

深度学习：入门、前沿、讨论

吴伟

lazyparser@gmail.com
中国科学院软件研究所

关于我

软件研究所 | 神经芯片 | 即时编译 | 机器人 | 人脸识别

开源贡献者 | HelloGCC | Mozilla | MXNet | TK1



http://img1.ph.126.net/4wrvmokOs9Xd_sQforoLGA==/674695519193009451.jpg

基础 | 前沿 | 讨论

深度学习的定义随着时间在演变

Deep Learning: machine learning algorithms based on learning multiple levels of representation / abstraction.

Deep Learning

NIPS'2015 Tutorial

Geoff Hinton, Yoshua Bengio & Yann LeCun



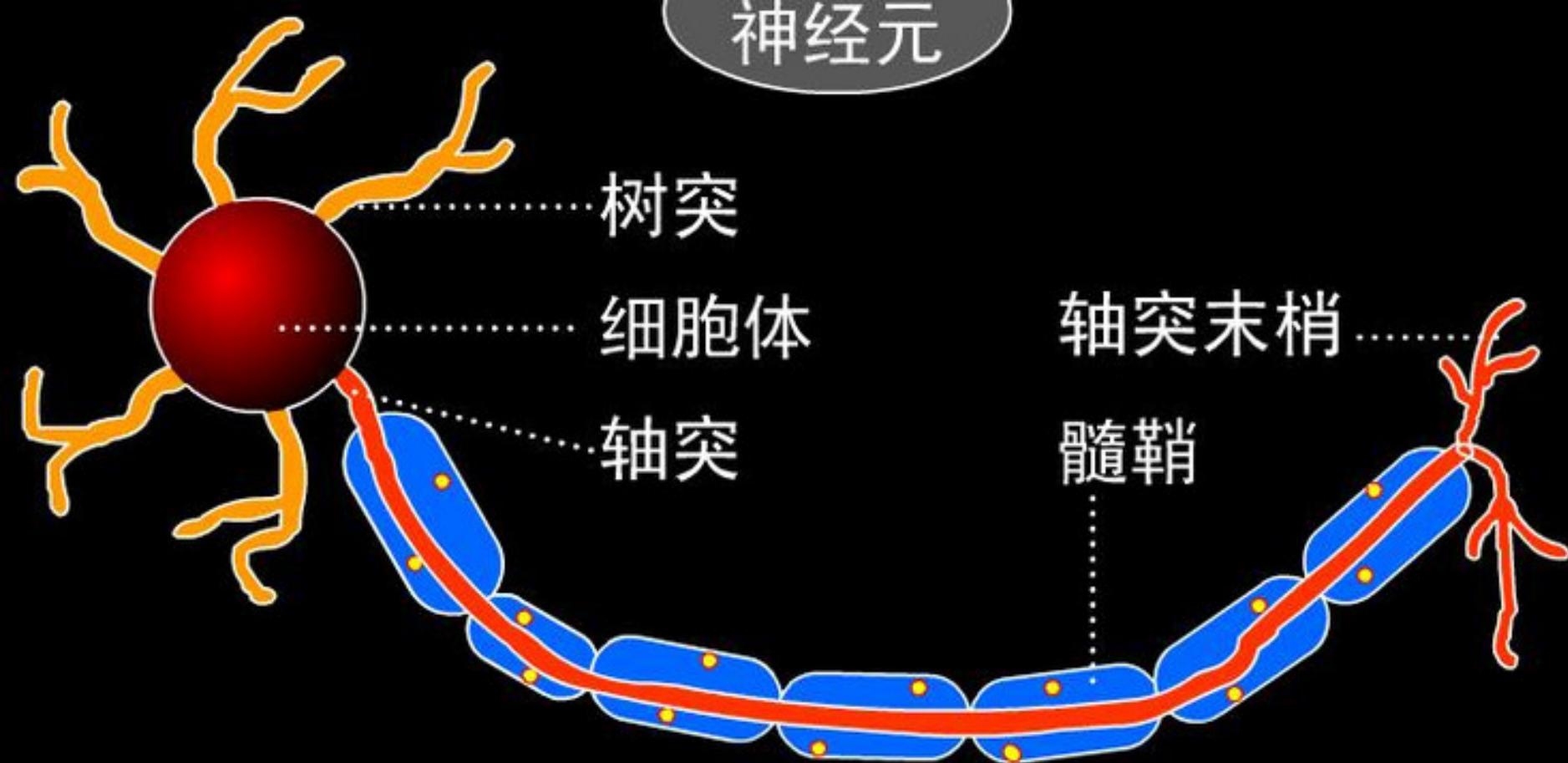
先缩小一下入门的范围

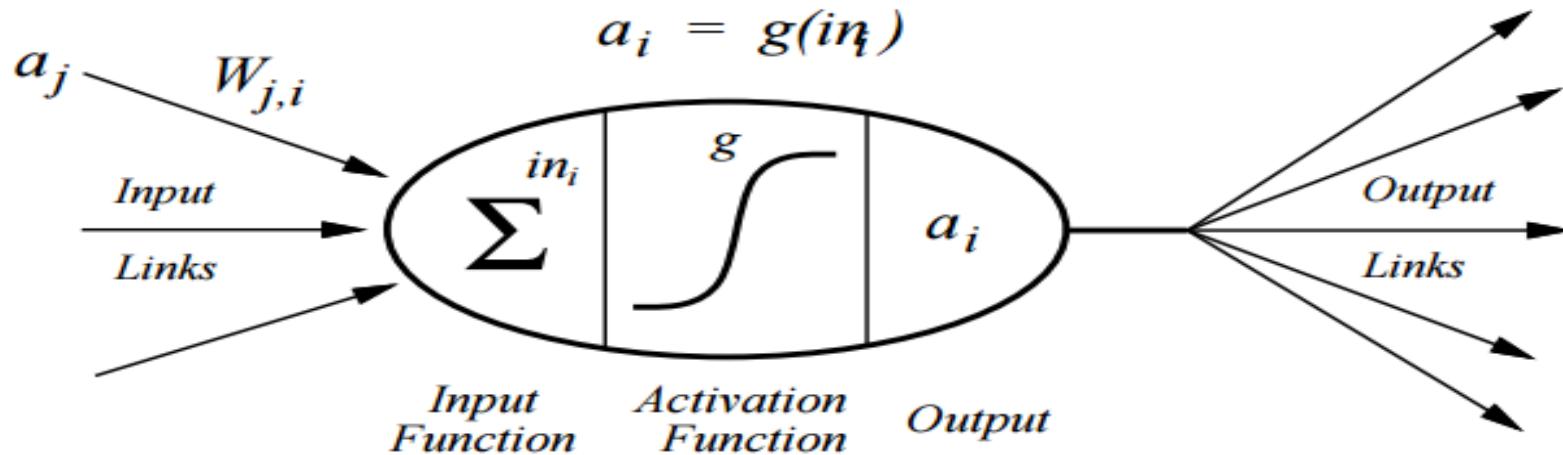
深度学习： 深度神经网络

先缩小一下入门的范围

深度学习： 深度**神经**网络

神经元





$$a_i = g\left(\sum_j W_{j,i} a_j\right)$$

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}, \mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}. \quad (1)$$

$$E(\mathbf{v}, \mathbf{h}) = - \sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j, \quad (2)$$

$$\begin{aligned} p(\mathbf{v}|\mathbf{h}) &= \prod_i p(v_i|\mathbf{h}) \quad \text{and} \quad p(v_i = 1|\mathbf{h}) = \text{sigm} \left(a_j + \sum_j h_j w_{ij} \right), \\ p(\mathbf{h}|\mathbf{v}) &= \prod_j p(h_j|\mathbf{v}) \quad \text{and} \quad p(h_j = 1|\mathbf{v}) = \text{sigm} \left(b_j + \sum_i v_i w_{ij} \right), \end{aligned} \quad (3)$$

$$\begin{aligned} p(v_i = x|\mathbf{h}) &= \frac{1}{\sigma_i \sqrt{2\pi}} \cdot e^{-\frac{(x - a_i - \sigma_i \sum_j w_{ij} h_j)^2}{2\sigma_i^2}}, \\ p(h_j = 1|\mathbf{v}) &= \text{sigm} \left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij} \right). \end{aligned} \quad (4)$$

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}, \mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}. \quad (1)$$

$$E(\mathbf{v}, \mathbf{h}) = - \sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j, \quad (2)$$

$$\begin{aligned} p(\mathbf{v}|\mathbf{h}) &= \prod_i p(v_i|\mathbf{h}) \quad \text{and} \quad p(v_i = 1|\mathbf{h}) = \text{sigm} \left(a_j + \sum_j h_j w_{ij} \right), \\ p(\mathbf{h}|\mathbf{v}) &= \prod_j p(h_j|\mathbf{v}) \quad \text{and} \quad p(h_j = 1|\mathbf{v}) = \text{sigm} \left(b_j + \sum_i v_i w_{ij} \right), \end{aligned} \quad (3)$$

$$\begin{aligned} p(v_i = x|\mathbf{h}) &= \frac{1}{\sigma_i \sqrt{2\pi}} \cdot e^{-\frac{(x - a_i - \sigma_i \sum_j w_{ij} h_j)^2}{2\sigma_i^2}}, \\ p(h_j = 1|\mathbf{v}) &= \text{sigm} \left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij} \right). \end{aligned} \quad (4)$$

故事背景

老板希望设计开发一款**人工智能**预测软件，
利用当天股价的涨跌来预测第二天房价的涨跌。

故事背景 找来小李实现这个app:

if 股价涨

then 显示房价涨；

else 显示房价不会涨；

产品原型

```
if (x > 0)
    y = 1
else /* x <= 0 */
    y = 0
```

产品原型

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

```
if (x > 0)
    y = 1
else /* x <= 0 */
    y = 0
```

产品原型（高级版）

```
if (x > 0)
```

```
y = x
```

```
else /* x <= 0 */
```

```
y = 0
```

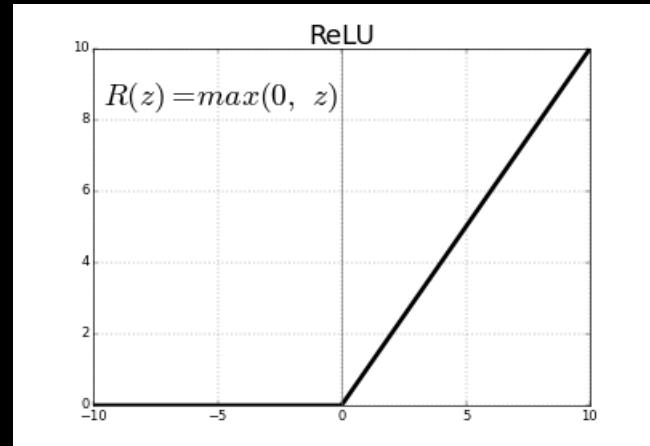
产品原型（高级版）

```
if (x > 0)
```

```
y = x
```

```
else /* x <= 0 */
```

```
y = 0
```



产品原型 1.0

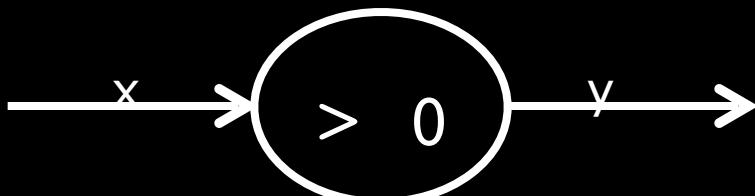
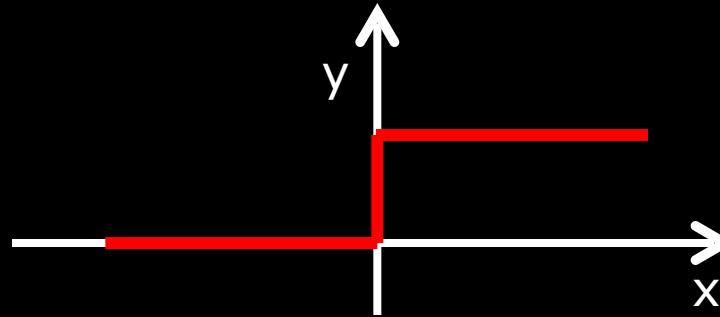
Heaviside

```
if (x > 0)
```

```
    y = 1
```

```
else
```

```
    y = 0
```



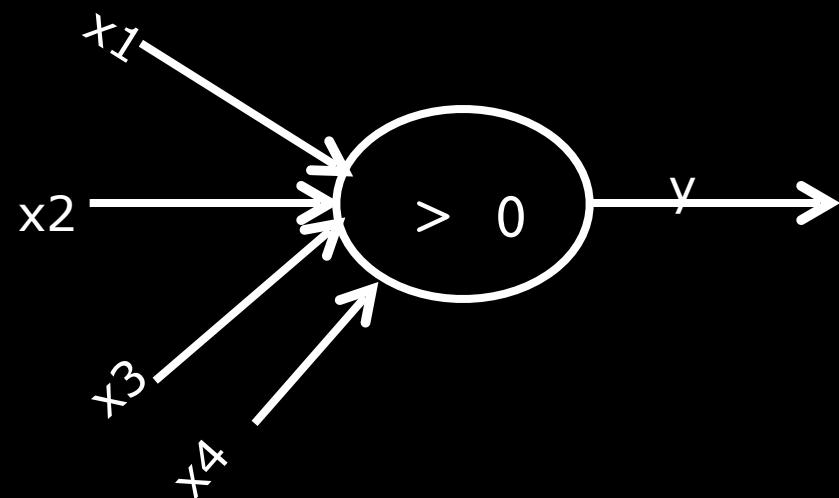
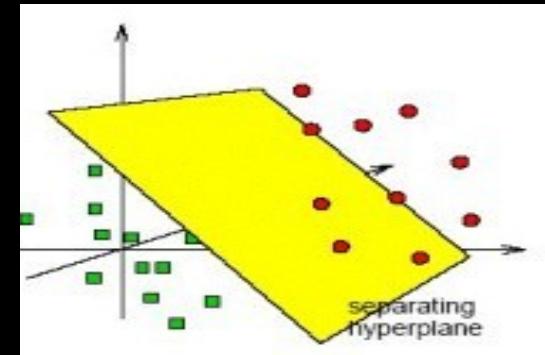
故事背景

原型做好了，但是显然，预测不够准：）

老板觉得应该把油价、汇率、金价都考虑进去...

产品原型 2.0

```
foo(x1,x2,x3,x4) {  
    return  
    x1+x2+x3+x4;  
}  
if (foo(x) > 0)  
    y = 1  
else  
    y = 0
```



故事背景

产品 2.0 完成，还是不准

老板：股价、油价、汇率、金价对房价的影响程度不一样，也就是权重 (weights) 是不同的。

故事背景

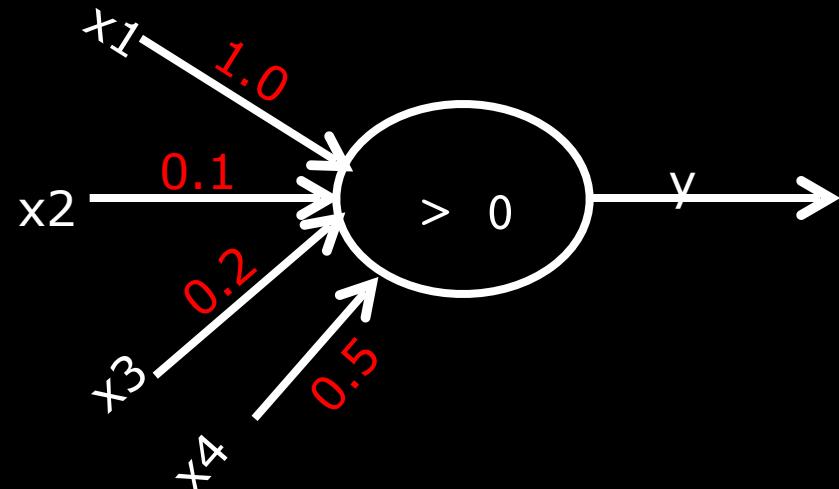
小李不知道权重怎么定，老板凭经验定了一套

股价、油价、汇率、金价的权重分别是

1.0, 0.1, 0.2, 0.5

产品原型 3.0

```
foo(x1,x2,x3,x4) {  
    return 1.0*x1 + 0.1*x2  
        + 0.2*x3 + 0.5*x4;  
}  
  
if (foo(x) > 0)  
    y = 1  
else  
    y = 0
```



故事背景

老板：长远看，基本面不变，房价也在缓慢上涨。

所以指标跌到一定程度，才能预测跌。

老板凭经验定了个 -0.5

产品原型 4.0

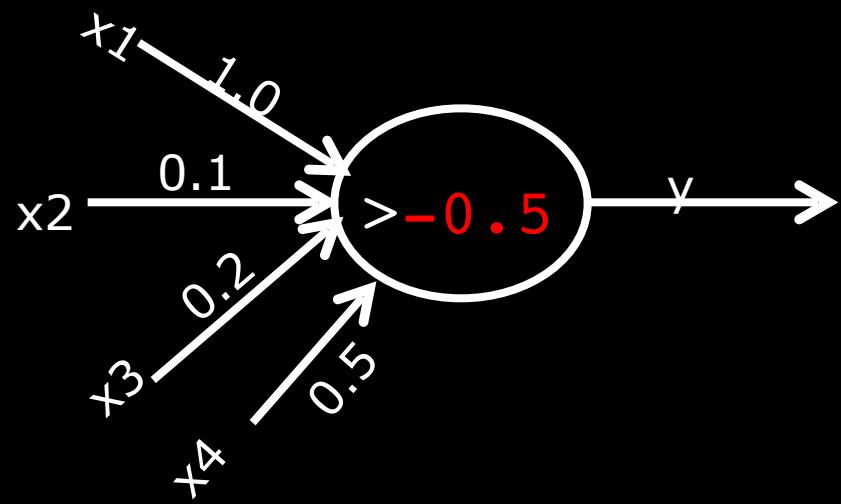
```
foo(x1,x2,x3,x4) {  
    return 1.0*x1 + 0.1*x2  
        + 0.2*x3 + 0.5*x4;  
}
```

```
if (foo(x) > -0.5)
```

```
    y = 1
```

