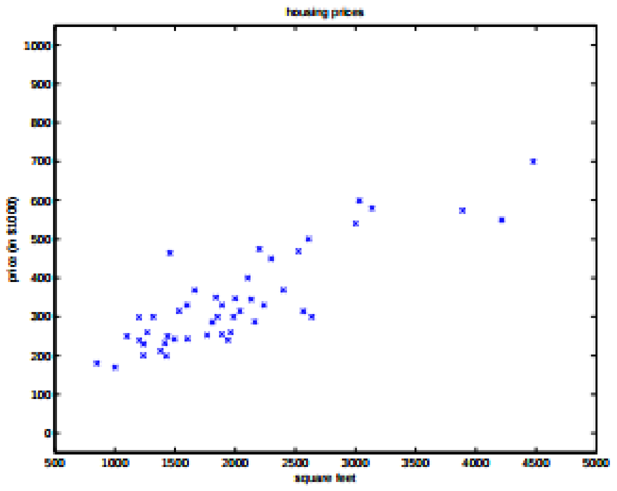
# 机器学习

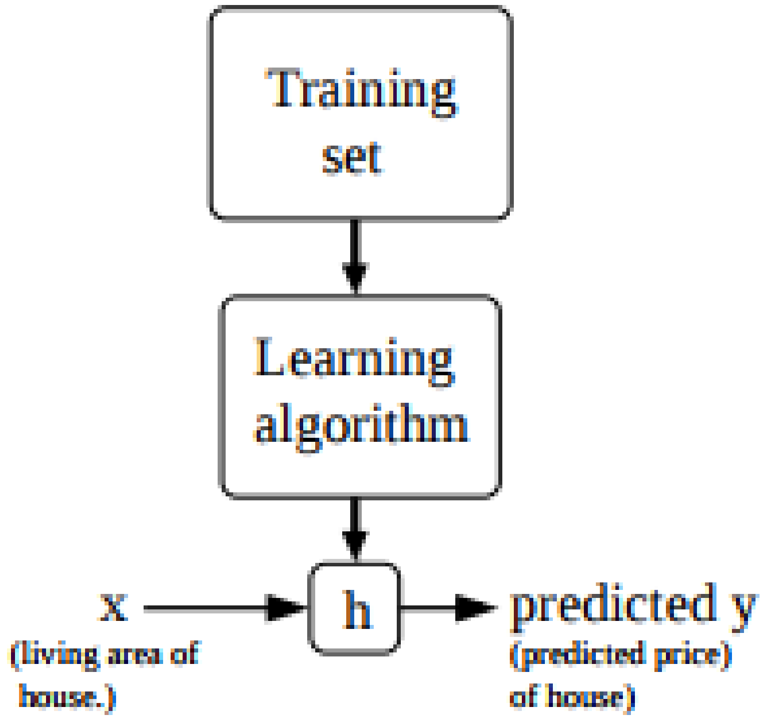
# 房价预测

| Living area ($\text{feet}^2$) | Price (1000\$s) |
|---|---|
| 2104 | 400 |
| 1600 | 330 |
| 2400 | 369 |
| 1416 | 232 |
| 3000 | 540 |
| ⋮ | ⋮ |

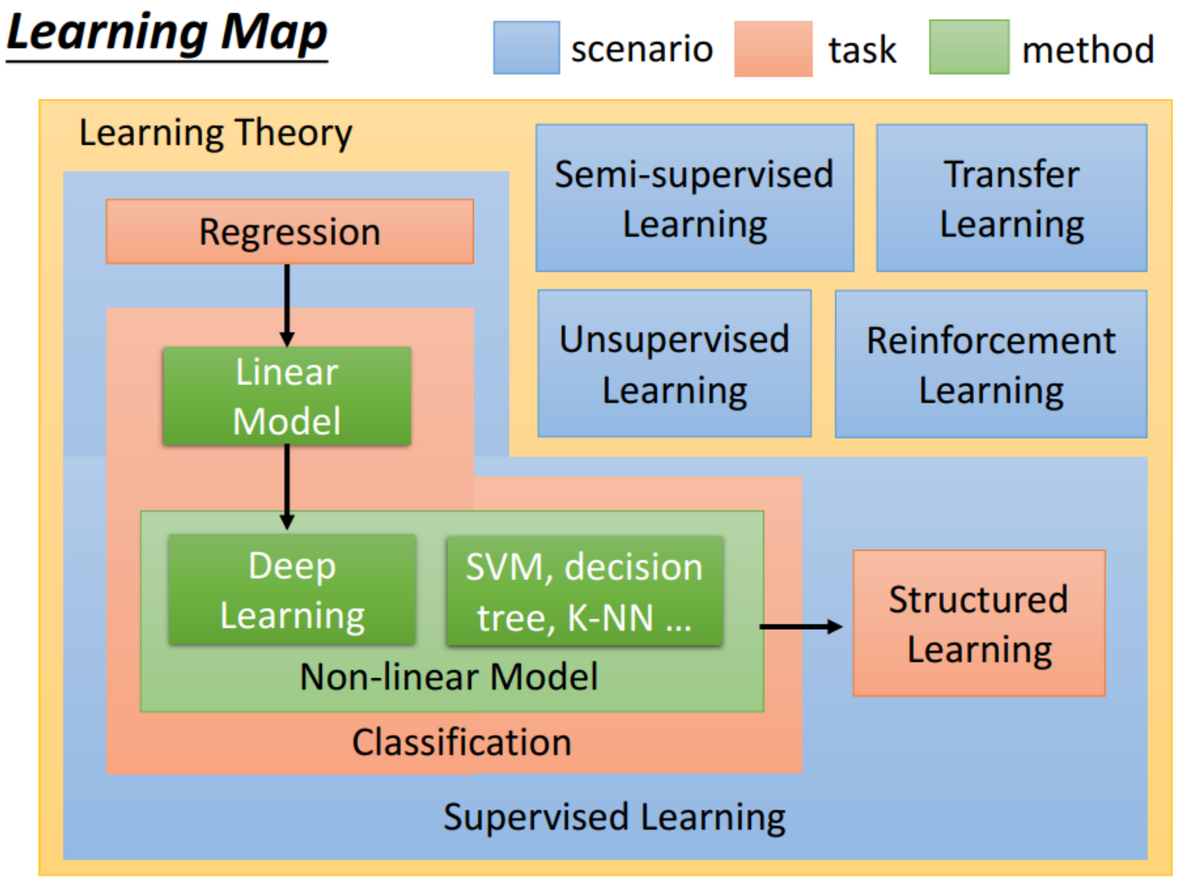
We can plot this data:



# 机器学习－模型

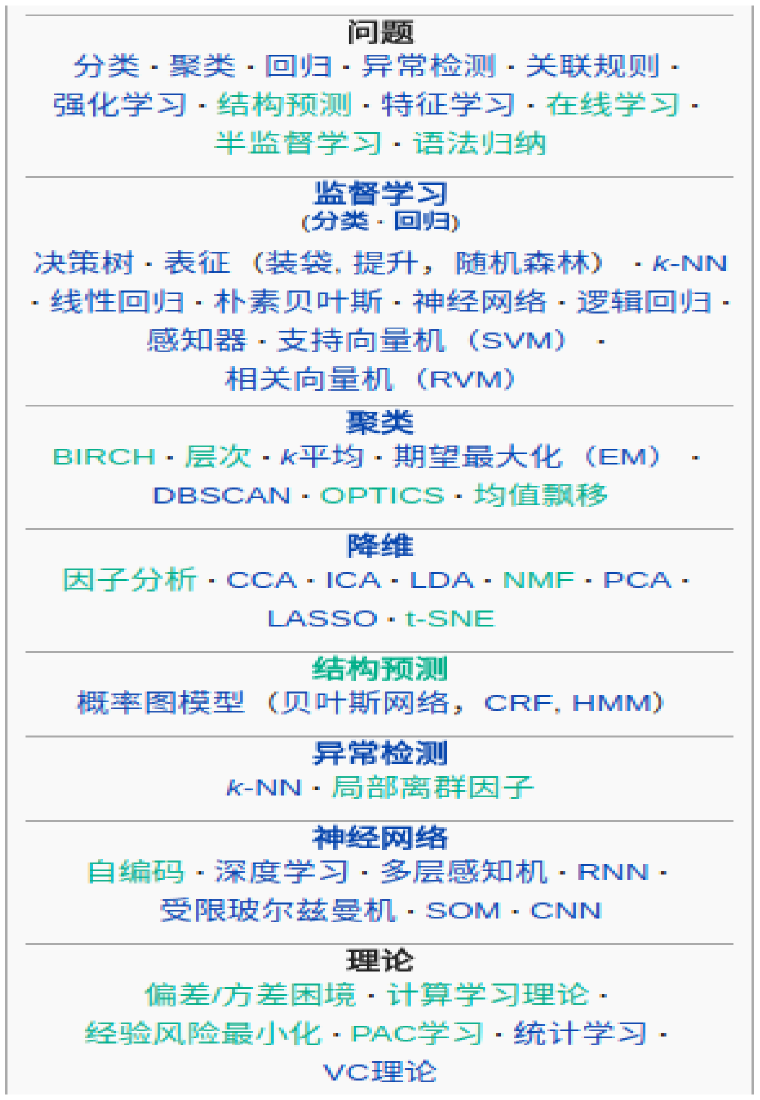

Training set
Learning algorithm
x
(living area of house.)
h
predicted y
(predicted price) of house)

# 机器学习



Learning Map
scenario
task
method
Learning Theory
Semi-supervised Learning
Transfer Learning
Unsupervised Learning
Reinforcement Learning
Regression
Linear Model
Deep Learning
SVM, decision tree, K-NN …
Non-linear Model
Classification
Structured Learning
Supervised Learning

# 机器学习



**问题**

分类 · 聚类 · 回归 · 异常检测 · 关联规则 · 强化学习 · 结构预测 · 特征学习 · 在线学习 · 半监督学习 · 语法归纳

**监督学习**
**（分类 · 回归）**

决策树 · 表征 （装袋，提升，随机森林） · *k*-NN · 线性回归 · 朴素贝叶斯 · 神经网络 · 逻辑回归 · 感知器 · 支持向量机 （SVM） · 相关向量机 （RVM）

**聚类**

BIRCH · 层次 · *k*平均 · 期望最大化 （EM） · DBSCAN · OPTICS · 均值飘移

**降维**

因子分析 · CCA · ICA · LDA · NMF · PCA · LASSO · t-SNE

**结构预测**

概率图模型 （贝叶斯网络，CRF，HMM）

**异常检测**

*k*-NN · 局部离群因子

**神经网络**

自编码 · 深度学习 · 多层感知机 · RNN · 受限玻尔兹曼机 · SOM · CNN

**理论**

偏差/方差困境 · 计算学习理论 · 经验风险最小化 · PAC学习 · 统计学习 · VC理论

## Linear regression

| Living area (feet$^2$) | #bedrooms | Price (1000$s) |
|---|---|---|
| 2104 | 3 | 400 |
| 1600 | 3 | 330 |
| 2400 | 3 | 369 |
| 1416 | 2 | 232 |
| 3000 | 4 | 540 |
| ⋮ | ⋮ | ⋮ |

目标函数：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

损失函数：

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

# 优化—梯度下降

参数更新：

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left( \sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}$$

# 凸优化－交叉熵（Cross entropy）

熵：

$$S(A) = -\sum_i P_A(x_i) log P_A(x_i)$$

ＫＬ离散度：

$$D_{KL}(A||B) = \sum_i P_A(x_i) log\left(\frac{P_A(x_i)}{P_B(x_i)}\right) = \sum_i P_A(x_i) log(P_A(x_i)) - P_A(x_i) log(P_B(x_i))$$

