用TK1加速人工智能在机器人中的应用

吴 伟
lazyparser@gmail.com
HelloGCC.org

关于我

中科院软件所临时工，HelloGCC工作组成员

目前在做面向某神经网络芯片的调试工具

做过编译、JIT优化、IDE扩展、软件测试等

喜欢给大大小小的开源项目提交小patch

网站： hellogcc.org，hellocompiler.com

IRC： #hellogcc on freenode

邮件列表： hellogcc@freelist.org

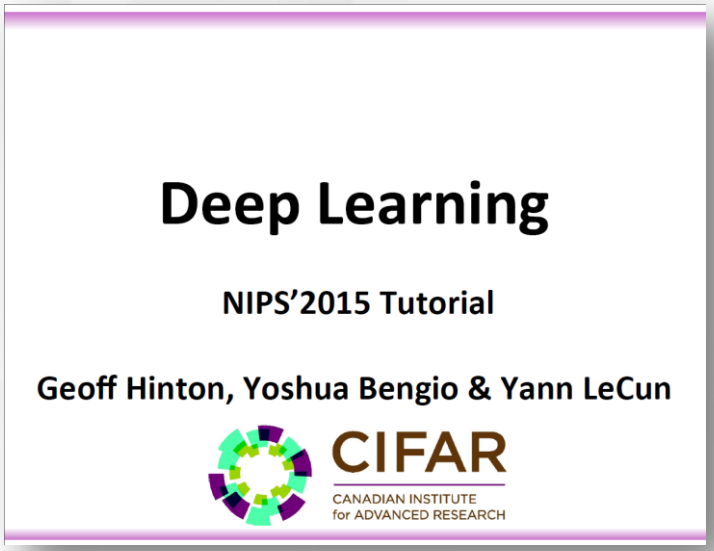
本次讨论的内容

深度学习的超简短入门

用Caffe/TensorFlow搭建神经网络

将TK1作为平台搭建智能应用的方法

先缩小一下入门的范围



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

先缩小一下入门的范围

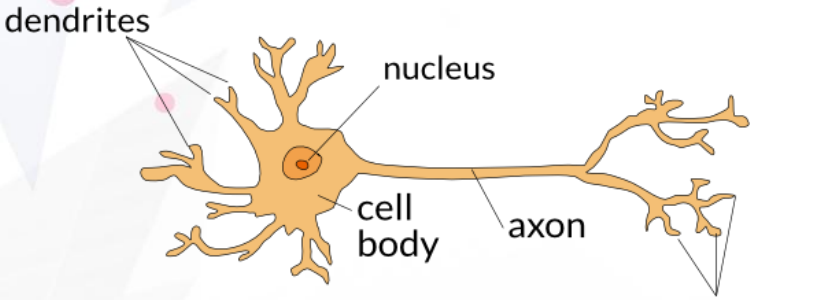
Deep Learning Neural Networks

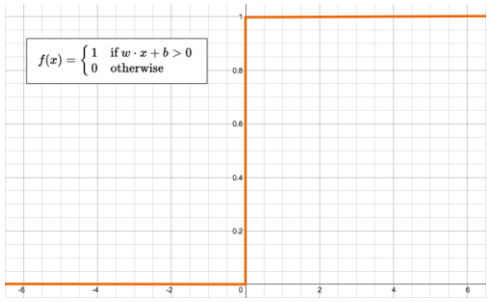
深度学习： 深度神经网络

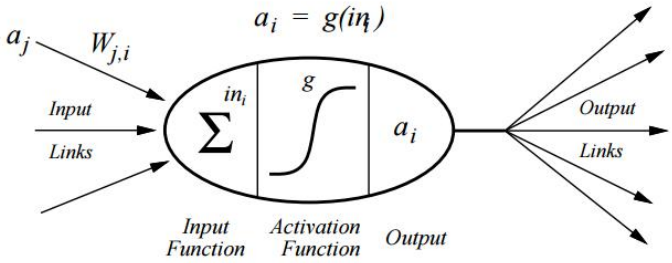
先缩小一下入门的范围

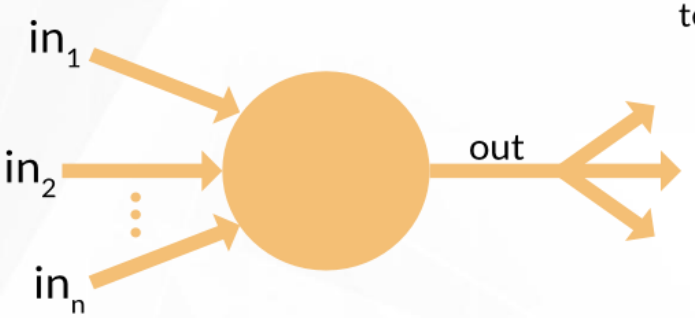
Deep Learning Neural Networks

深度学习： 深度神经网络

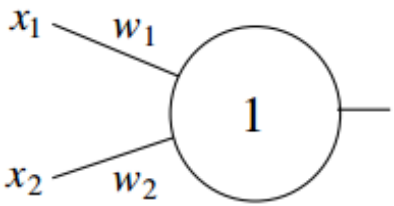


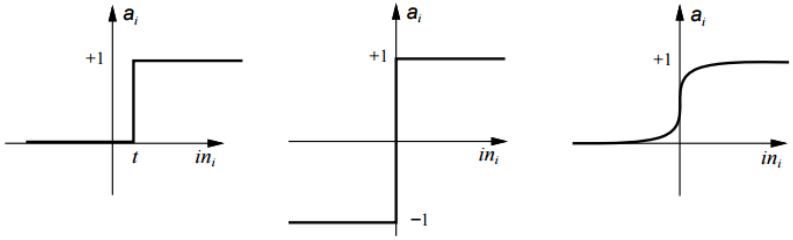






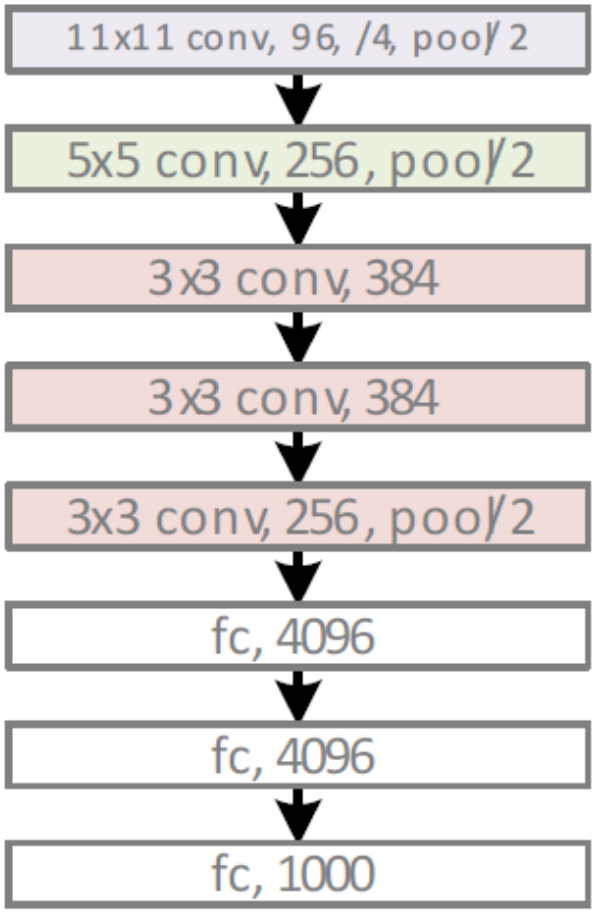
$$a_i = g(\sum_j W_{j,i} a_j)$$

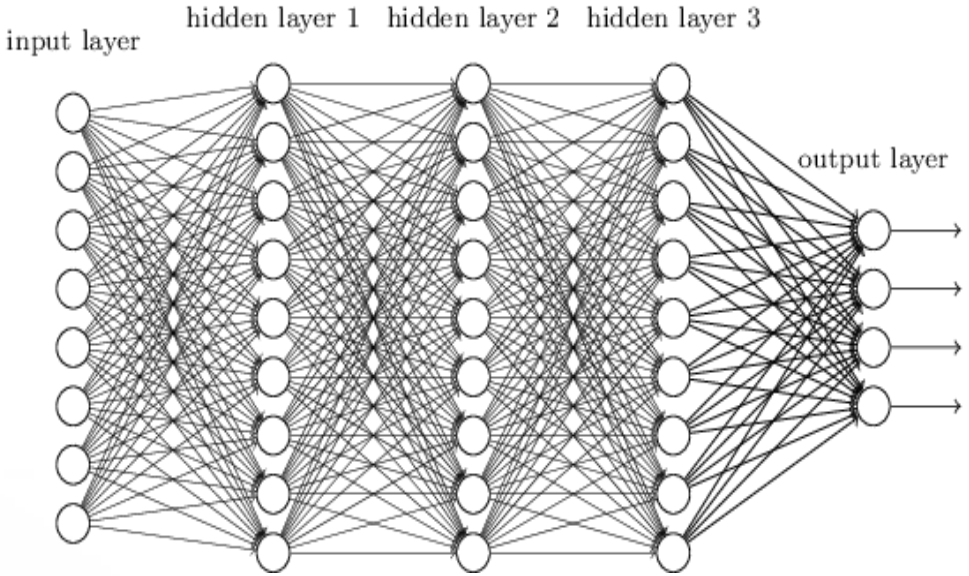


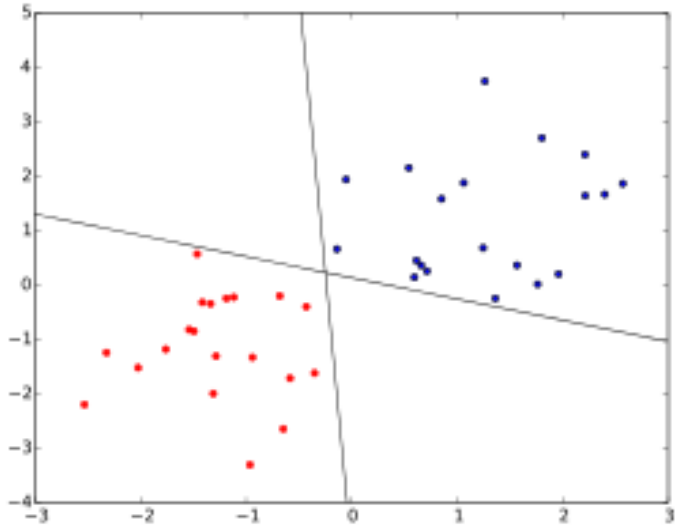


https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-f01/www/handouts/110601.pdf

Credit: https://page.mi.fu-berlin.de/rojas/neural/chapter/K4.pdf https://appliedgo.net/perceptron/







Credit: https://en.wikipedia.org/wiki/Perceptron

