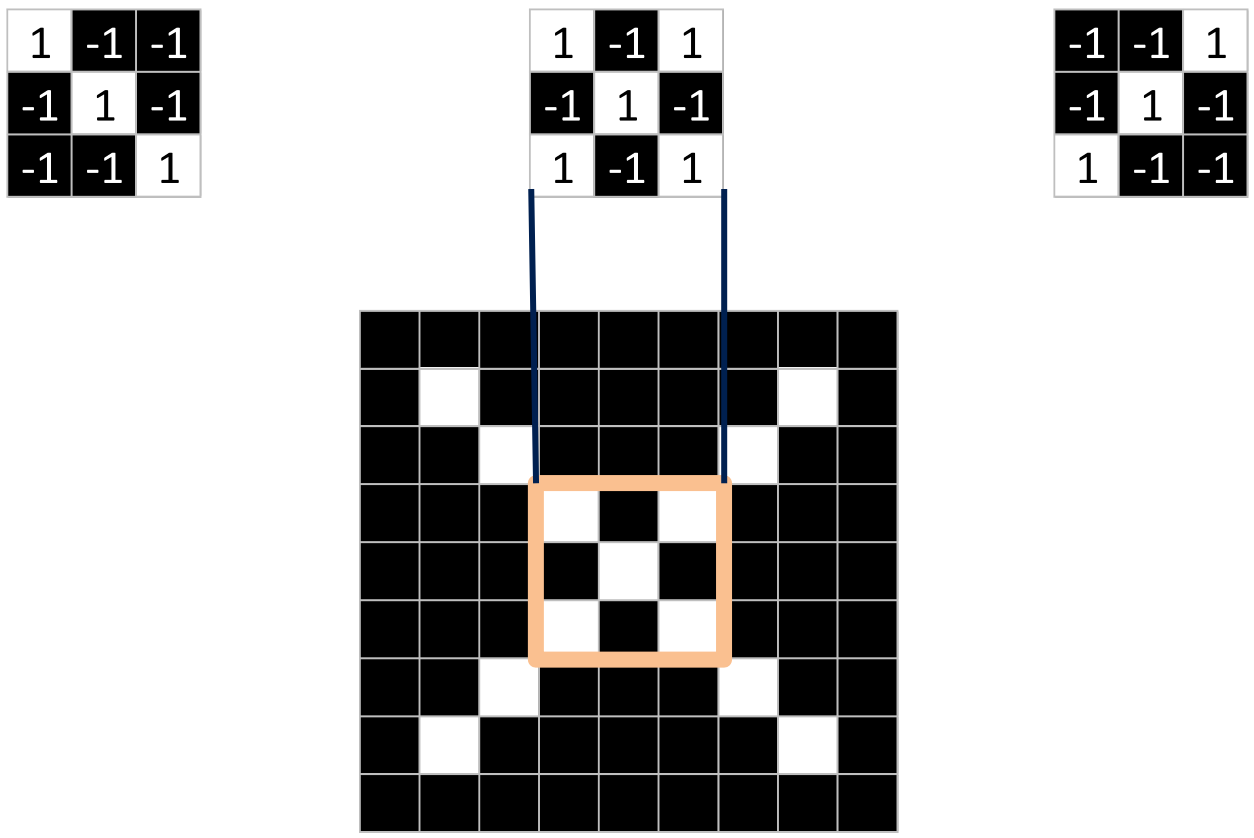
1	-1	1
-1	1	-1
1	-1	1

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

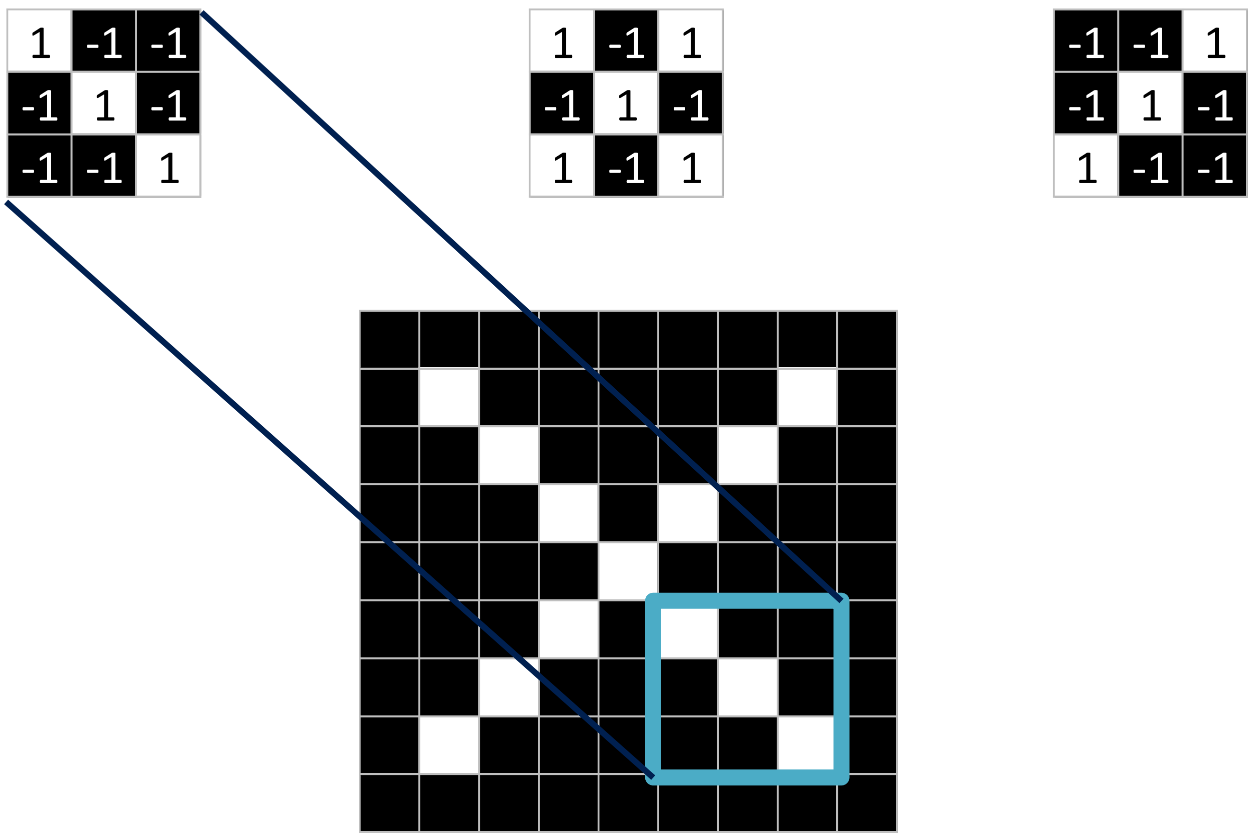

1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

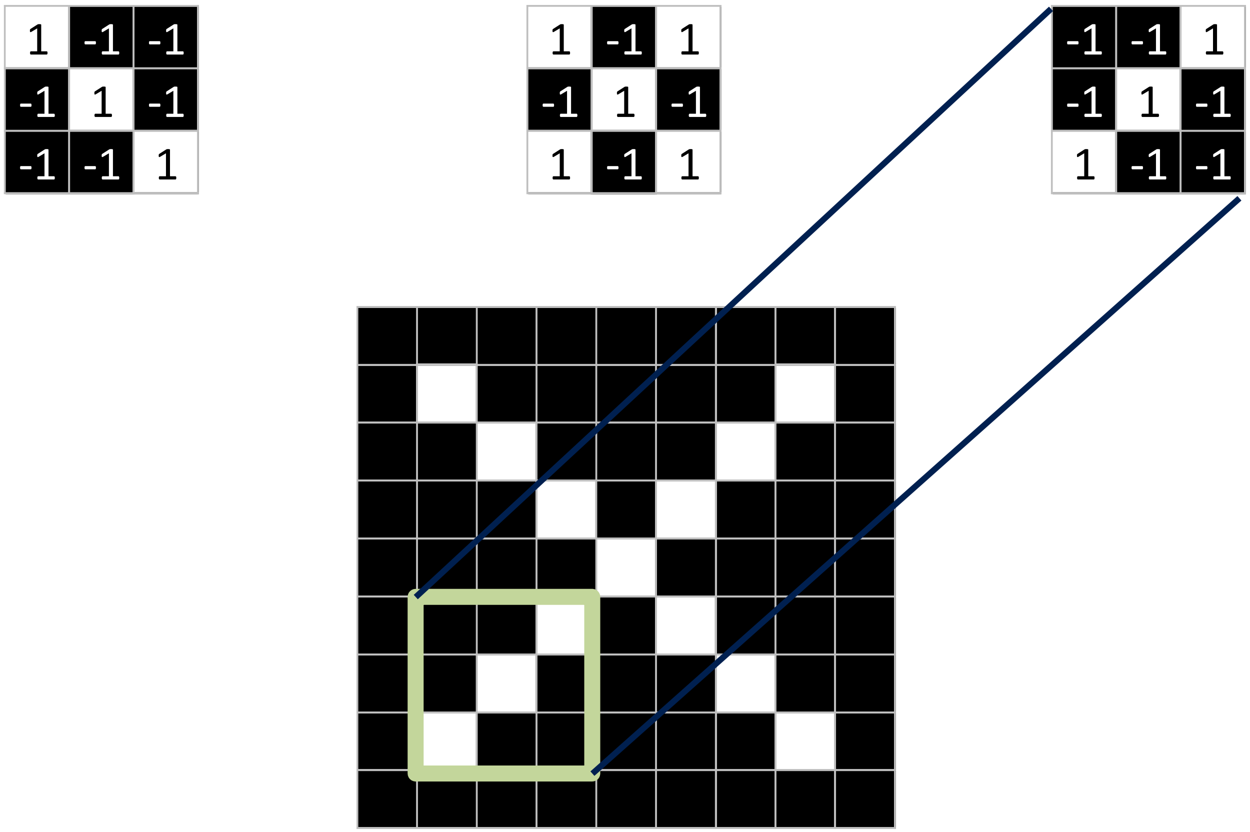


1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1



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


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


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

Filtering: The math behind the match

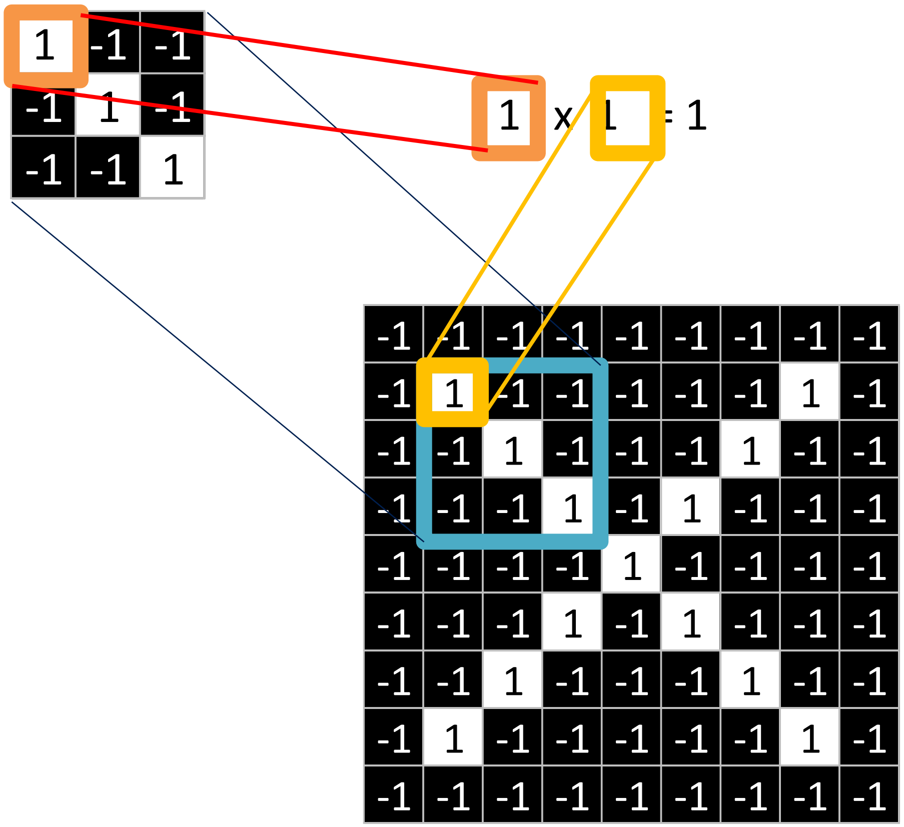
1	-1	-1
-1	1	-1
-1	-1	1

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

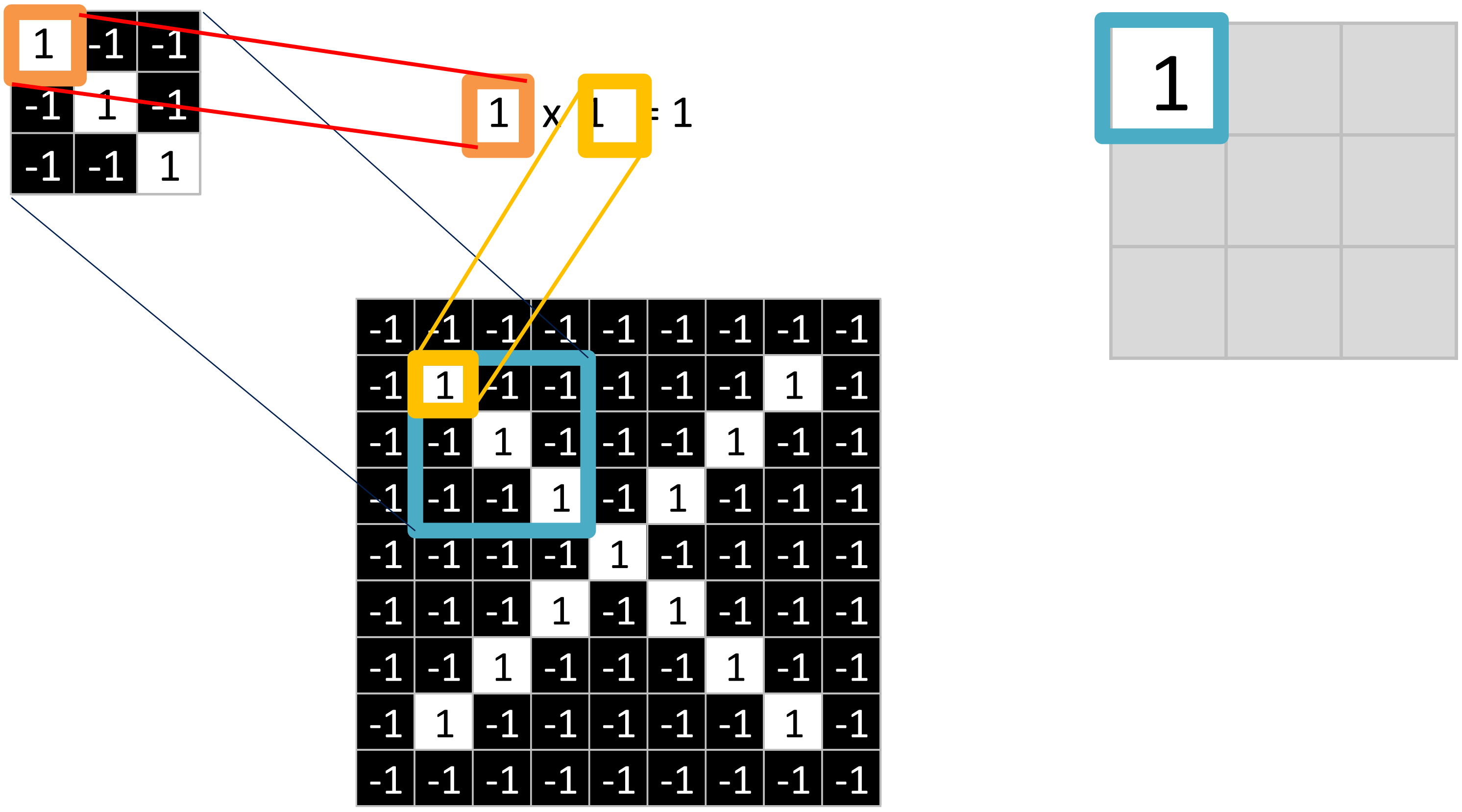
Filtering: The math behind the match

1. Line up the feature and the image patch.
2. Multiply each image pixel by the corresponding feature pixel.
3. Add them up.
4. Divide by the total number of pixels in the feature.