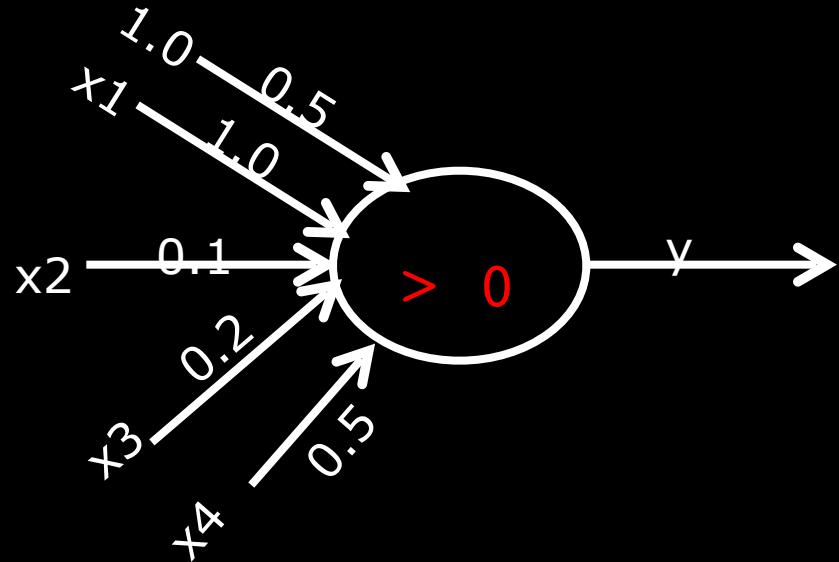
```
else
```

```
    y = 0
```



产品原型 4.1

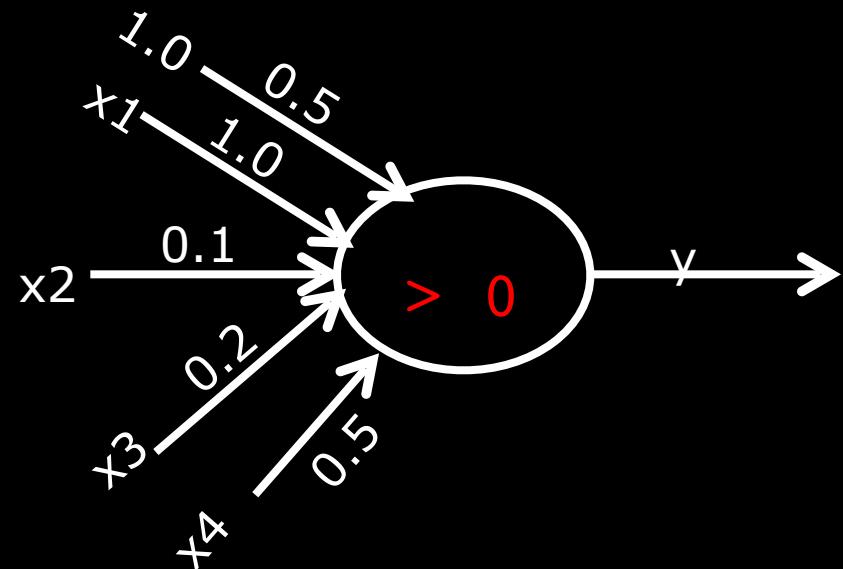
```
foo(x1,x2,x3,x4) {  
    return 0.5*x1 + 1.0*x1 + 0.1*x2  
        + 0.2*x3 + 0.5*x4;  
}  
  
if (foo(x) > 0)  
    y = 1  
else  
    y = 0
```



产品原型 4.2

```
w = [+0.5, 1.0, 0.1, 0.2, 0.5]
x = [ 1, x1, x2, x3, x4 ]
foo(x) { return sum(dot_mult(w,x)) }

if (foo(x) > 0)
    y = 1
else
    y = 0
```

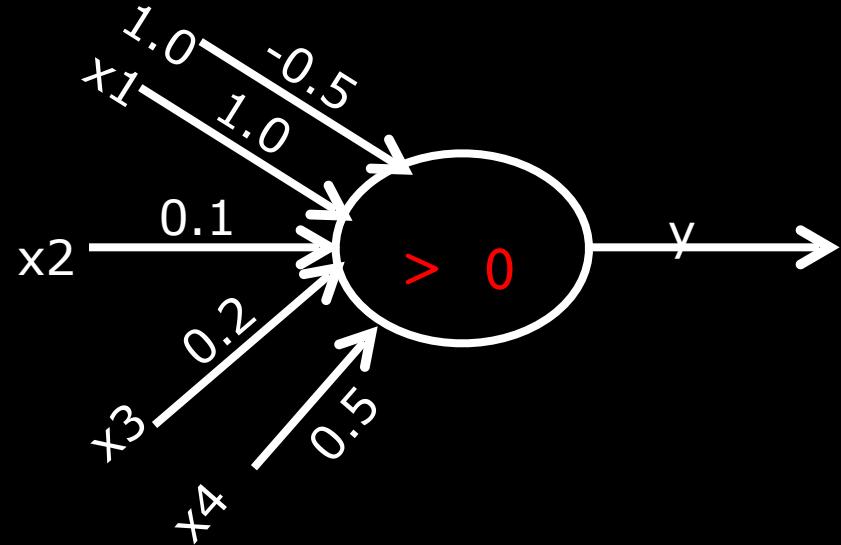


产品原型 4.2

```
w = [+0.5, 1.0, 0.1, 0.2, 0.5]
x = [    1,  x1,  x2,  x3,  x4 ]
foo(x) { return sum(dot_mult(w,x)) }
```

```
if (foo(x) > 0)
    y = 1
else
    y = 0
```

激活函数
Activation function

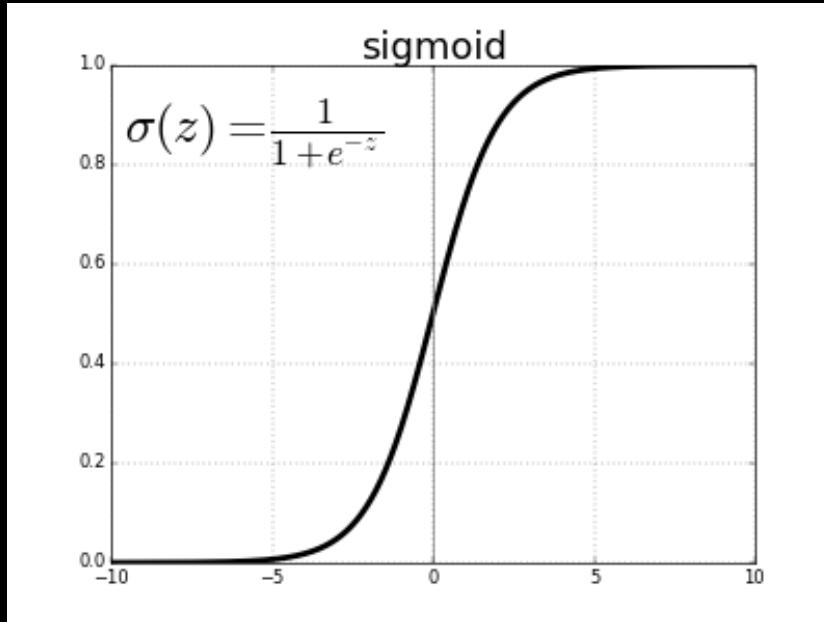


故事背景

迭代四个大版本了，还是不准。

老板：线性关系不能有效的表示这种复杂的关系，
我们要用“非线性关系”，用最流行的Sigmoid。

Sigmoid 函数（两种解释，类比HTML5）



神经网络用Sigmoid函数，其实是为了后面反向传播算法的时候计算微分方便

产品原型 5.0

```
w = [+0.5, 1.0, 0.1, 0.2, 0.5]  
x = [ 1, x1, x2, x3, x4 ]
```

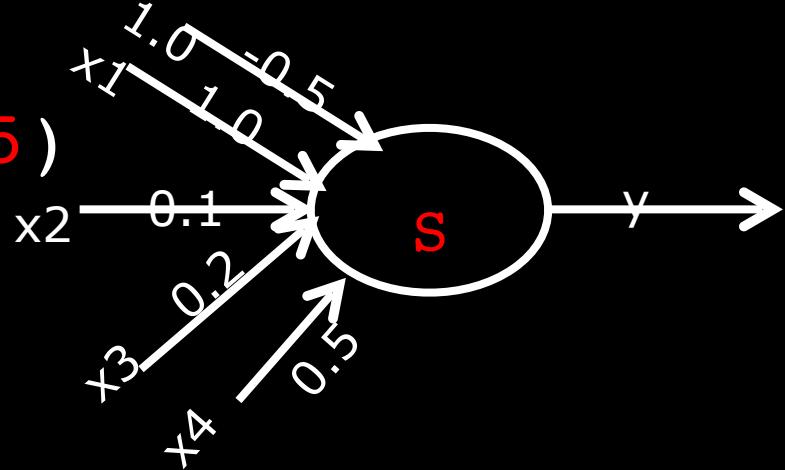
```
foo(x) { return sum(dot_mult(w,x)) }
```

```
if (sigmoid(foo(x))>0.5)
```

```
    y = 1
```

```
else
```

```
    y = 0
```



激活 (Activation) 函数大家族

Name	Plot	Equation	Derivative (with respect to x)	Range	Order of continuity	Monotonic	Derivative Monotonic	Approximates identity near the origin
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}	Yes	No	No
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞	Yes	No	No
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞	Yes	No	Yes
Arctan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$	C^∞	Yes	No	Yes
Softsign [7][8]		$f(x) = \frac{x}{1 + x }$	$f'(x) = \frac{1}{(1 + x)^2}$	$(-1, 1)$	C^1	Yes	No	Yes
Rectifier (ReLU) ^[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$	C^0	Yes	Yes	No
Parameteric Rectified Linear Unit (PReLU) ^[10]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes iff $\alpha \geq 0$	Yes	Yes iff $\alpha = 1$
Exponential Linear Unit (ELU) ^[11]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$	C^1 when $\alpha = 1$	Yes iff $\alpha \geq 0$	Yes iff $0 \leq \alpha \leq 1$	Yes iff $\alpha = 1$
SoftPlus ^[12]		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, \infty)$	C^∞	Yes	Yes	No
Bent identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2 + 1}} + 1$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes
SoftExponential ^[13]		$f(\alpha, x) = \begin{cases} -\frac{\log_e(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1 - \alpha(x + \alpha)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes iff $\alpha = 0$
Sinusoid		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$	C^∞	No	No	Yes

故事背景

5.0版还是不准。

老板：一定是模型太简单了，要加层。

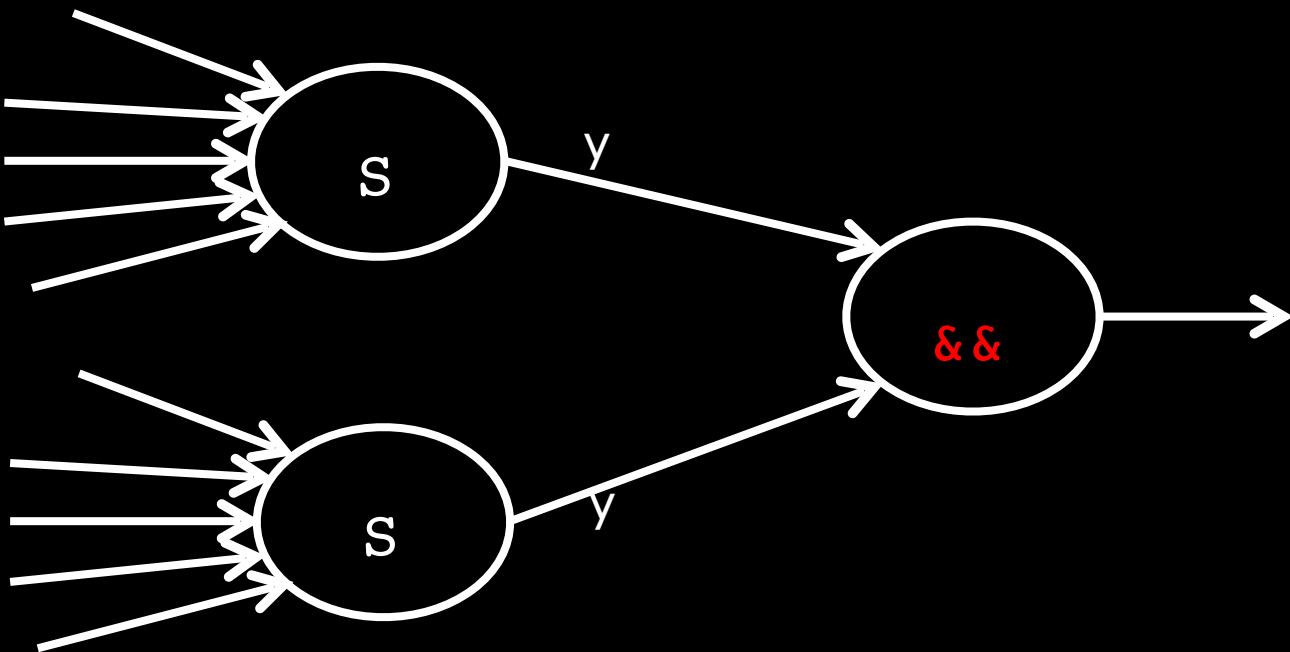
现在深度学习，层数动辄“一百多层”。

故事背景

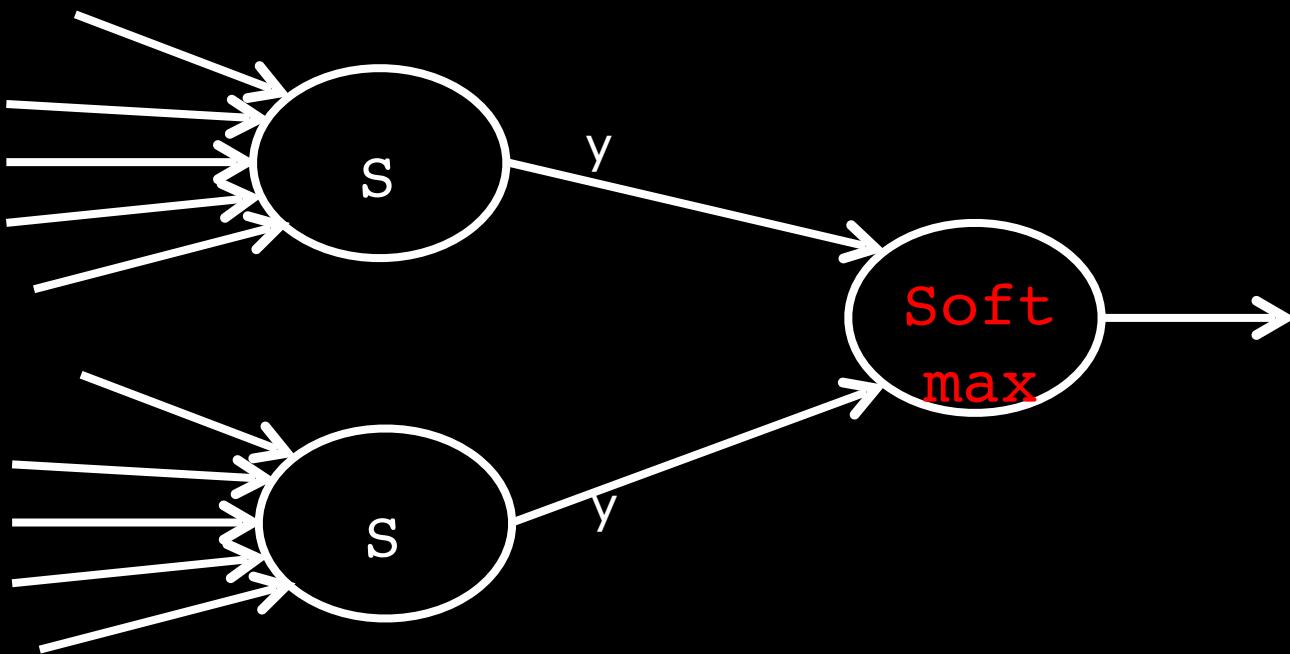
老板：买房乃大宗消费，考虑甚多，不会“一刀切”

应该用多个不同的计算公式，综合给出预测结果。

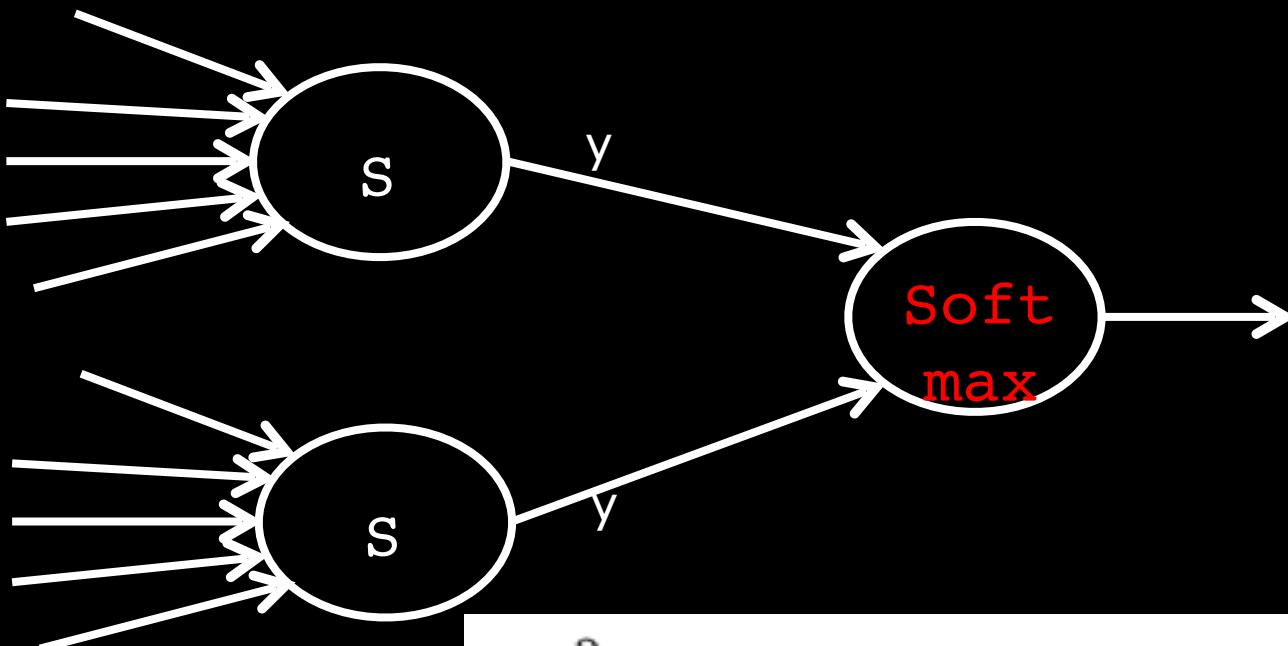
小李很快复制粘贴出来了6.0版，全说涨才涨



照猫画虎将最后的投票函数改成了 softmax



照猫画虎将最后的投票函数改成了 softmax



$$\frac{\partial}{\partial q_k} \sigma(\mathbf{q}, i) = \dots = \sigma(\mathbf{q}, i)(\delta_{ik} - \sigma(\mathbf{q}, k))$$

故事背景

有神经元，有层，是神经网络了！但是.....