交叉熵：

$$H(A, B) = -\sum_i P_A(x_i) log(P_B(x_i))$$

## 凸优化

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

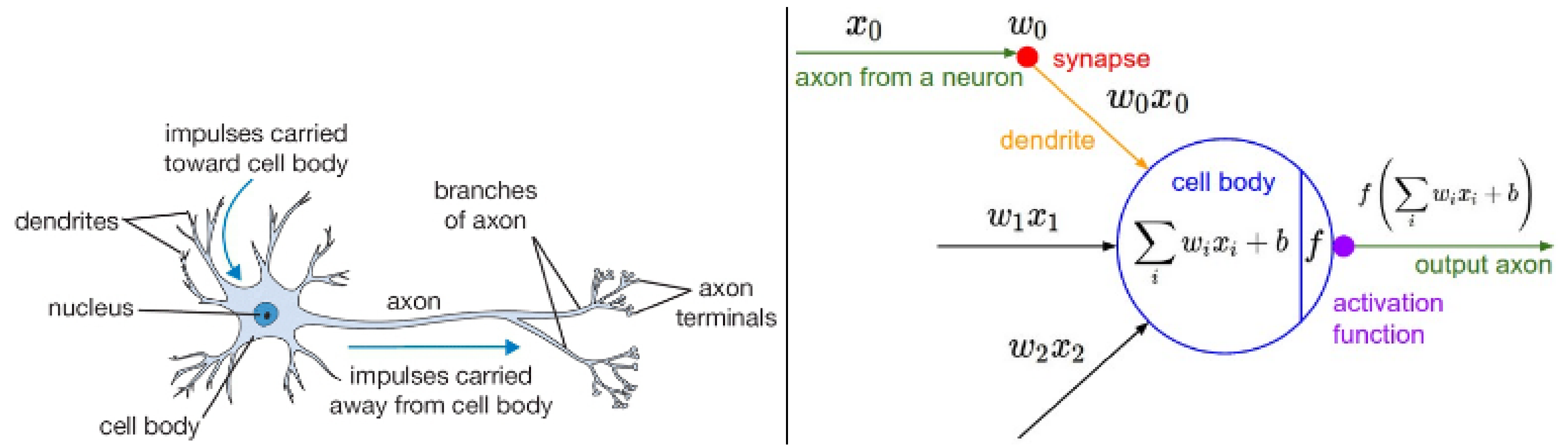
where $f$ is a convex function, $g_i$ are convex functions, and $h_i$ are affine functions, and $x$ is the optimization variable.

# 深度学习－1

## Ups and downs of Deep Learning

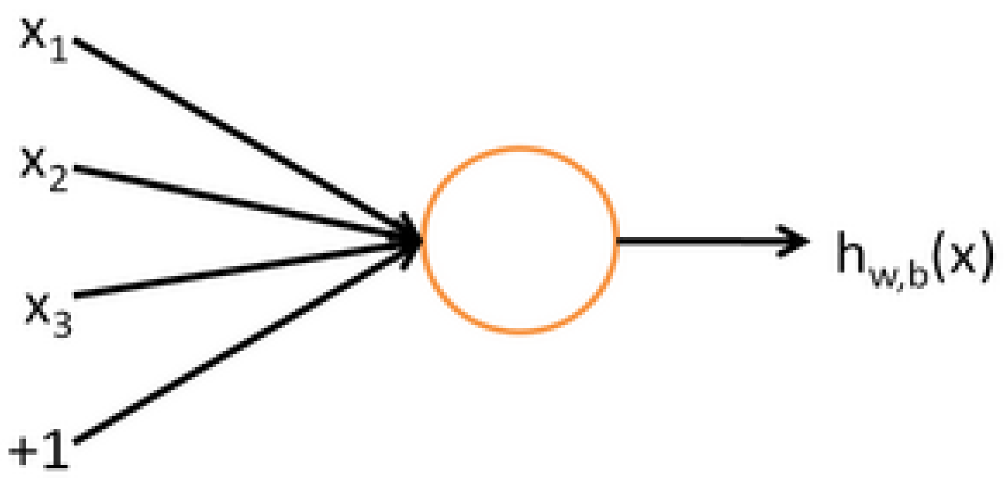
- 1958: Perceptron (linear model)
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
  - Do not have significant difference from DNN today
- 1986: Backpropagation
  - Usually more than 3 hidden layers is not helpful
- 1989: 1 hidden layer is “good enough”, why deep?
- 2006: RBM initialization (breakthrough)
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC image competition

# 模型



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

# 神经网络－1



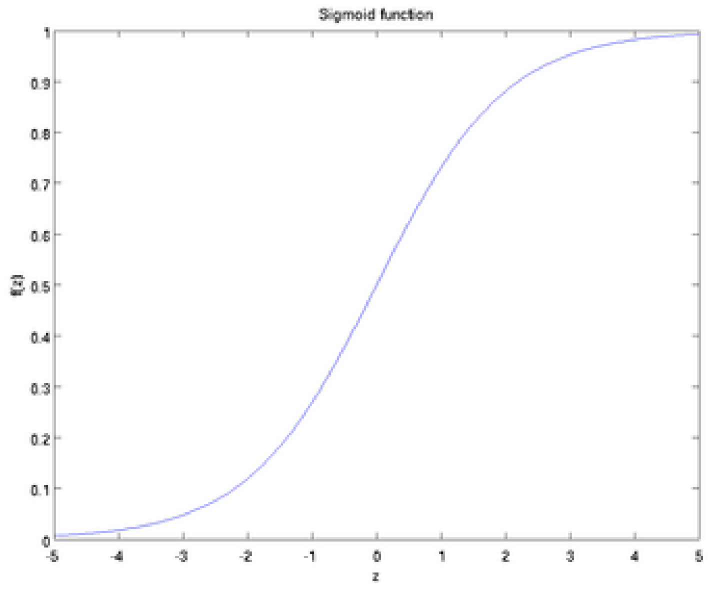
目标函数：$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$