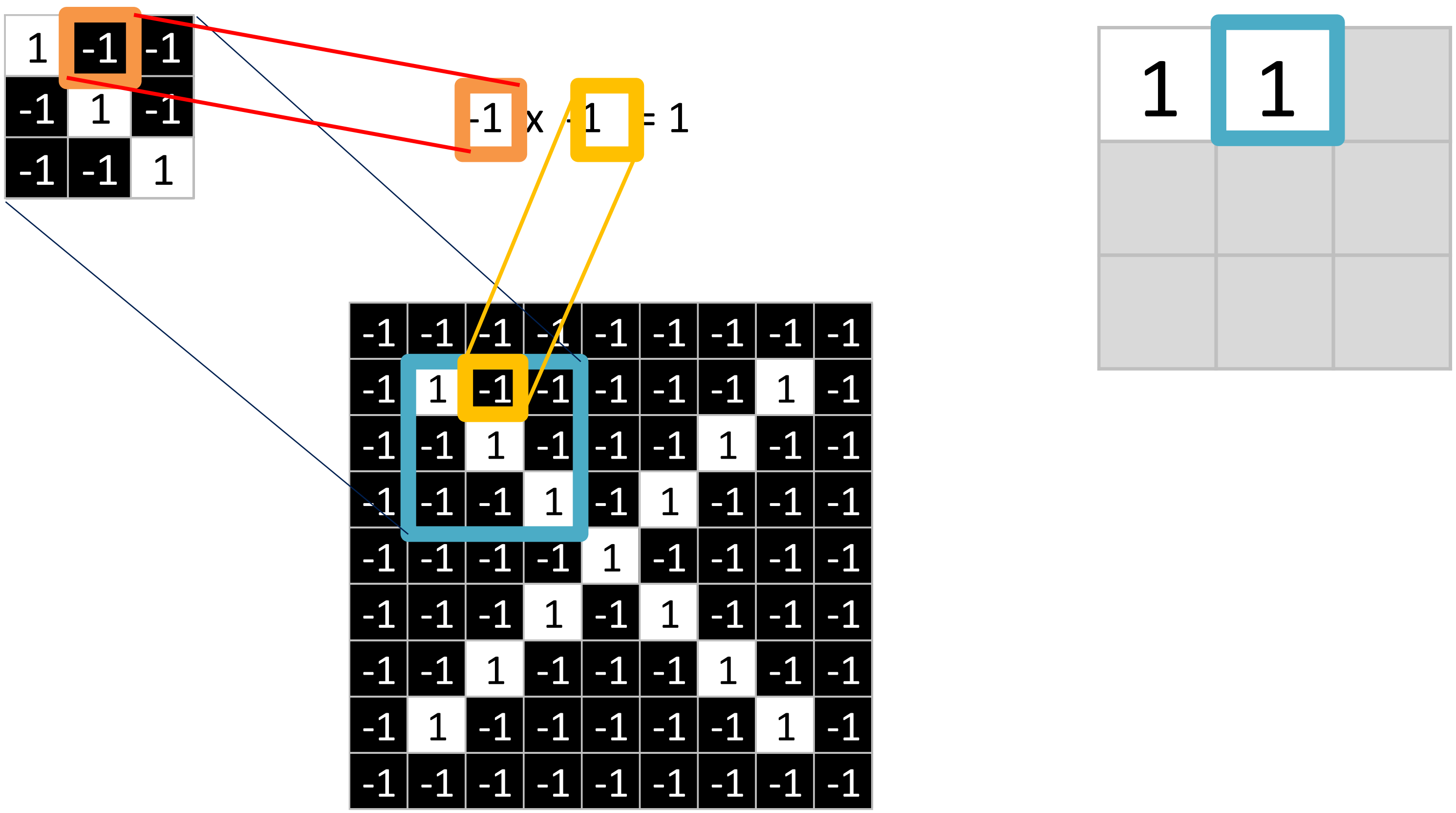
Filtering: The math behind the match



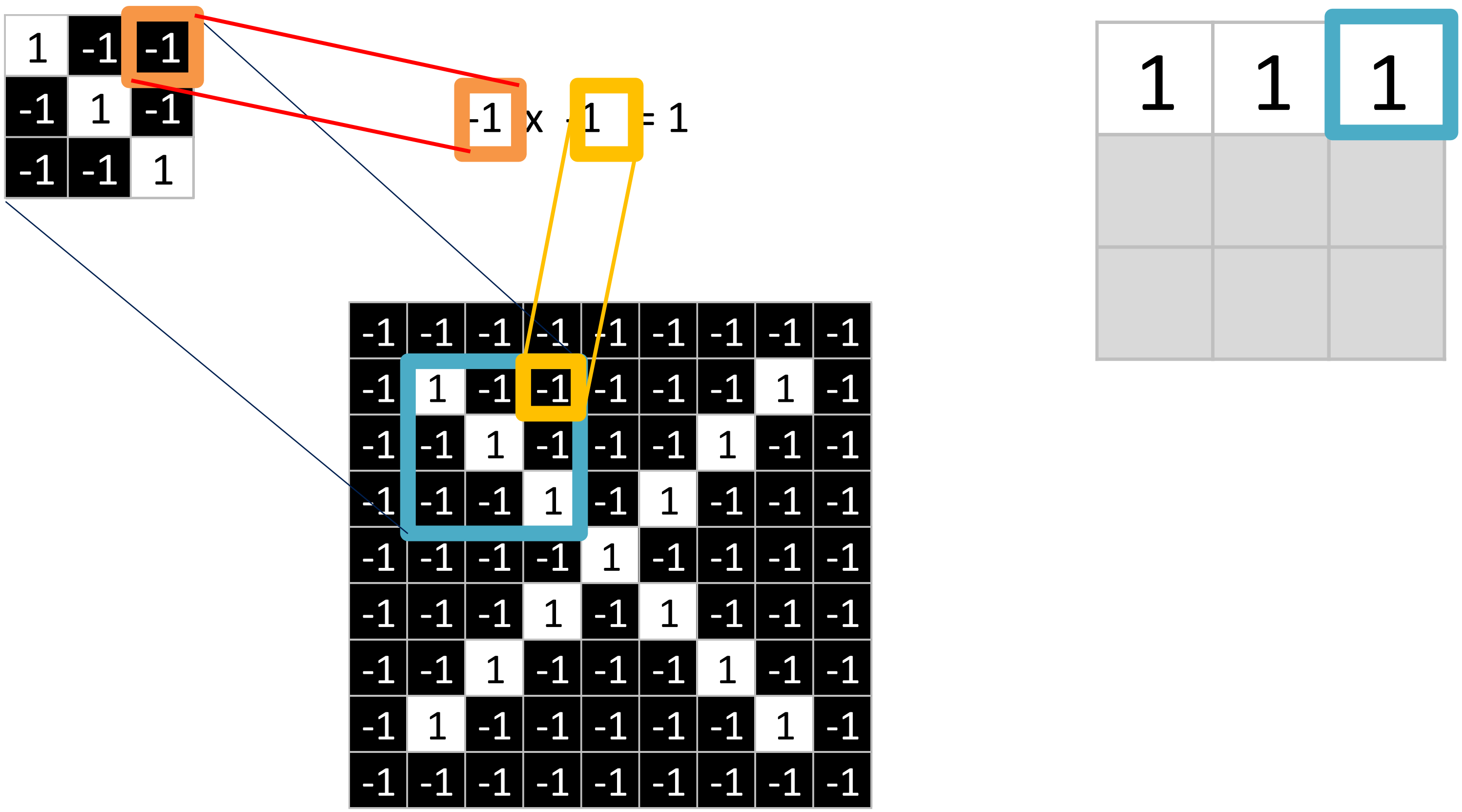
Filtering: The math behind the match



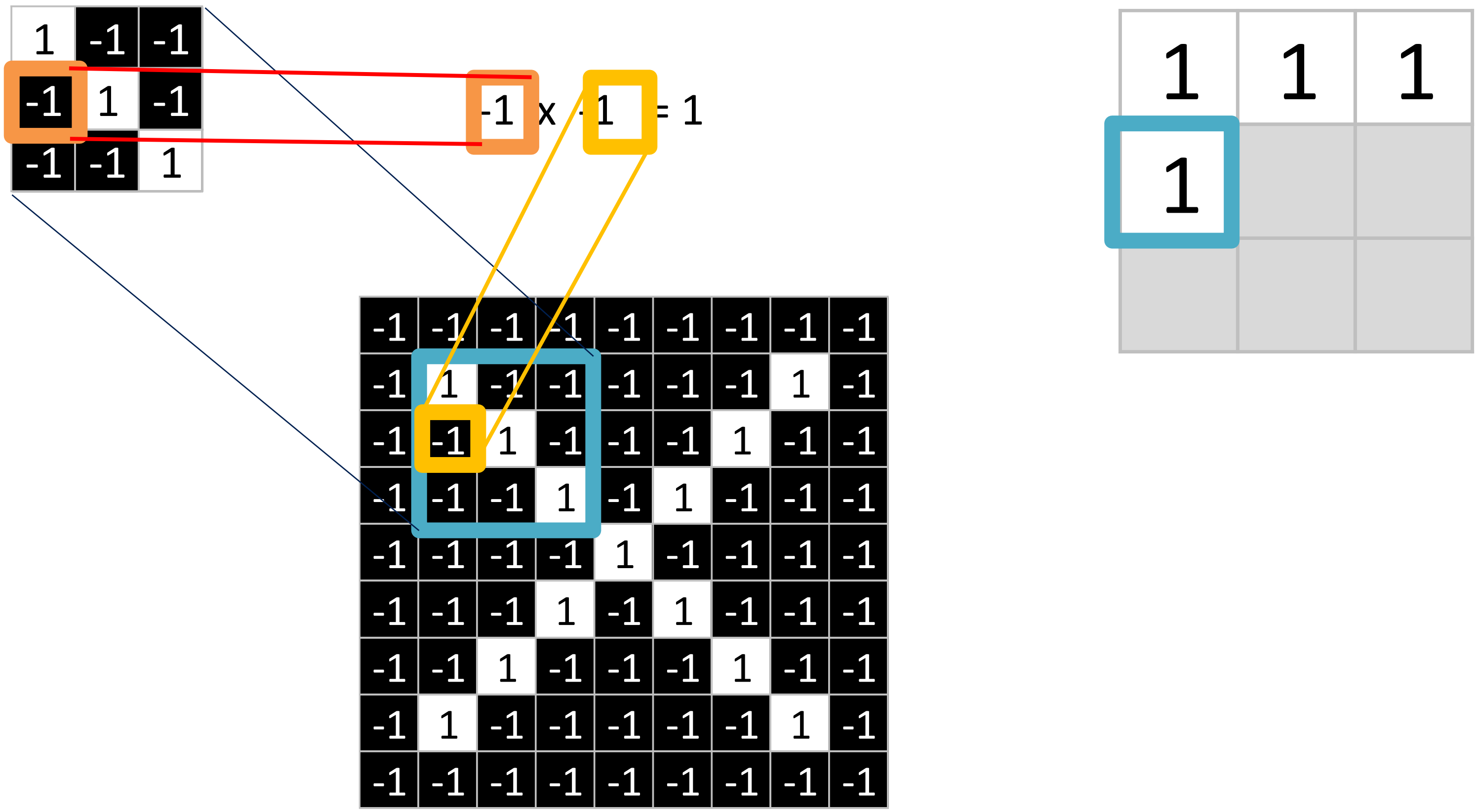
Filtering: The math behind the match



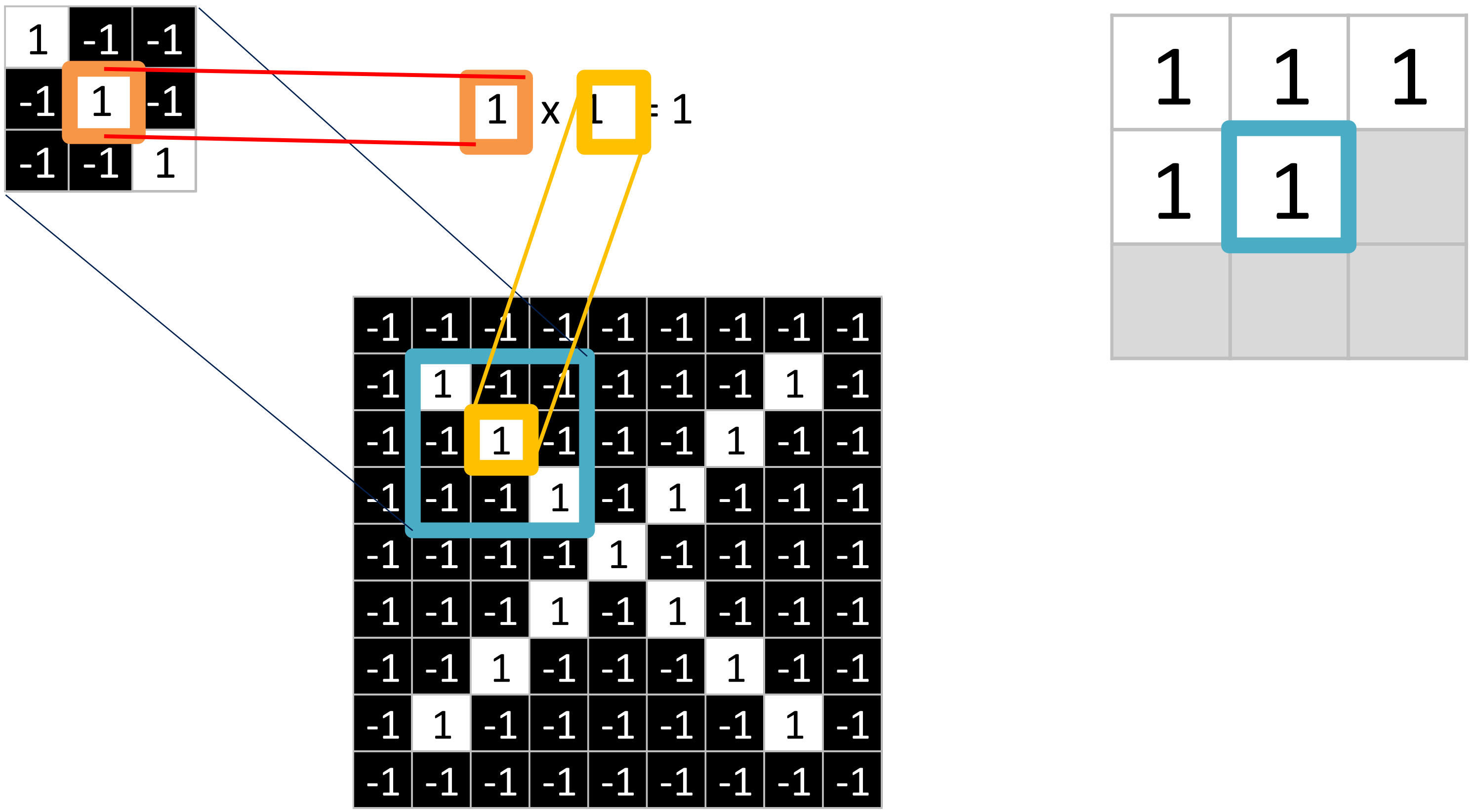
Filtering: The math behind the match



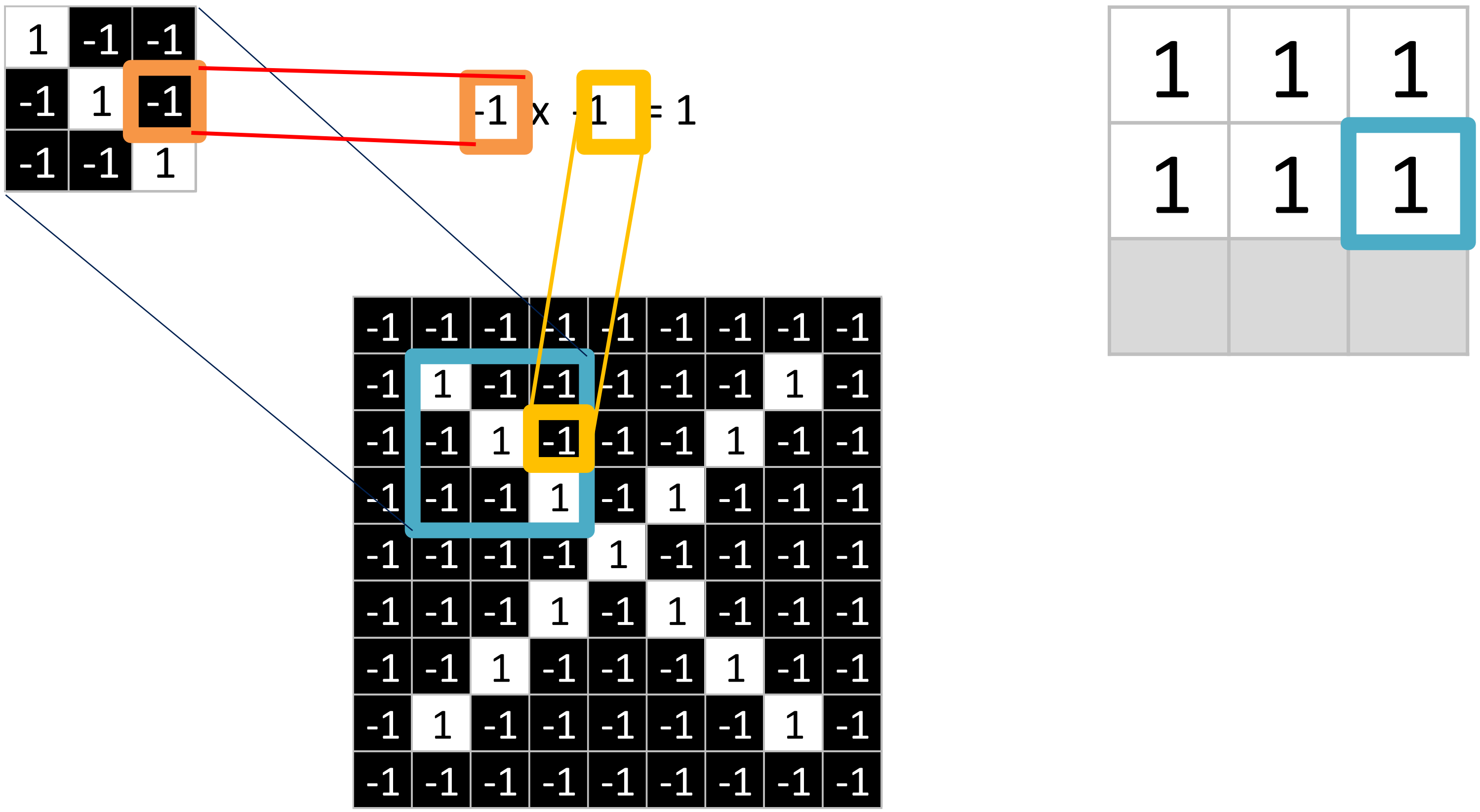
Filtering: The math behind the match



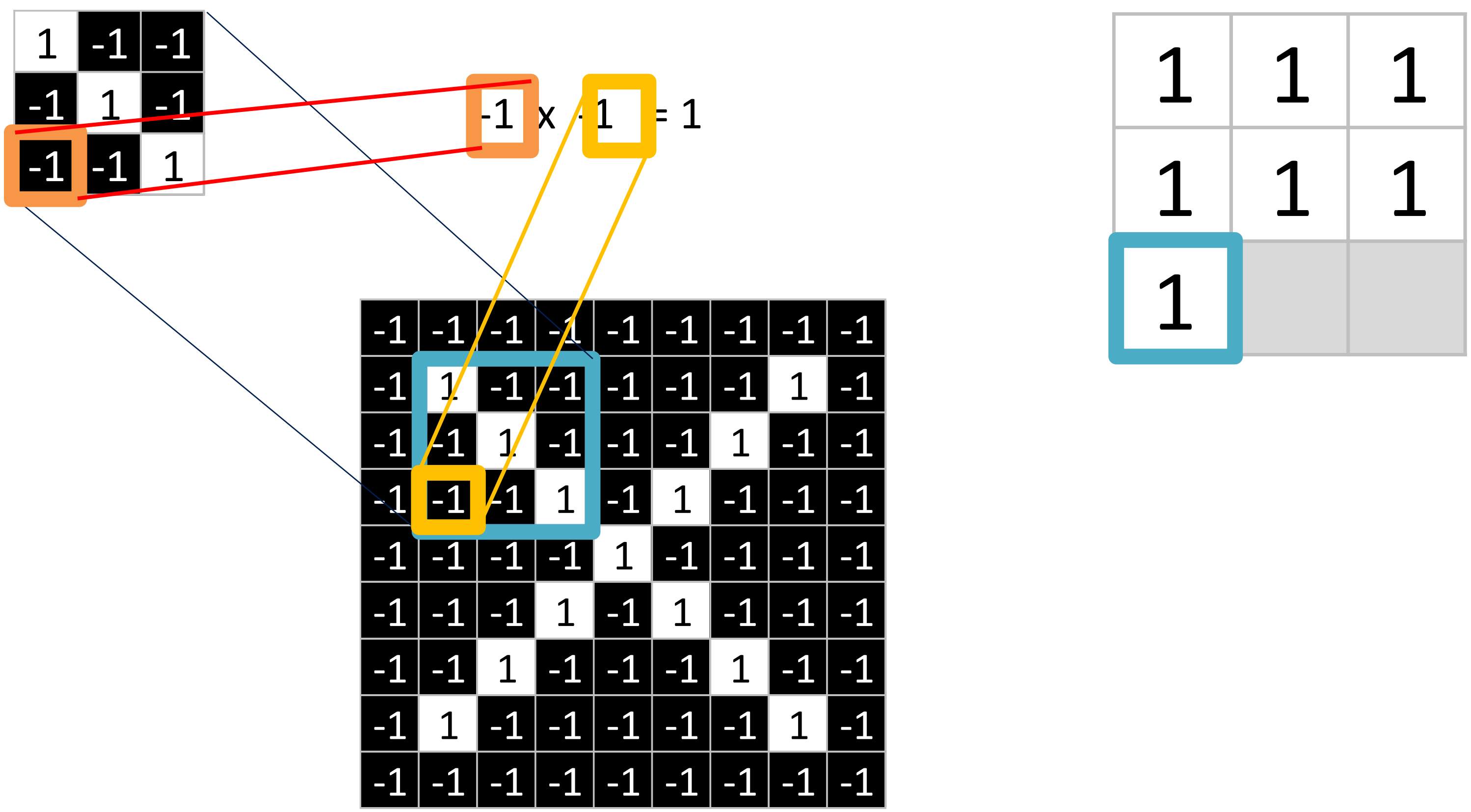
Filtering: The math behind the match



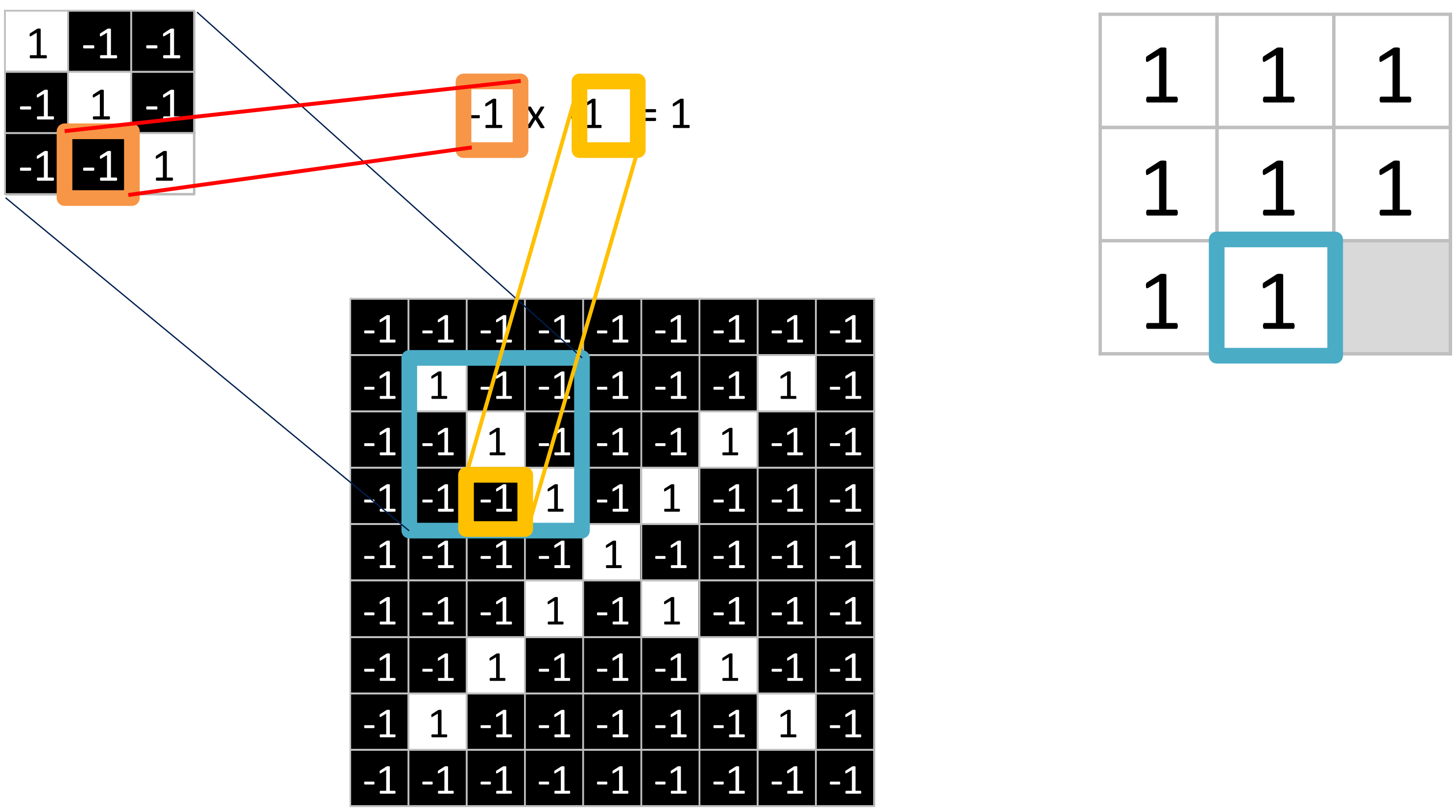
Filtering: The math behind the match



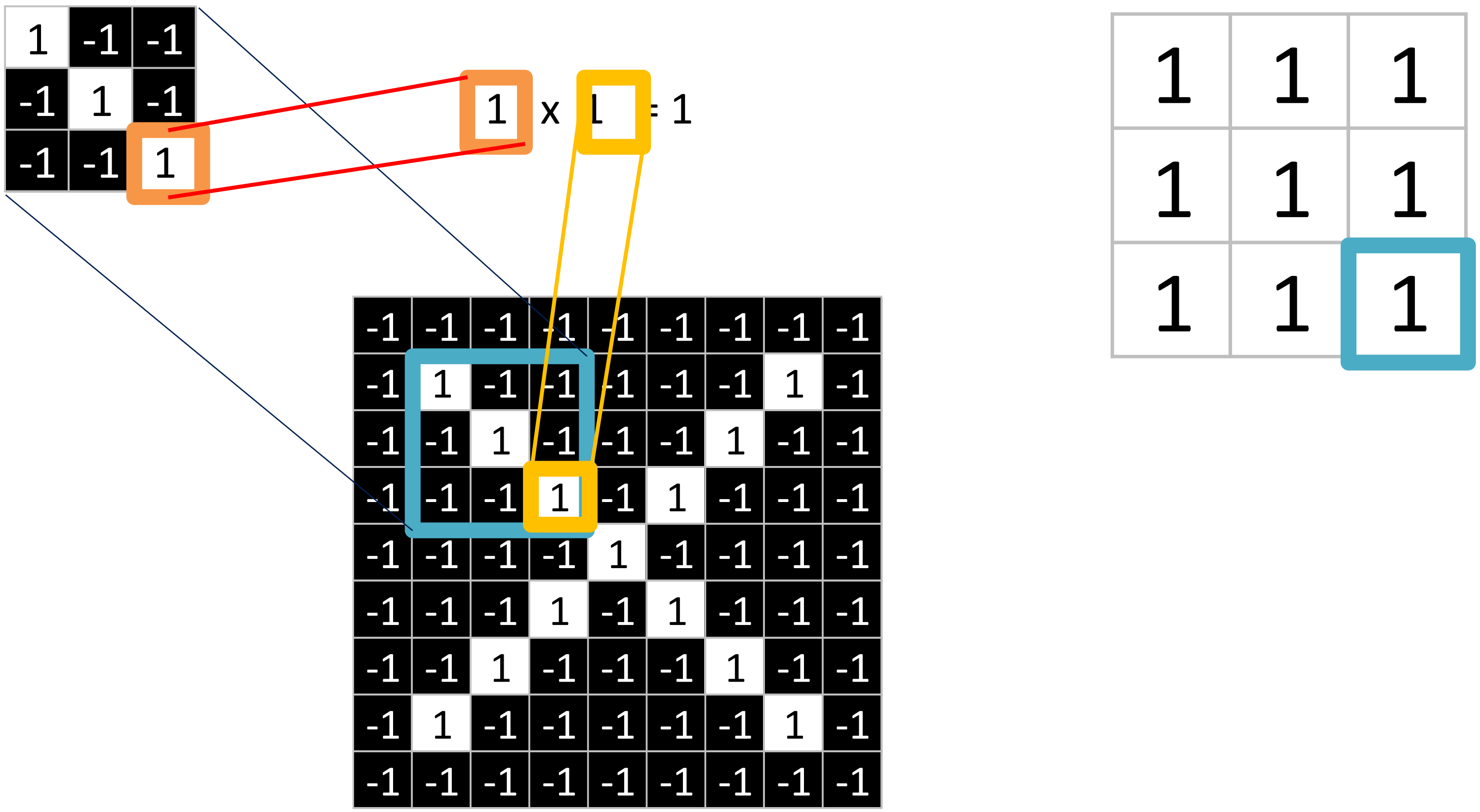
Filtering: The math behind the match



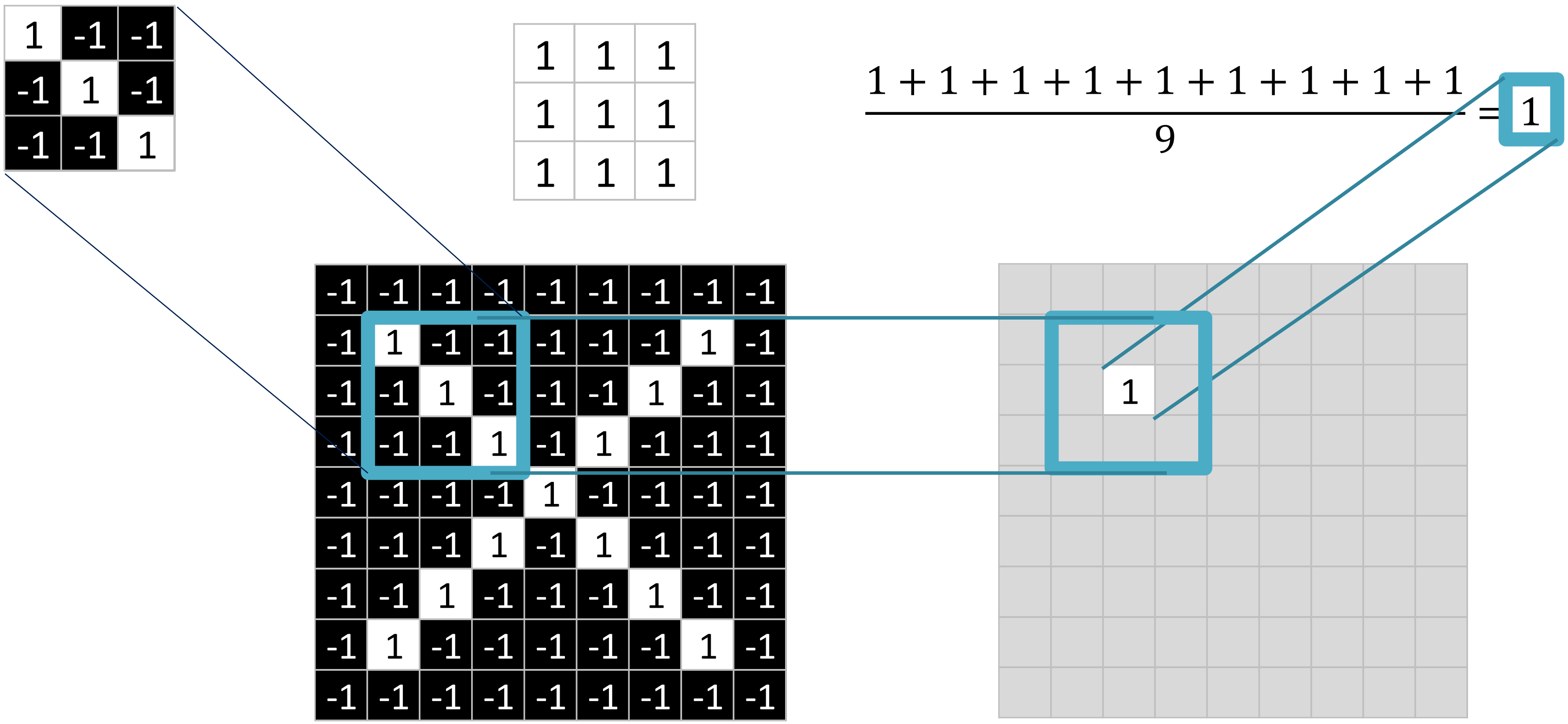
Filtering: The math behind the match



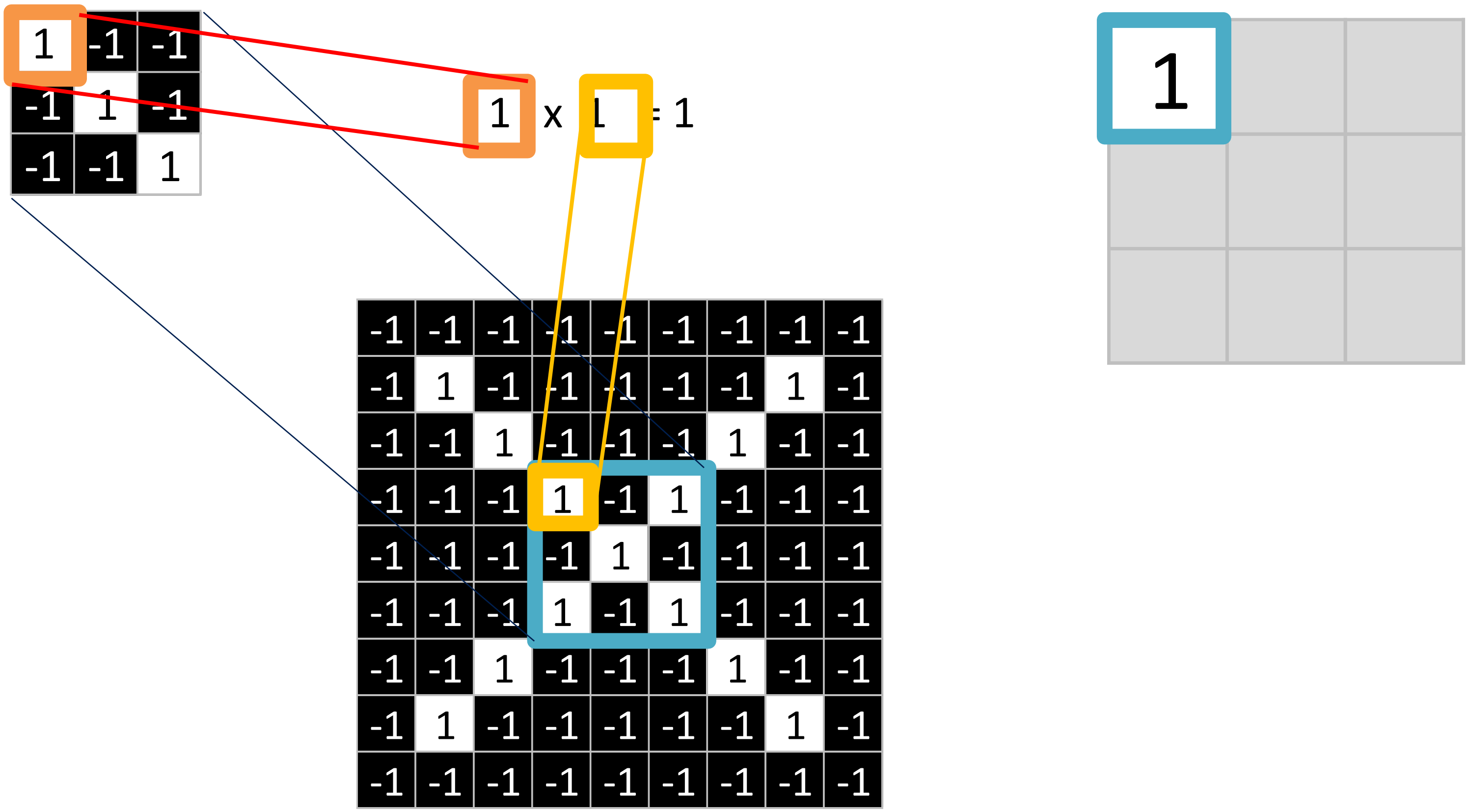
Filtering: The math behind the match



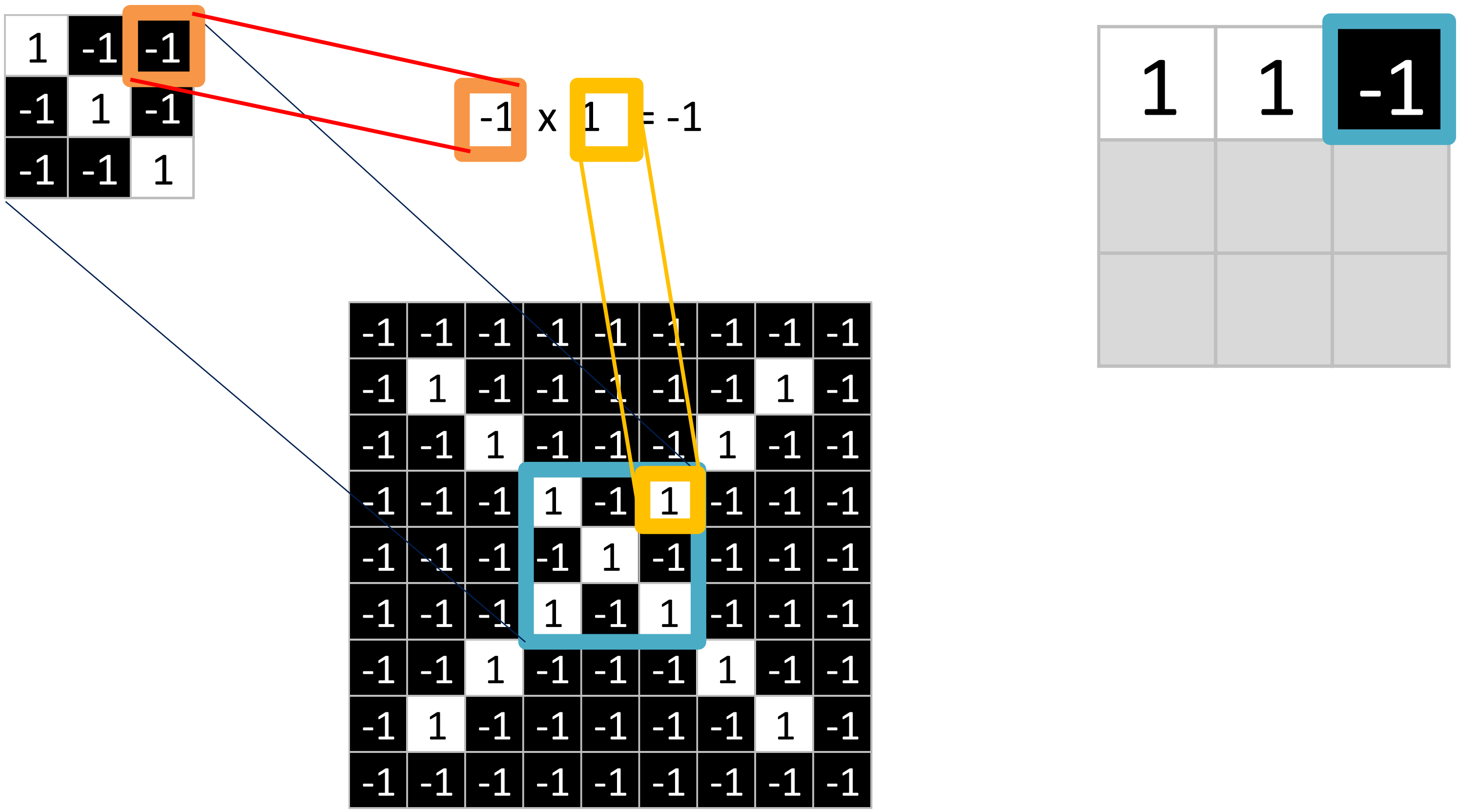
Filtering: The math behind the match



Filtering: The math behind the match