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

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

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

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

$$p(v_i = x|\mathbf{h}) = \frac{1}{\sigma_i\sqrt{2\pi}} \cdot e^{-\frac{\left(x - a_i - \sigma_i \sum_j w_{ij} h_j\right)^2}{2\sigma_i^2}},$$

$$p(h_j = 1|\mathbf{v}) = \text{sigm}\left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij}\right). \tag{4}$$

本次讨论的目标

> 一直从事底层软件/嵌入式的工作，数学和英语功底都不太好，也没有接触过机器学习。最近看到深度学习（人工智能）非常火，想要了解一下，不知道应该从何入门？

本次讨论的目标

> 一直从事底层软件/嵌入式的工作，数学和英语功底都不太好，也没有接触过机器学习。最近看到深度学习（人工智能）非常火，想要了解一下，不知道应该从何入门？

真正零基础，最浅的深度学习入门

~~现场编一个~~故事背景

某公司老板张总，有一天突然有了一个灵感：

设计开发一个**人工智能**软件，利用当天股价的涨跌来预测第二天房价的涨跌。

~~现场编一个~~故事背景

某公司老板张总，有一天突然有了一个灵感：

设计开发一个**人工智能**软件，利用当天股价的涨跌来预测第二天房价的涨跌。

于是找来工程师小李，要求实现这个app：

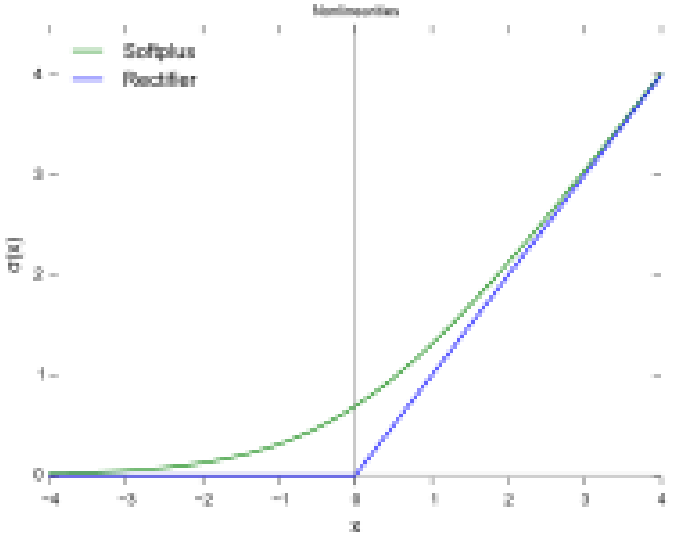
```
if 股价涨 then 显示房价涨;
else 显示房价不会涨;
```

产品原型

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

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

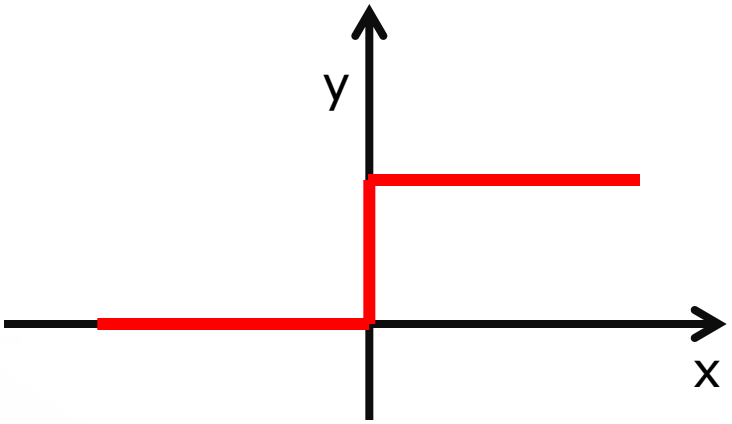
产品原型（高级版）

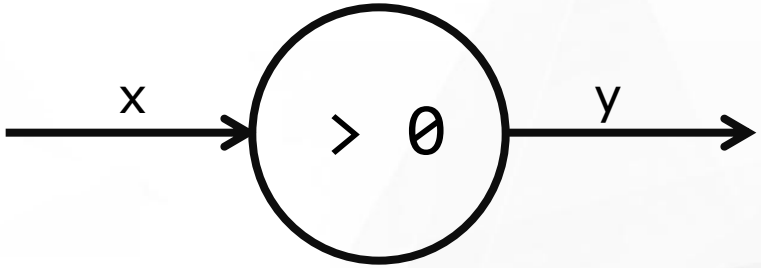



```
if (x > 0)
    y = x
else /* x <= 0 */
    y = 0
```