两个输入的神经元，权重一模一样.....

老板让小李自己也编一套放进去

故事背景

现在小李手里有神经网络（模型），有房价、
股价、油价、汇率和金价的历史数据（训练集），
和老板给的经验权重（初始值）。

小李编了几个权重，预测都不准。接下来怎么办？

大神出场：反向传播算法&BP神经网络

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California,
San Diego, La Jolla, California 92093, USA

† Department of Computer Science, Carnegie-Mellon University,
Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure¹.

more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

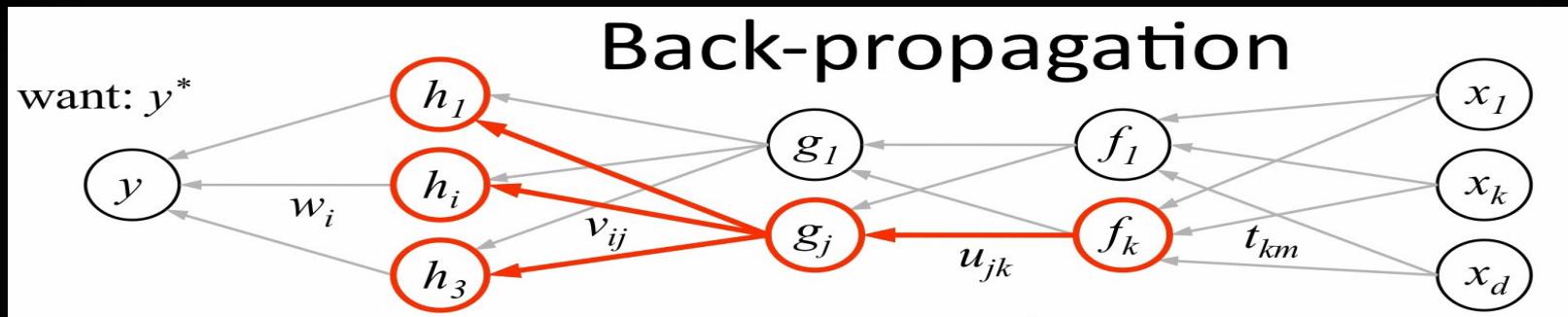
The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input x_i to unit i is a linear function of the outputs

思路：用历史数据来修正权重

11.1	11.2	11.3	11.4	11.30
股价	涨	跌	涨	涨
汇率	跌	涨	跌	跌
金价	涨	跌	涨	涨
油价	跌	涨	跌	跌
房价 (有监督)	涨	涨	涨	涨	涨

(不理解没关系，该算法已经不用自己实现了)



<https://www.youtube.com/watch?v=An5z8lR8asY>

<https://i.ytimg.com/vi/An5z8lR8asY/maxresdefault.jpg>

进一步学习：吴恩达在Coursera上的机器学习课程

Backpropagation algorithm

> Training set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m \leftarrow $(\underline{x}^{(i)}, \underline{y}^{(i)})$

Set $a^{(1)} = \underline{x}^{(i)}$

Perform forward propagation to compute $\underline{a}^{(l)}$ for $l = 2, 3, \dots, L$

Using $\underline{y}^{(i)}$, compute $\delta^{(L)} = \underline{a}^{(L)} - \underline{y}^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

(use to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)

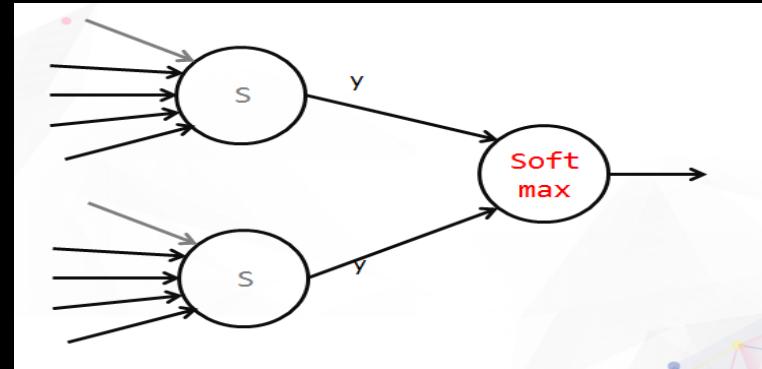
$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + \delta^{(l+1)} (\underline{a}^{(l)})^T.$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

故事背景

训练完的效果确实比以前好了很多，但是.....

权重还是都是一样的嘛！



不起眼但是很重要：随机初始化

随机化在训练中经常需要被使用到

Backprop in Practice Y LeCun

- Use ReLU non-linearities
- Use cross-entropy loss for classification
- Use Stochastic Gradient Descent on minibatches
- Shuffle the training samples (← very important)
- Normalize the input variables (zero mean, unit variance)
- Schedule to decrease the learning rate
- Use a bit of L1 or L2 regularization on the weights (or a combination)
 - ▶ But it's best to turn it on after a couple of epochs
- Use "dropout" for regularization
- Lots more in [LeCun et al. "Efficient Backprop" 1998]
- Lots, lots more in "Neural Networks, Tricks of the Trade" (2012 edition) edited by G. Montavon, G. B. Orr, and K-R Müller (Springer)
- More recent: Deep Learning (MIT Press book in preparation)

故事背景

准确率上去了。老板很开心，问小李权重的含义。

小李表示不知道，于是去问各路大神.....



“不知道”



“很复杂”

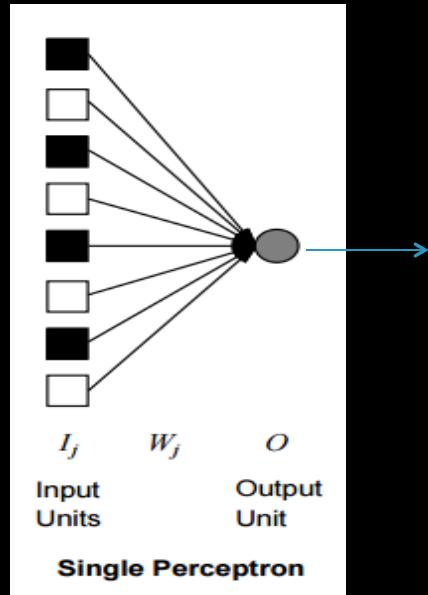


“不好说”

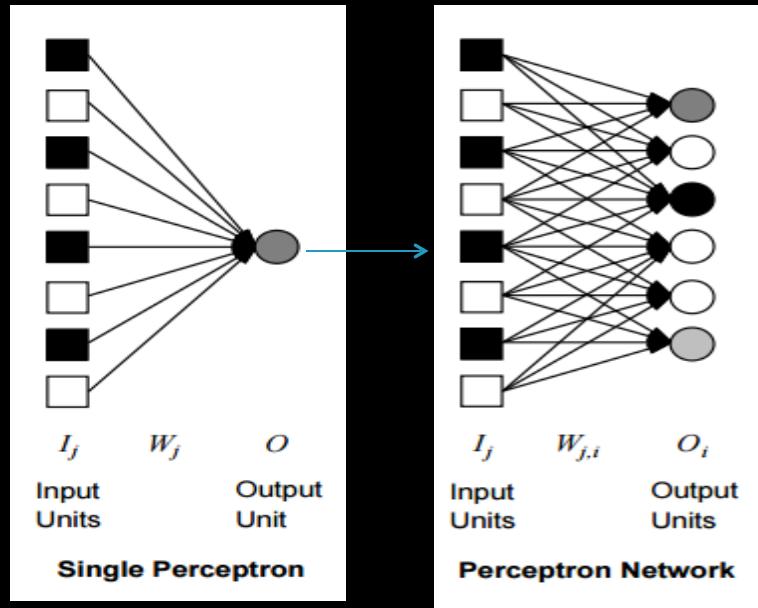


“有点难”

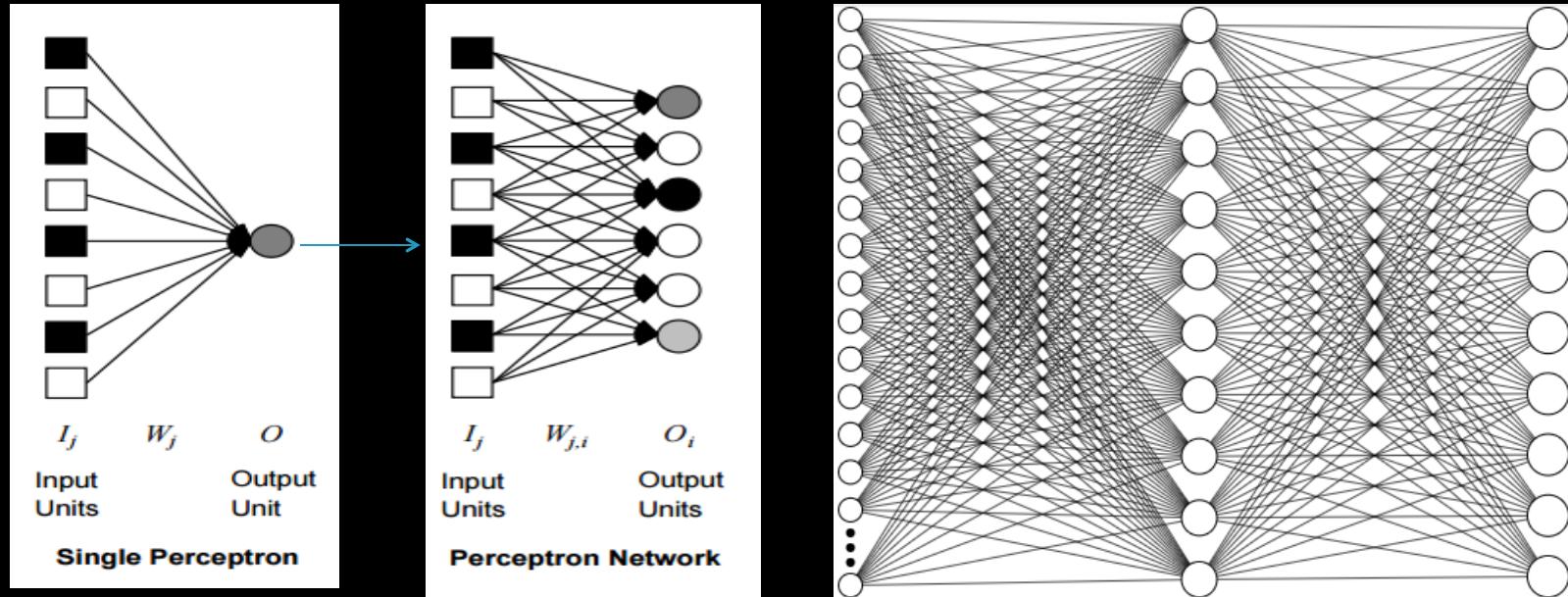
算法在手，层数你有



算法在手，层数你有



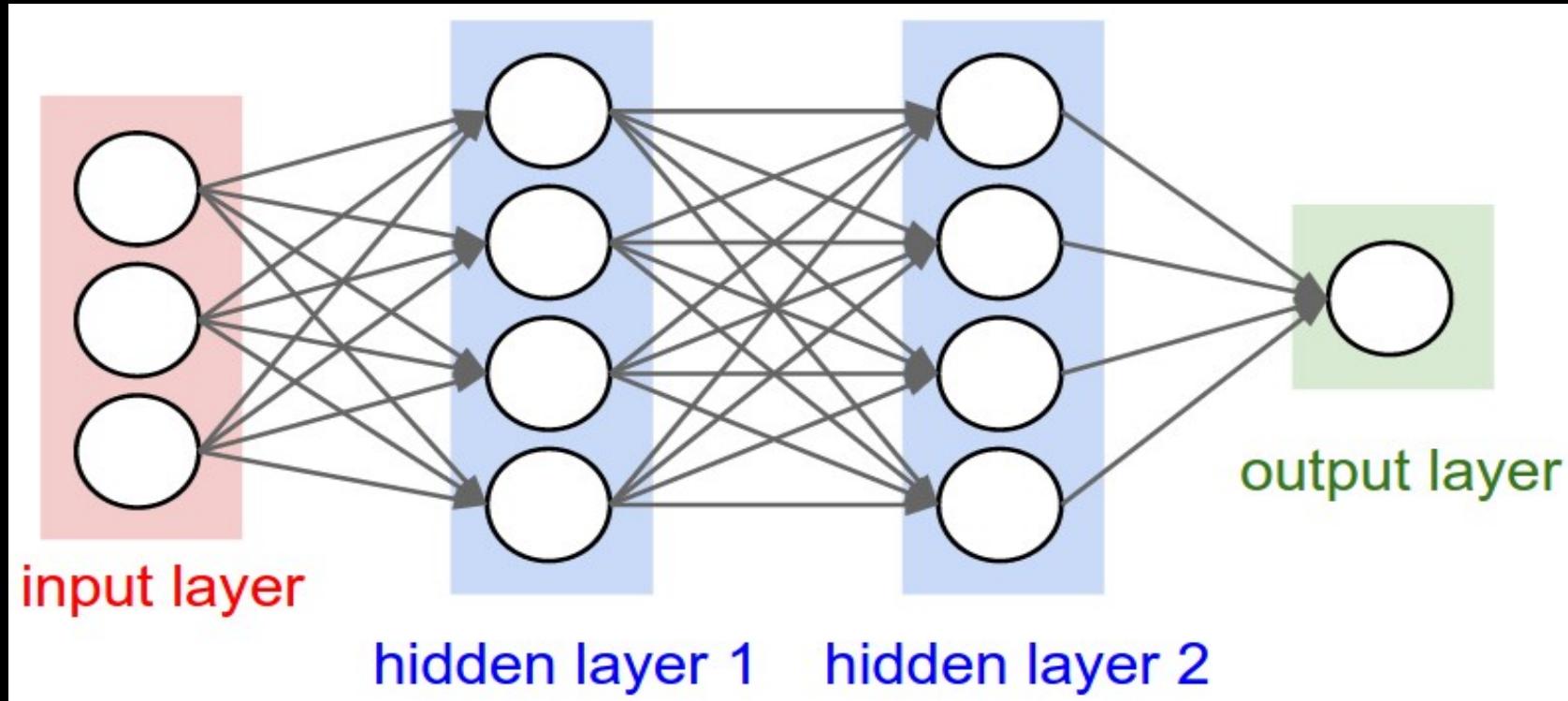
算法在手，层数你有



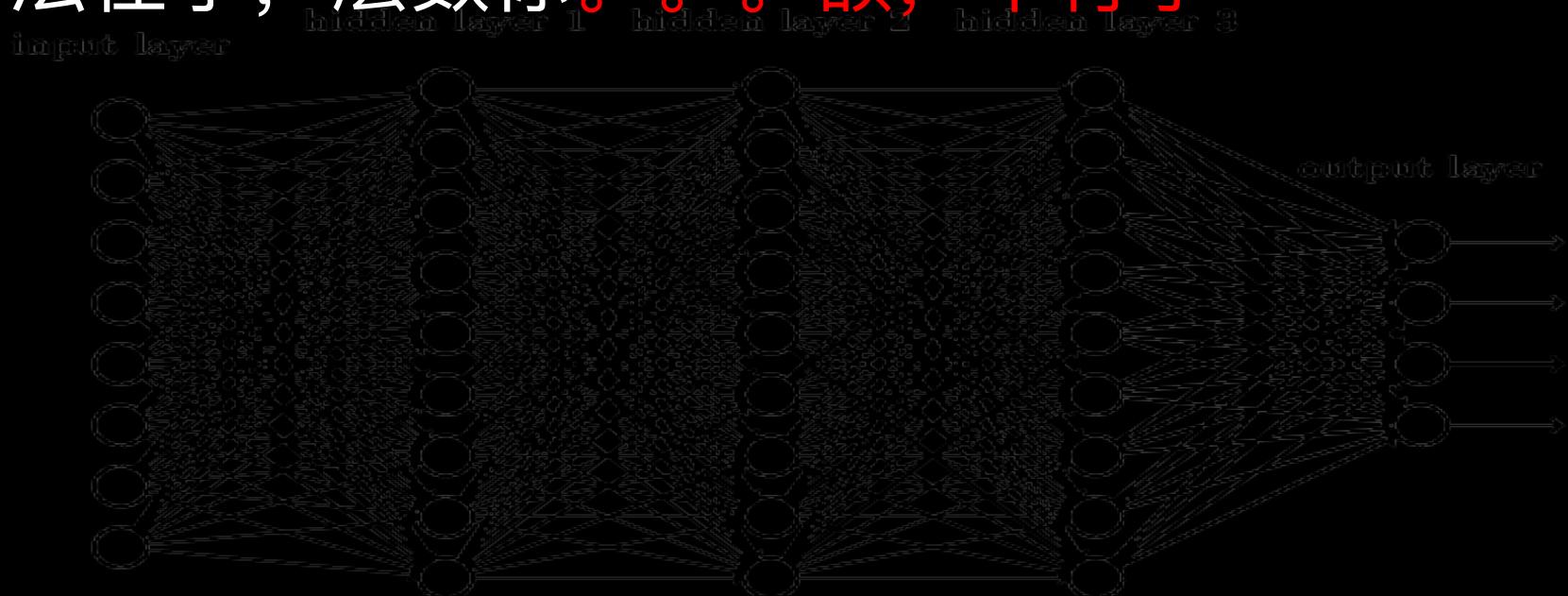
https://en.wikipedia.org/wiki/Artificial_neural_network