Filtering: The math behind the match



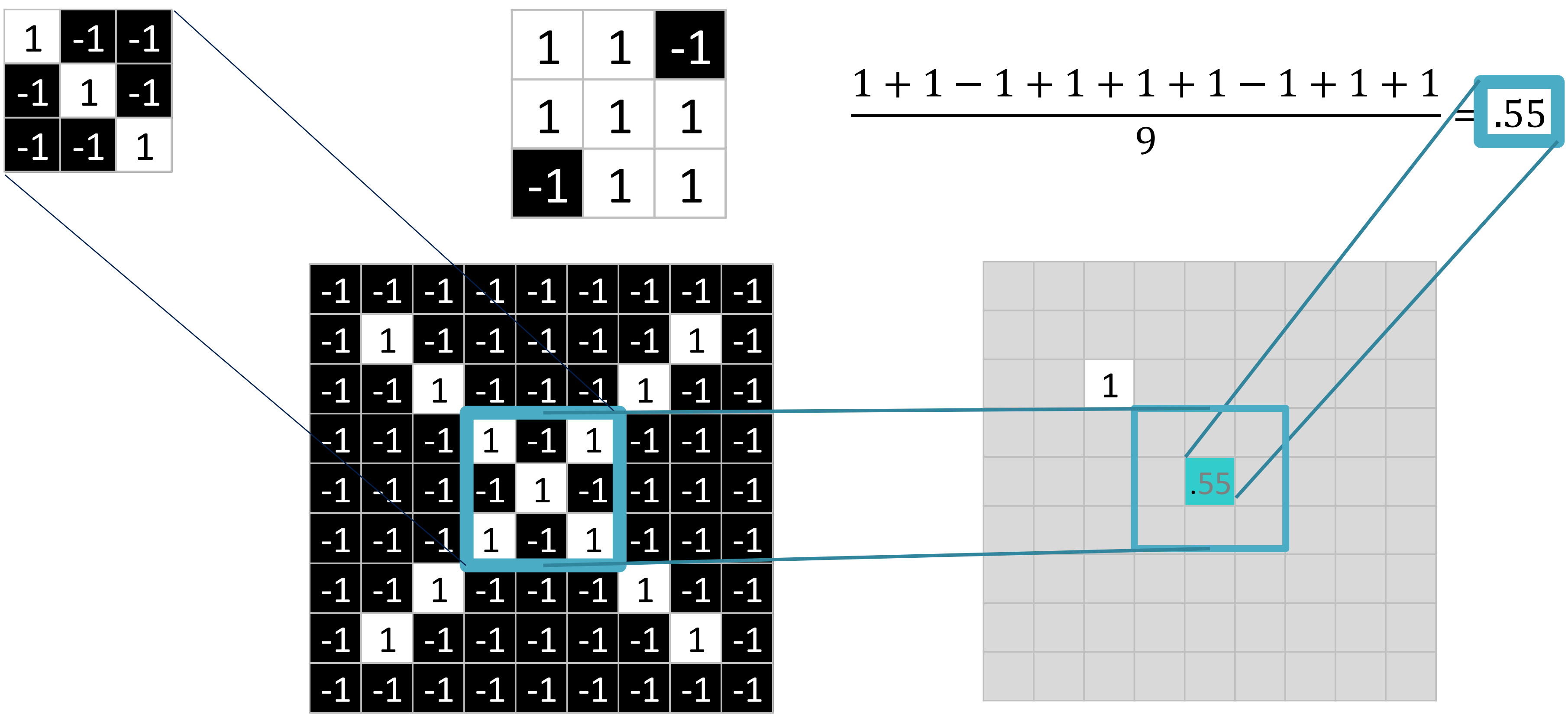
Filtering: The math behind the match

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

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

1	1	-1
1	1	1
-1	1	1

Filtering: The math behind the match



Convolution: Trying every possible match

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

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

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Convolution: Trying every possible match

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

$\otimes$

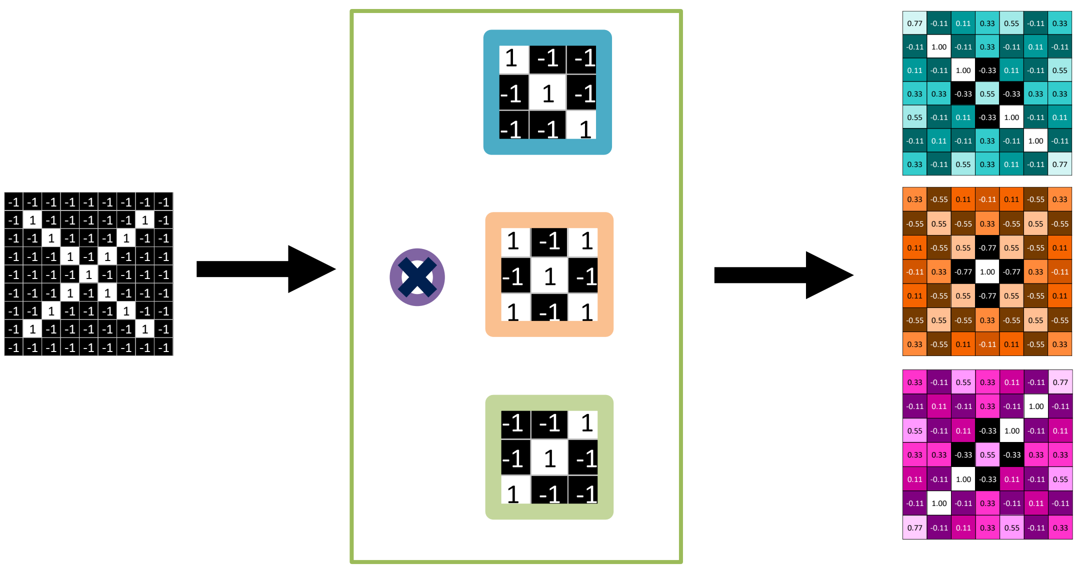
1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Convolution layer

One image becomes a stack of filtered images

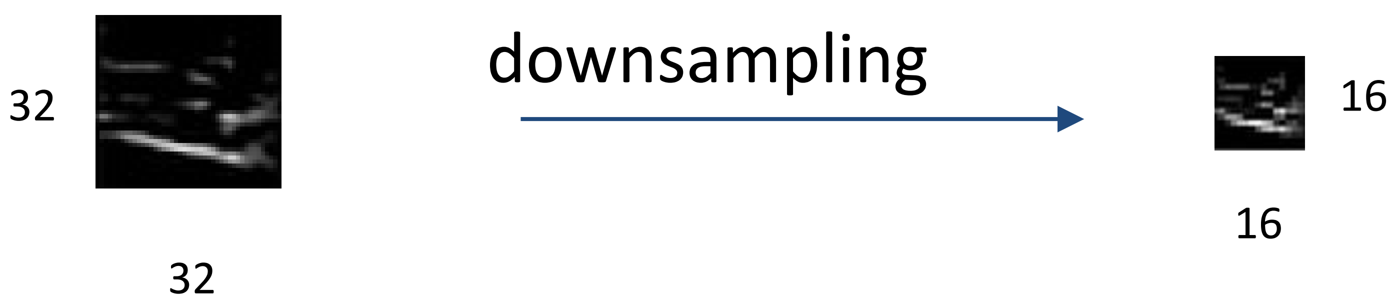


outline

- Convolutional layer (module)
  - Convolution operation
  - Filters
  - Convolution module in a network
- Pooling layer (module)