产品原型

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





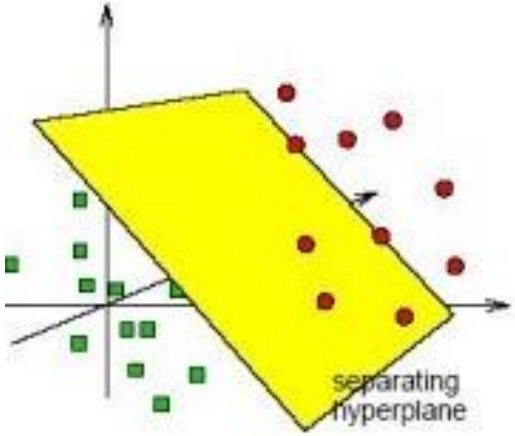
故事背景

小李很快做出来了产品的原型，但是显然，这个软件是预测不准的……

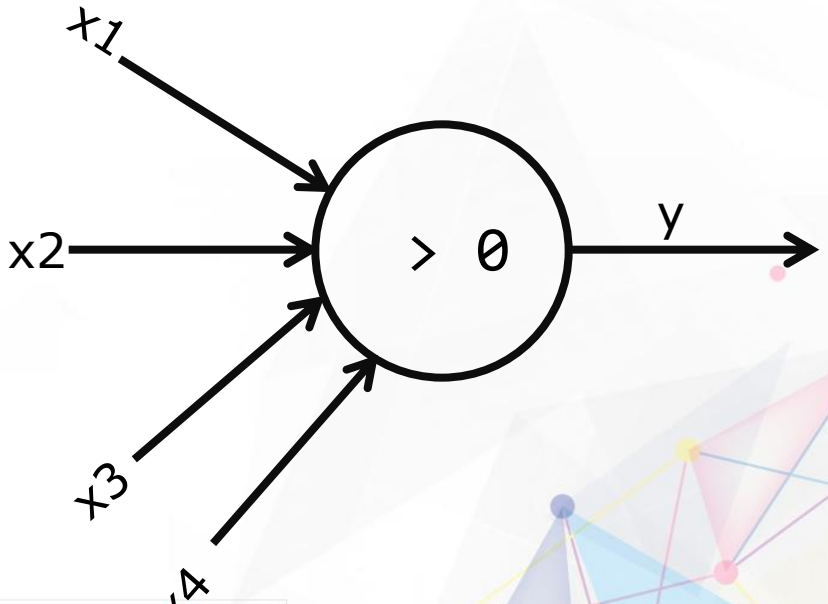
张总思考了一下，觉得应该把油价、汇率、金价都考虑进去，这样预测就准了。

产品原型

```
foo(x1,x2,x3,x4) {
    return x1+x2+x3+x4;
}
```




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



图片来源：http://taop.marchtea.com/07.02.svm.html

故事背景

小李很快迭代出了产品，但是显然，
这个软件还是预测不准的……

张总思考了一下，觉得股价、油价、汇率、金价
对房价的影响是不一样的，也就是权重（weights）
是不同的。

故事背景

小李明白权重的意思，但不知道具体怎么取值。

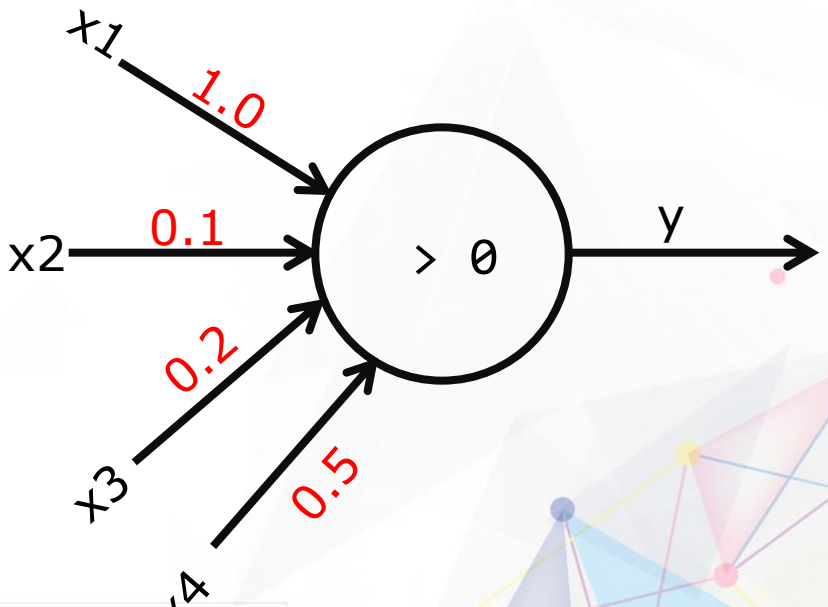
张总根据经验，告诉小李：

股价、油价、汇率、金价，权重分别是

1.0, 0.1, 0.2, 0.5

产品原型

```
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

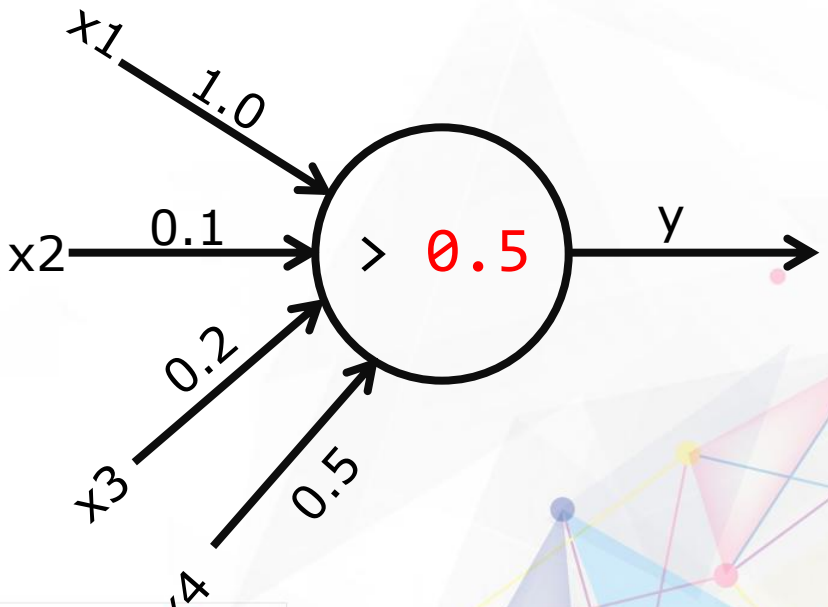
张总进一步觉得，长远来看，就算市场基本面不变，房价也在缓慢上涨。所以只有指标看跌到一定程度，才能预测说是房价要跌。

张总根据经验，告诉小李：

只有指标达到了0.5了，才能算是要跌。

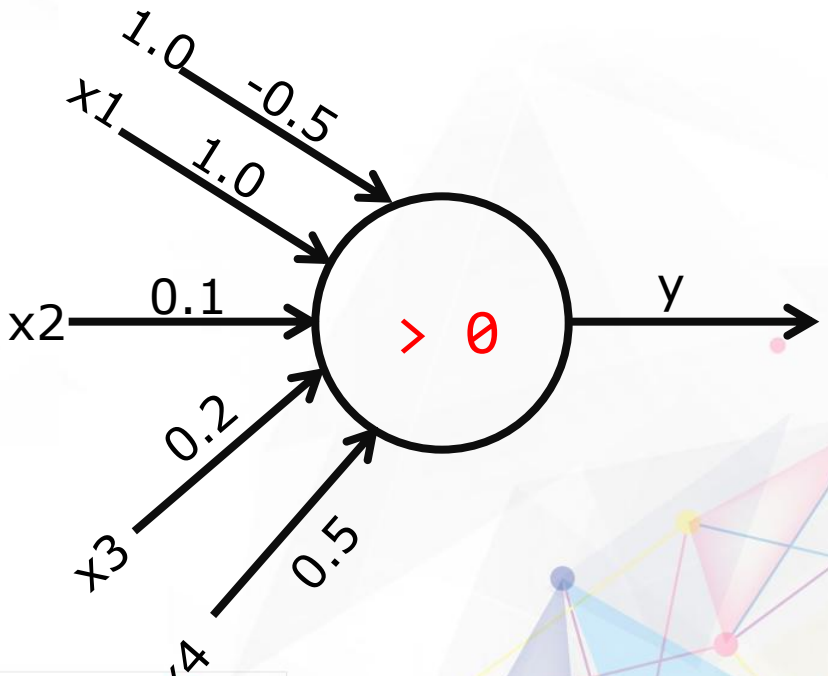
产品原型

```
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
```



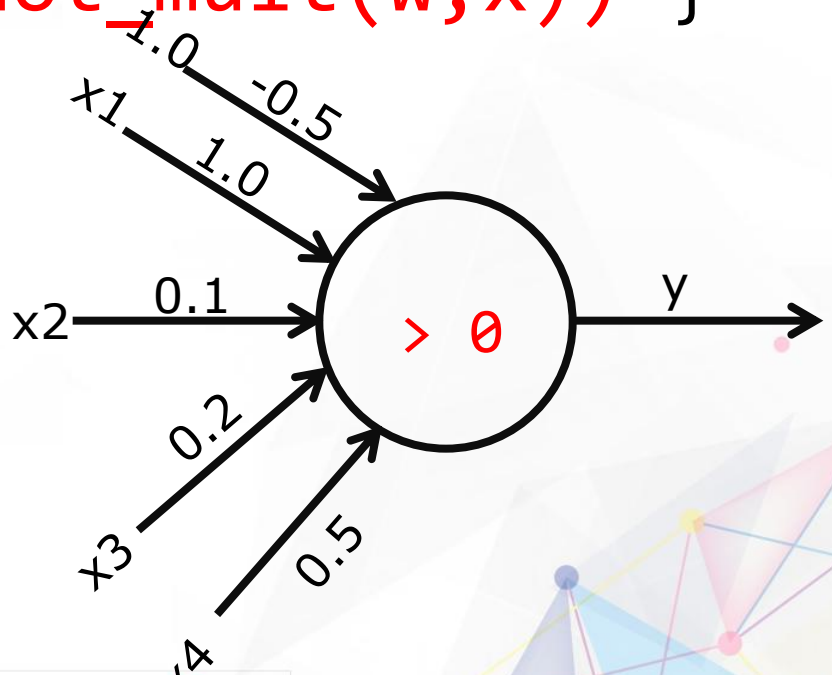
产品原型-小李重构了一下

```
foo(x1,x2,x3,x4) {
      return -0.5*1 + 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-小李重构了一下

```
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
```