<https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-f01/www/handouts/110601.pdf>

算法在手，层数你有



算法在手，层数你。。。额，不行了



不行了：反向传播算法的局限性

梯度消失

局部最优

过学习

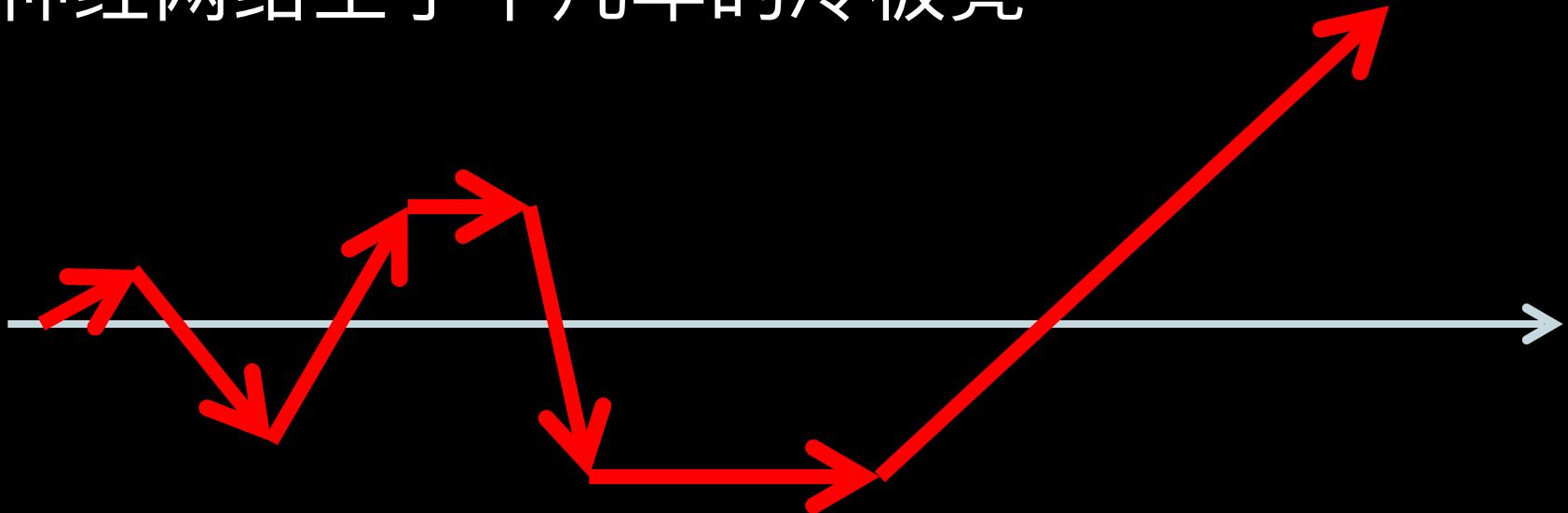
过学习 (Overfit) 和欠学习 (Underfit)