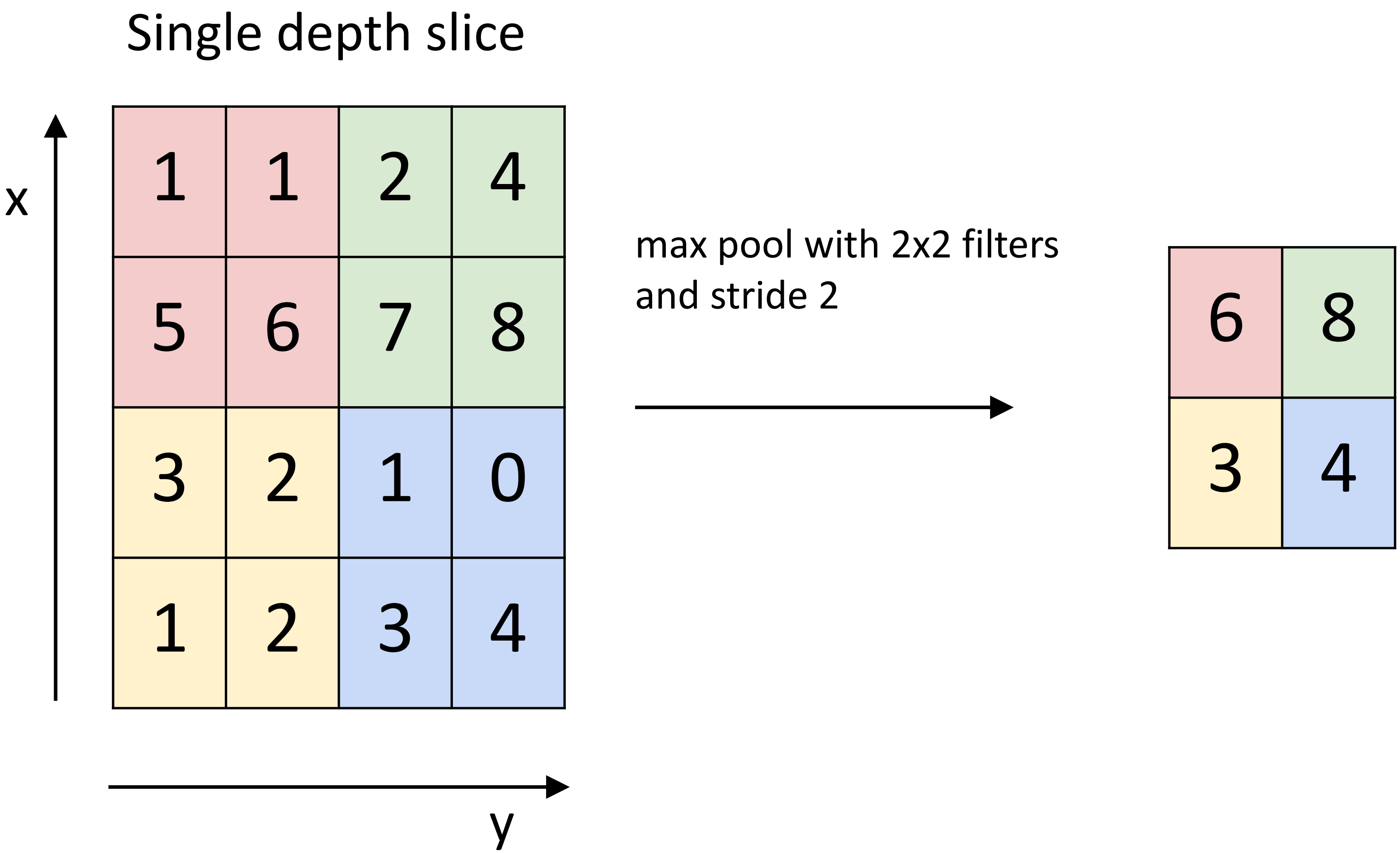
POOLING Layer

- In ConvNet architectures, **Conv** layers are often followed by **Pooling** layers
  - makes the representations smaller and more manageable without losing too much information.
  - Invariant in region.



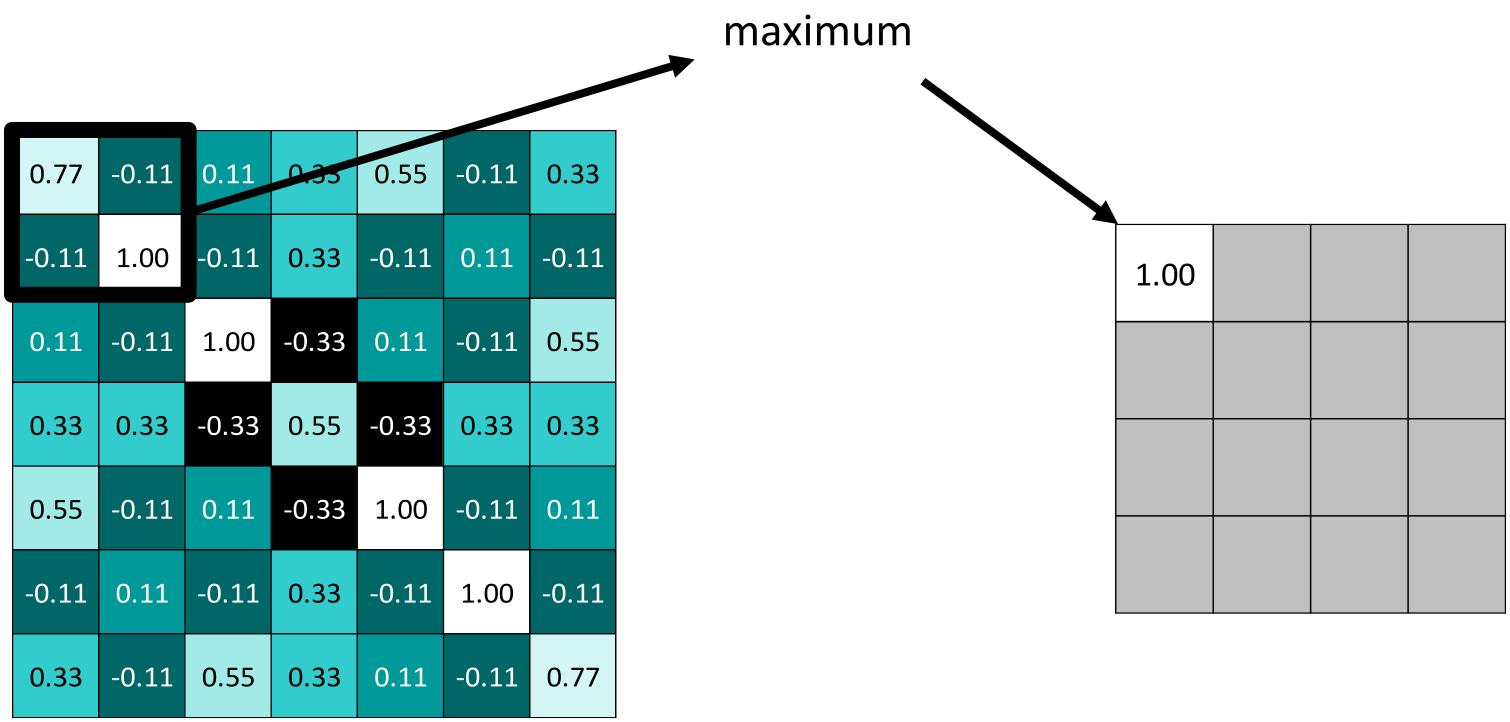
Source: Andrej Karpathy & Fei-Fei Li

MAX POOLING

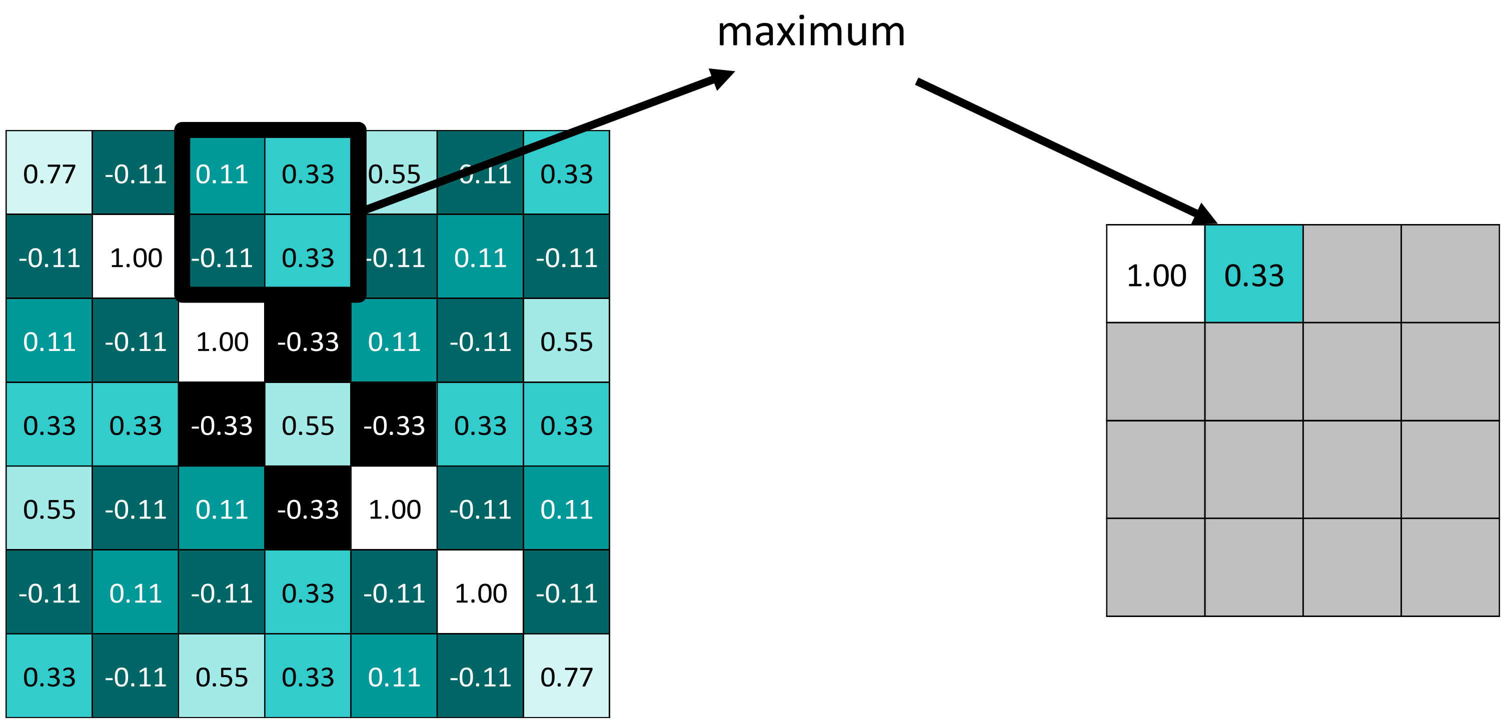


Source: Andrej Karpathy & Fei-Fei Li

Max Pooling


maximum
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
1.00

Max Pooling


maximum
0.77 -0.11 0.11 0.33 0.55 0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
1.00 0.33

Max Pooling

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

maximum

1.00	0.33	0.55	

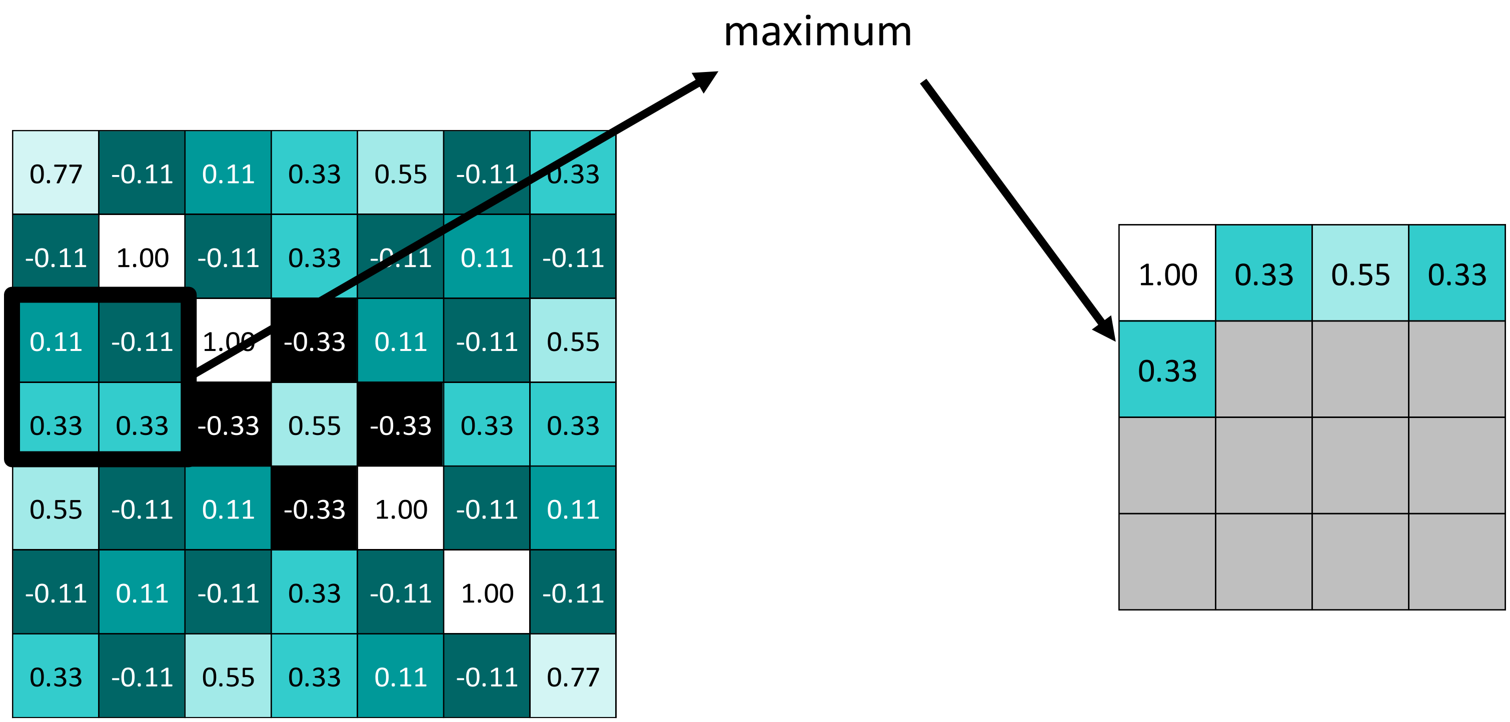
Max Pooling

maximum

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33

Max Pooling


maximum
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
1.00 0.33 0.55 0.33
0.33

Max Pooling

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

max pooling

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

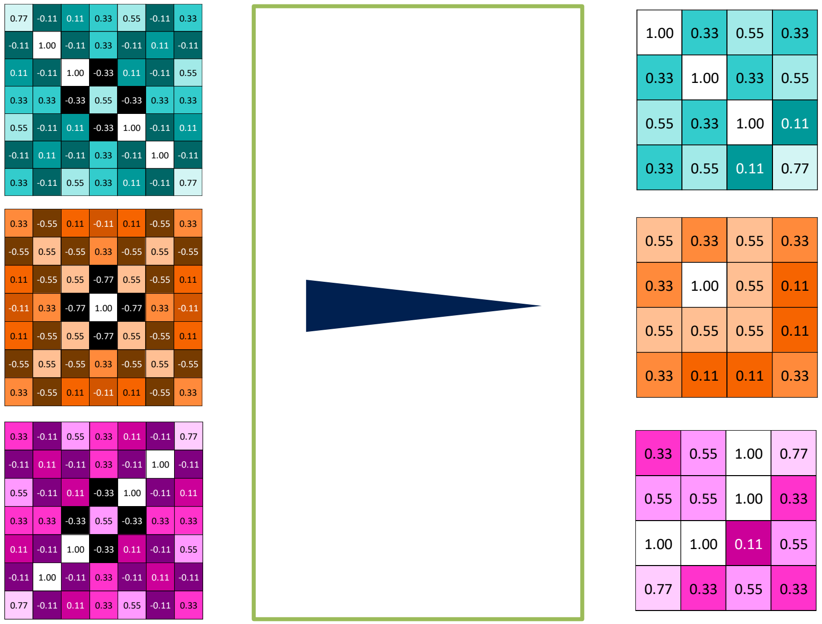
0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

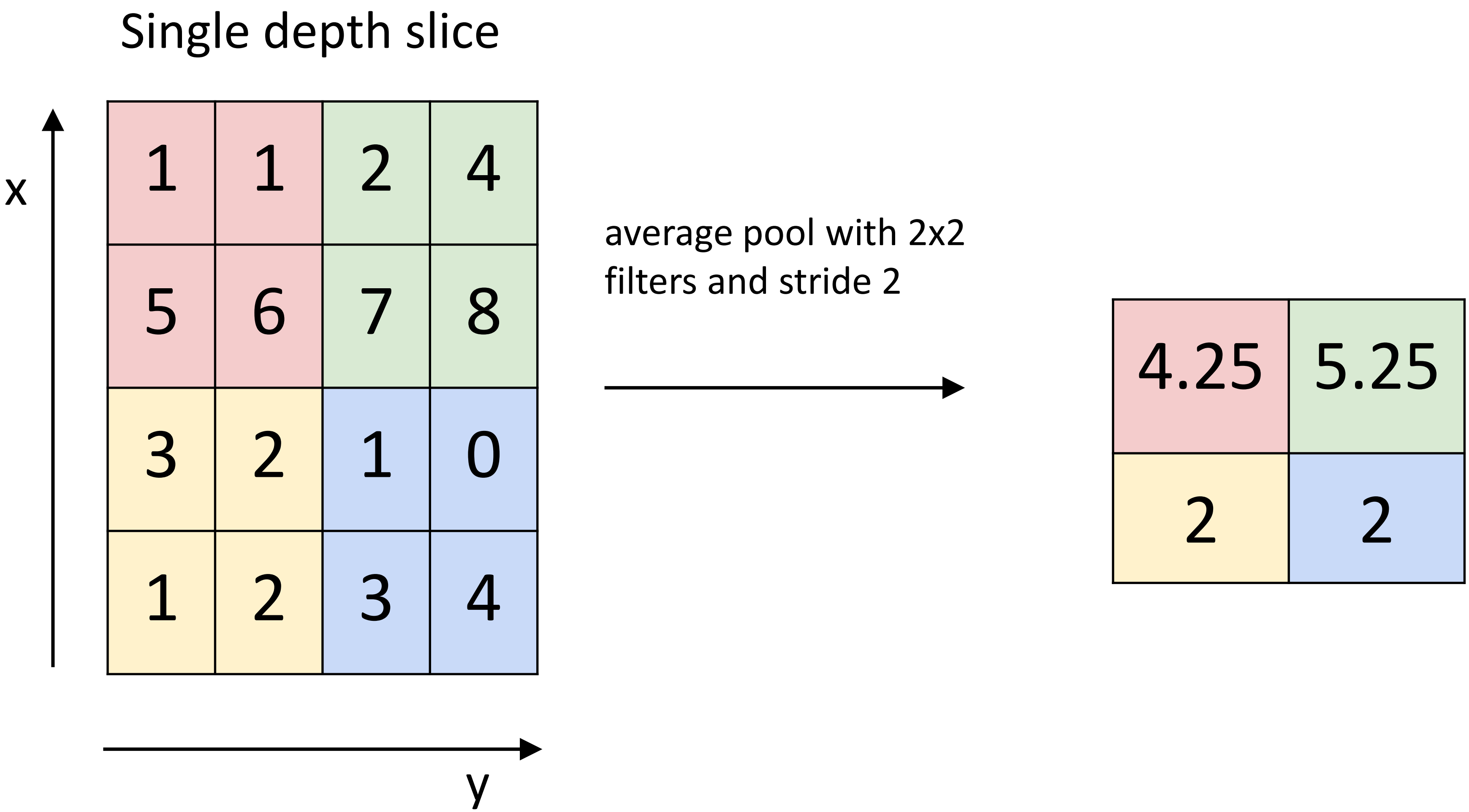
0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

Max Pooling layer

A stack of images becomes a stack of smaller images.