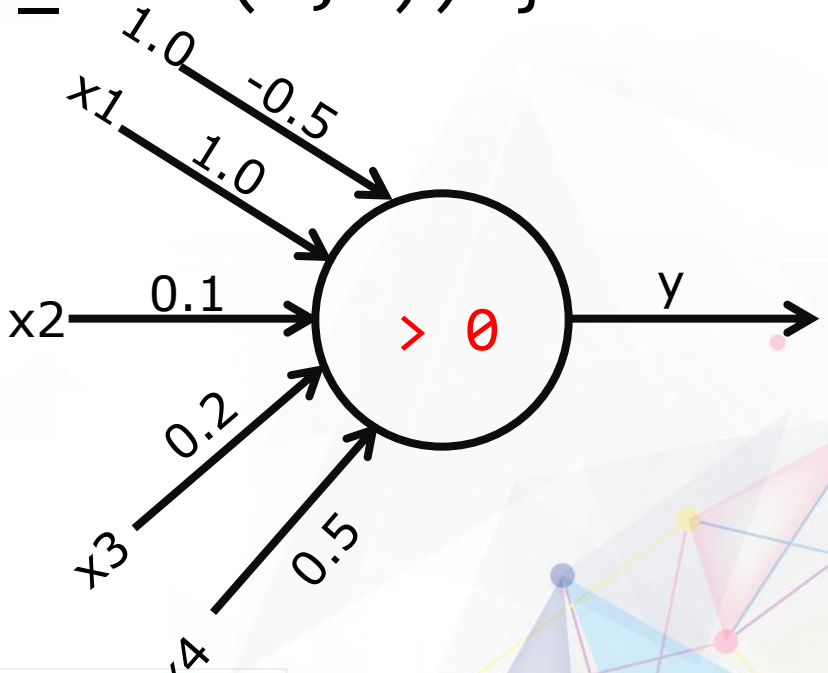


产品原型-小李重构了一下

```
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

软件迭代了四个版本了，测起来还是不准。

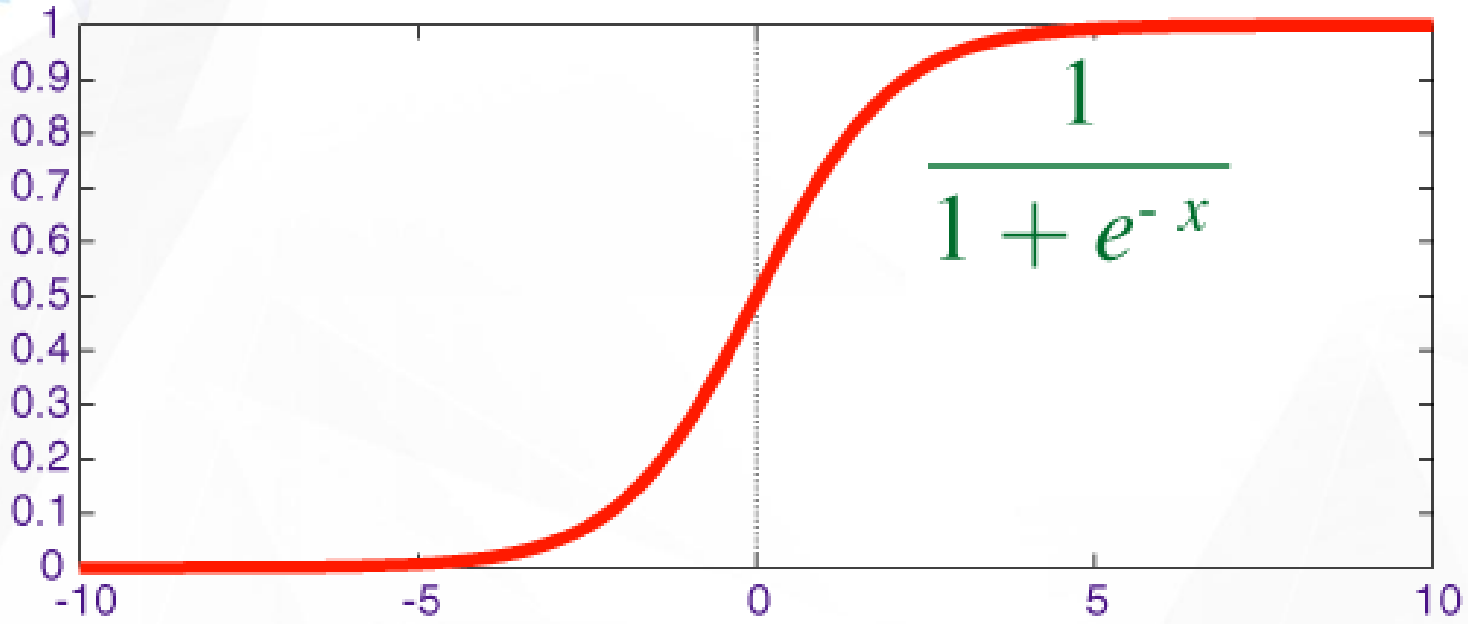
张总读了微信收藏里的深度学习文章，告诉小李：

这个关系很复杂，我们之前用的是线性关系，

不能有效的表示这种复杂的关系，

我们要上“非线性关系”，用最流行的Sigmoid。

Sigmoid 函数（两种解释，类比HTML5）

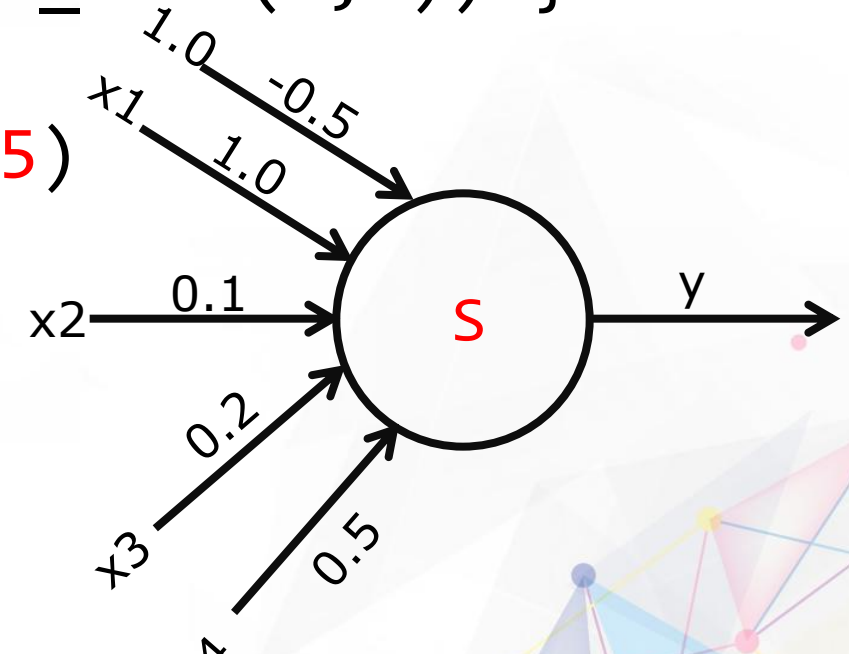


（神经网络用Sigmoid函数，其实是为了后面反向传播算法的时候计算微分方便）

https://www.quora.com/What-is-the-sigmoid-function-and-what-is-its-use-in-machine-learnings-neu

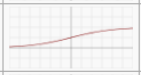
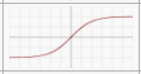
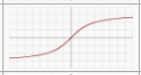
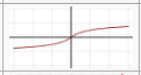
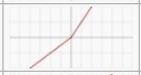
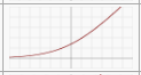
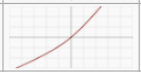
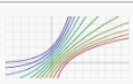
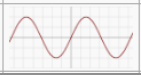
产品原型-小李重构了一下