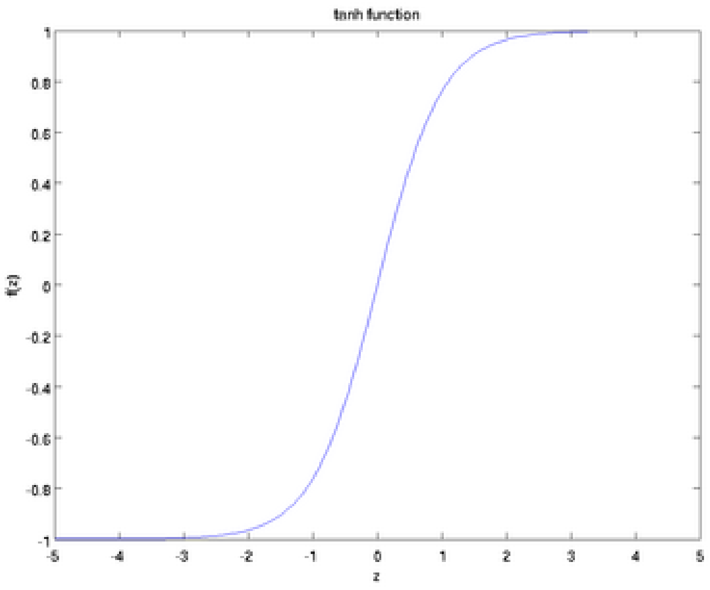
激活函数：$$f(z) = \frac{1}{1+\exp(-z)}.$$

# 神经网络－ 2

Sigmoid:

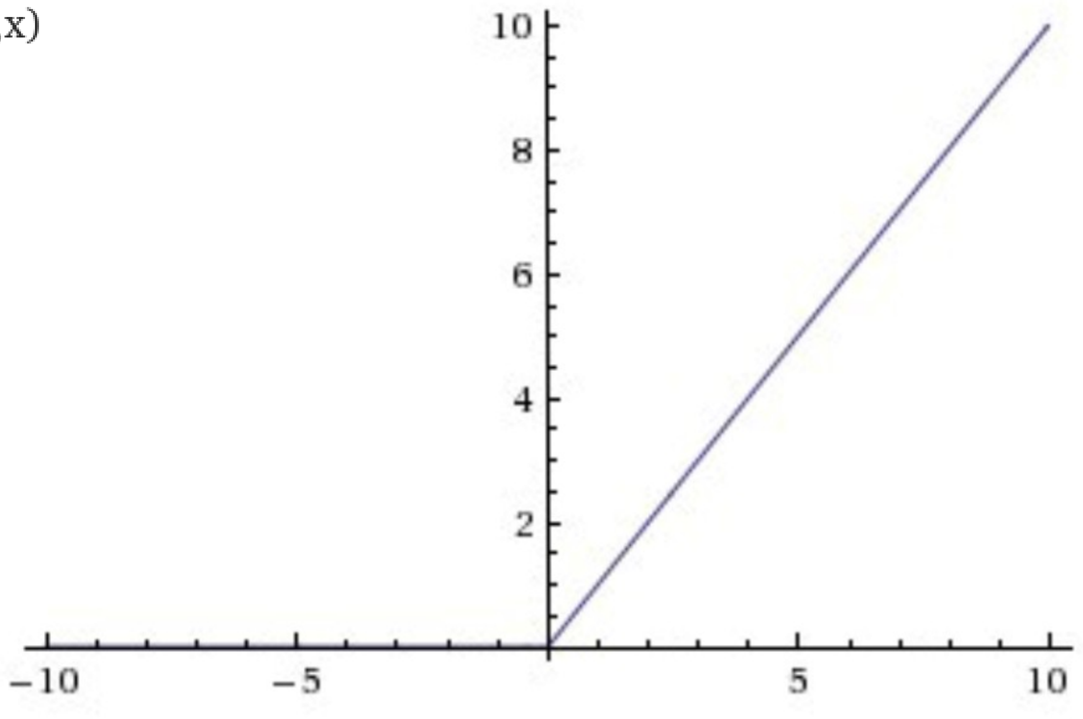
Tanh:



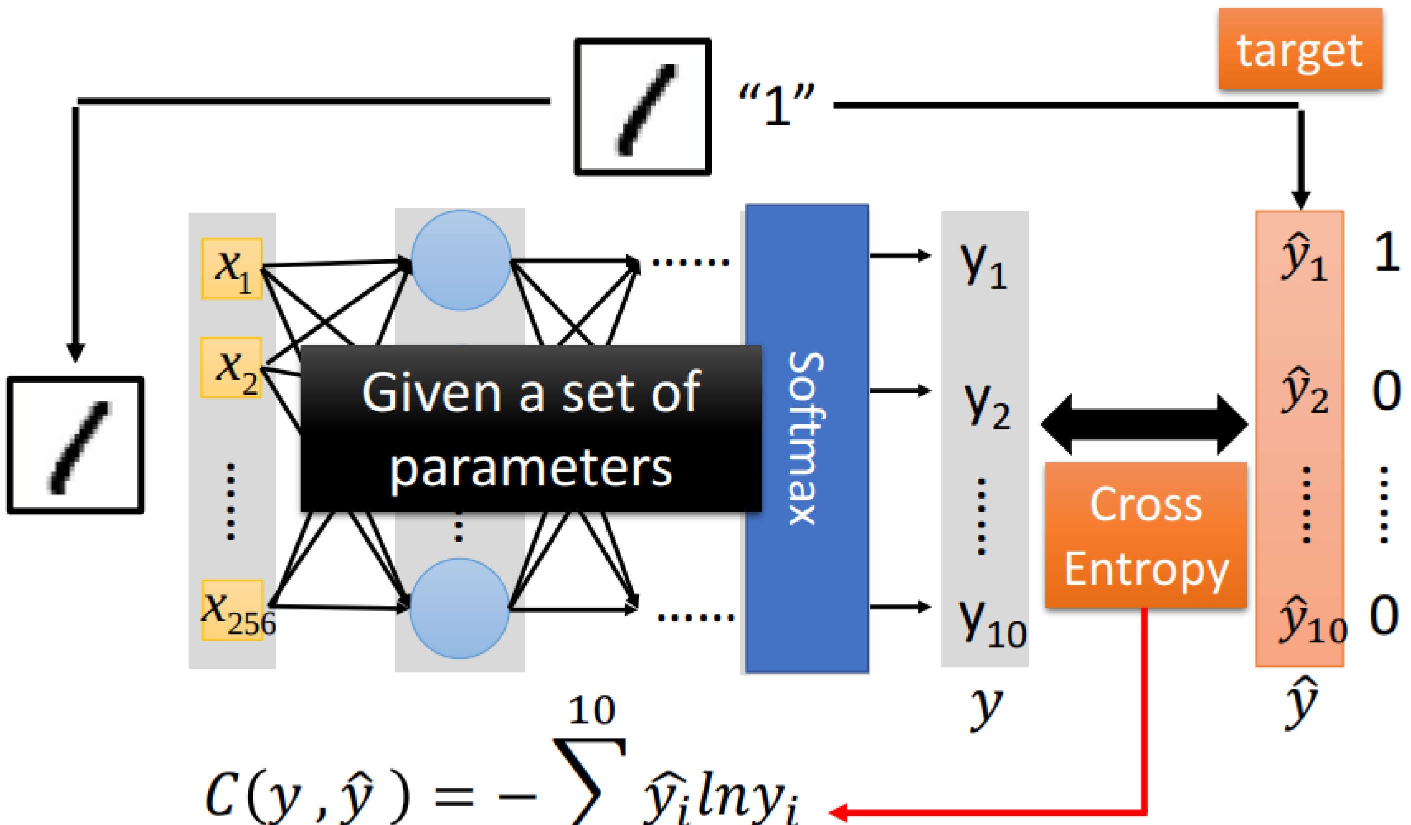


# 神经网络－ Relu

$A(x) = \max(0,x)$

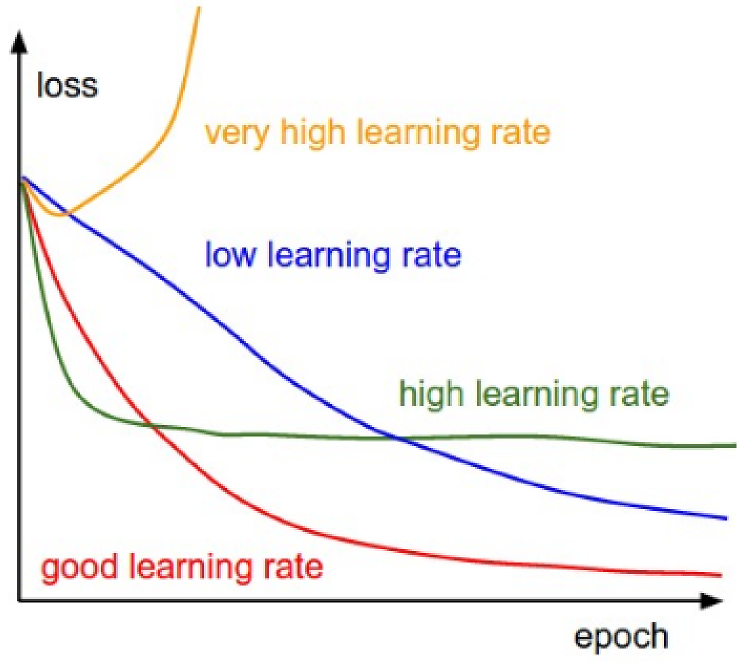


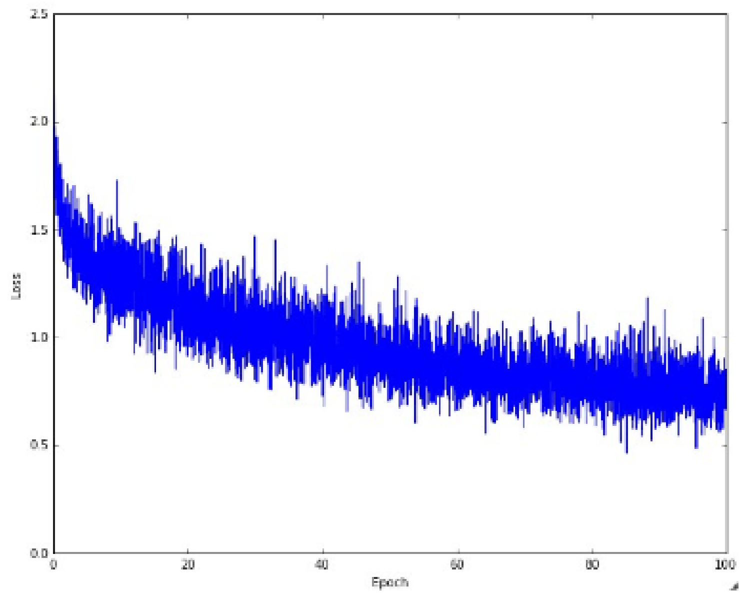
# 深度学习－3



$$C(y,\hat{y}) = -\sum_{i=1}^{10} \hat{y}_i \ln y_i$$

# 深度学习－损失评估

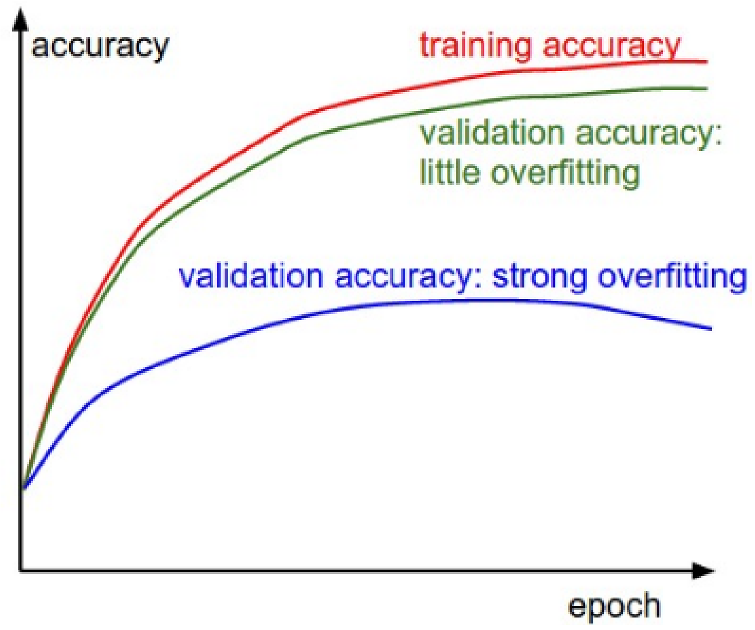

loss
very high learning rate
low learning rate
high learning rate