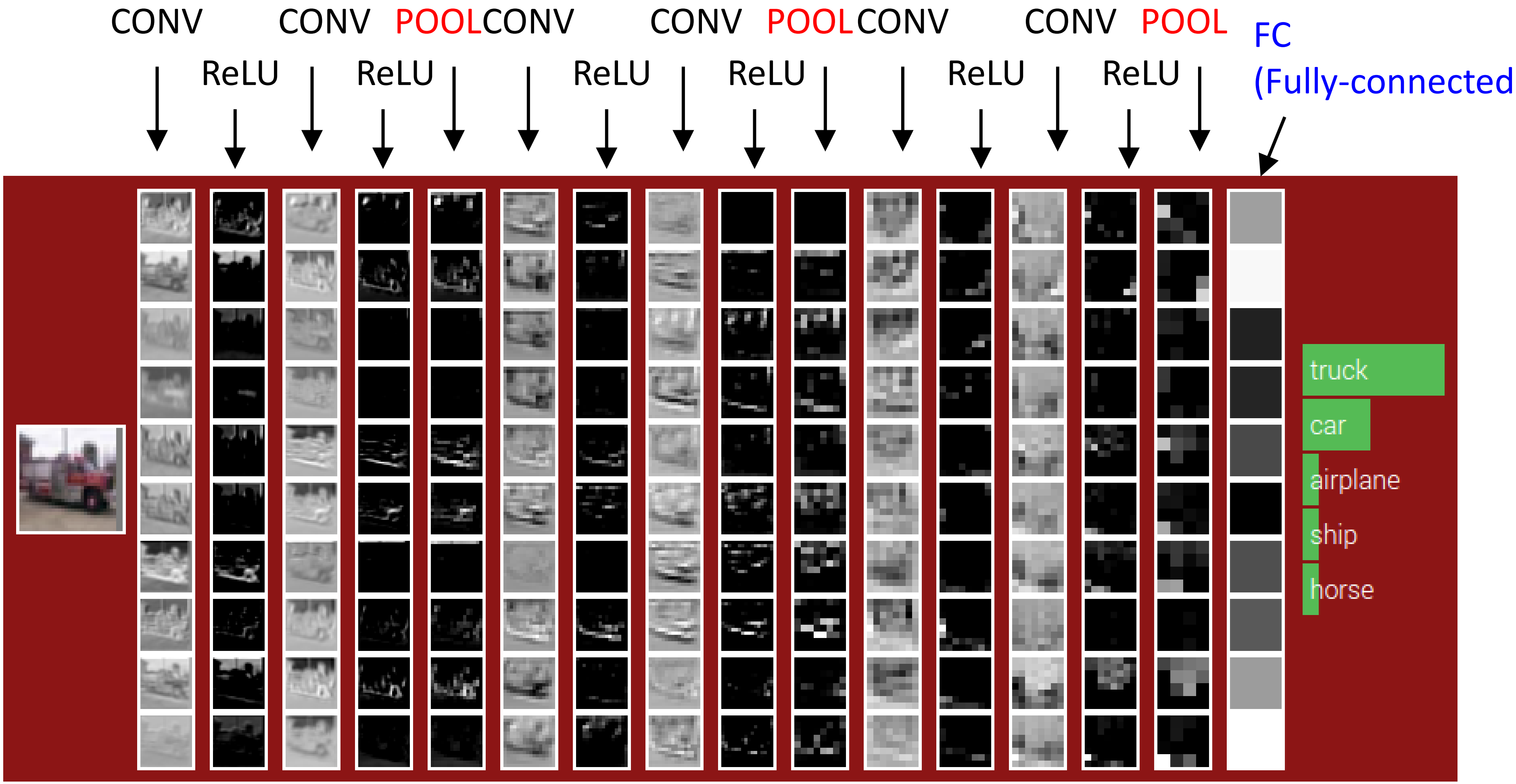


Average POOLING



Source: Andrej Karpathy & Fei-

CNN: Intuitive example

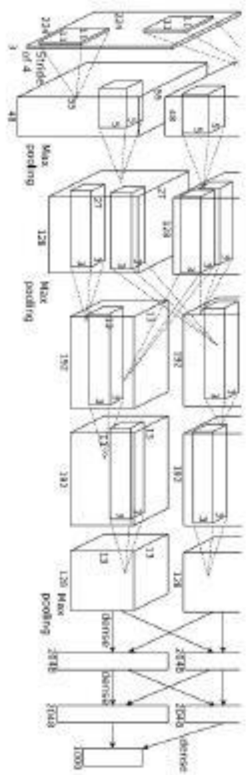


Source: Andrej Karpathy & Fei-Fei Li

Famous Net Architecture

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

Year 2014

GoogLeNet

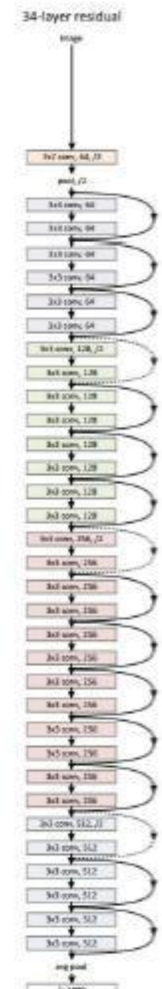
[Szegedy arxiv 2014]

VGG

[Simonyan arxiv 2014]

Year 2015

MSRA



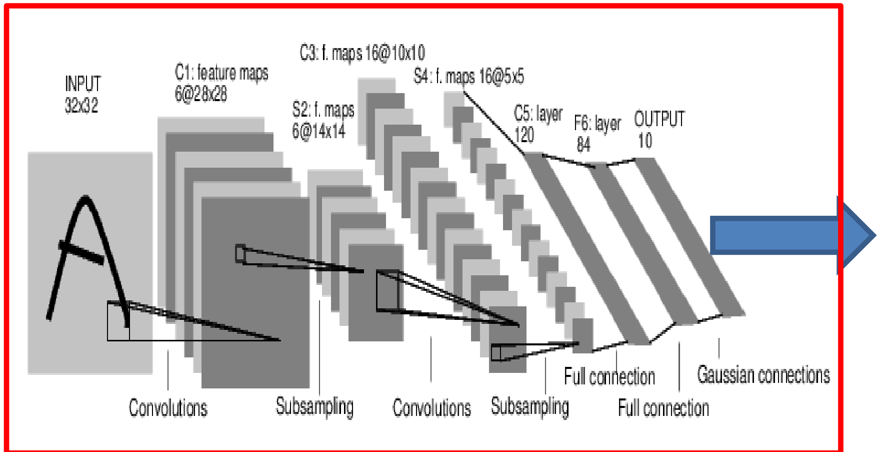
主要内容

- 多层感知机(Multi-layer Perceptron, MLP)
  - 反向传播算法(Back-propagation)
- 卷积神经网络 （Convolutional Neural Network, CNN）
  - 卷积操作和卷积层
  - 池化 （Pooling）
- 循环神经网络（Recurrent Neural Network, RNN）
  - 建模和训练
  - LSTM模型

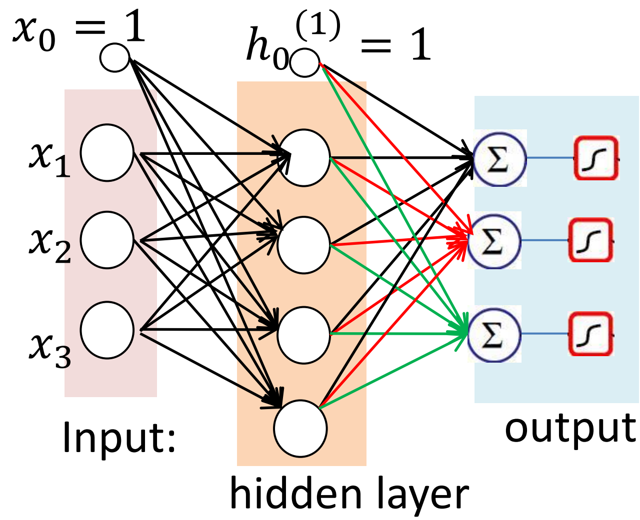
outline

- Recurrent Neural Network
  - Modeling
  - Training
- Long Short Term Memory (LSTM)
  - Motivation
  - Modeling
- Application
  - Generate article

Classification: MLP and CNN

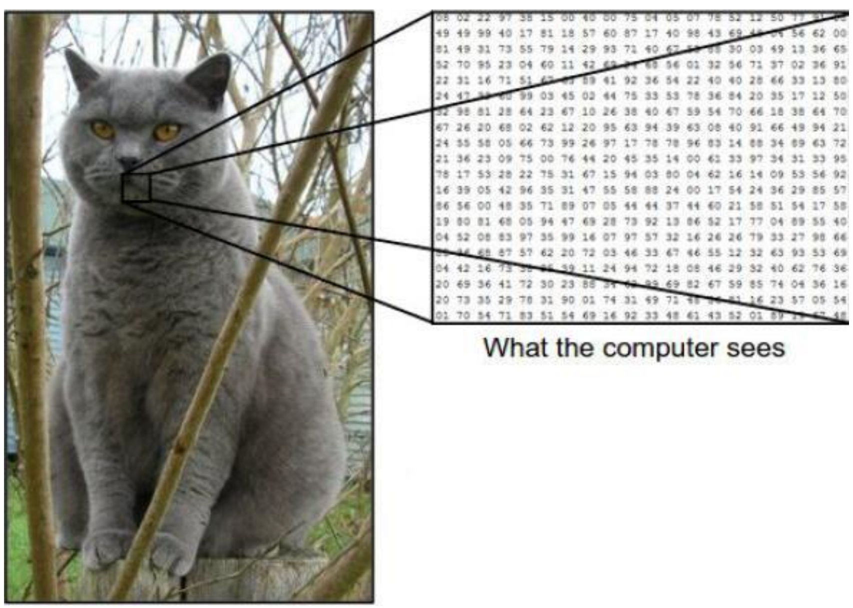


Convolution (卷积)



$(1, 0, 0)^T$

(x_1 , x_2 , x_3)

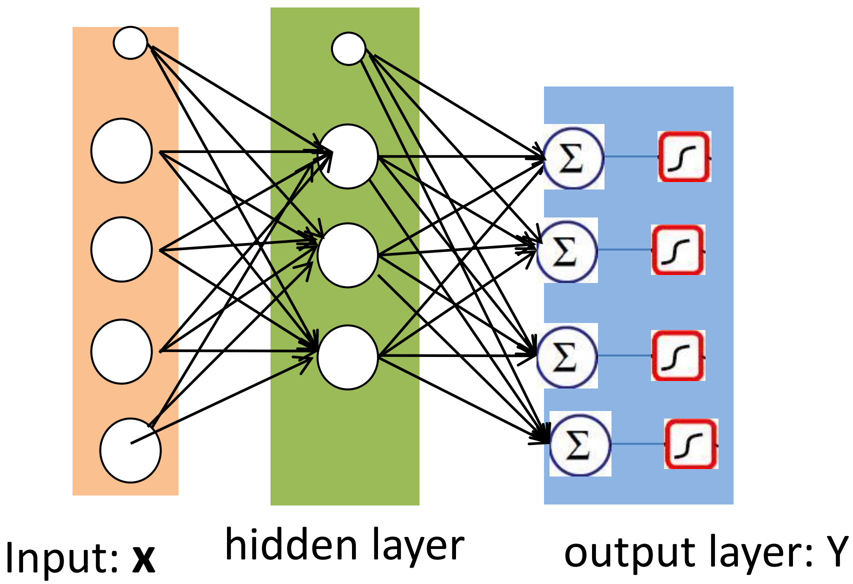


I go to cinema, and

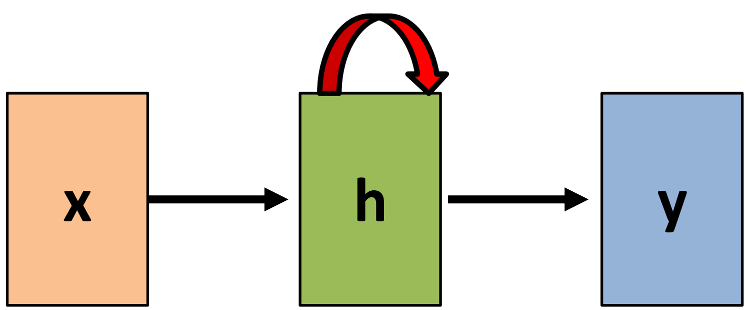
I book a ticket

One example-modeling: motivation

- **Task**: **Character-level language model**
  - **example**
    - Vocabulary [h,e,l,o]
    - Training sequence “hello”

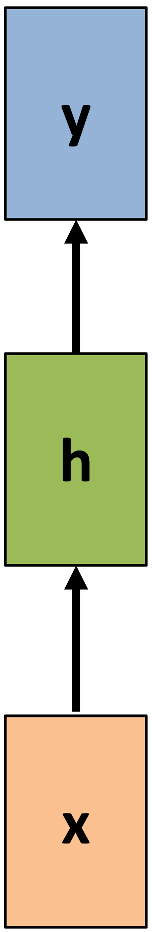


- Data representation：
  - X: {h, e, l, l}
  - Y: {e, l , l, o}



Modeling

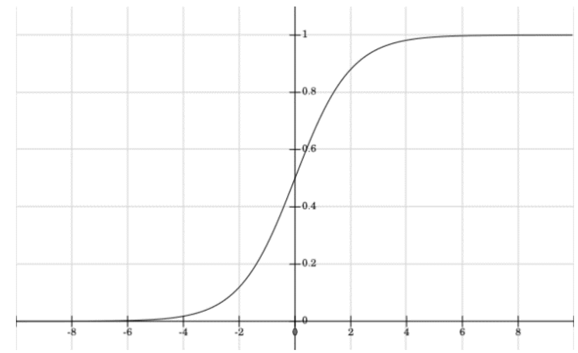
- MLP → RNN



$$\boldsymbol{y} = \boldsymbol{W_{yh}h}$$

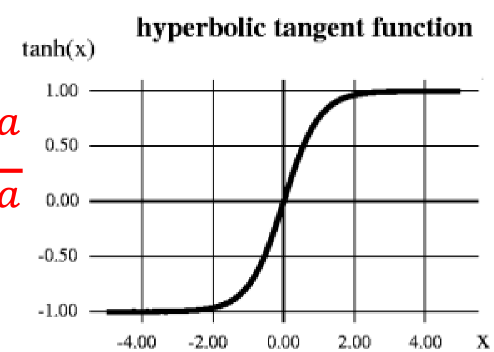
$$\boldsymbol{y} = \sigma(\boldsymbol{W_{yh}h})$$

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



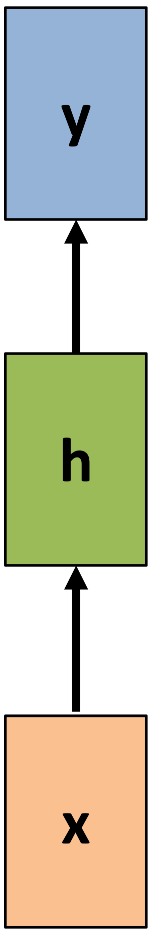
$$\boldsymbol{h} = \sigma(\boldsymbol{W_{hx}x})$$

$$\tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$



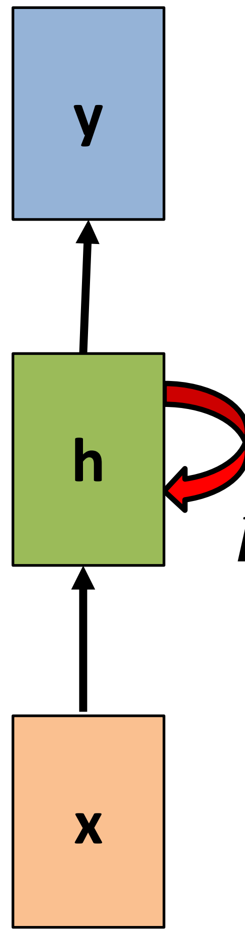
Modeling

- MLP → RNN



$$y = \boldsymbol{W}_{yh}\boldsymbol{h}$$

$$\boldsymbol{h} = tanh(\boldsymbol{W}_{hx}\boldsymbol{x})$$

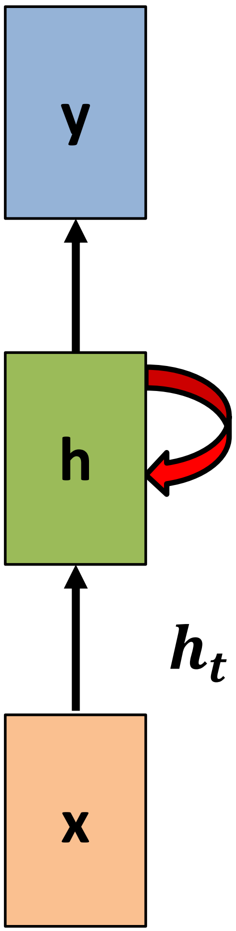


$$y_t = \boldsymbol{W}_{yh}\boldsymbol{h}_t$$

$$\boldsymbol{h}_t = tanh(\boldsymbol{W}_{hx}\boldsymbol{x} + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1})$$

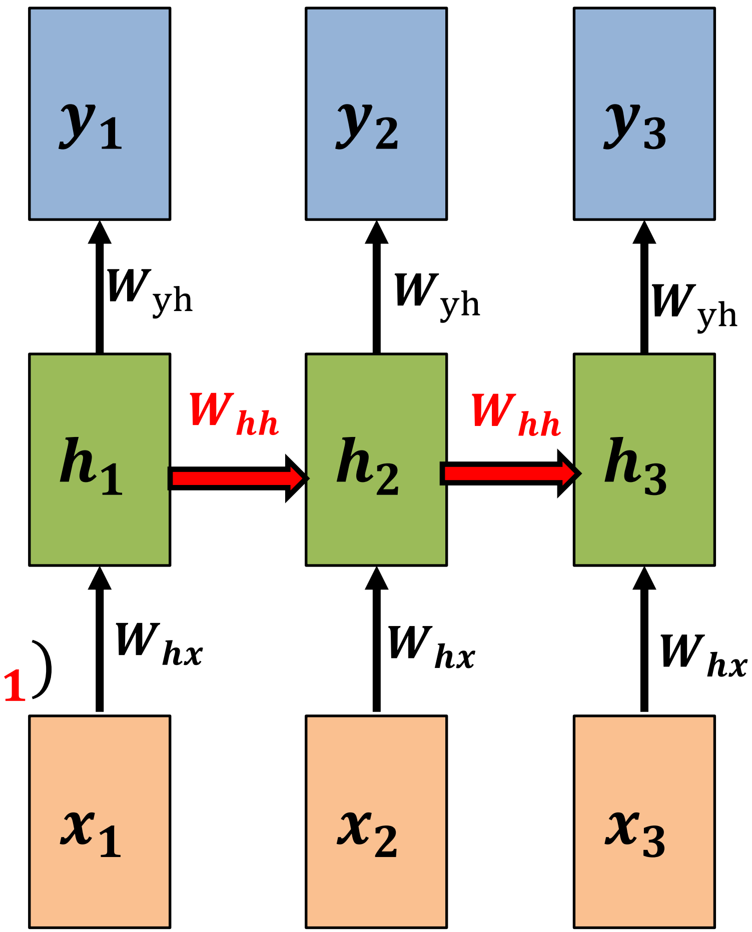
Modeling

- RNN-unrolling(展开)



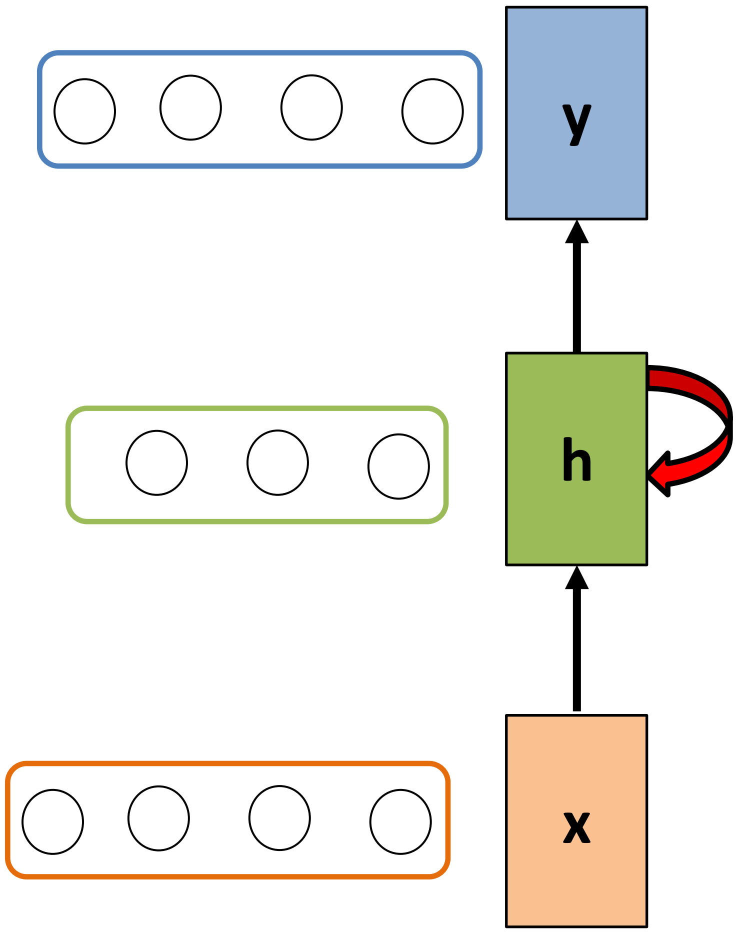
$$y_t = W_{yh}h_t$$

$$h_t = tanh(W_{hx}x + W_{hh}h_{t-1})$$



Example

- Character-level language model



  - Training sequence:
    - “Hello”
  - Presentation:

X: {h, e, l, l}

Y: {e, l , l, o}

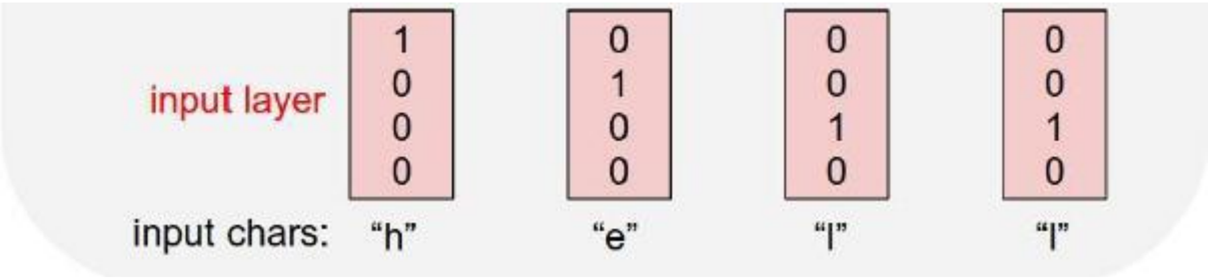
One-hot aka
one-of-K encoding

Example

- Examples:
  - Training sequence:
    - “Hello”
  - Presentation:

  X: {h, e, l, l}

  Y: {e, l , l, o}

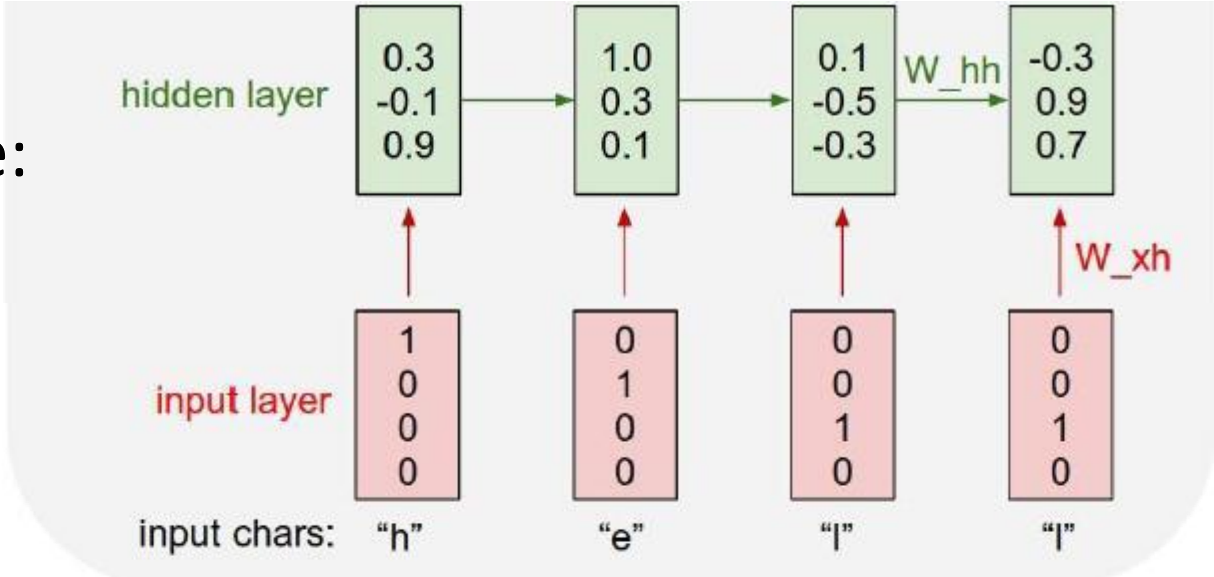


Example

- Examples:
  - Training sequence:
    - “Hello”
  - Presentation:

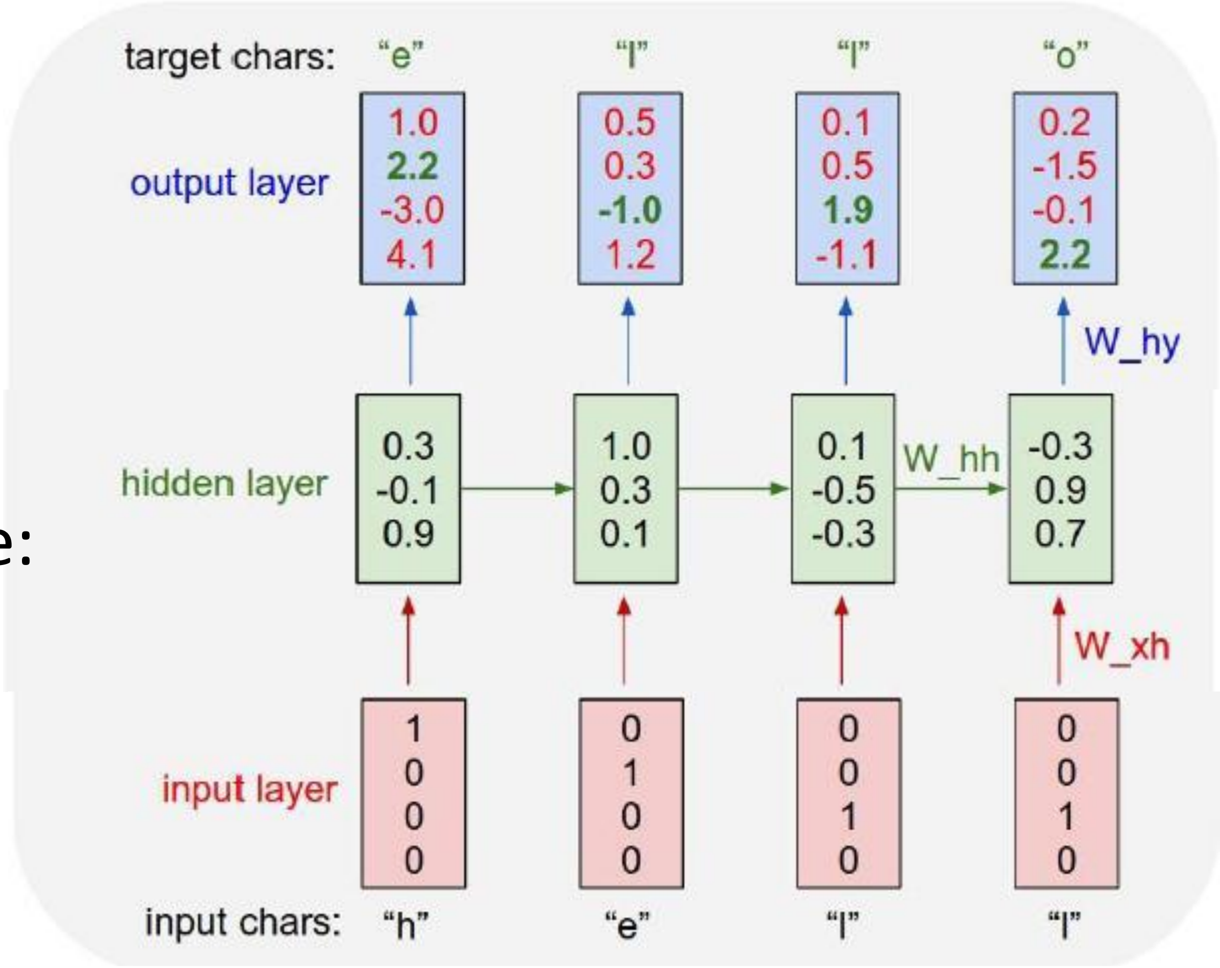
  X: {h, e, l, l}

  Y: {e, l , l, o}



Example

- Examples:



  - Training sequence:
    - "Hello"
  - Presentation:

X: {h, e, l, l}

Y: {e, l , l, o}

outline

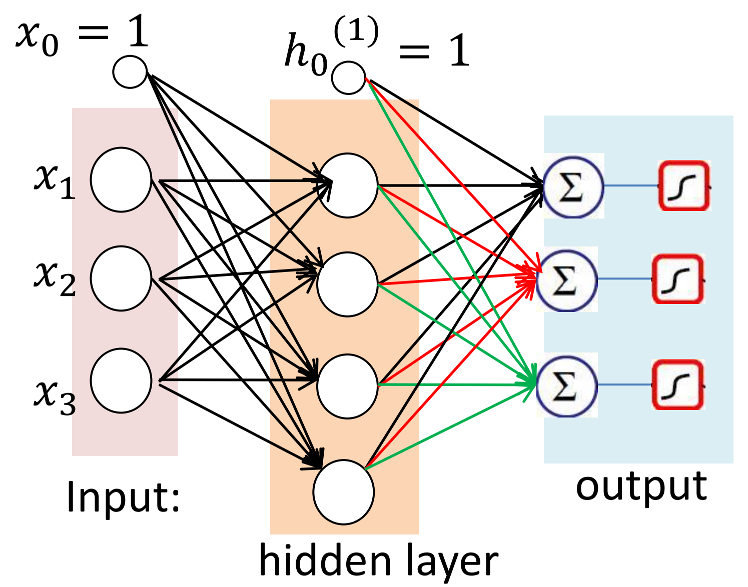
- Recurrent Neural Network
  - Modeling
  - Training
- Long Short Term Memory (LSTM)
  - Motivation
  - Modeling
- Application
  - Generate article

Neural Network

- Training Algorithm

  - 0.初始化权重 $\boldsymbol{W}^{(0)}$
  - 1. 前向过程：
    - 1.1根据输入$\boldsymbol{x}$，计算输出值 $\boldsymbol{y}$
    - 1.2.计算损失函数值$L(\mathbf{y}, \hat{\mathbf{y}})$。
  - 2.后向传播
    - 计算$\frac{\mathrm{d}\,\mathrm{L}}{\mathbf{y}}$
    - 后向传播直到计算$\frac{\mathrm{d}\,\mathrm{L}}{\mathbf{x}}$
  - 3.计算梯度$\frac{d\,L}{\boldsymbol{d}\,\boldsymbol{W}}$
  - 4.更新梯度

$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta \frac{d\,\mathrm{L}}{\boldsymbol{d}\,\boldsymbol{W}^{(t)}}$$

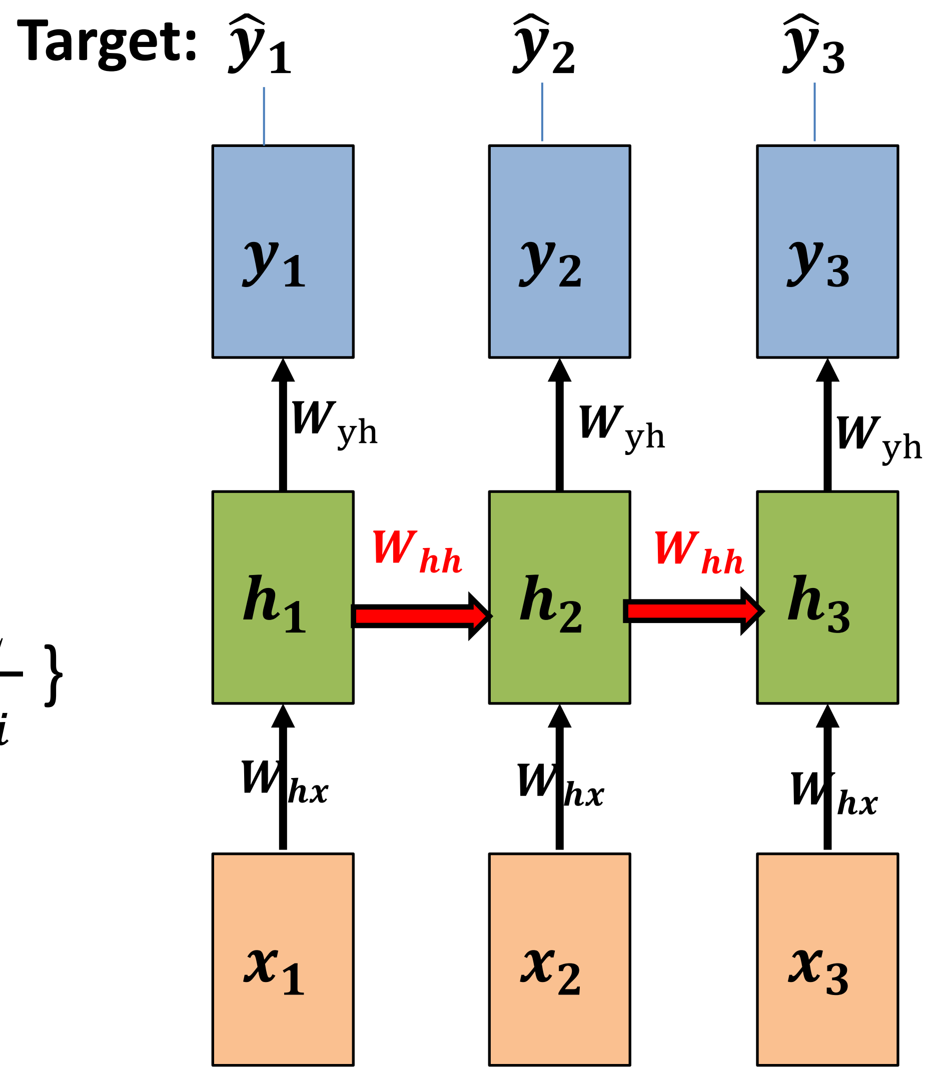


$(1, 0, 0)^T$

Training

- learning
  - Sequence length=3
- Back-propagation

$$-\left\{\frac{\mathrm{d\,L}}{\boldsymbol{d\,y_i}} \Rightarrow \frac{\mathrm{d\,L}}{\boldsymbol{d\,h_i}} \Rightarrow \frac{\mathrm{d\,L}}{\boldsymbol{d\,x_i}}\right\}$$

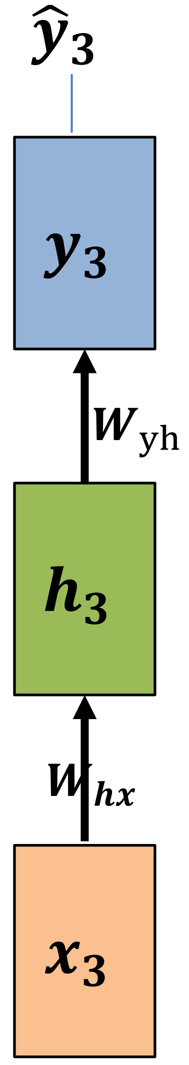


Training

- Back-propagation

Target:

$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_3} = \frac{\mathrm{d\,L}}{\boldsymbol{d}\,\boldsymbol{y}_3}\,\frac{\mathrm{d}\boldsymbol{y}_3}{d\boldsymbol{h}_3}$$



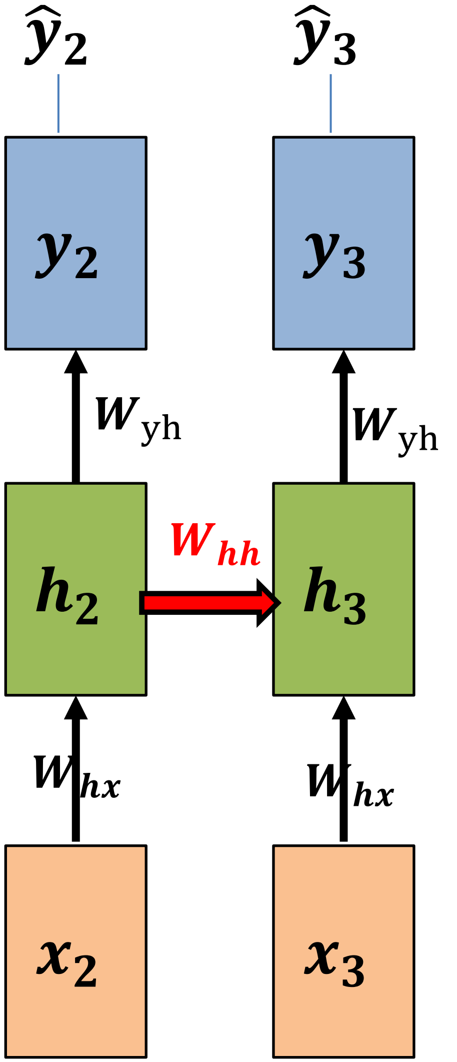
Training

- Back-propagation

Target:

$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_3} = \frac{\mathrm{d\,L}}{\boldsymbol{d}\,\boldsymbol{y}_3}\,\frac{\mathrm{d}\boldsymbol{y}_3}{d\boldsymbol{h}_3}$$

$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_2} = \frac{\mathrm{d\,L}}{\boldsymbol{d}\,\boldsymbol{y}_2}\,\frac{\mathrm{d}\boldsymbol{y}_2}{d\boldsymbol{h}_2} + \frac{\mathrm{d\,L}}{\boldsymbol{d}\,\boldsymbol{y}_3}\,\frac{\mathrm{d}\boldsymbol{y}_3}{d\boldsymbol{h}_3}\,\frac{\mathrm{d}\boldsymbol{h}_3}{d\boldsymbol{h}_2}$$



Training

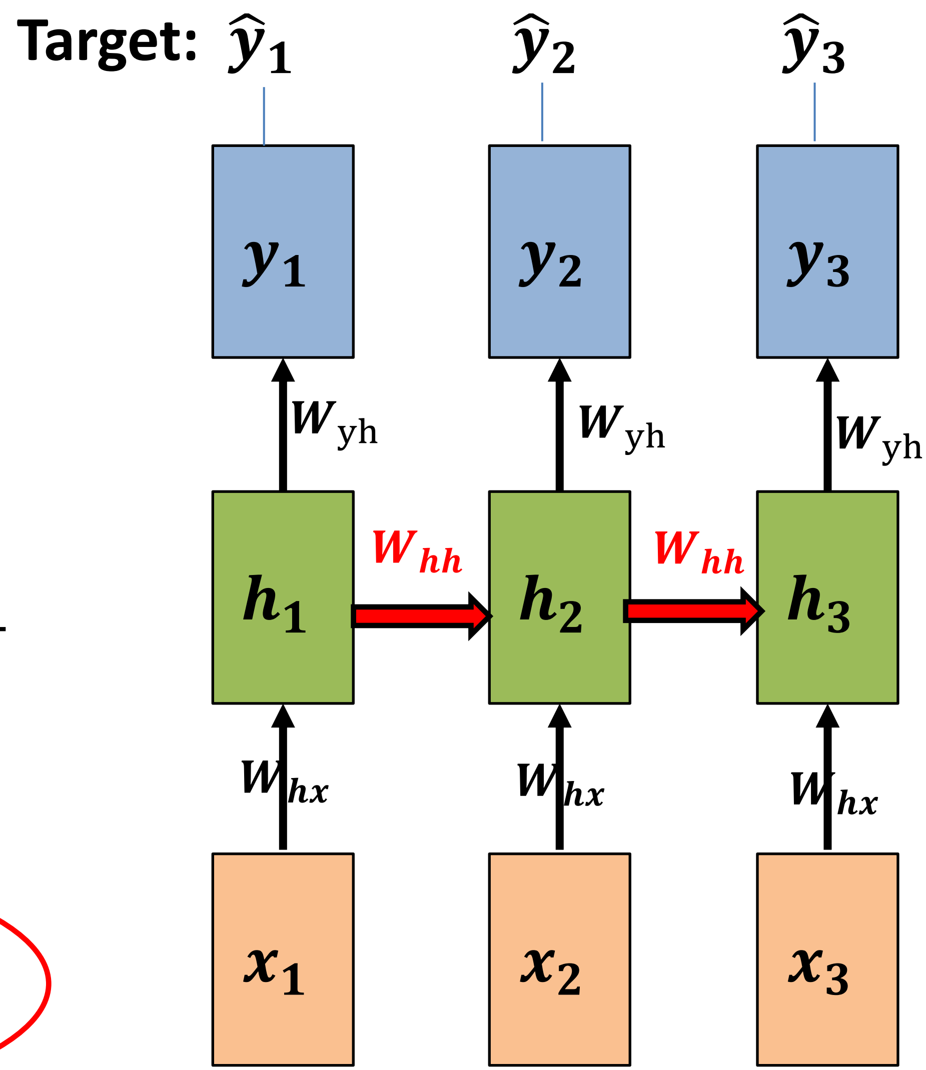
- Back-propagation

$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_3} = \frac{\mathrm{d\,L}}{d\,\boldsymbol{y}_3}\,\frac{\mathrm{d}\boldsymbol{y}_3}{d\boldsymbol{h}_3}$$

$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_2} = \frac{\mathrm{d\,L}}{d\,\boldsymbol{y}_2}\,\frac{\mathrm{d}\boldsymbol{y}_2}{d\boldsymbol{h}_2} + \frac{\mathrm{d\,L}}{d\,\boldsymbol{y}_3}\,\frac{\mathrm{d}\boldsymbol{y}_3}{d\boldsymbol{h}_3}\,\frac{\mathrm{d}\boldsymbol{h}_3}{d\boldsymbol{h}_2}$$

$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_1} = \frac{\mathrm{d\,L}}{d\,\boldsymbol{y}_1}\,\frac{\mathrm{d}\boldsymbol{y}_1}{d\boldsymbol{h}_1} + \frac{\mathrm{d\,L}}{d\,\boldsymbol{y}_2}\,\frac{\mathrm{d}\boldsymbol{y}_2}{d\boldsymbol{h}_2}\,\frac{\mathrm{d}\boldsymbol{h}_2}{d\boldsymbol{h}_1} + \frac{\mathrm{d\,L}}{d\,\boldsymbol{y}_3}\,\frac{\mathrm{d}\boldsymbol{y}_3}{d\boldsymbol{h}_3}\,\frac{\mathrm{d}\boldsymbol{h}_3}{d\boldsymbol{h}_2}\,\frac{\mathrm{d}\boldsymbol{h}_2}{d\boldsymbol{h}_1}$$