```
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
```



外传：激活函数（Activation Func）

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

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		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	C^∞	Yes	No	Yes
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$	$(-1, 1)$	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]		$f(\alpha, x) = \begin{cases} -\frac{\log_e(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x}-1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1-\alpha(\alpha+x)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes iff $\alpha = 0$
Sinusoid		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$	C^∞	No	No	Yes

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

故事背景

软件迭代了五个版本了，测起来还是不准。

张总觉得之所以测不准，还是因为模型太简单了。现在深度学习，动辄就说“一百多层”，看看自己，一层都没有，肯定不行。

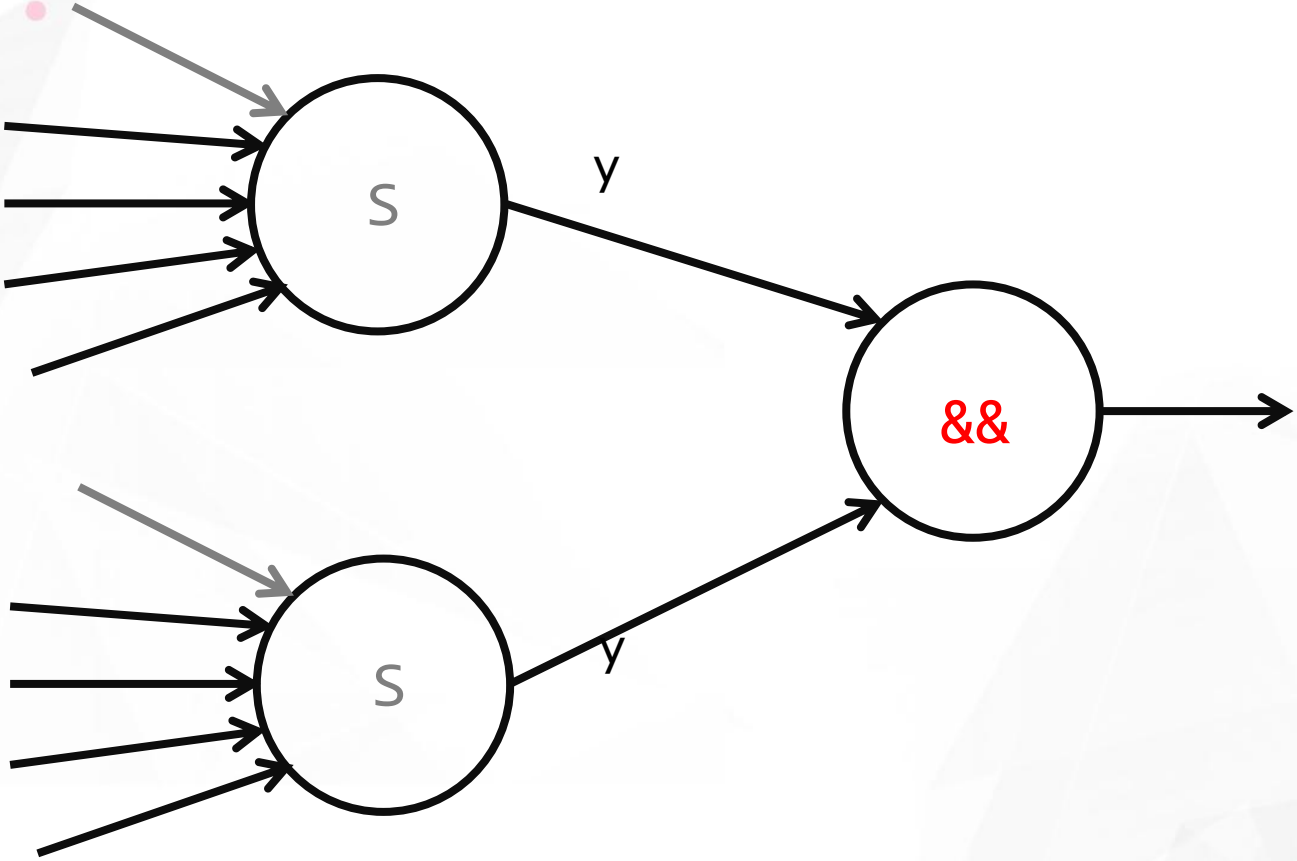
故事背景

一天早晨，张总突然想到，买房对谁都是大事，房价预测岂能用一个公式“一刀切”？

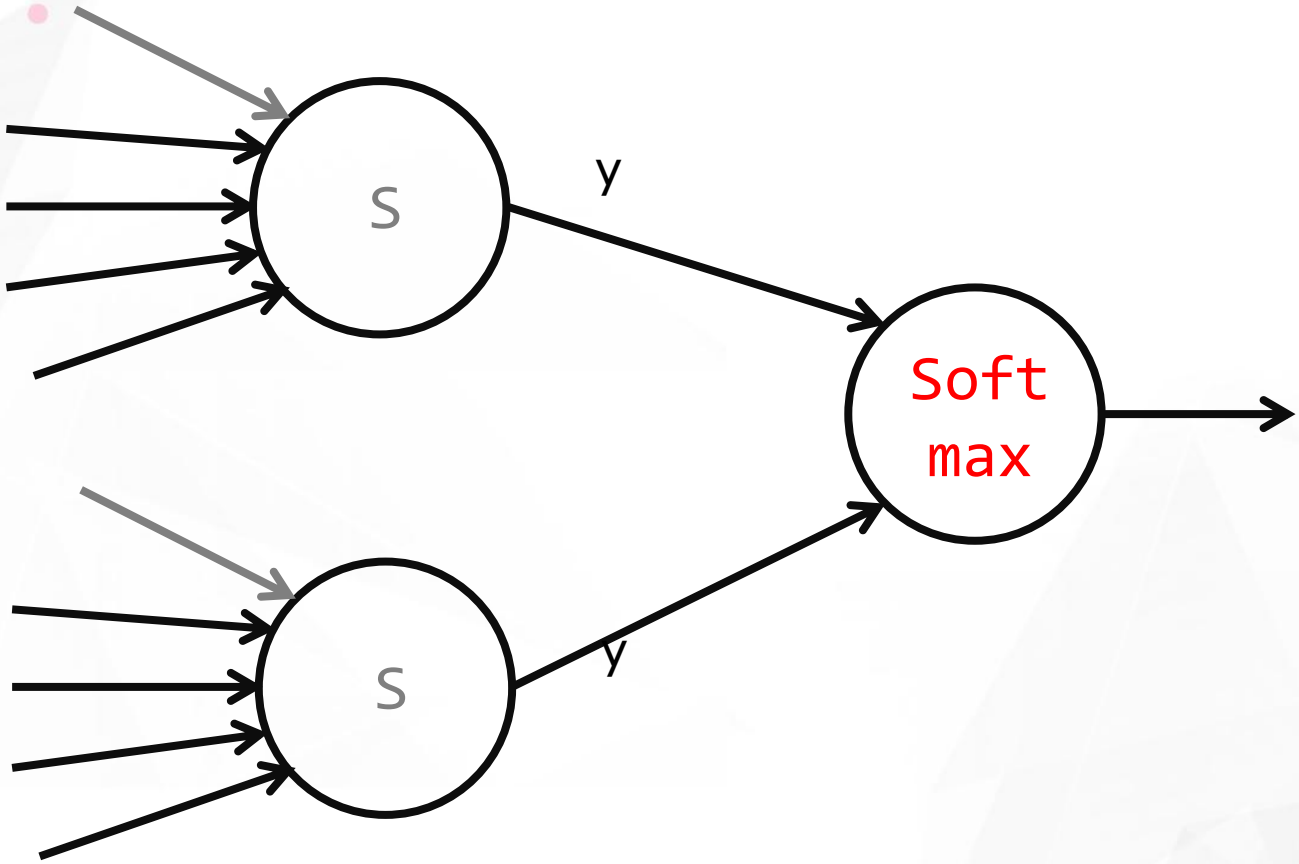