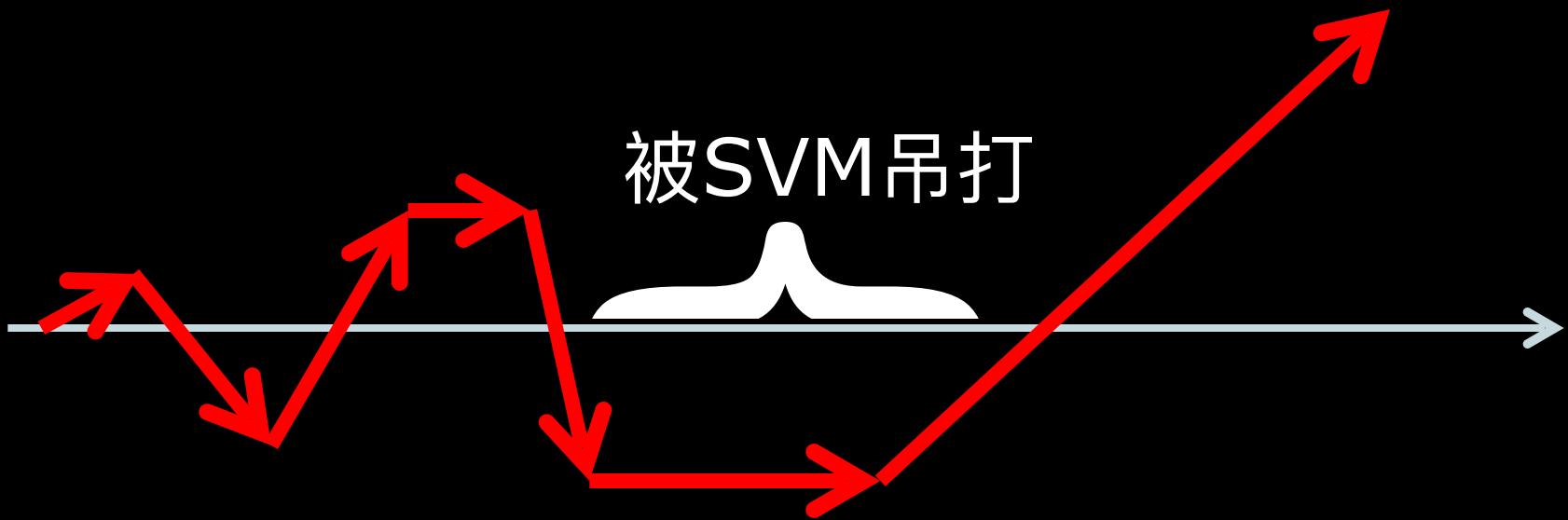
给我四个参数，我能拟合出来一头大象。
给我五个，我就能让大象的鼻子动起来。

-- 冯·诺依曼



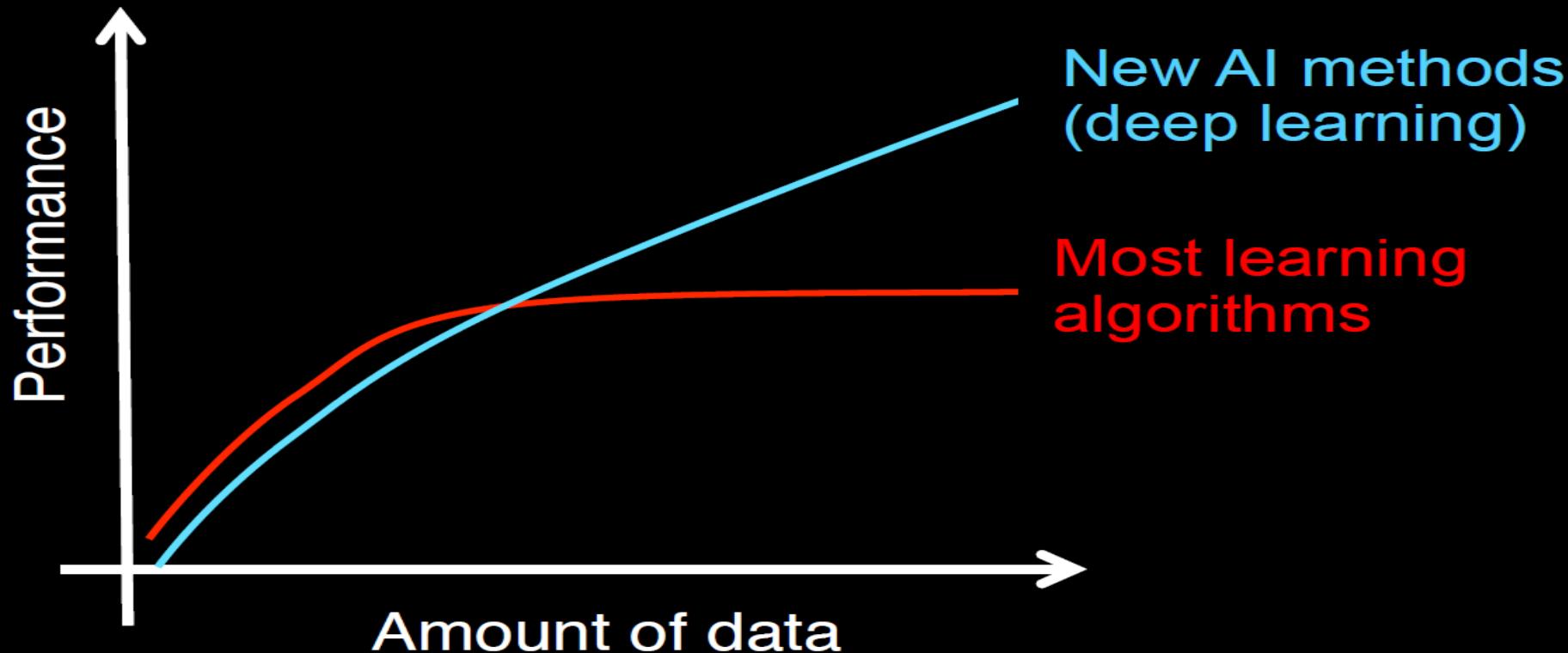
神经网络坐了十几年的冷板凳

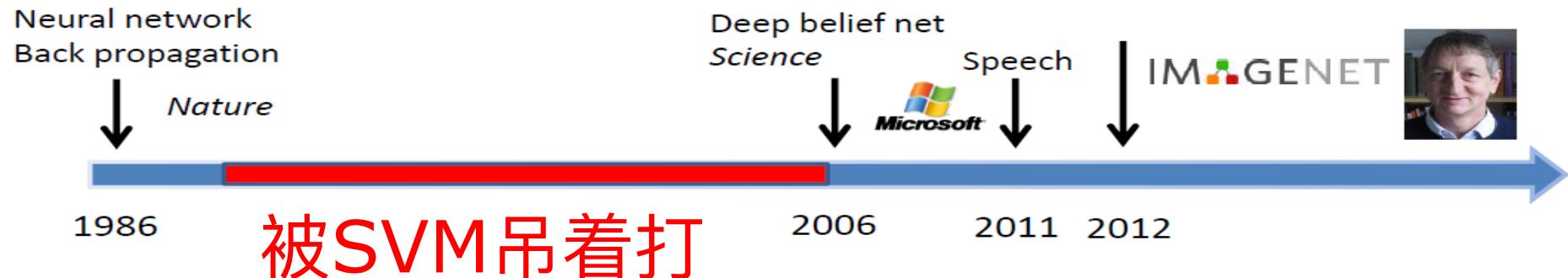




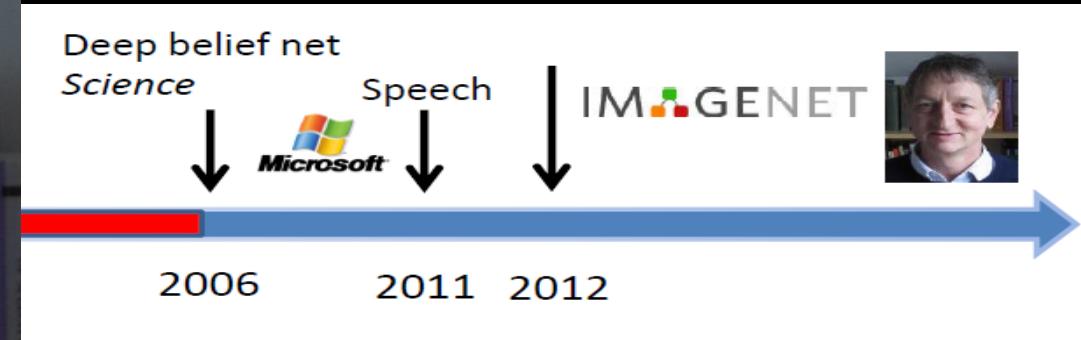
被SVM吊打

Data and machine learning





Credit: Xiaogang Wang, The Chinese University of Hong Kong



面壁十年图破壁

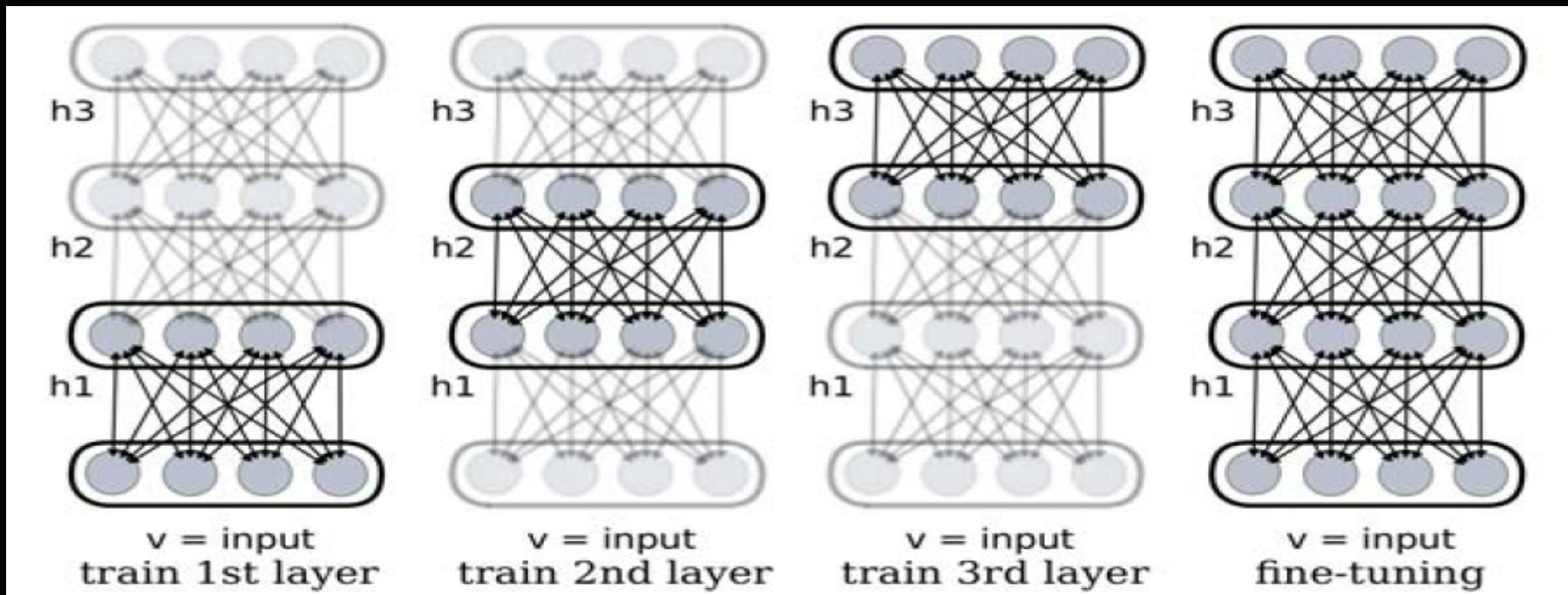
我Hinton悟到了

一！层！一！层！来！



Credit: Xiaogang Wang, The Chinese University of Hong Kong

一层一层先预训练参数，最后反向传播



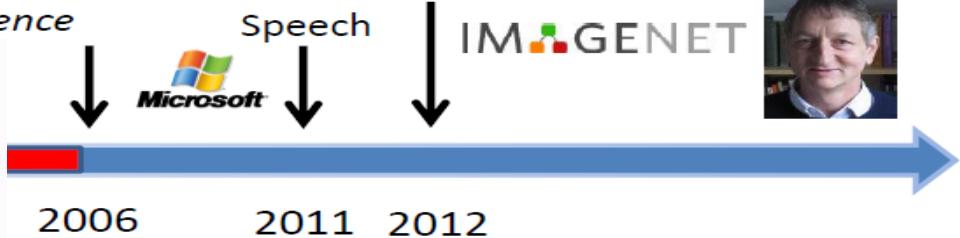
Neural network
Back propagation



1986



Deep belief net
Science *Speech*



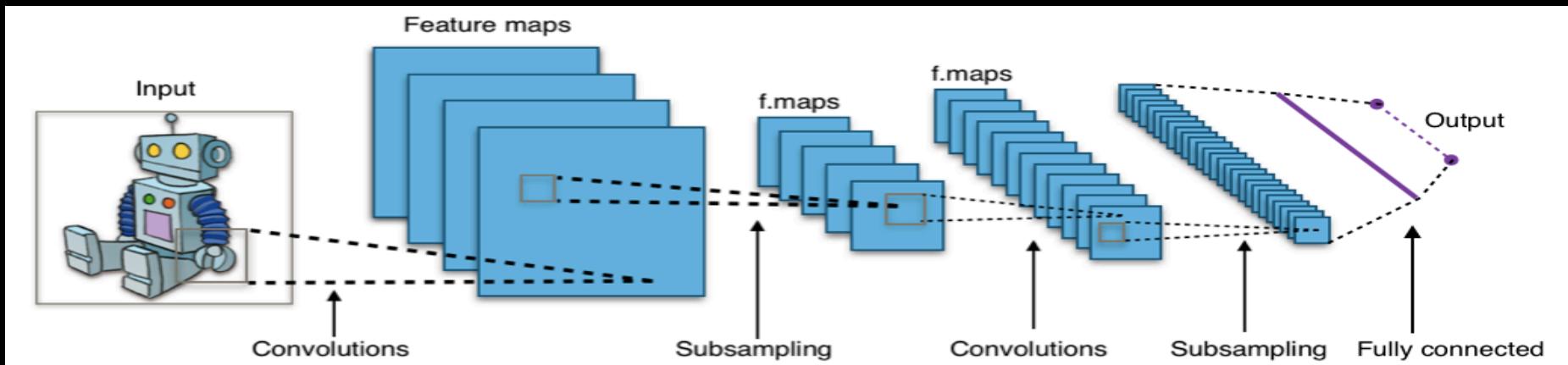
减少训练权重的规模

试试我乐康卷 😊

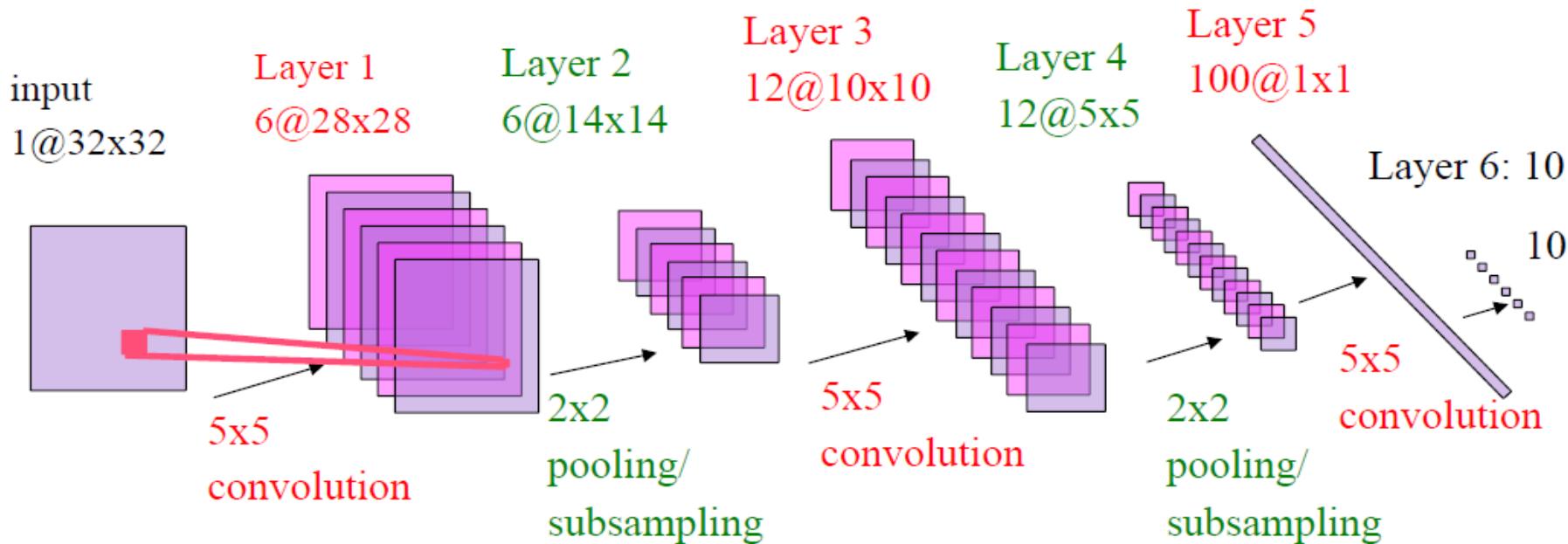


Credit: Xiaogang Wang, The Chinese University of Hong Kong

乐康卷：大体架构



乐康卷：参数配置 LeNet5

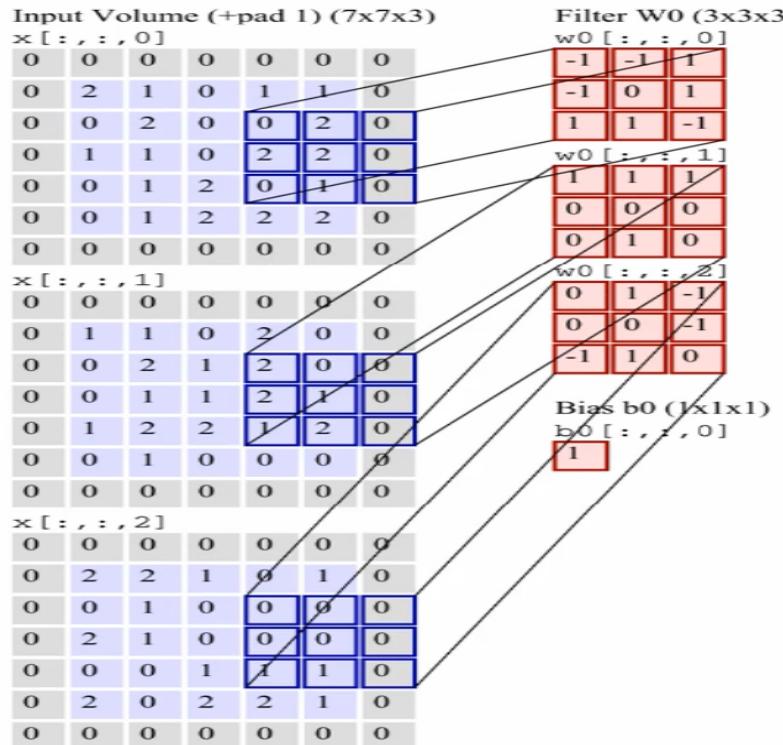




Multiple Convolutions

乐康卷：具体实例（来自LeCun）

Y LeCun



Filter W1 (3x3x3)

$$w1[:, :, 0]$$

0	-1	-1
0	-1	-1
-1	0	1

$$w1[:, :, 1]$$

0	-1	-1
1	1	0
-1	1	-1

$$w1[:, :, 2]$$

1	0	0
1	-1	0
-1	1	1

Bias b1 (1x1x1)

$$b1[:, :, 0]$$

0							
---	--	--	--	--	--	--	--

Output Volume (3x3x2)

$$o[:, :, 0]$$

-2	3	2
4	11	2
6	2	2

$$o[:, :, 1]$$

-3	-5	-2
-8	-1	0
-7	-10	-3

toggle movement

Animation: Andrej Karpathy <http://cs231n.github.io/convolutional-networks/>

乐康卷：具体实例（来自LeCun大神）

Input Volume (+pad 1) (7x7x3)

$\times [:, :, 0]$

0	0	0	0	0	0	0
0	2	1	0	1	1	0
0	0	2	0	0	2	0
0	1	1	0	2	2	0
0	0	1	2	0	1	0
0	0	1	2	2	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

$w0 [:, :, 0]$

-1	-1	1
-1	0	1
1	1	-1
1	1	1
0	0	0
0	1	0

不行了：反向传播算法的局限性

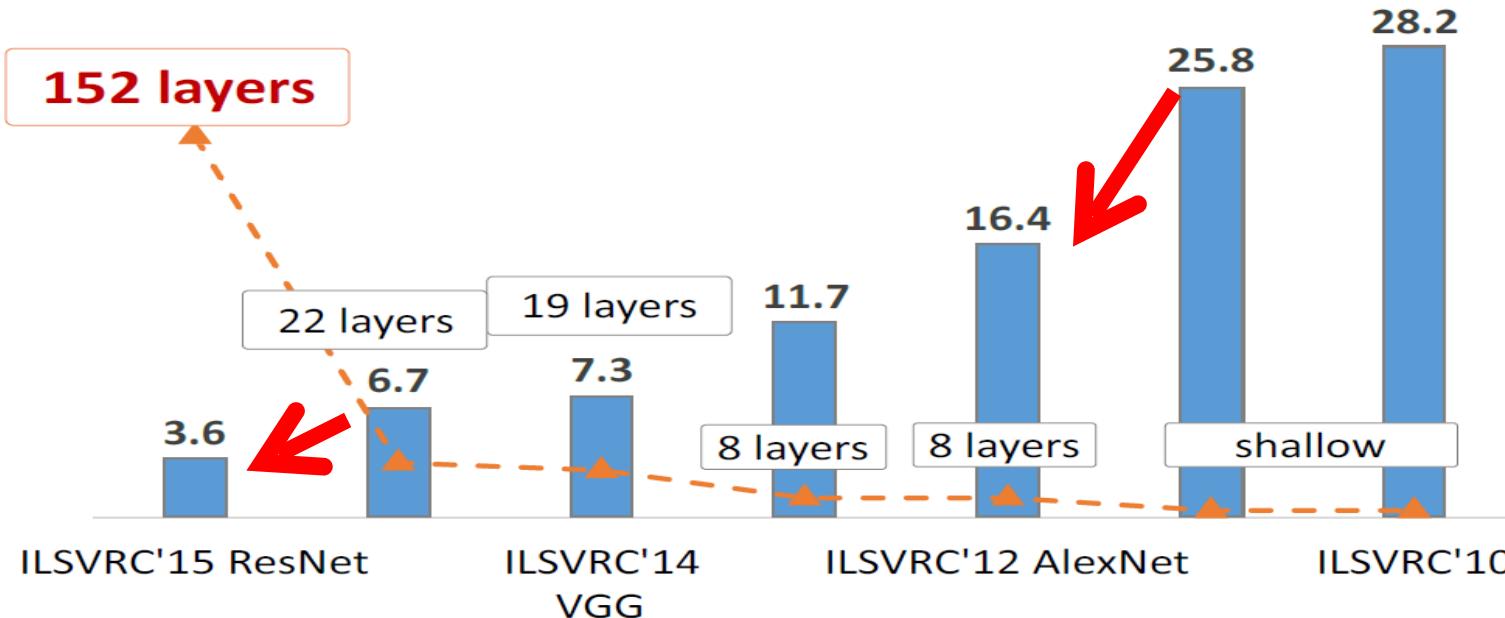
梯度消失 - 逐层预训练

局部最优 - 多次随机化

过学习 - 大数据集

算法和模型都完备后，PhD们的心开始躁动起来

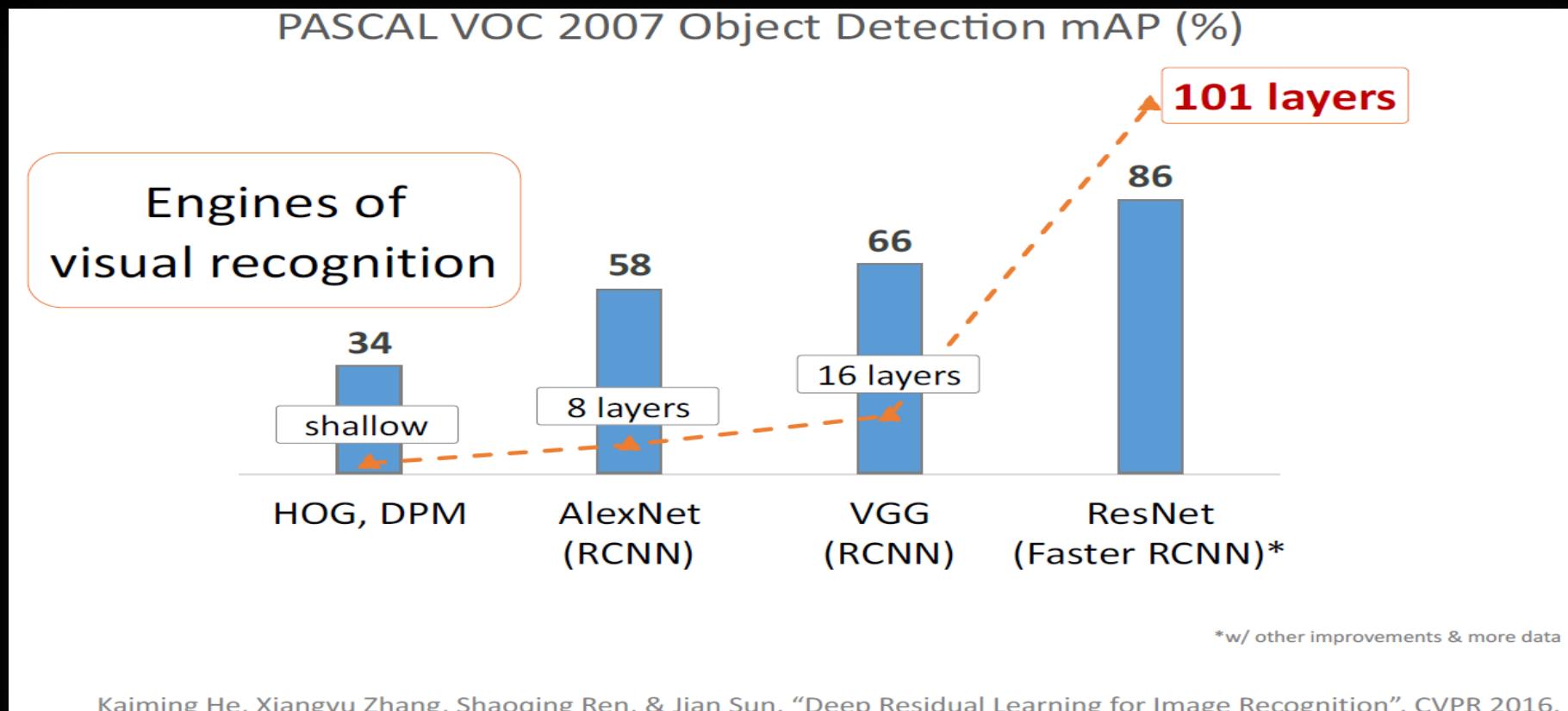
ImageNet Classification top-5 error (%)



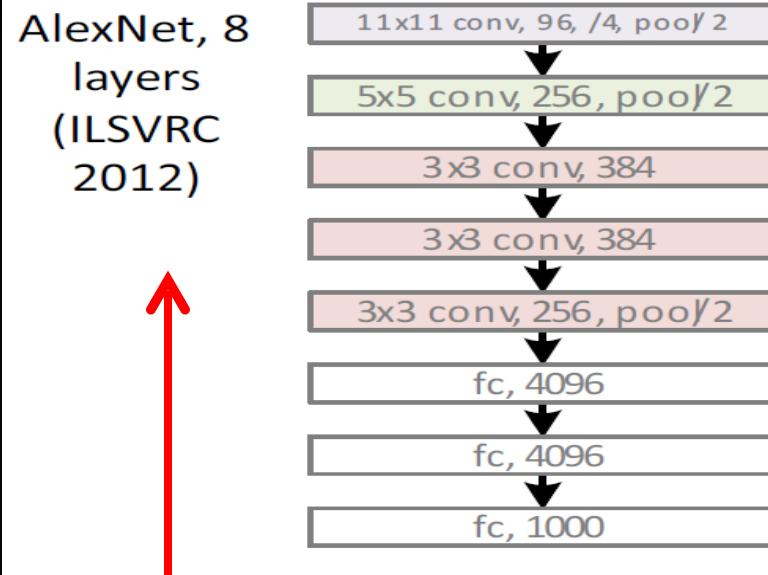
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Credit: Bernt Schiele - schiele@mpi-inf.mpg.de. Mario Fritz - mfritz@mpi-inf.mpg.de . <https://www mpi-inf.mpg.de/hlcv>

算法和模型都完备后，PhD们的心开始躁动起来



算法和模型都完备后， PhD们的心开始躁动起来

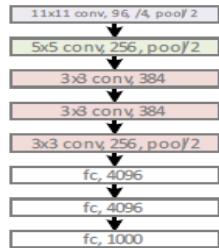


Caffe、mxnet中有现成模型

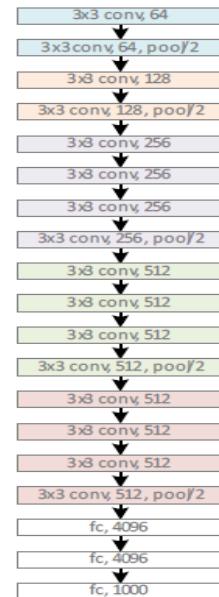
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

算法和模型都完备后， PhD们的心开始躁动起来

AlexNet, 8
layers
(ILSVRC
2012)



VGG, 19
layers
(ILSVRC
2014)



GoogleNet, 22
layers
(ILSVRC 2014)



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

算法和模型都完备后，PhD们的心开始躁动起来

Revolution of Depth

AlexNet, 8
layers
(ILSVRC
2012)



VGG, 19
layers
(ILSVRC
2014)



ResNet, 152
layers
(ILSVRC 2015)



算法和模型都完备后， PhD们的心开始躁动起来

README.md

Deep Residual Networks with 1K Layers

By Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.

Microsoft Research Asia (MSRA).

Table of Contents

- 1. Introduction
- 2. Notes
- 3. Usage

Introduction

This repository contains re-implemented code for the paper "Identity Mappings in Deep Residual Networks" (<http://arxiv.org/abs/1603.05027>). This work enables training quality 1k-layer neural networks in a super simple way.

Acknowledgement: This code is re-implemented by Xiang Ming from Xi'an Jiaotong University for the ease of release.