good learning rate
epoch


2.5
2.0
1.5
1.0
0.5
0.0
Loss
0
20
40
60
80
100
Epoch

# 准确度

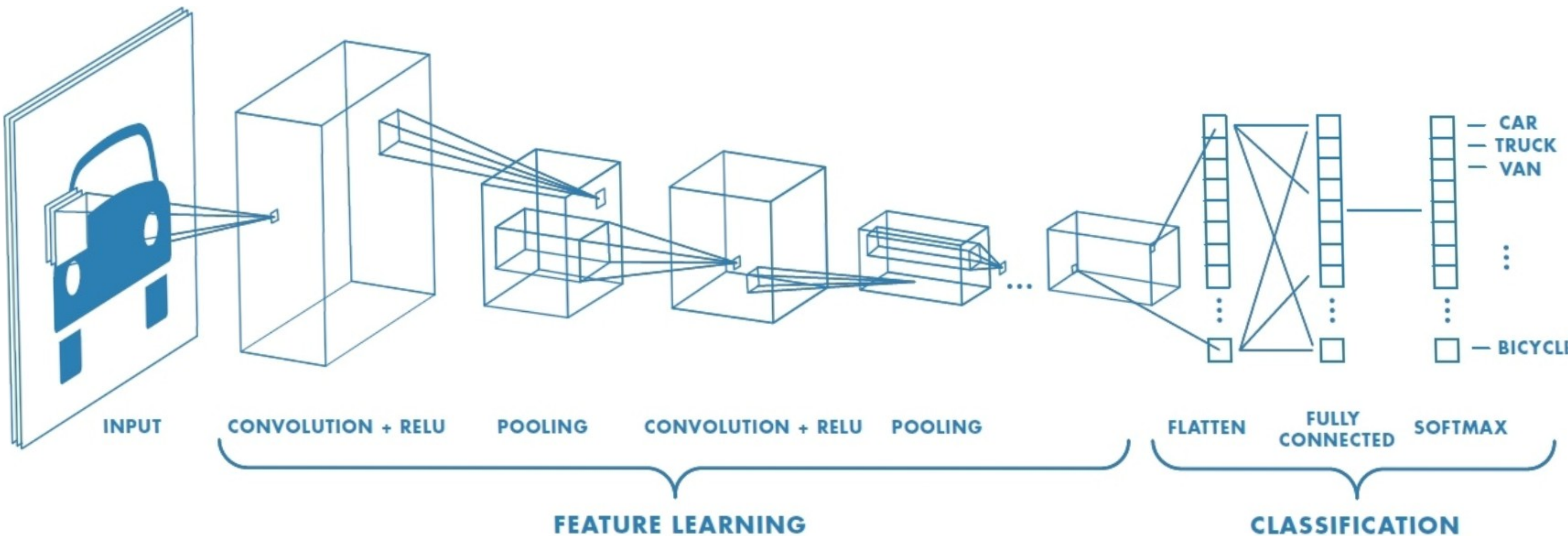
## Train/Val accuracy

The second important quantity to track while training a classifier is the validation/training accuracy. This plot can give you valuable insights into the amount of overfitting in your model:

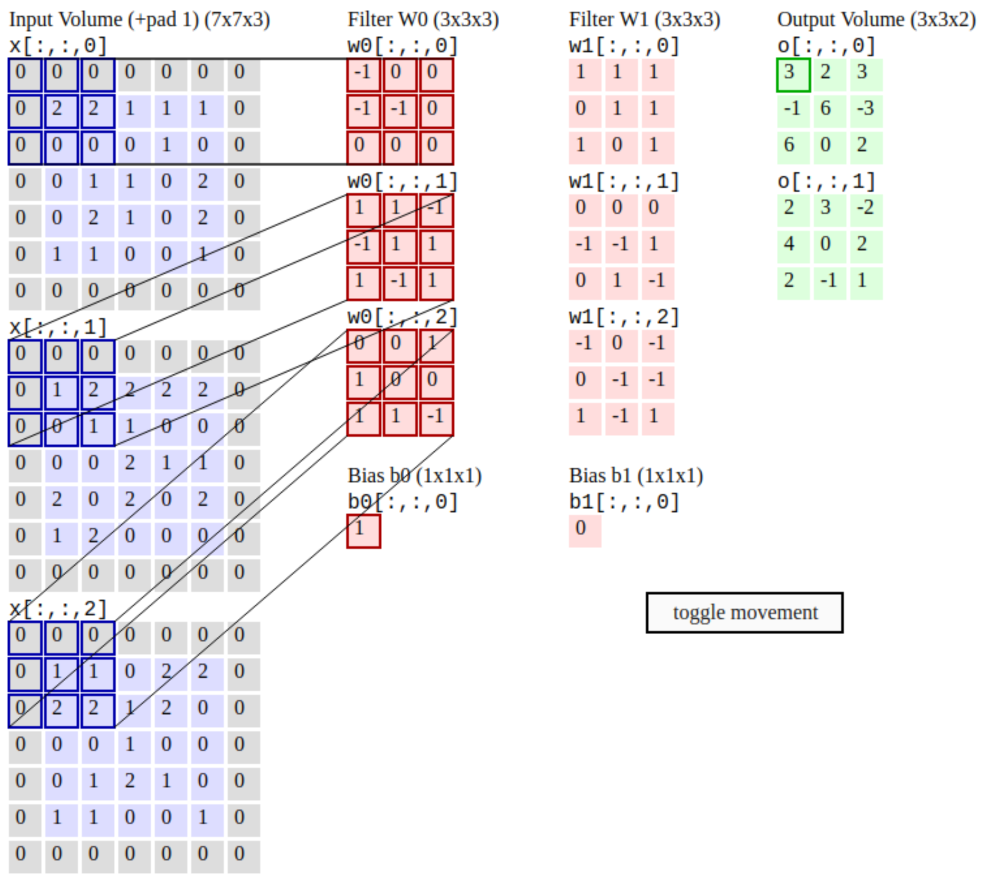


The gap between the training and validation accuracy indicates the amount of overfitting. Two possible cases are shown in the diagram on the left. The blue validation error curve shows very small validation accuracy compared to the training accuracy, indicating strong overfitting (note, it's possible for the validation accuracy to even start to go down after some point). When you see this in practice you probably want to increase regularization (stronger L2 weight penalty, more dropout, etc.) or collect more data. The other possible case is when the validation accuracy tracks the training accuracy fairly well. This case indicates that your model capacity is not high enough: make the model larger by increasing the number of parameters.

# CNN


INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
CAR
TRUCK
VAN
BICYCL
FEATURE LEARNING
CLASSIFICATION

# CNN-conv



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 2 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 0 | 0 | 2 | 1 | 0 | 2 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,1]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 2 | 2 | 2 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 2 | 1 | 1 | 0 |
| 0 | 2 | 0 | 2 | 0 | 2 | 0 |
| 0 | 1 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,2]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 2 | 2 | 0 |
| 0 | 2 | 2 | 1 | 2 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 2 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Filter W0 (3x3x3)

w0[:,:,0]

| | | |
|---|---|---|
| -1 | 0 | 0 |
| -1 | -1 | 0 |
| 0 | 0 | 0 |

w0[:,:,1]

| | | |
|---|---|---|
| 1 | 1 | -1 |
| -1 | 1 | 1 |
| 1 | -1 | 1 |

w0[:,:,2]

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | -1 |

Bias b0 (1x1x1)

b0[:,:,0]

| |
|---|
| 1 |

Filter W1 (3x3x3)

w1[:,:,0]

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

w1[:,:,1]

| | | |
|---|---|---|
| 0 | 0 | 0 |
| -1 | -1 | 1 |
| 0 | 1 | -1 |

w1[:,:,2]

| | | |
|---|---|---|
| -1 | 0 | -1 |
| 0 | -1 | -1 |
| 1 | -1 | 1 |

Bias b1 (1x1x1)

b1[:,:,0]

| |
|---|
| 0 |

Output Volume (3x3x2)

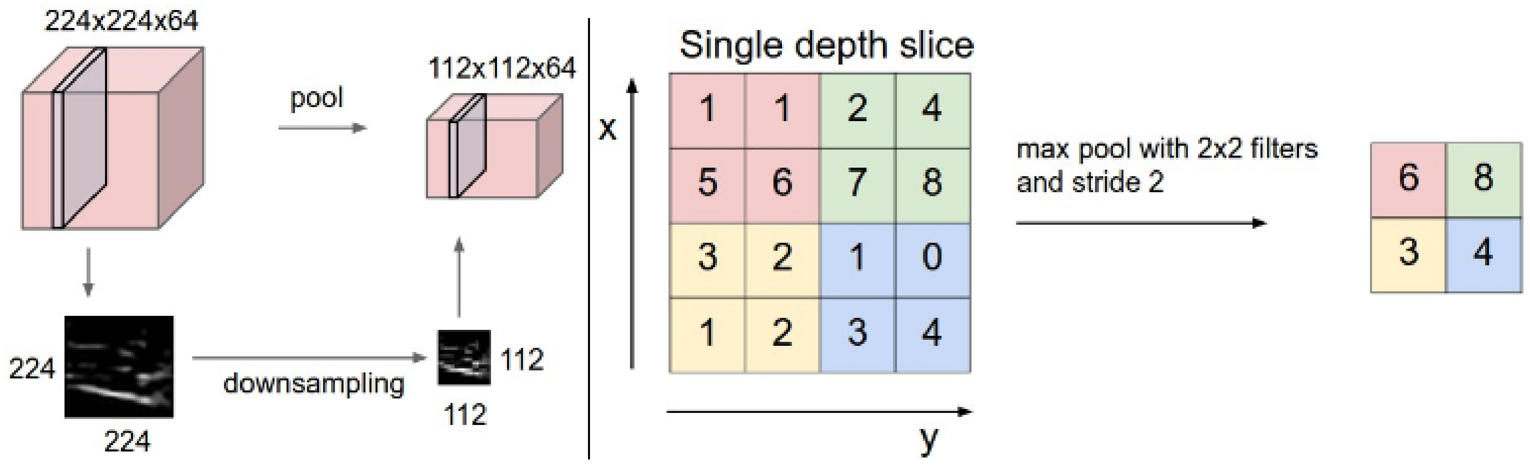
o[:,:,0]

| | | |
|---|---|---|
| 3 | 2 | 3 |
| -1 | 6 | -3 |
| 6 | 0 | 2 |

o[:,:,1]

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

# CNN-pooling


224x224x64
pool
112x112x64
224
224
downsampling
112
112
Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters and stride 2
6 8
3 4