于是把小李找来，告诉他：

我们要进一步完善，用多个不同的计算公式，综合最终的结果给出预测结果。

小李很快复制粘贴出来了结果， 全说涨才涨



照猫画虎将最后的投票函数改成了 softmax



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

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

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……

张总将这个问题交给了小李，让小李自己去完成。

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），也有张总给的经验权重（初始值）。

小李编了几个权重，效果都不好。接下来怎么办，小李也不知道了……

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 procedure[1].

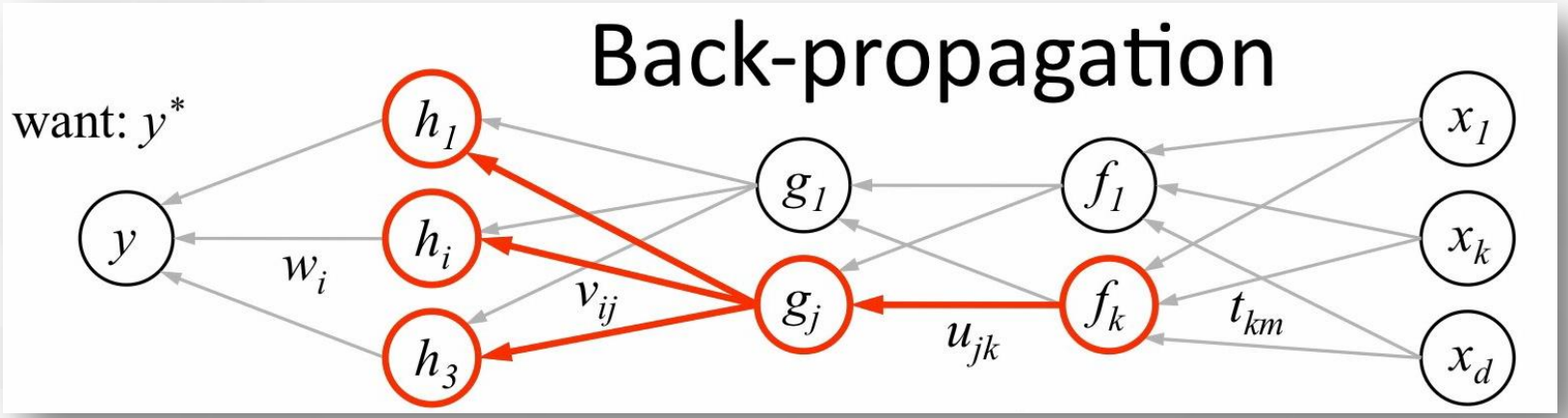
more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined 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.

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

（不理解没关系，该算法已经不用自己实现了）

Backpropagation algorithm

- Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

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

For $i = 1$ to m

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

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