See Also: Re-implementations of ResNet-200 [a] on ImageNet from Facebook AI Research (FAIR):
<https://github.com/facebook/fb.resnet.torch/tree/master/pretrained>

1001层DRN

故事背景

老板（激动的）：我们加到两千层试试！

小李：.....

小结与思考

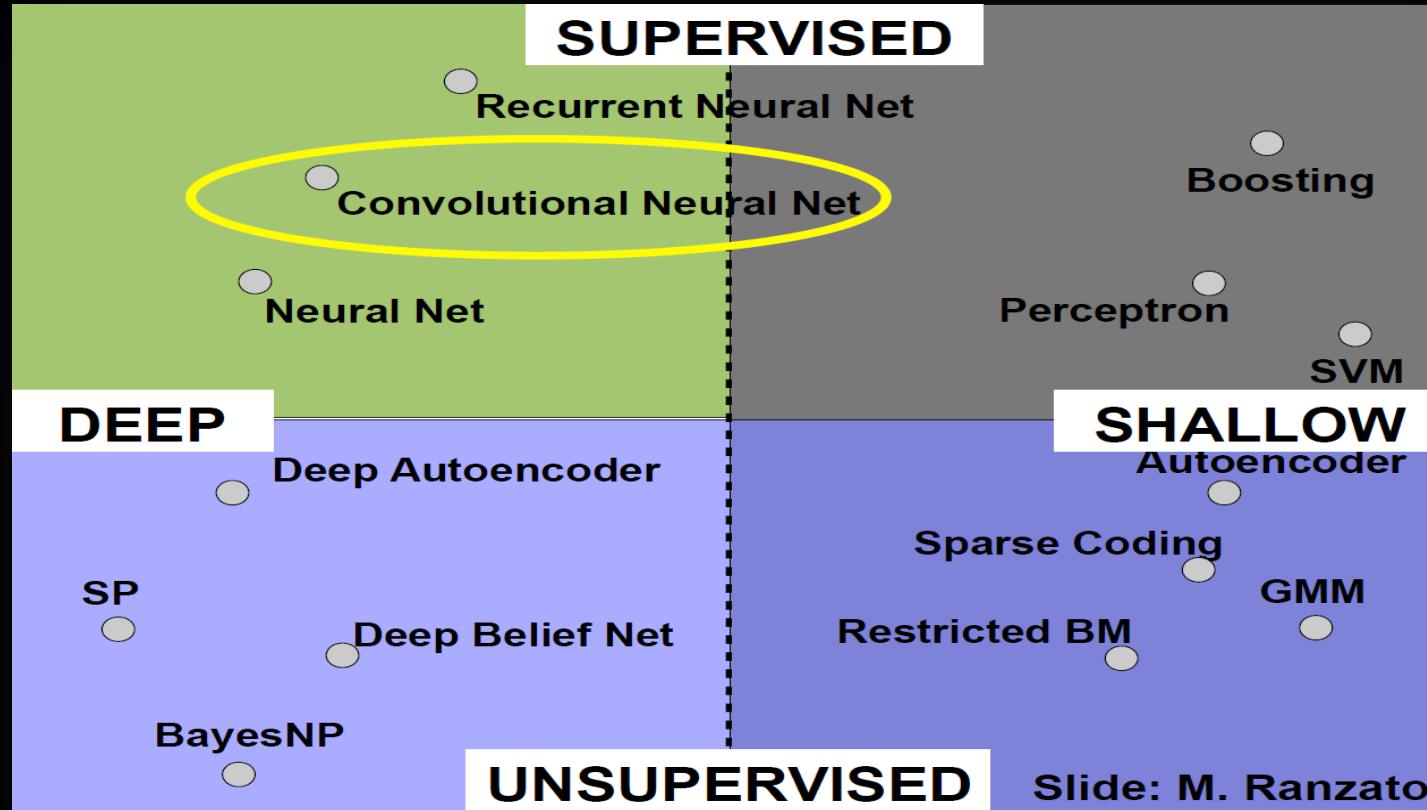
为什么深度学习会在图像、音频、NLP领域成功？

单纯的加层就可以提高预测的效果么？

扩大训练数据的数据规模就可以提高预测的效果么？

最重要的，如果一开始就针对的错误的问题.....

没有提到的跟机器学习有关的内容



没有提到的跟机器学习有关的内容

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}, \mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}. \quad (1)$$

$$E(\mathbf{v}, \mathbf{h}) = - \sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j, \quad (2)$$

$$p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i|\mathbf{h}) \quad \text{and} \quad p(v_i = 1|\mathbf{h}) = \text{sigm} \left(a_j + \sum_j h_j w_{ij} \right),$$

$$p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v}) \quad \text{and} \quad p(h_j = 1|\mathbf{v}) = \text{sigm} \left(b_j + \sum_i v_i w_{ij} \right), \quad (3)$$

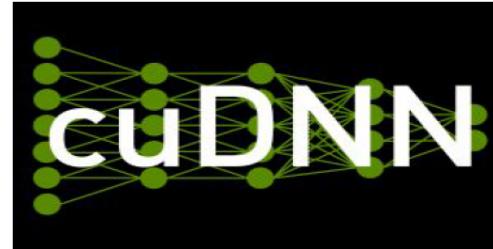
$$p(v_i = x|\mathbf{h}) = \frac{1}{\sigma_i \sqrt{2\pi}} \cdot e^{-\frac{(x - a_i - \sigma_i \sum_j w_{ij} h_j)^2}{2\sigma_i^2}},$$
$$p(h_j = 1|\mathbf{v}) = \text{sigm} \left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij} \right). \quad (4)$$

想要学习深度学习，如何上手？

现在学习深度学习是相对容易的

Deep-Learning Package Zoo

- Torch
- Caffe
- Theano (Keras, Lasagne)
- CuDNN
- Tensorflow
- Mxnet
- Etc.



theano
dmlc
mxnet



入门阶段已经不需要自己写算法，写模型即可

Deep-Learning Package Design Choices

- Model specification: **Configuration file** (e.g. Caffe, DistBelief, CNTK) versus **programmatic generation** (e.g. Torch, Theano, Tensorflow)
- For programmatic models, choice of high-level language: Lua (Torch) vs. Python (Theano, Tensorflow) vs others.
- We chose to work with **python** because of rich community and library infrastructure.

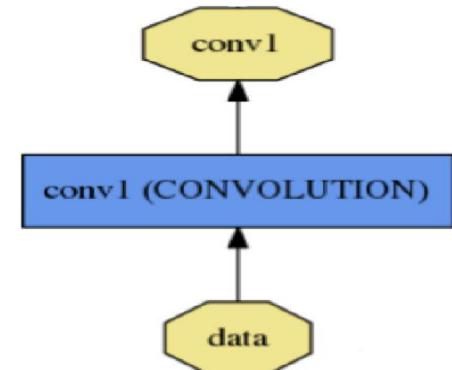
以两个最常用的包为例



A Caffe Layer

```
name: "conv1"
type: CONVOLUTION
bottom: "data"
top: "conv1"
convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
        type: "xavier"
    }
}
```

} name, type, and the connection structure (input blobs and output blobs)
layer-specific parameters

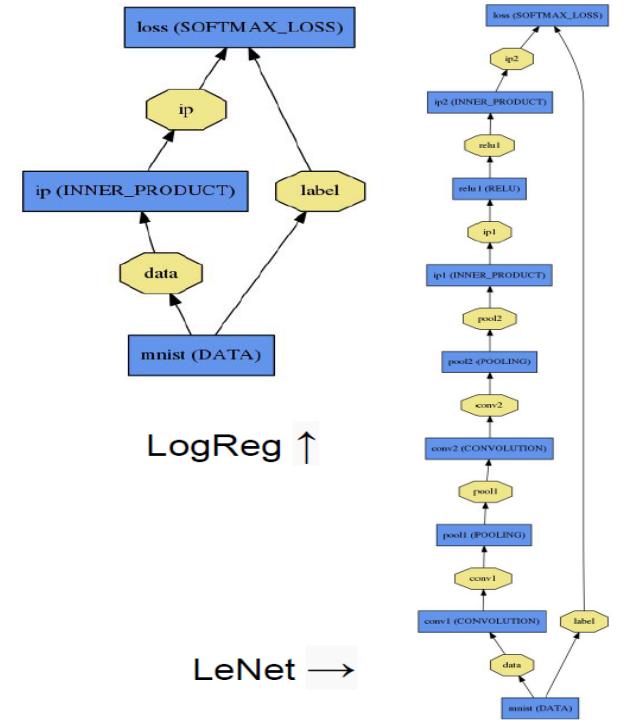


*Link to the [Google Protobuffer Documentation](#)

A Caffe Network

- A network is a set of layers connected as a DAG:

```
name: "dummy-net"  
  
layers { name: "data" ... }  
layers { name: "conv" ... }  
layers { name: "pool" ... }  
... more layers ...  
layers { name: "loss" ... }
```



MNIST: 深度学习领域的 Hello World

THE MNIST DATABASE

of handwritten digits

[Yann LeCun](#), Courant Institute, NYU

[Corinna Cortes](#), Google Labs, New York

[Christopher J.C. Burges](#), Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

[train-images-idx3-ubyte.gz](#): training set images (9912422 bytes)

[train-labels-idx1-ubyte.gz](#): training set labels (28881 bytes)

[t10k-images-idx3-ubyte.gz](#): test set images (1648877 bytes)

[t10k-labels-idx1-ubyte.gz](#): test set labels (4542 bytes)

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

Misclassified examples

4	3	2	1	5	4	2	3	6	1
9	8	7	5	7	6	3	2	3	4
8	5	4	3	0	9	4	6	5	1
9	2	1	3	3	9	0	0	2	6
4	7	9	4	2	7	4	9	9	9
8	4	8	3	8	6	8	3	3	9
1	9	6	0	6	7	0	1	4	2
2	8	4	7	7	1	9	6	6	5

Caffe

Deep learning framework
by the [BVLC](#)

Created by
[Yangqing Jia](#)
Lead Developer
[Evan Shelhamer](#)

 [View On GitHub](#)

Training LeNet on MNIST with Caffe

We will assume that you have Caffe successfully compiled. If not, please refer to the [Installation page](#). In this tutorial, we will assume that your Caffe installation is located at `CAFFE_ROOT`.

Prepare Datasets

You will first need to download and convert the data format from the MNIST website. To do this, simply run the following commands:

```
cd $CAFFE_ROOT
./data/mnist/get_mnist.sh
./examples/mnist/create_mnist.sh
```

If it complains that `wget` or `gunzip` are not installed, you need to install them respectively. After running the script there should be two datasets, `mnist_train_lmdb`, and `mnist_test_lmdb`.

LeNet: the MNIST Classification Model

Before we actually run the training program, let's explain what will happen. We will use the [LeNet](#) network, which is known to work well on digit classification tasks. We will use a slightly different version from the original LeNet implementation, replacing the sigmoid activations with Rectified Linear Unit (ReLU) activations for the neurons.

Caffe

Deep learning framework
by the [BVLC](#)

Created by
[Yangqing Jia](#)
Lead Developer
[Evan Shelhamer](#)

[View On GitHub](#)

Caffe Model Zoo

Lots of researchers and engineers have made Caffe models for different tasks with all kinds of architectures and data. These models are learned and applied for problems ranging from simple regression, to large-scale visual classification, to Siamese networks for image similarity, to speech and robotics applications.

To help share these models, we introduce the model zoo framework:

- A standard format for packaging Caffe model info.
- Tools to upload/download model info to/from Github Gists, and to download trained `.caffemodel` binaries.
- A central wiki page for sharing model info Gists.

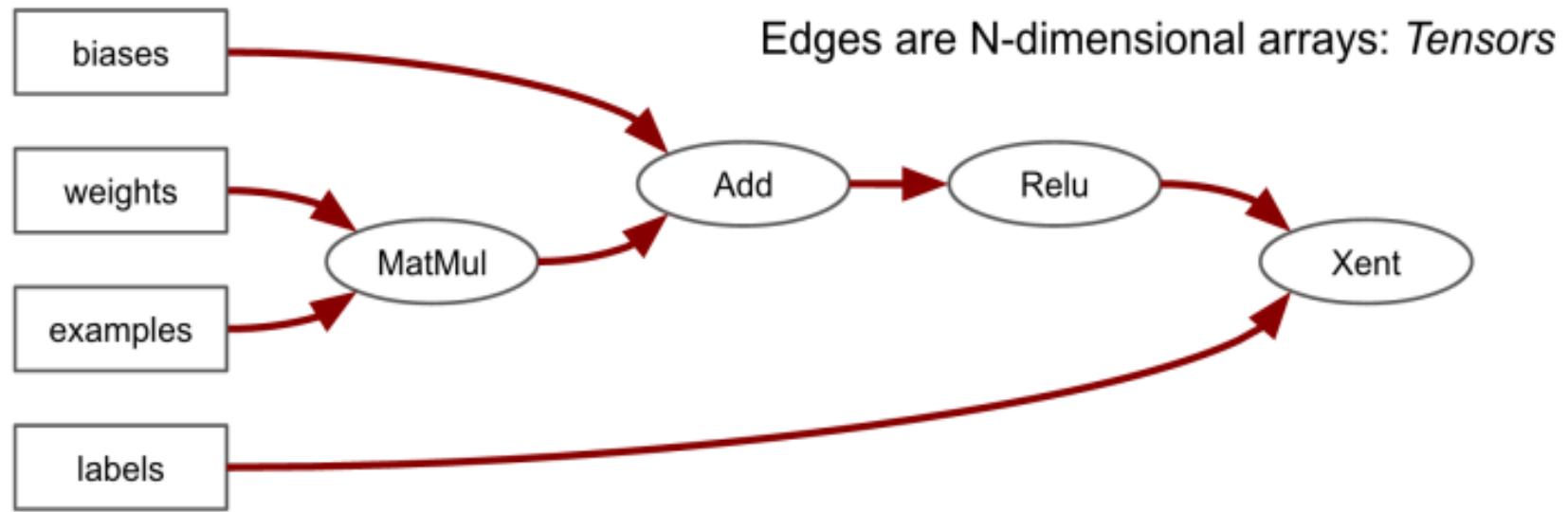
Where to get trained models

First of all, we bundle BVLC-trained models for unrestricted, out of the box use.

See the [BVLC model license](#) for details. Each one of these can be downloaded by running `scripts/download_model_binary.py <dirname>` where `<dirname>` is specified below:

TensorFlow: 搞清楚Tensor和Flow就行了

TensorFlow is an open source software library for numerical computation using data flow graphs.



HELLO WORLD...

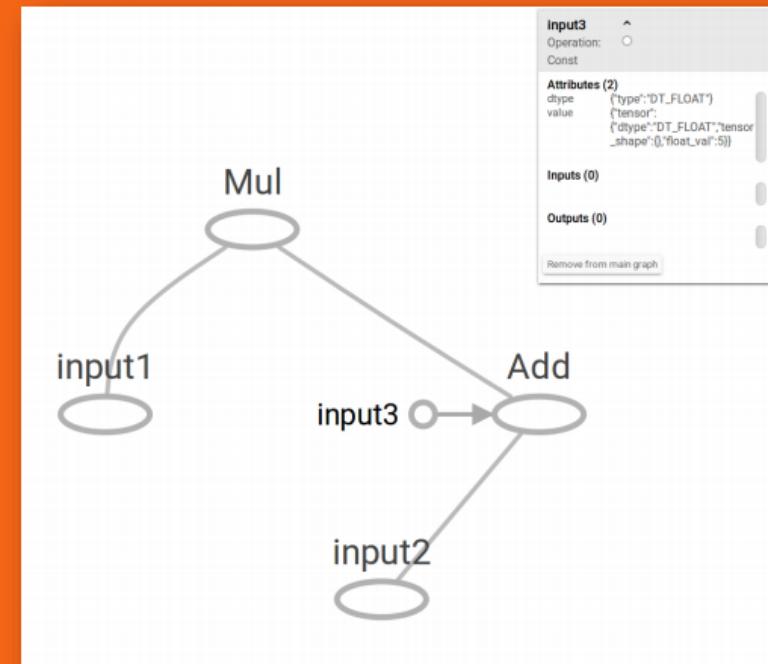
$$mul = i_1 * (i_2 + i_3)$$

```
import tensorflow as tf

input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)
input3 = tf.constant(5.0)
intermed = tf.add(input2, input3)
mul = tf.mul(input1, intermed)

with tf.Session() as sess:
    result = sess.run(
        [mul, intermed],
        feed_dict={input1:[1.0],
                   input2:[2.0]})

print(result)
```



Version: r0.11 ▾

Basic Neural Networks

[MNIST For ML Beginners](#)

[About this tutorial](#)

[The MNIST Data](#)

[Softmax Regressions](#)

[Implementing the Regression](#)

[Training](#)

[Evaluating Our Model](#)

[Deep MNIST for Experts](#)

[About this tutorial](#)

[Setup](#)

[Load MNIST Data](#)

[Start TensorFlow InteractiveSession](#)

[Build a Softmax Regression Model](#)

[Placeholders](#)

[Variables](#)

[Predicted Class and Loss Function](#)

MNIST For ML Beginners

This tutorial is intended for readers who are new to both machine learning and TensorFlow. If you already know what MNIST is, and what softmax (multinomial logistic) regression is, you might prefer this [faster paced tutorial](#). Be sure to [install TensorFlow](#) before starting either tutorial.

When one learns how to program, there's a tradition that the first thing you do is print "Hello World." Just like programming has Hello World, machine learning has MNIST.

MNIST is a simple computer vision dataset. It consists of images of handwritten digits like these:



It also includes labels for each image, telling us which digit it is. For example, the labels for the above images are 5, 0, 4, and 1.

In this tutorial, we're going to train a model to look at images and predict what digits they are. Our goal isn't to train a really elaborate model that achieves state-of-the-art performance -- although we'll give you code to do that later! -- but rather to dip a toe into using TensorFlow. As such, we're going to start with a very simple model, called a Softmax Regression.