# CNN

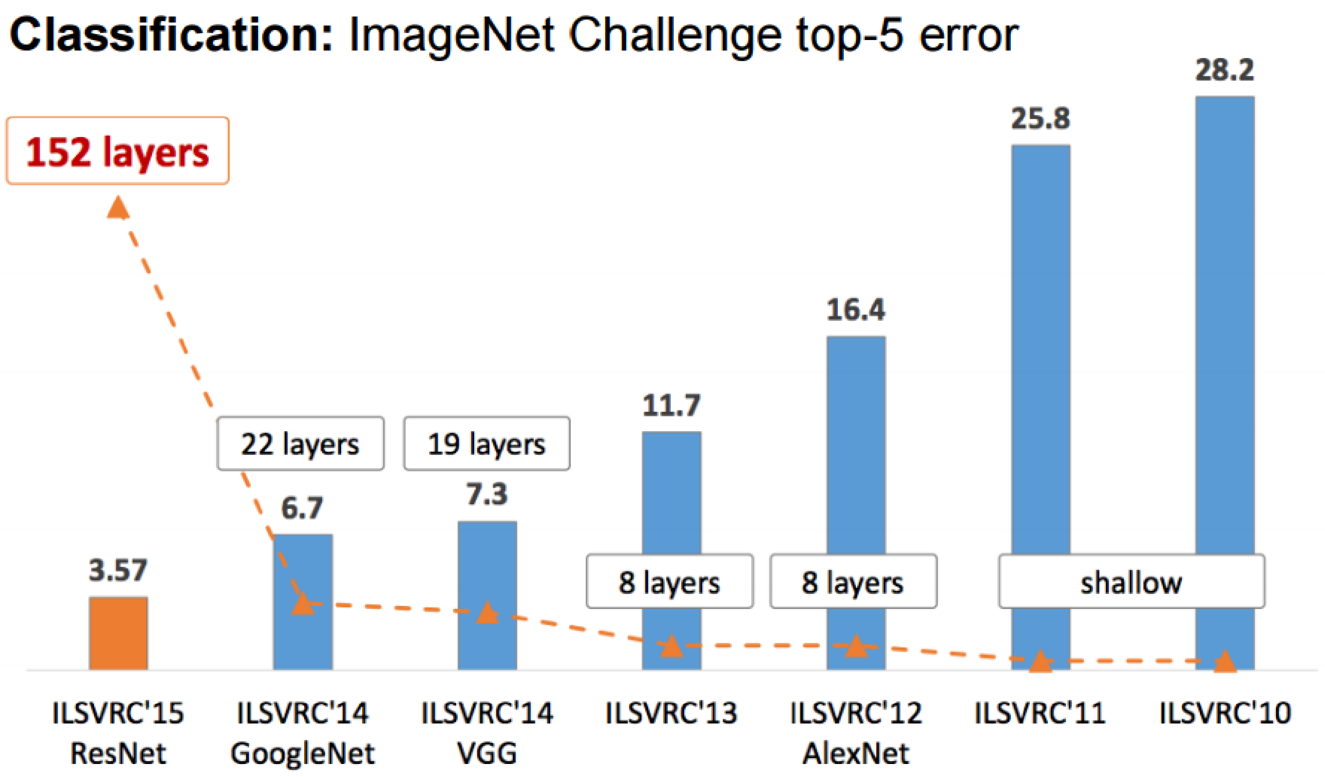
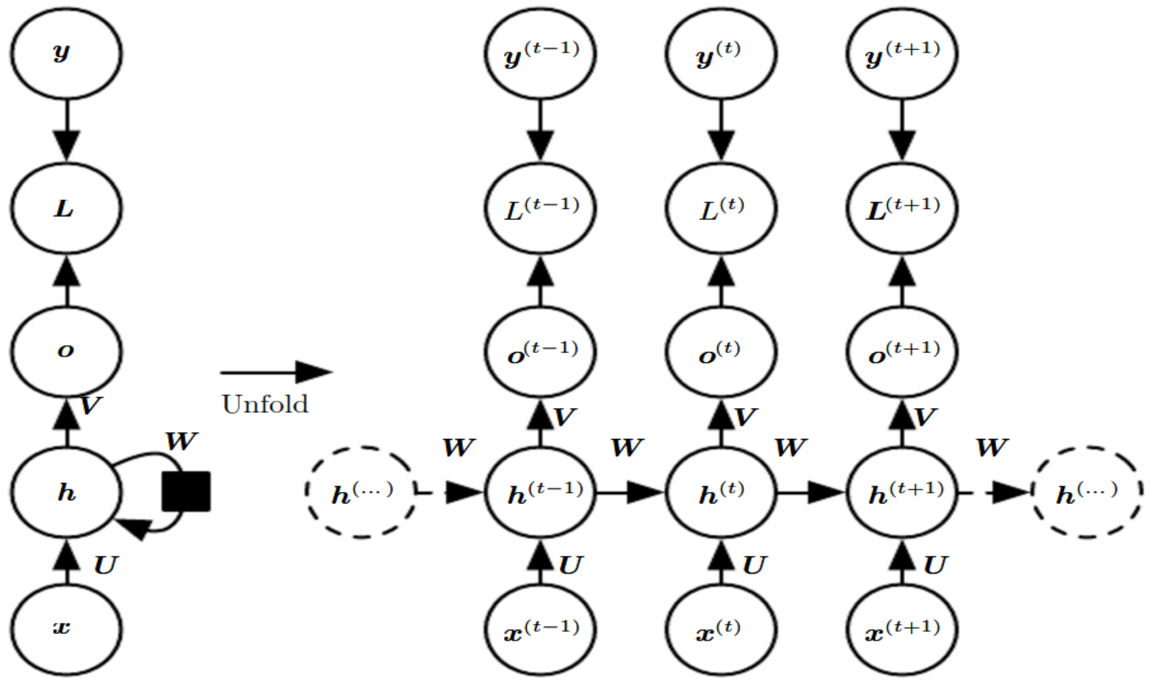


Figure source: [Kaiming He]

# RNN



$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}),$$

# RNN

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \tag{10.8}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}), \tag{10.9}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \tag{10.10}$$

$$\hat{\boldsymbol{y}}^{(t)} = \operatorname{softmax}(\boldsymbol{o}^{(t)}), \tag{10.11}$$

- T h a n k s