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

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

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

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

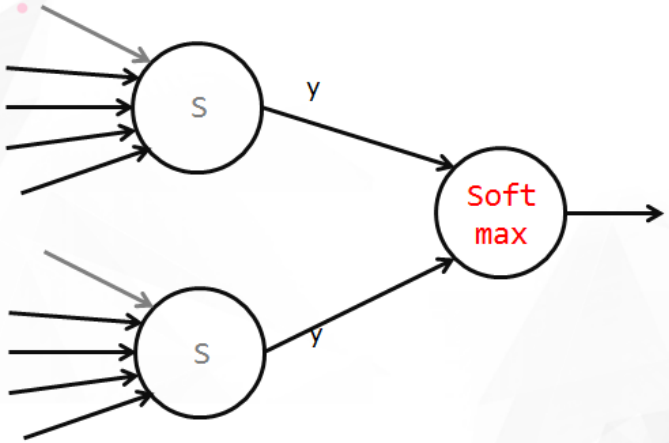
$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

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

故事背景

小李激动的训练完，效果确实比以前好了很多，但是……

权重还是都是一样的嘛！



解法：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

小李用了反向传播算法，准确率上去了。

老板很开心，让他加层，并问小李里面权重的意思。

小李表示不知道，于是去问各路大神.....

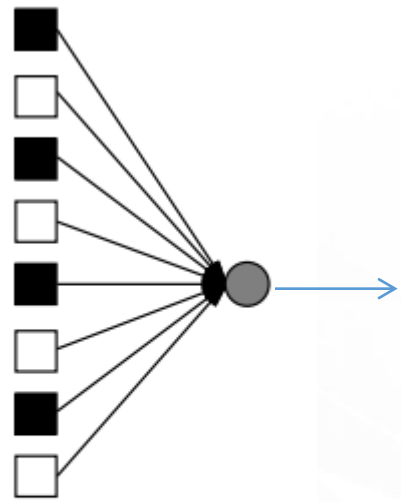
"不知道"

"很复杂"

"不好说"

"有点难"

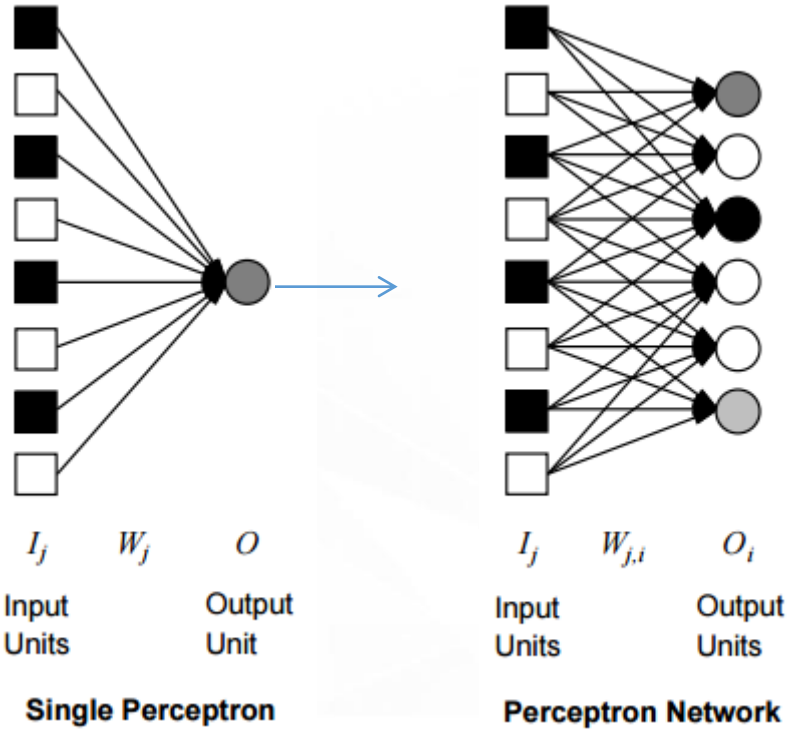
算法在手，层数你有



Single Perceptron

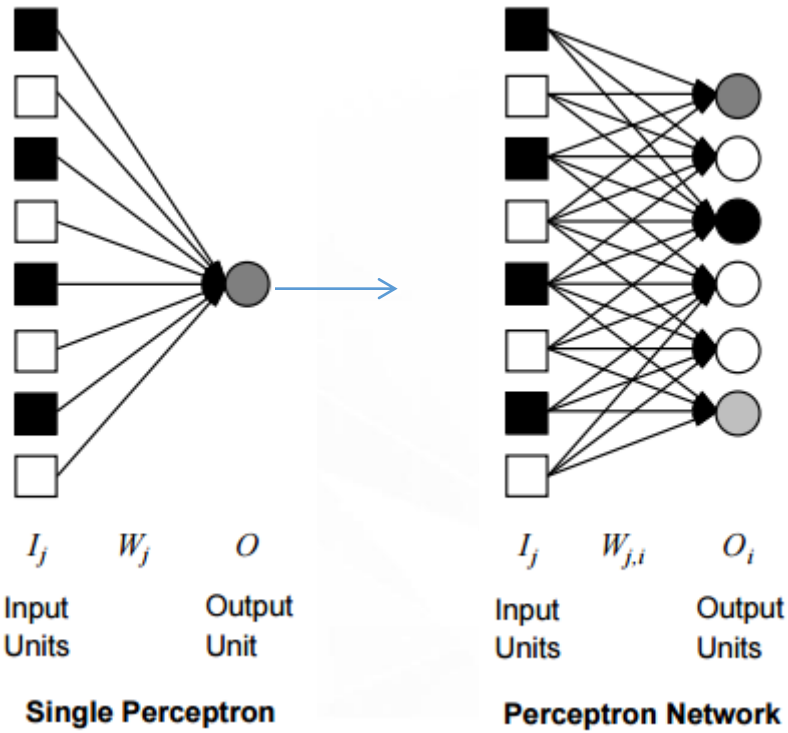
https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-f01/www/handouts/110601.pdf

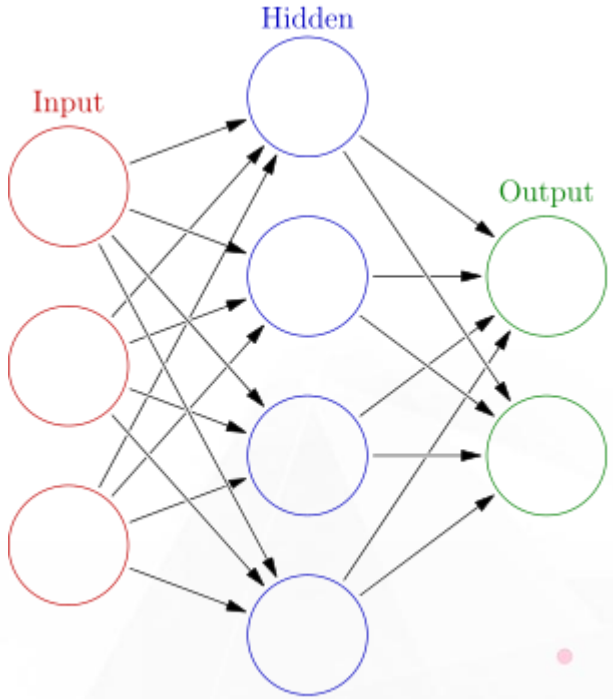
算法在手，层数你有



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

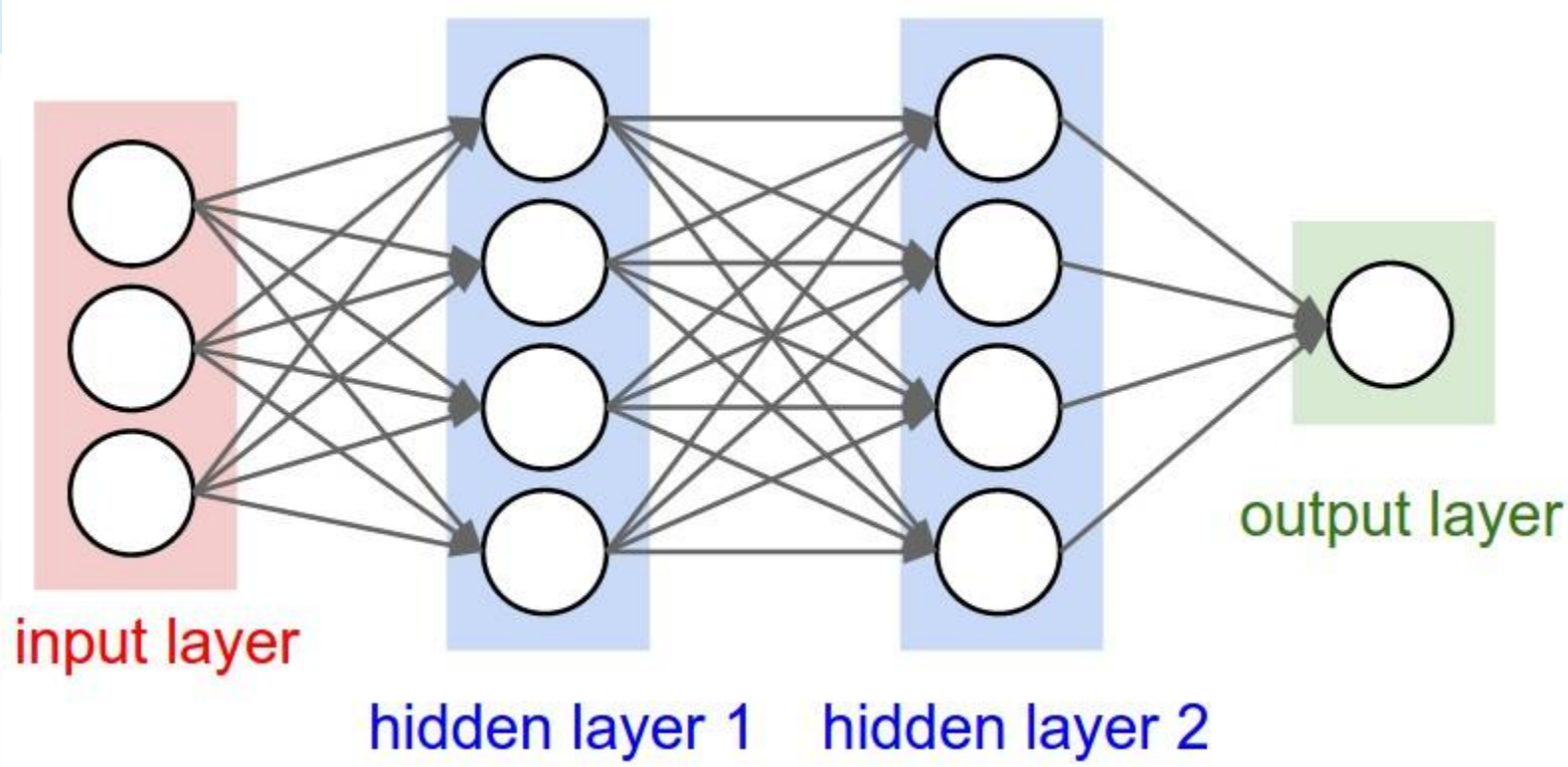