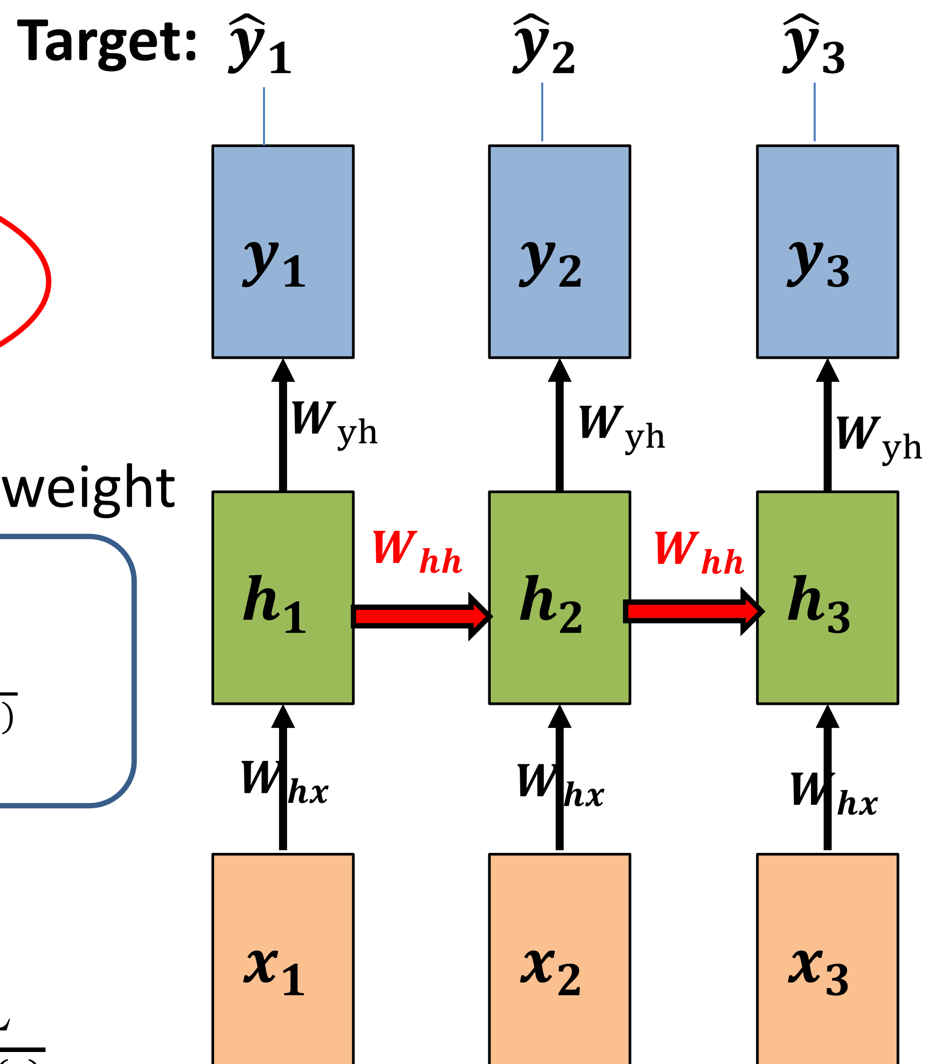
$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_t} = \sum_{s=t}^{T=3} \frac{\mathrm{d\,L}}{d\,\boldsymbol{y}_s}\,\frac{\mathrm{d}\boldsymbol{y}_s}{d\boldsymbol{h}_s}\,\frac{\mathrm{d}\boldsymbol{h}_s}{d\boldsymbol{h}_t}$$



Training

- Gradient r.t Weight

$$\frac{\mathrm{d\,L}}{d\,\boldsymbol{h}_t} = \sum_{s=t}^{T} \frac{\mathrm{d\,L}}{\boldsymbol{d}\ \boldsymbol{y}_s} \frac{\mathrm{d}\boldsymbol{y}_s}{d\boldsymbol{h}_s} \frac{\mathrm{d}\boldsymbol{h}_s}{d\boldsymbol{h}_t}$$



- Calculate gradient respect to weight

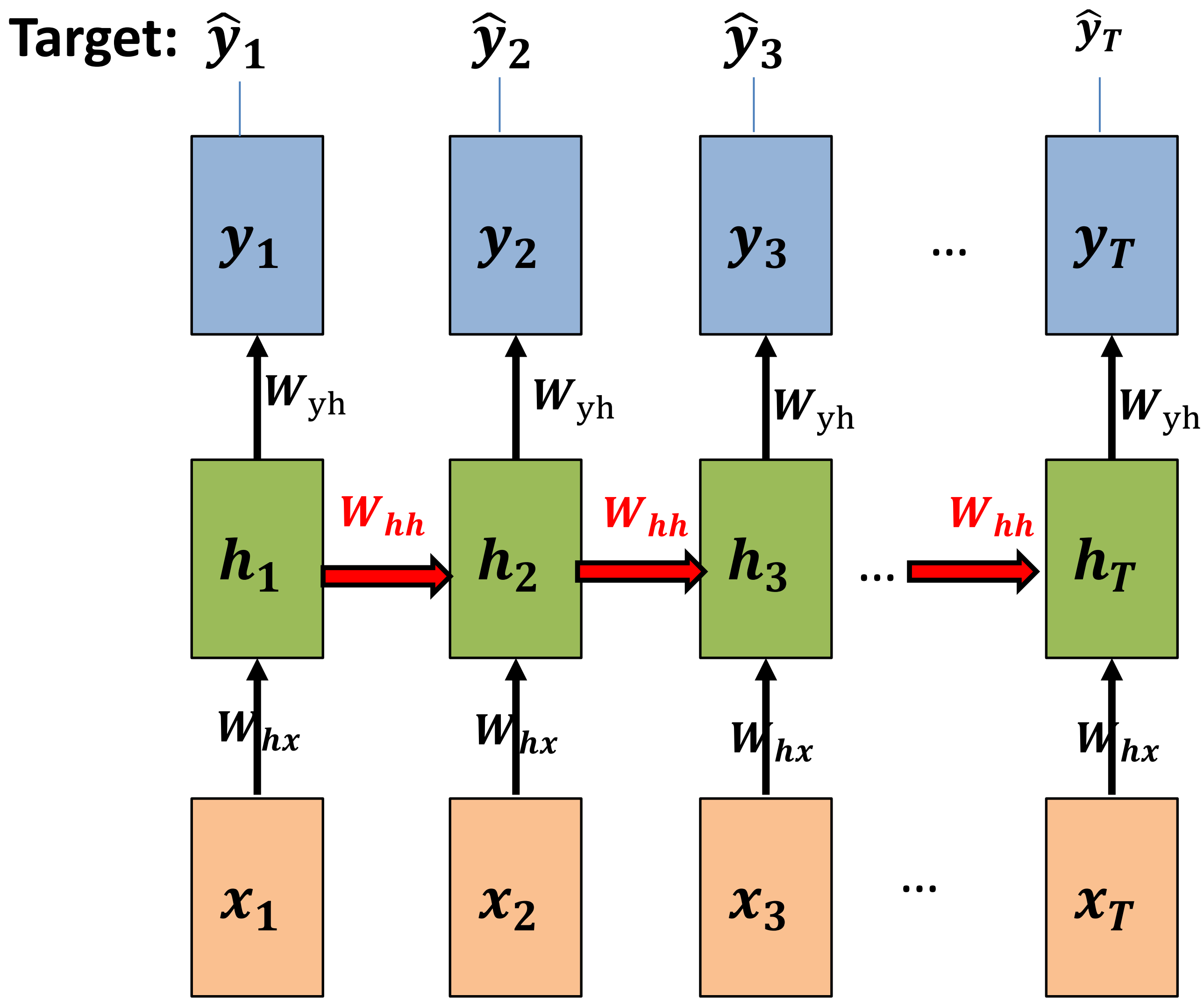
$$\frac{\mathrm{d\,L}}{d\mathbf{W}_{\mathrm{yh}}{}^{(\mathrm{i})}} \qquad \frac{\mathrm{d\,L}}{d\mathbf{W}_{\mathrm{hh}}{}^{(\mathrm{i})}} \qquad \frac{\mathrm{d\,L}}{d\mathbf{W}_{\mathrm{hx}}{}^{(\mathrm{i})}}$$

- Update weight

$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta \sum_{s=1}^{T} \frac{d\,\mathrm{L}}{\boldsymbol{d}\,\boldsymbol{W}^{(s)}}$$

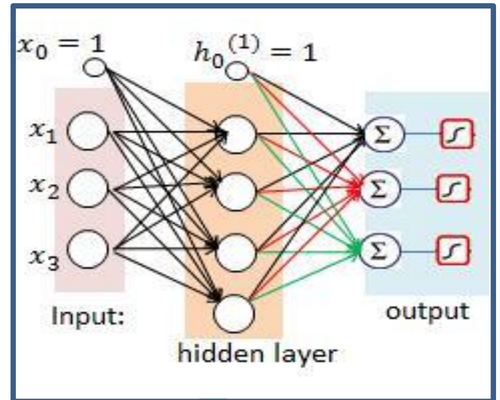
RNN

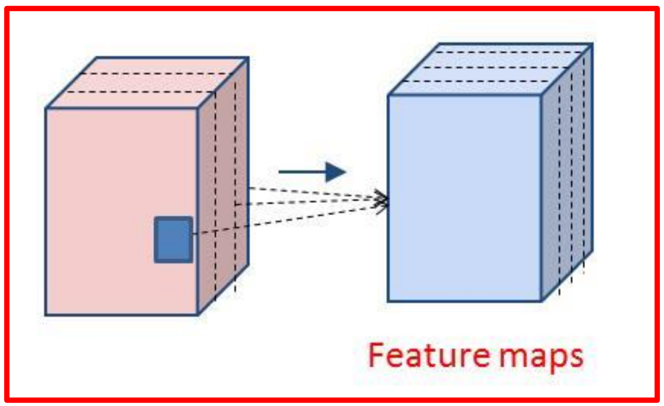
- longer
- deeper



Relation: MLP, CNN and RNN

x$\in R^{d_{in}}$ $W \in R^{d_{out} \times d_{in}}$ y$\in R^{d_{out}}$



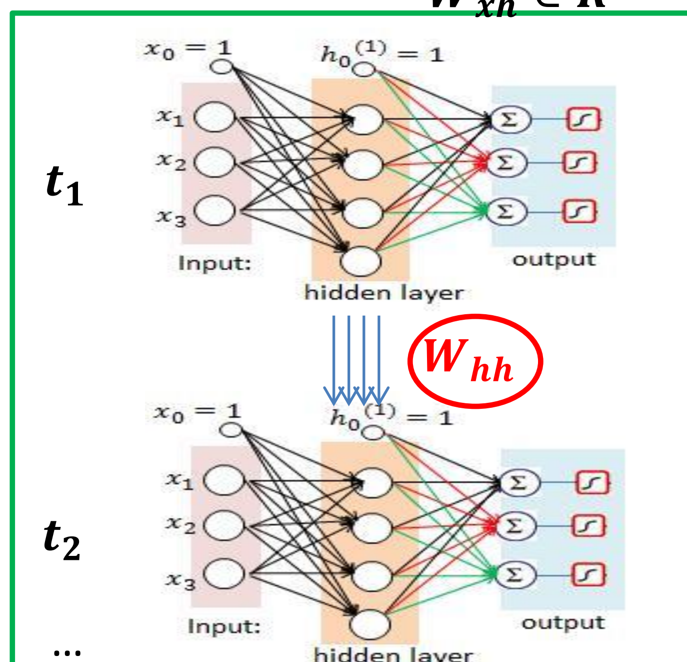


X$\in R^{d_{in} \times h \times w}$

W$\in R^{d_{out} \times d_{in} \times F_h \times F_w}$

Y$\in R^{d_{out} \times h \times w}$

$W_{xh} \in R^{d_{out} \times d_{in}}$



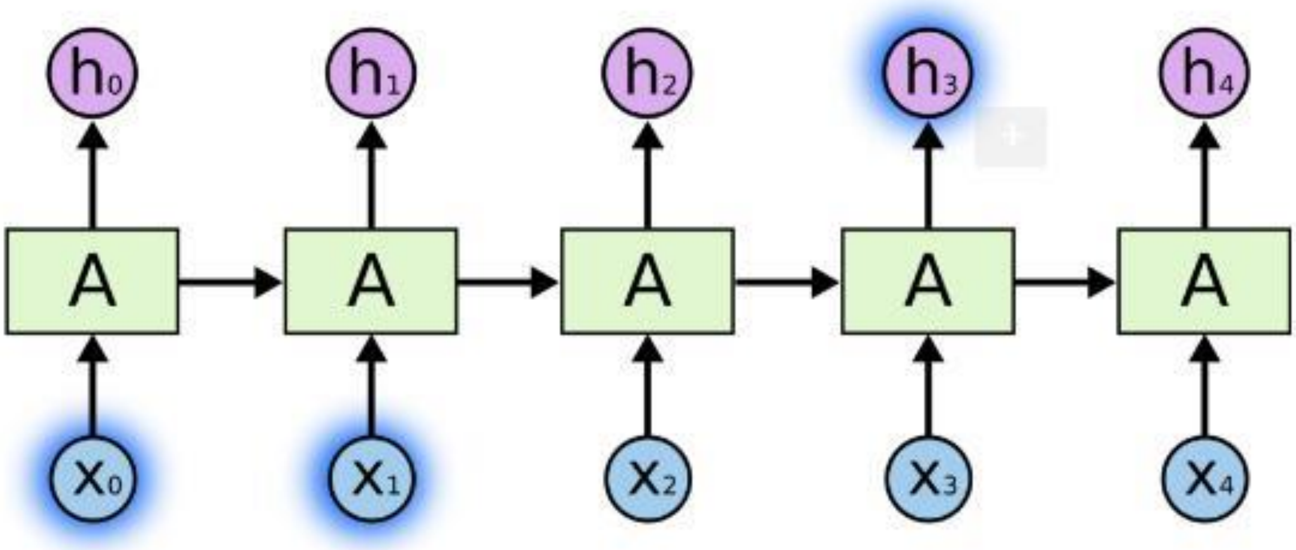
outline

- Recurrent Neural Network
  - Modeling
  - Training
- Long Short Term Memory (LSTM)
  - Motivation
  - Modeling
- Application
  - Generate article

Long Short Term Memory (LSTM)

- Motivation

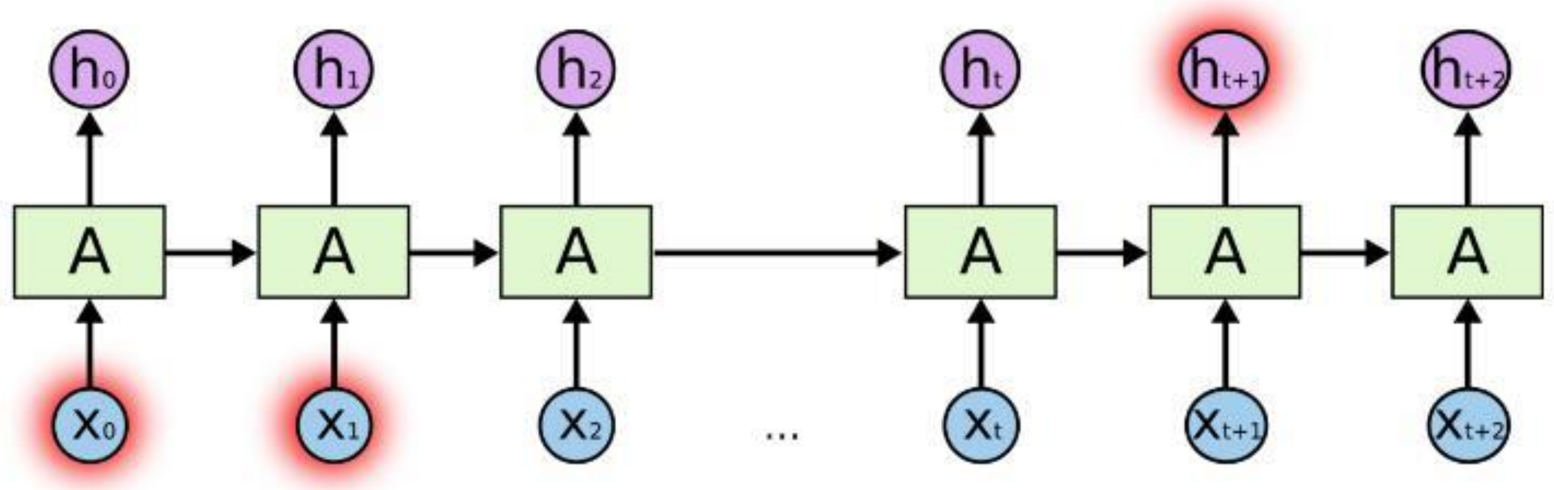
“the clouds are in the *sky*,”



Long Short Term Memory (LSTM)

- Motivation

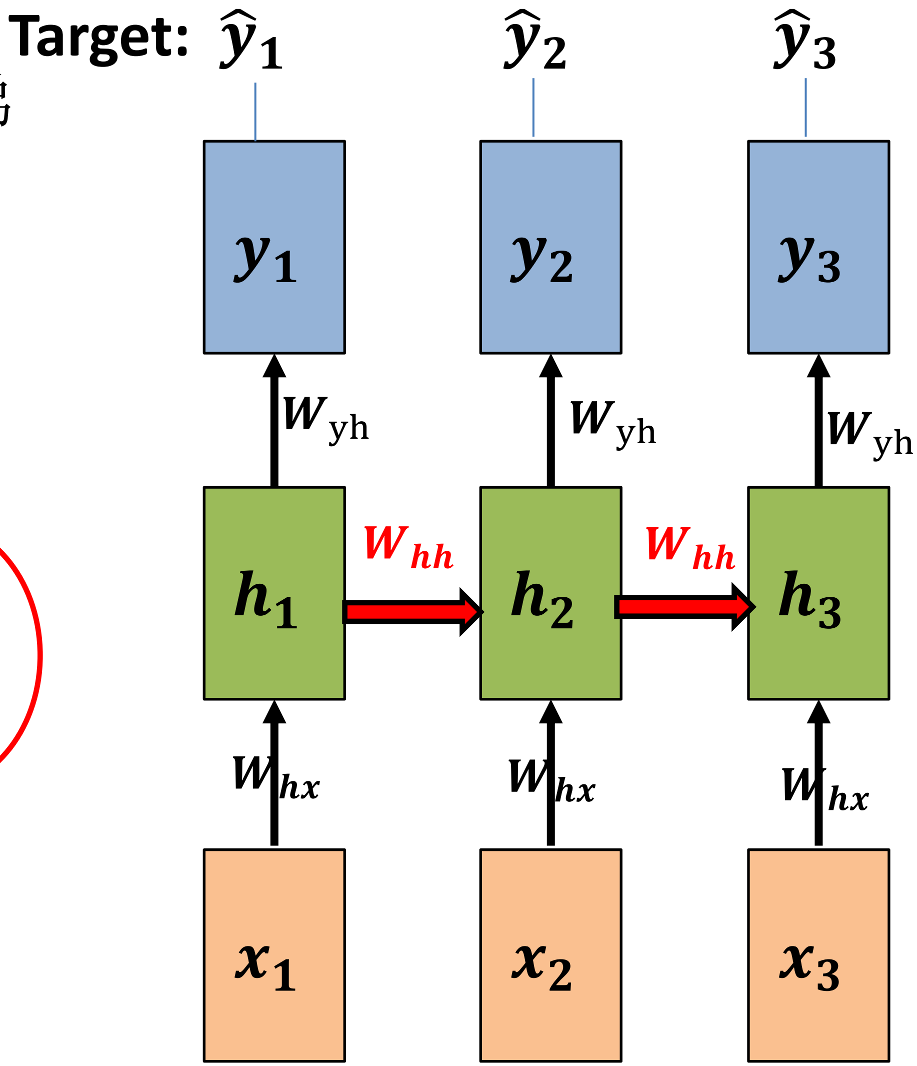
"I grew up in France… I speak fluent *French*."