The actual code for this tutorial is very short, and all the interesting stuff happens in just three lines. However, it is very important to understand the ideas behind it: both how TensorFlow works and the core machine learning concepts. Because of this, we are going to very carefully work through the code.

README.md

TensorFlow Models

This repository contains machine learning models implemented in [TensorFlow](#). The models are maintained by their respective authors.

To propose a model for inclusion please submit a pull request.

Models

- [autoencoder](#) -- various autoencoders
- [inception](#) -- deep convolutional networks for computer vision
- [namigner](#) -- recognize and generate names
- [neural_gpu](#) -- highly parallel neural computer
- [privacy](#) -- privacy-preserving student models from multiple teachers
- [resnet](#) -- deep and wide residual networks
- [slim](#) -- image classification models in TF-Slim
- [swivel](#) -- the Swivel algorithm for generating word embeddings
- [syntaxnet](#) -- neural models of natural language syntax
- [textsum](#) -- sequence-to-sequence with attention model for text summarization.
- [transformer](#) -- spatial transformer network, which allows the spatial manipulation of data within the network
- [im2txt](#) -- image-to-text neural network for image captioning.

深度学习怎么玩， 取决于你多有钱

GPU Acceleration

And FPGA、TPU、寒武纪

-gpu N flag tells caffe which gpu to use

Alternatively, specify solver_mode: GPU in solver.prototxt

16.2s

10x

163s

Tesla K40m, cuDNN v3-RC
ECC off, autoboot on

Intel® Xeon® E5-2698 v3 @
2.60GHz, 3.6GHz turbo, (16
cores total), HT off

Benchmark: Train Caffenet model, 20 iterations, 256x256 images, mini-batch size 256

基础 | 前沿 | 讨论

前沿 | NPU | GAN | 自编程

语言处理 | 语音识别 | 图像识别 | 视频理解

自然语言处理 | TL;DR

语音识别 | 热火朝天

语音识别 | 百度,讯飞,思必驰

谷歌,微软,苹果,IBM,亚马逊

语音识别 | GMM-HMM (李开复)

语音识别 | Warp-CTC (百度)

Connectionist Temporal Classification

语音识别 | FSMN, DFCNN (讯飞)

前馈型序列记忆网络 | 深度全序列卷积神经网络

语音识别 | ResNet+ Σ (微软)

NIST 2000 Switchboard WER 5.9%

TTS | WaveNet (谷歌)

图像识别 | 成果最突出显著

图像识别 | MNIST



图像识别 | IMGENET

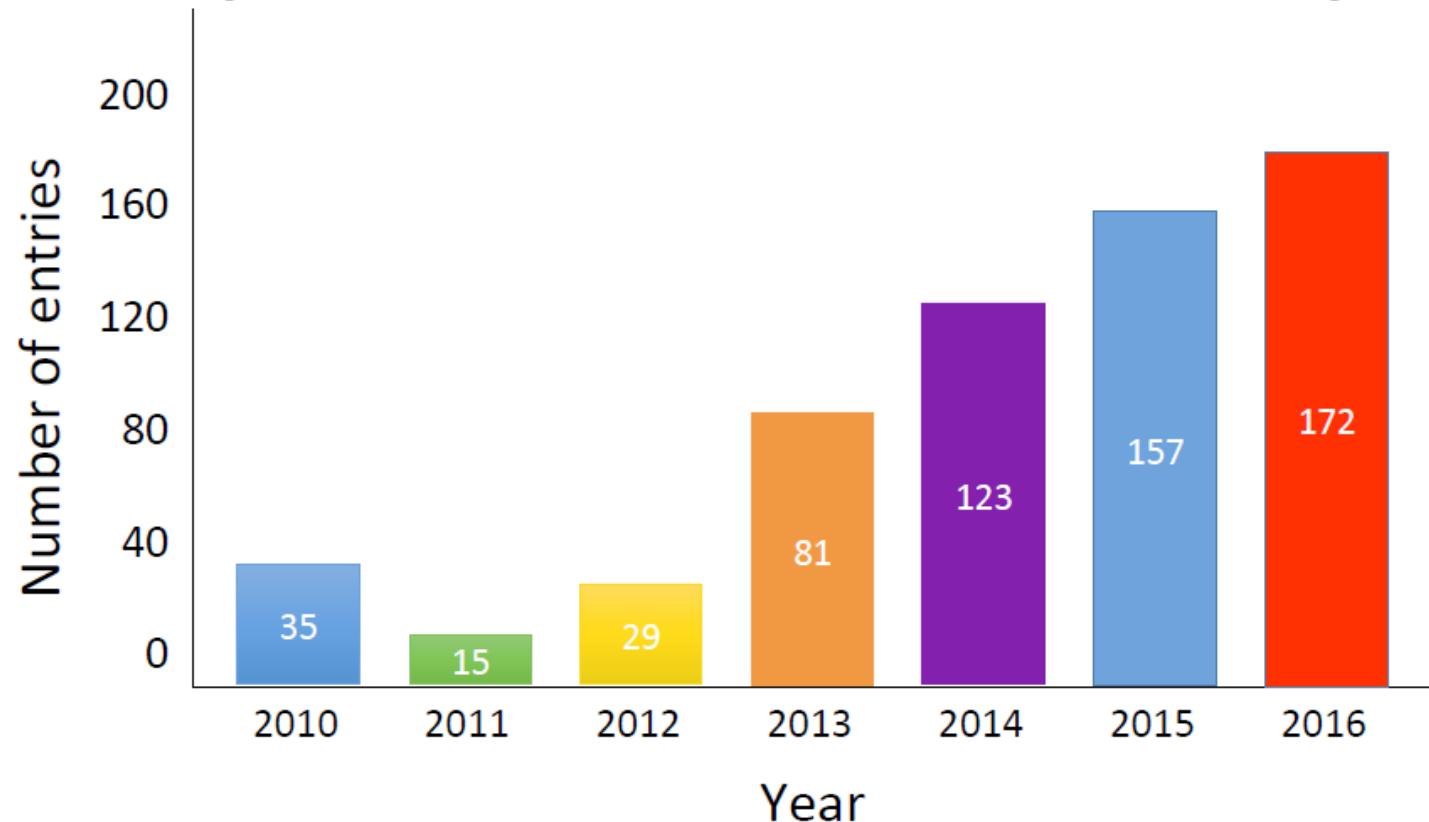
WordNet | 21K | 14M | 1M



图像识别 | ILSVRC

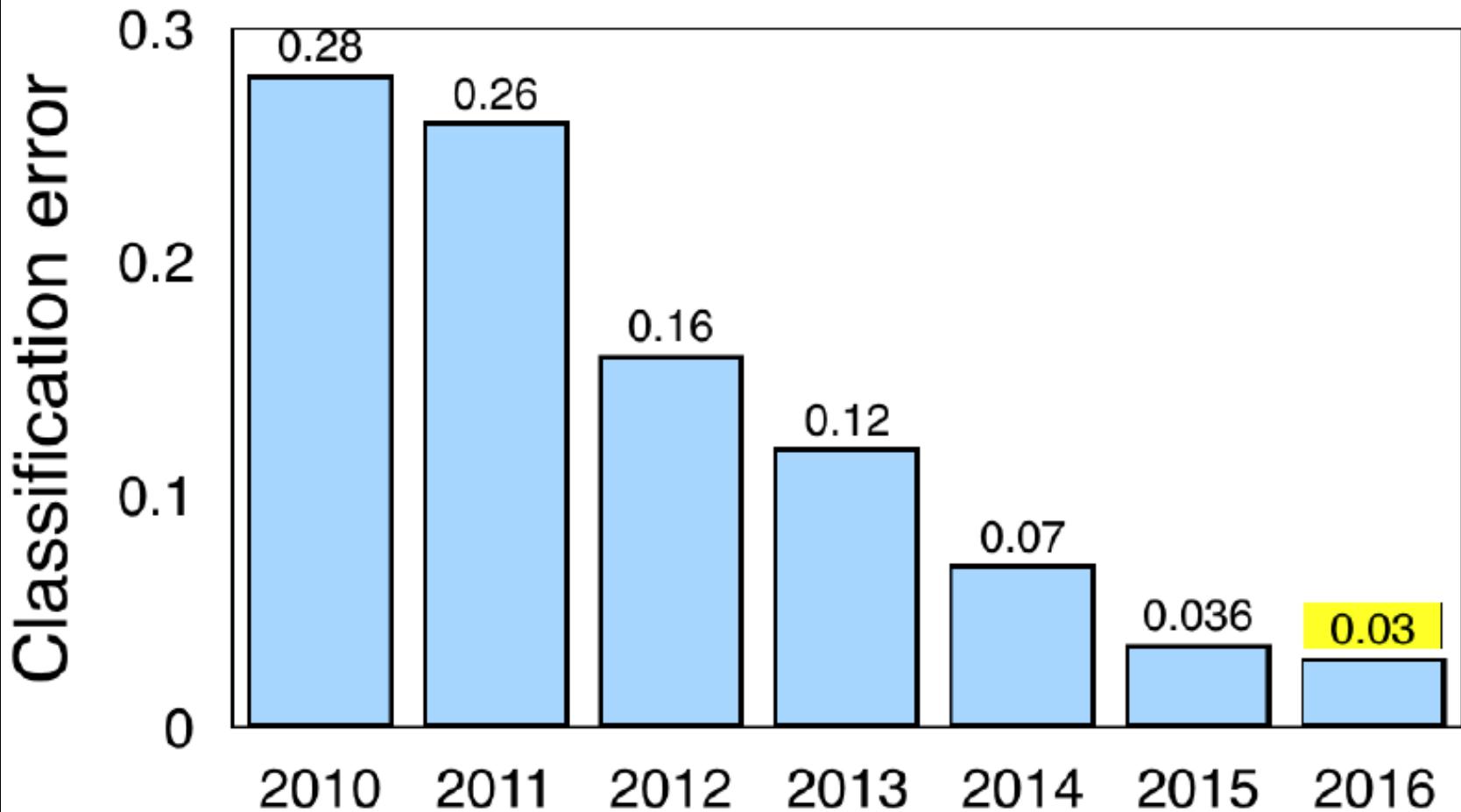
始于2010, 里程碑

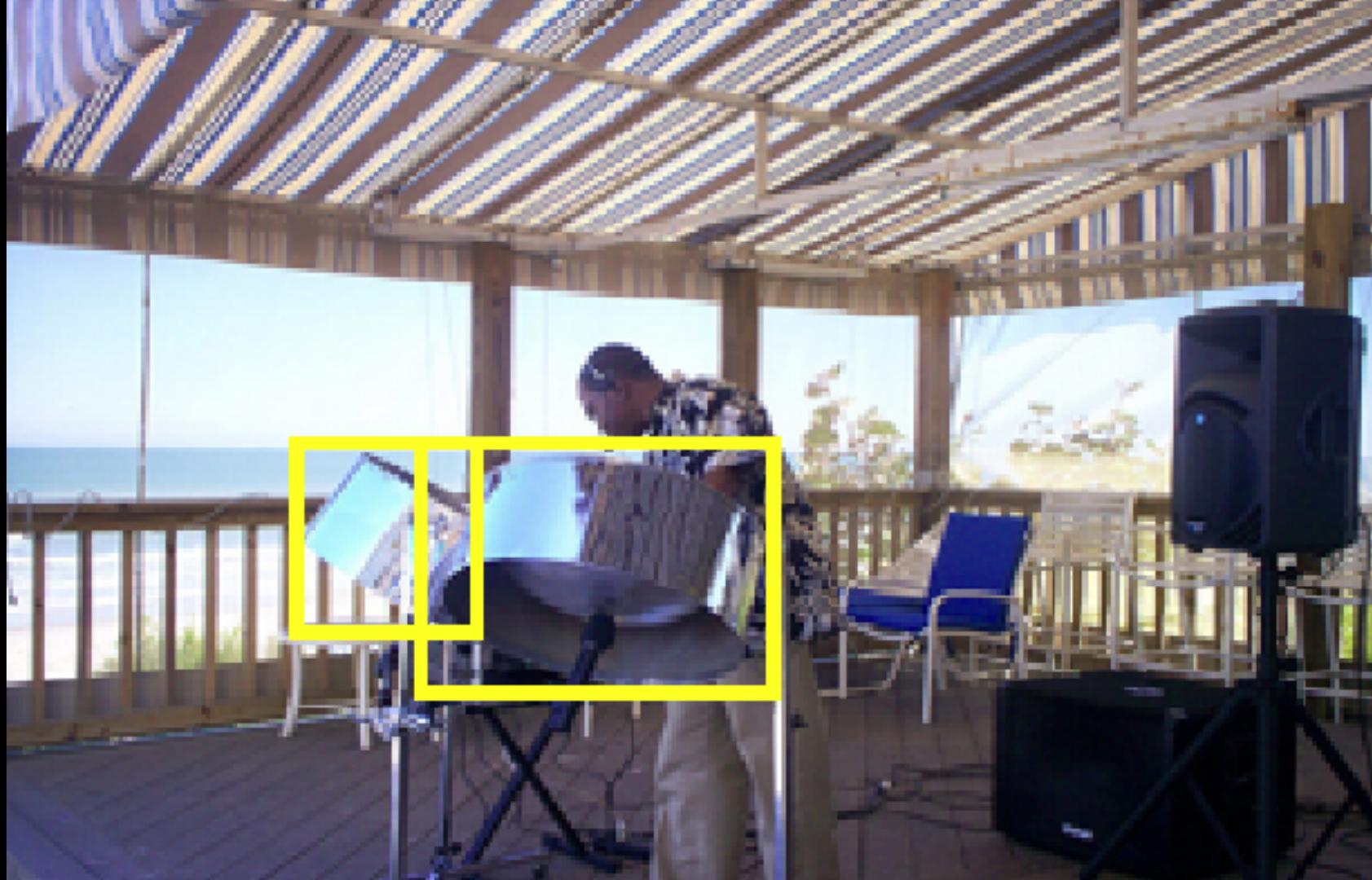
Participation in ILSVRC over the years



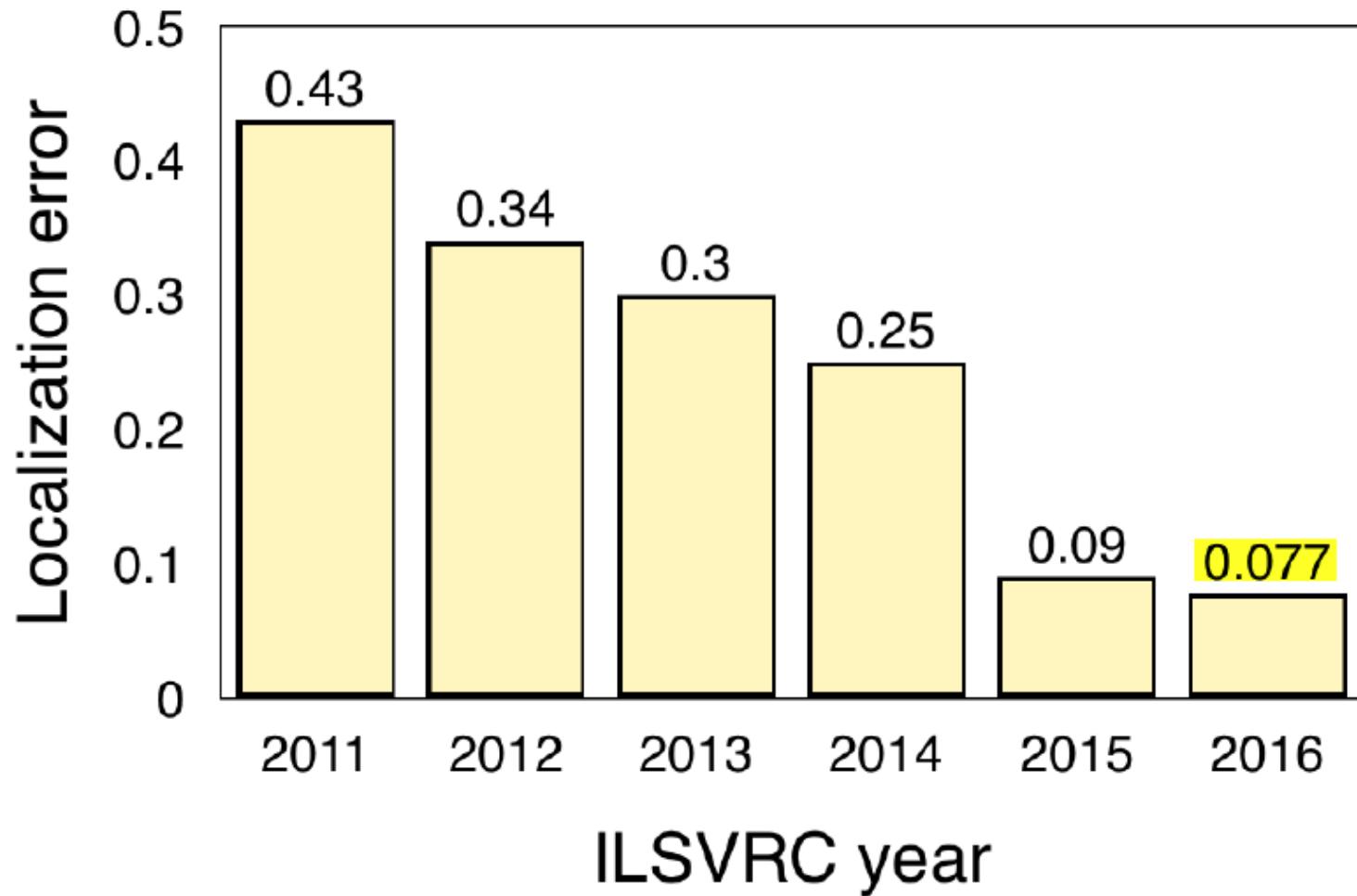


Classification



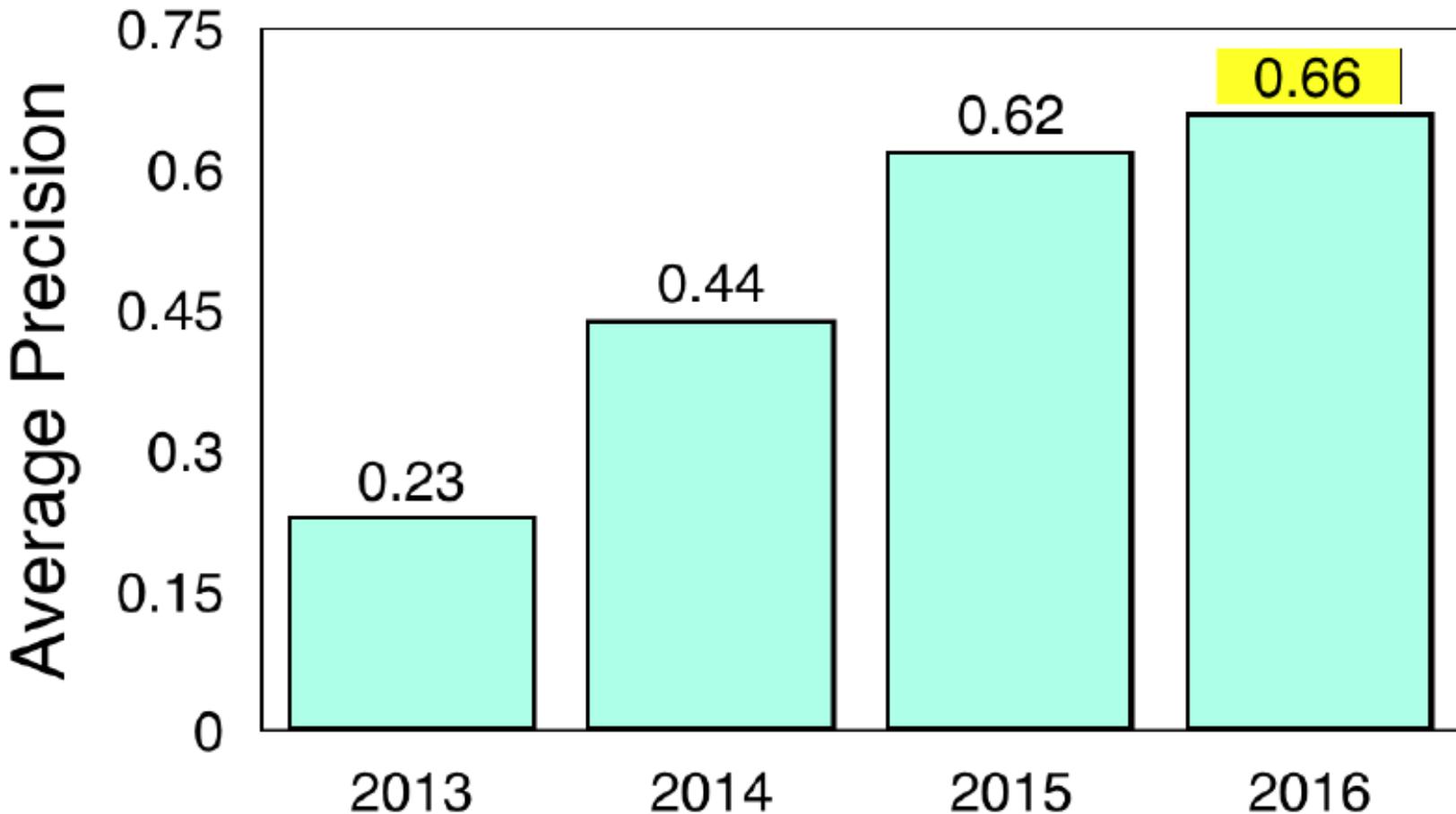


Localization





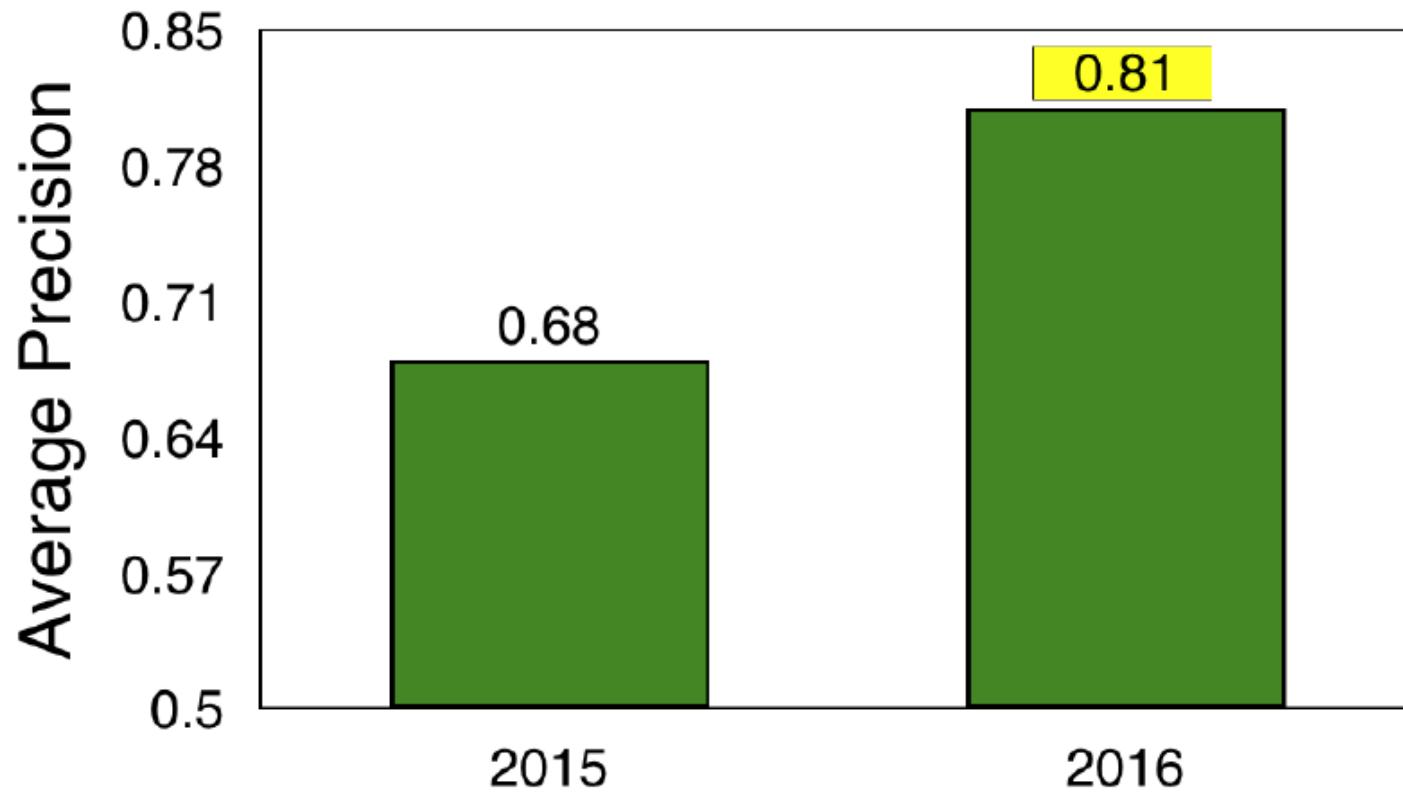
Detection



Fully annotated 30 object classes across 6,278 snippets (train+test)



Video Detection



图像识别 | Beyond ImageNet

Sense Classification & Parsing

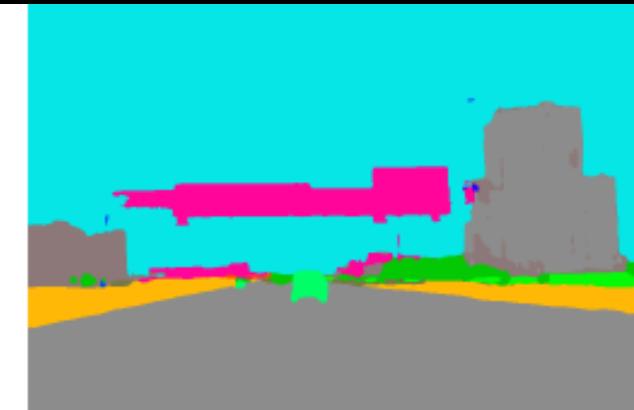


Image Captioning

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



视频分析 | 门槛在慢慢降低

视频预测 | 效果还比较诡异

人脸识别 | 进入混战模式



SECURITY
CHINA

2016 Security China

中国国际社会公共安全产品博览会

2016 China International Exhibition on Public Safety and Security

中文版 | English | Русский



OK



服务经济民生

展示安防技术

构建智慧城市

推进平安中国

网站首页

关于展会

展会活动

展会新闻

参展服务

观众服务

交通及住宿

下载中心

联系我们

主办单位：中国安全防范产品行业协会

展览日期：2016年10月25-28日

展览地点：中国国际展览中心(新馆)

智造平安城市！ 旷视智能安防解决方案——天眼系统

Face++ 旷视



SenseFace-TX1

硬件

NVIDIA Tegra X1

核心板仅一张信用卡大小

质量约75g

功耗6.5 ~ 15W

功能

多人脸检测

人脸关键点提取

人脸质量评估

动态跟踪

性能

1080P监控视频

25hz实时处理



SENSEFACE

抓拍人数 1009

告警人数 46

视频路数 4

属性

风格

人脸检测
100% 100% 100%



定位 展会1号相机
10-25 2016



正在播放

定位 展会1号相机

任务 人脸识别系统1



定位 展会2号相机

任务 人脸识别系统2



定位 展会3号相机

任务 人脸识别系统3



定位 展会4号相机

任务 人脸识别分析



SENSEFACE

抓拍人数 1031 告警人数 48

视频路数 4

2016-10-25 11:06:16

属性 风格

任务 人脸识别系统1

10月25日 2016年 六月

性别 年龄 种族 表情 状态

男 26岁 黄种人 微笑 闭眼

女 25岁 黄种人 微笑 闭眼

男 19岁 黄种人 微笑 闭眼

女 18岁 黄种人 微笑 闭眼

男 23岁 黄种人 微笑 闭眼

女 22岁 黄种人 闭眼 张嘴

姓名 梁博 性别 男 所在库 安博展员工库

展会1号相机 2016-10-25 11:06:16

展会2号相机 2016-10-25 11:06:16

展会3号相机 2016-10-25 11:06:16

展会4号相机 2016-10-25 11:06:16

DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Yaniv Taigman

Ming Yang

Marc'Aurelio Ranzato

Lior Wolf

Facebook AI Research
Menlo Park, CA, USA

{yaniv, mingyang, ranzato}@fb.com

Tel Aviv University
Tel Aviv, Israel

wolf@cs.tau.ac.il

Abstract

In modern face recognition, the conventional pipeline consists of four stages: detect \Rightarrow align \Rightarrow represent \Rightarrow classify. We revisit both the alignment step and the representation step by employing explicit 3D face modeling in order to apply a piecewise affine transformation, and derive a face representation from a nine-layer deep neural network. This deep network involves more than 120 million parameters using several locally connected layers without weight sharing, rather than the standard convolutional layers. Thus we trained it on the largest facial dataset to-date, an identity labeled dataset of four million facial images belonging to more than 4,000 identities. The learned representa-

toward tens of thousands of appearance features in other recent systems [5, 7, 2].

The proposed system differs from the majority of contributions in the field in that it uses the deep learning (DL) framework [3, 21] in lieu of well engineered features. DL is especially suitable for dealing with large training sets, with many recent successes in diverse domains such as vision, speech and language modeling. Specifically with faces, the success of the learned net in capturing facial appearance in a robust manner is highly dependent on a very rapid 3D alignment step. The network architecture is based on the assumption that once the alignment is completed, the location of each facial region is fixed at the pixel level. It is therefore possible to learn from the raw pixel RGB values,

OpenFace

News

Overview

Posts About OpenFace

Notable Relevant Projects

Citations

Acknowledgements

Licensing

OpenFace



Star

5,879

Free and open source face recognition with deep neural networks.

News

- 2016-09-15: We presented OpenFace in the [Data \(after\)Lives](#) art exhibit at the University of Pittsburgh and have released the code as [Demo 4: Real-time Face Embedding Visualization](#).
- 2016-08-09: [New blog post: \(Face\) Image Completion with Deep Learning in TensorFlow](#). ([OpenFace group discussion on it](#))
- 2016-06-01: [OpenFace tech report released](#)
- 2016-01-19: OpenFace 0.2.0 released! See [this blog post](#) for more details.

OpenFace is a Python and [Torch](#) implementation of face recognition with deep neural networks and is based on the CVPR 2015 paper [FaceNet: A Unified Embedding for Face Recognition and Clustering](#) by Florian Schroff, Dmitry Kalenichenko, and James Philbin at Google. Torch allows the network to be executed on a CPU or with CUDA.

Crafted by Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan.

- The code is available on GitHub at [cmusatyalab/openface](#).
- [API Documentation](#)
- Join the [cmu-openface group](#) or the [gitter chat](#) for discussions and installation issues.
- Development discussions and bugs reports are on the [issue tracker](#).

FaceNet: A Unified Embedding for Face Recognition and Clustering

Florian Schroff

fschroff@google.com

Google Inc.

Dmitry Kalenichenko

dkalenichenko@google.com

Google Inc.

James Philbin

jphilbin@google.com

Google Inc.

Abstract

Despite significant recent advances in the field of face recognition [10, 14, 15, 17], implementing face verification and recognition efficiently at scale presents serious challenges to current approaches. In this paper we present a system, called FaceNet, that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.

Our method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. To train, we use triplets of roughly aligned



README.md

FaceNet implementation in Tensorflow

This is a TensorFlow implementation of the face recognizer described in the paper "[FaceNet: A Unified Embedding for Face Recognition and Clustering](#)". The project also uses ideas from the paper "[A Discriminative Feature Learning Approach for Deep Face Recognition](#)" as well as the paper "[Deep Face Recognition](#)" from the Visual Geometry Group at Oxford.

The accuracy measured on the LFW test set is ~0.98 when trained on a combination of FaceScrub and CASIA-WebFace.

Inspiration

The code is heavily inspired by the [OpenFace](#) implementation.

Training data

The [FaceScrub](#) dataset and the [CASIA-WebFace](#) dataset have been used for training. This training set consists of total of 536 685 images over 11105 identities.

Pre-processing

基础 | 前沿 | 讨论

评价 | 泡泡越大，失望越大 (AGI)

评价 | 能解决的问题是有限的

评价 | 自动编程，呵呵哒

评价 | 警惕伪科学和各类歧视

Automated Inference on Criminality using Face Images

Xiaolin Wu

McMaster University

Shanghai Jiao Tong University

xwu510@gmail.com

Xi Zhang

Shanghai Jiao Tong University

zhangxi_19930818@sjtu.edu.cn

Abstract

We study, for the first time, automated inference on criminality based solely on still face images, which is free of any biases of subjective judgments of human observers. Via supervised machine learning, we build four classifiers (logistic regression, KNN, SVM, CNN) using facial images of 1856 real persons controlled for race, gender, age and facial expressions, nearly half of whom were convicted criminals, for discriminating between criminals and non-criminals. All four classifiers perform consistently well and empirically establish the validity of automated face-induced

management science, criminology, etc.

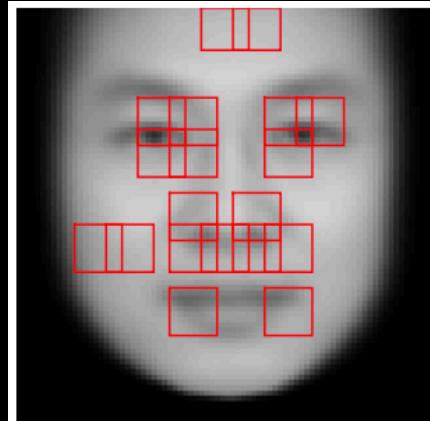
In all cultures and all periods of recorded human history, people share the belief that the face alone suffices to reveal innate traits of a person. Aristotle in his famous work Prior Analytics asserted, "It is possible to infer character from features, if it is granted that the body and the soul are changed together by the natural affections". Psychologists have known, for as long as a millennium, the human tendency of inferring innate traits and social attributes (e.g., the trustworthiness, dominance) of a person from his/her facial appearance, and a robust consensus of individuals' inferences . These are the facts found through numerous studies



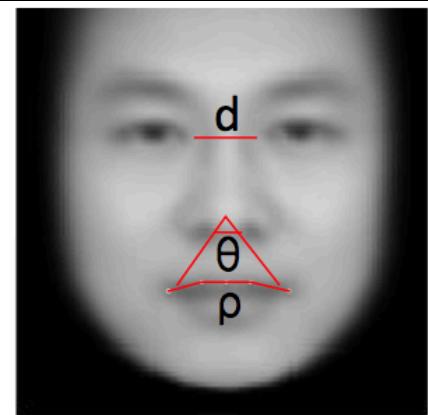
(a) Three samples in criminal ID photo set S_c .



(b) Three samples in non-criminal ID photo set S_n .



(a)



(b)

	Mean		Variance	
	criminal	non-criminal	criminal	non-criminal
ρ	0.5809	0.4855	0.0245	0.0187
d	0.3887	0.4118	0.0202	0.0144
θ	0.2955	0.3860	0.0185	0.0130

评价 | 新交互开启新攻击面

欢迎各位一起学习讨论



零数学基础深度学习入门

吴伟

lazyparser@gmail.com
中国科学院软件研究所