LSTM

- Motivation
  - Gradient explosion (梯度爆炸)
  - Gradient vanishing(梯度弥散)

$$\frac{\mathrm{d}\,\mathrm{L}}{d\,\boldsymbol{h}_t}=\sum_{s=t}^{T}\frac{\mathrm{d}\,\mathrm{L}}{\boldsymbol{d}\,\boldsymbol{y}_s}\frac{\mathrm{d}\boldsymbol{y}_s}{d\boldsymbol{h}_s}\frac{\mathrm{d}\boldsymbol{h}_s}{d\boldsymbol{h}_t}$$



outline

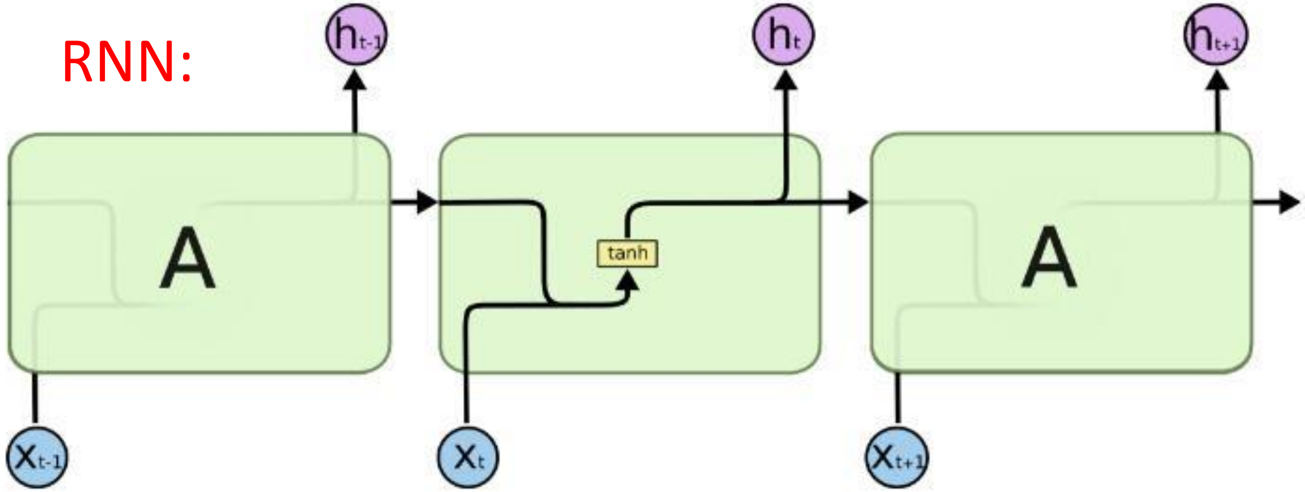
- Recurrent Neural Network
  - Modeling
  - Training
- Long Short Term Memory (LSTM)
  - Motivation
  - Modeling
- Application
  - Generate article

LSTM

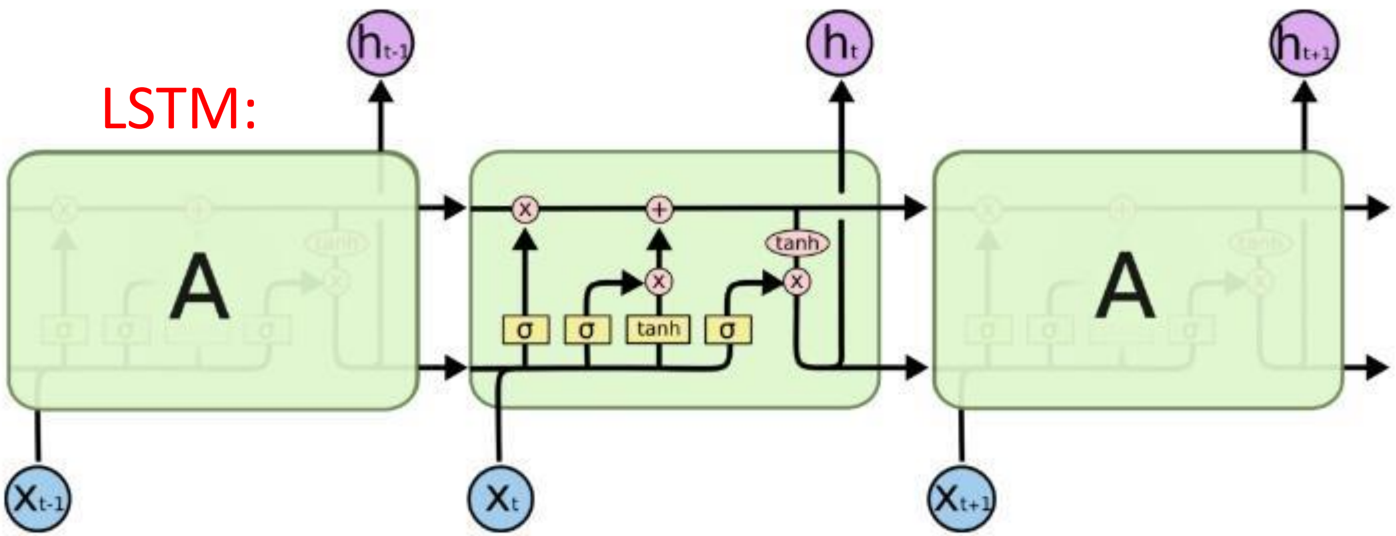
- Modeling

$$\boldsymbol{h_t} = tanh(\boldsymbol{W_{hx}}\boldsymbol{x} + \boldsymbol{W_{hh}}\boldsymbol{h_{t-1}})$$

RNN:

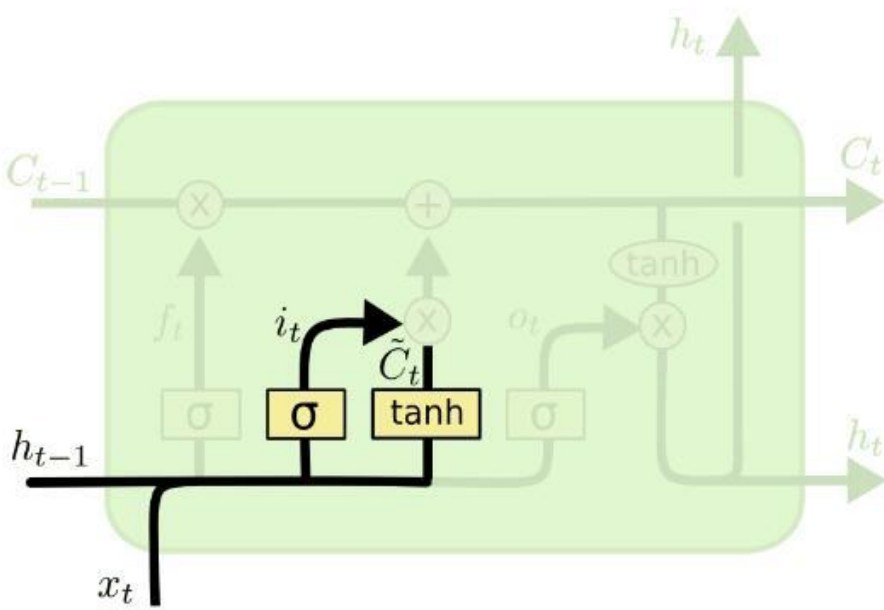


LSTM:



LSTM

- Modeling
  - Input gate
  - Input information

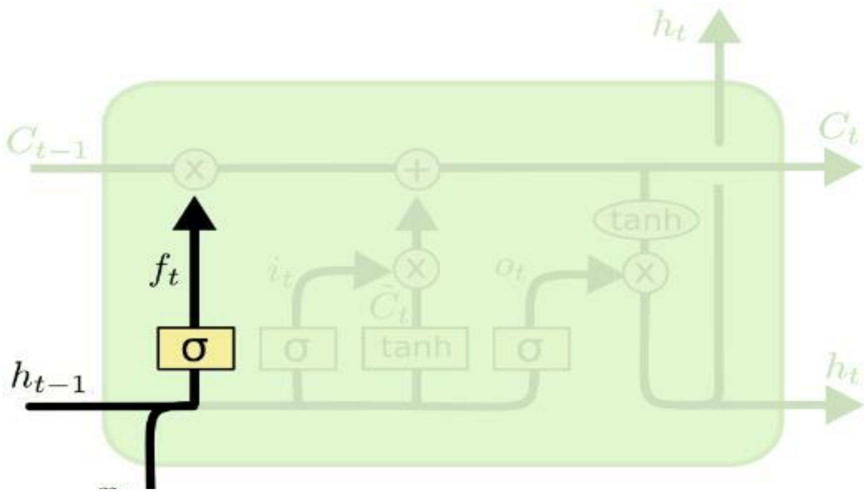


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

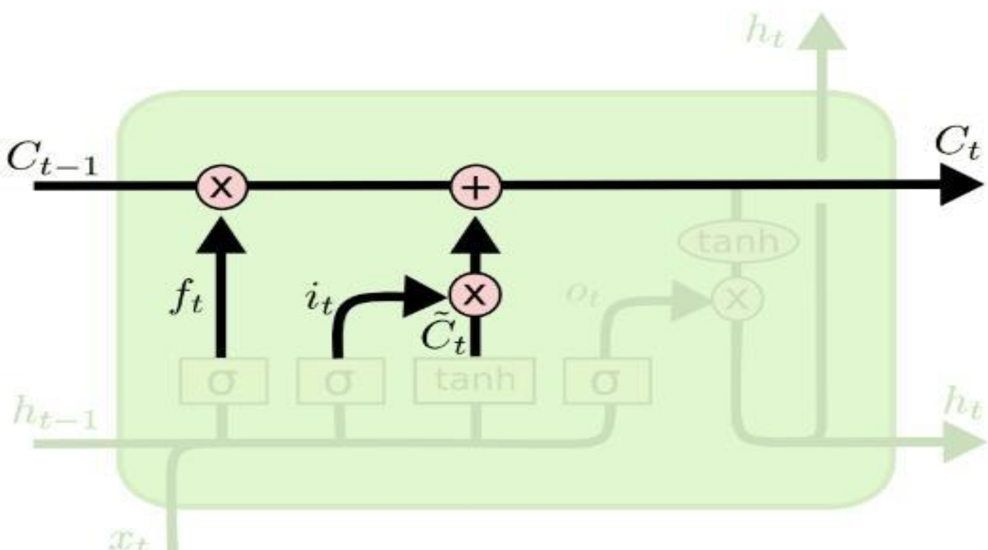
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM

- Modeling



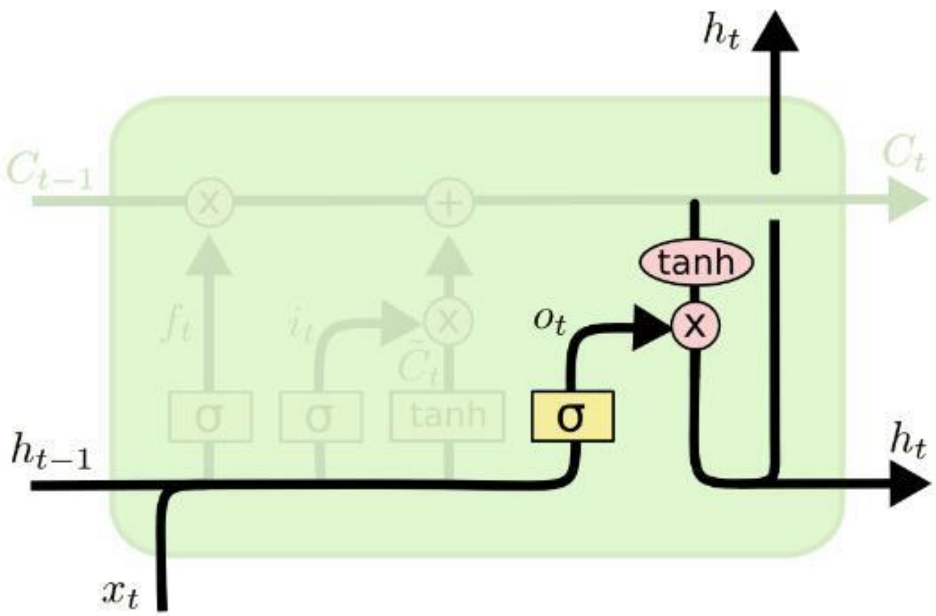
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



$$C_t = \overbrace{f_t}^{\text{Forget Gate}} * \underbrace{C_{t-1}}_{\text{Previous Info}} + \overbrace{i_t}^{\text{Input Gate}} * \underbrace{\tilde{C}_t}_{\text{Input Info}}$$

LSTM

- Modeling
  - Output gate
  - Output information

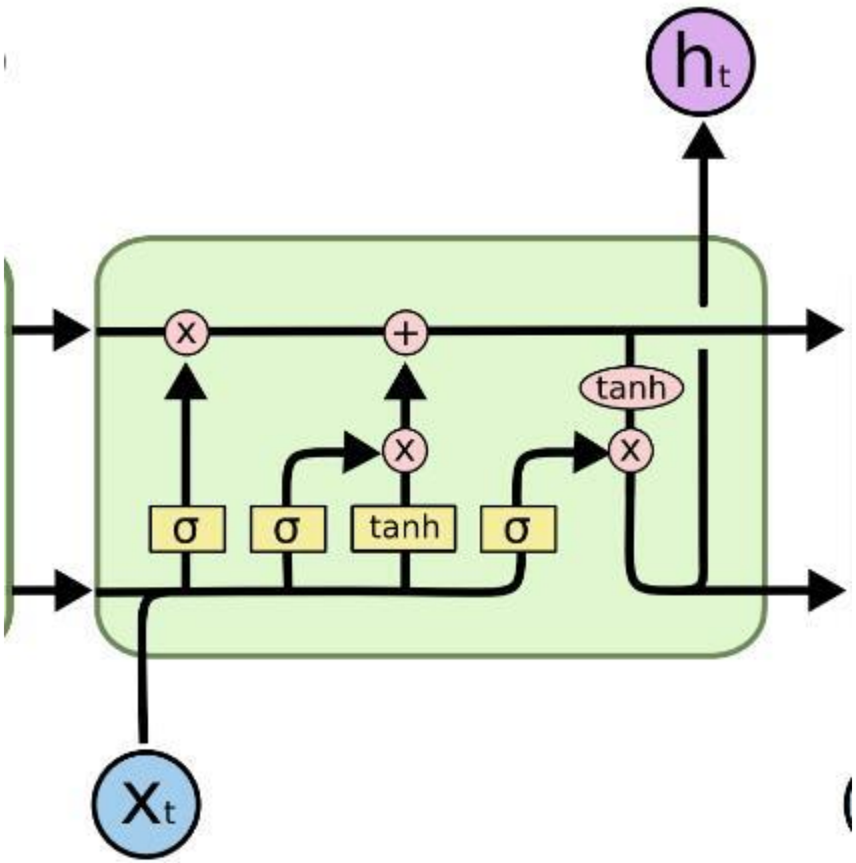


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM

- Modeling



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

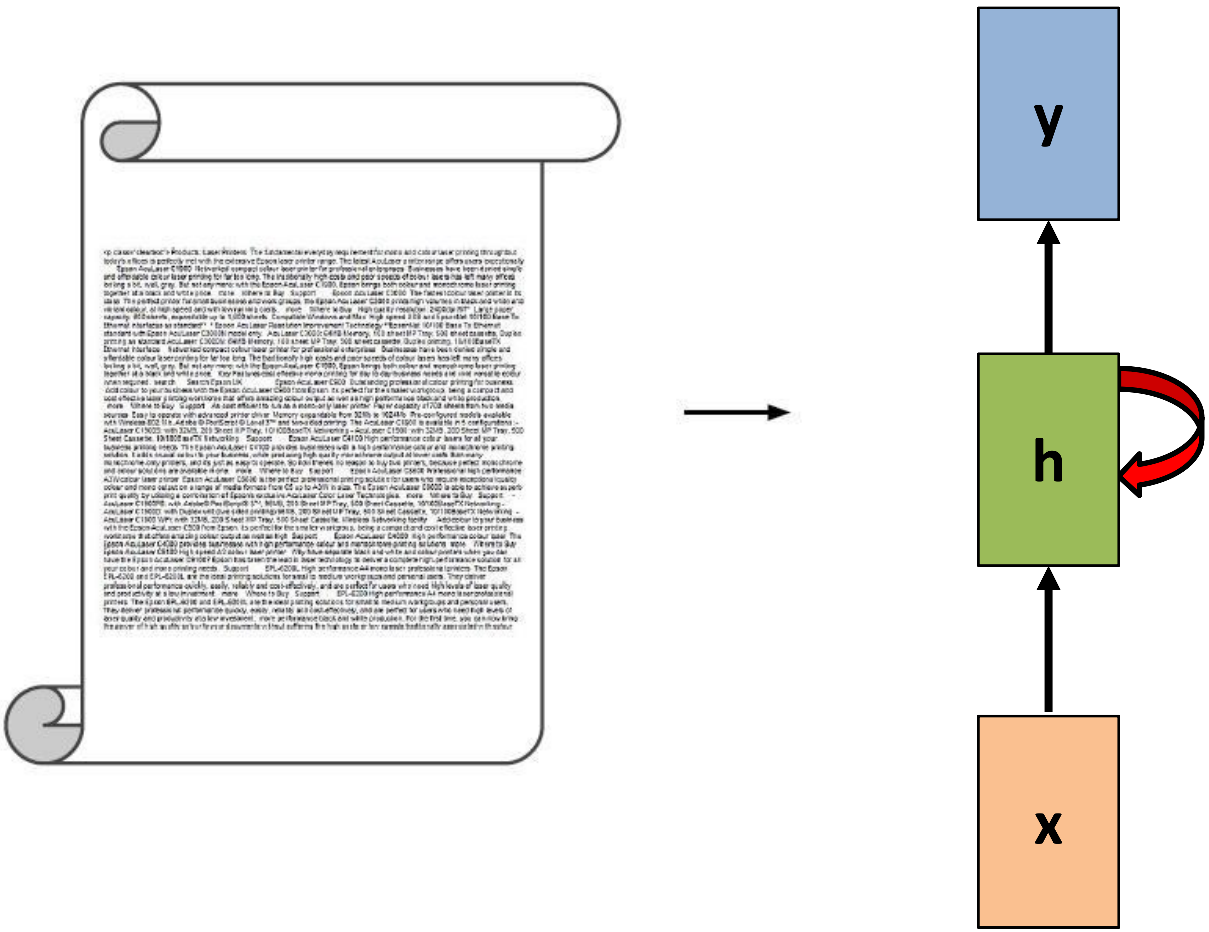
$$h_t = o_t * \tanh\left(C_t\right)$$

outline

- Recurrent Neural Network
  - Modeling
  - Training
- Long Short Term Memory (LSTM)
  - Motivation
  - Modeling
- Application
  - Generate article

RNN-application

- Generate article



RNN-application

- Generate article

Sonnet 116 – Let me not ...

by William Shakespeare

Let me not to the marriage of true minds
        Admit impediments. Love is not love
Which alters when it alteration finds,
        Or bends with the remover to remove:
O no! it is an ever-fixed mark
        That looks on tempests and is never shaken;
It is the star to every wandering bark,
        Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
        Within his bending sickle's compass come:
Love alters not with his brief hours and weeks,
        But bears it out even to the edge of doom.
If this be error and upon me proved,
        I never writ, nor no man ever loved.

RNN-application

- Generate article

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

RNN-application

- Generate article

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.
```

RNN-application

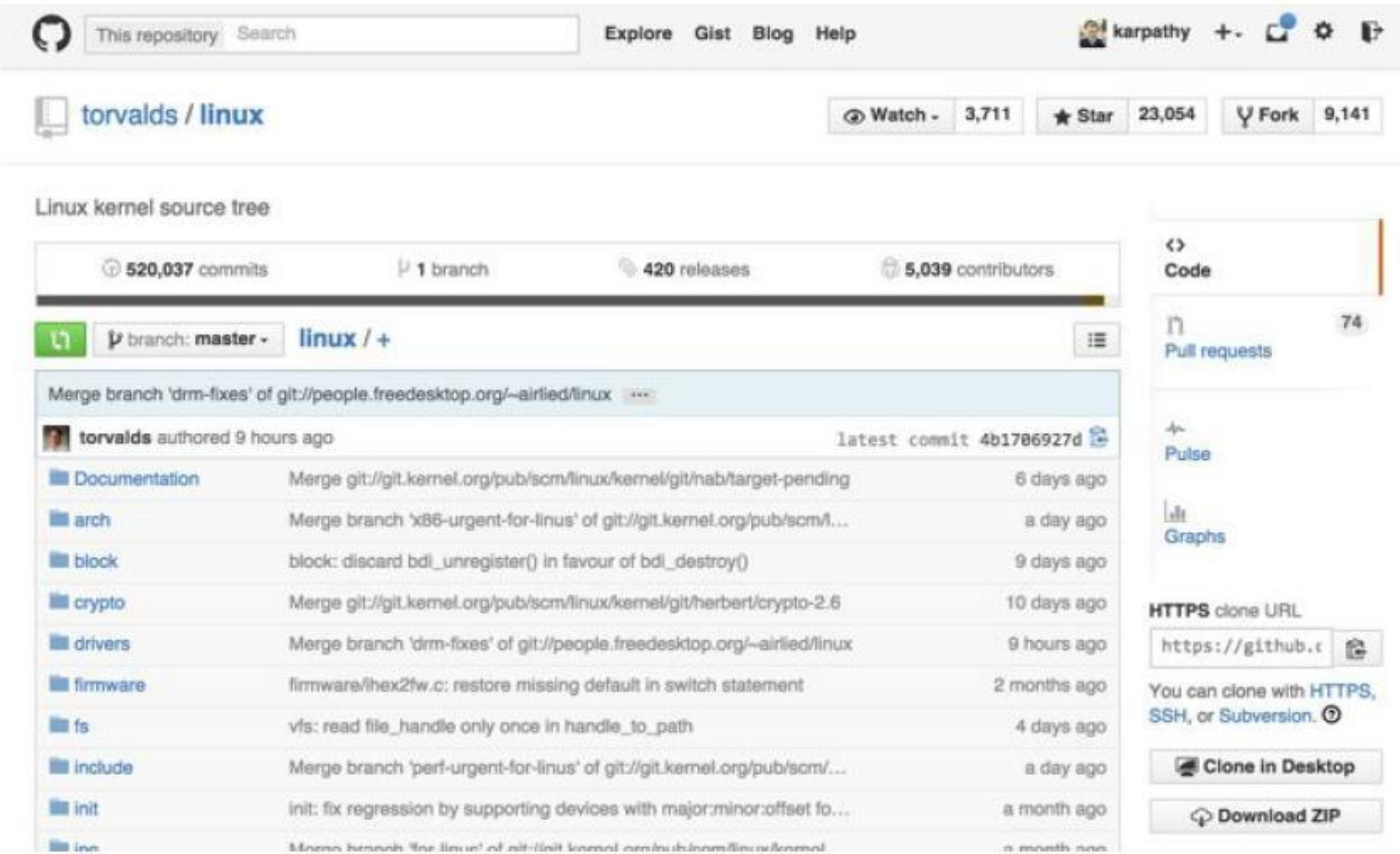
- Generate article

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

RNN-application

- Generate C code



RNN-application

- Generated C code

```c
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated C code

RNN-application

- Generated C code

```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)     (func)

#define SWAP_ALLOCATE(nr)     (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0));   \
  if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
          pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
  PUT_PARAM_RAID(2, sel) = get_state_state();
  set_pid_sum((unsigned long)state, current_state_str(),
           (unsigned long)-1->lr_full; low;
}
```