https://en.wikipedia.org/wiki/Artificial_neural_network

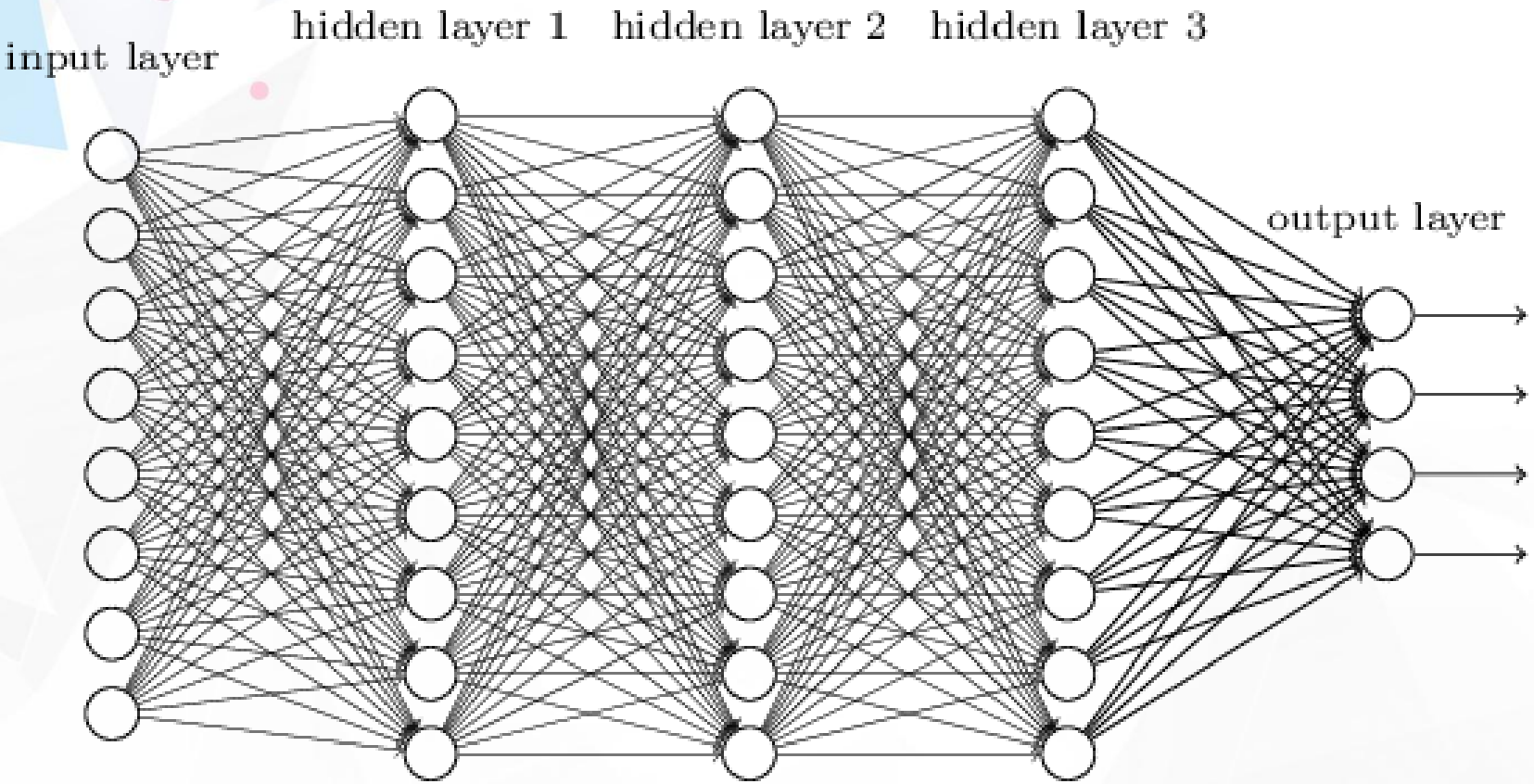
https://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-f01/www/handouts/110601.pdf

算法在手，层数你有



http://cs231n.github.io/neural-networks-1/

算法在手，层数你。。。额，不行了



http://neuralnetworksanddeeplearning.com/chap6.html

不行了：反向传播算法的局限性

梯度消失

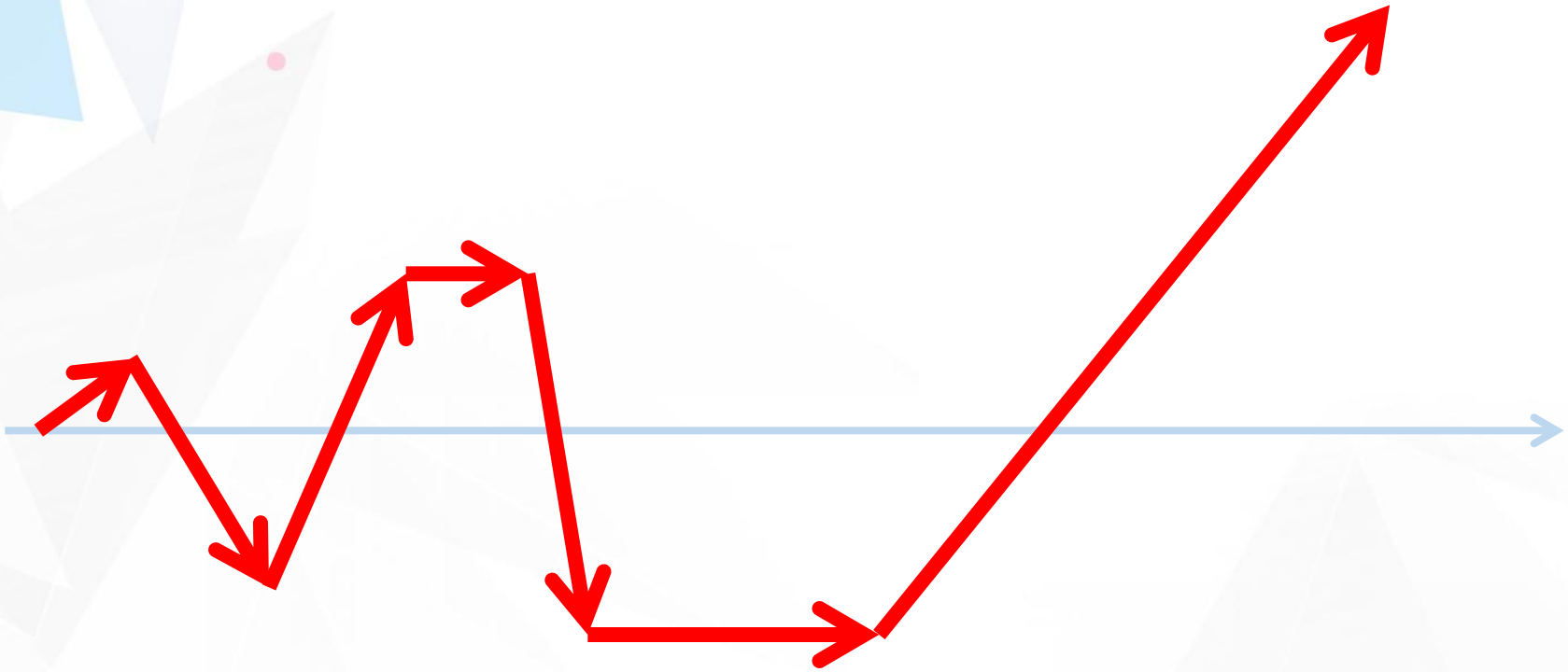
局部最优

过学习

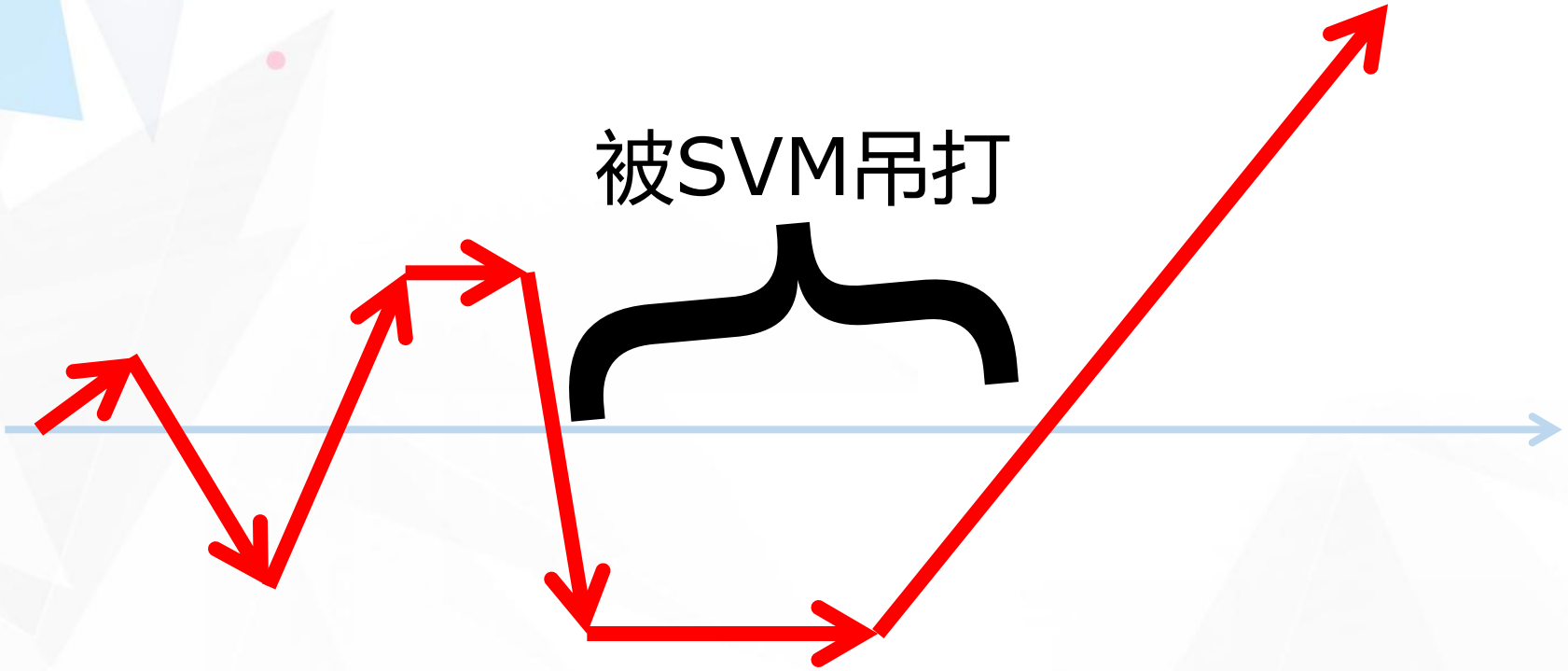
关于过学习（Overfit）和欠学习（Underfit）

> 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

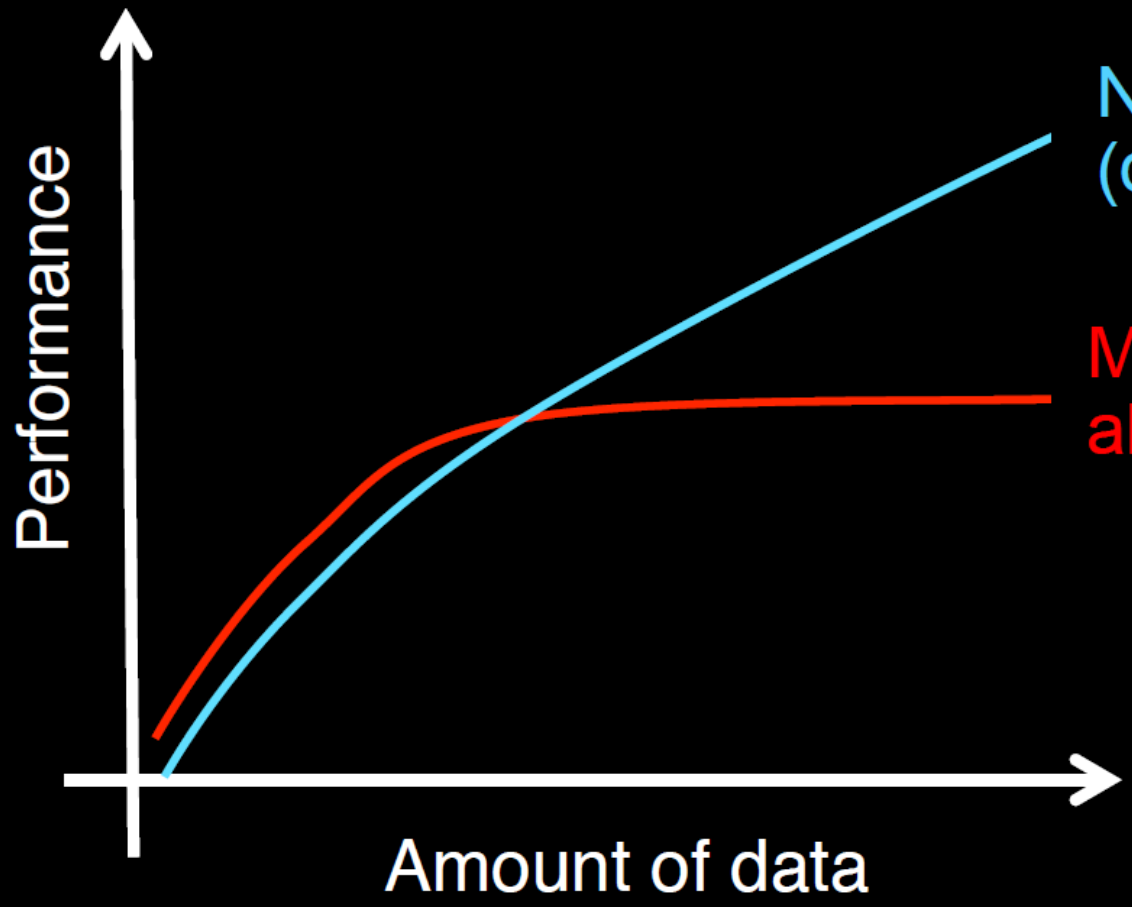
神经网络坐了十几年的冷板凳



神经网络坐了十几年的冷板凳



Data and machine learning



Andrew Ng

Neural network
Back propagation

Nature

Deep belief net
Science

Microsoft

Speech

IMAGENET

1986 被SVM吊着打 2006 2011 2012

Credit: Xiaogang Wang, The Chinese University of Hong Kong

Neural network
Back propagation

Nature

Deep belief net
Science

Microsoft

Speech

IMAGENET

1986

06

2011 2012

面壁十年图破壁

我Hinton悟到了

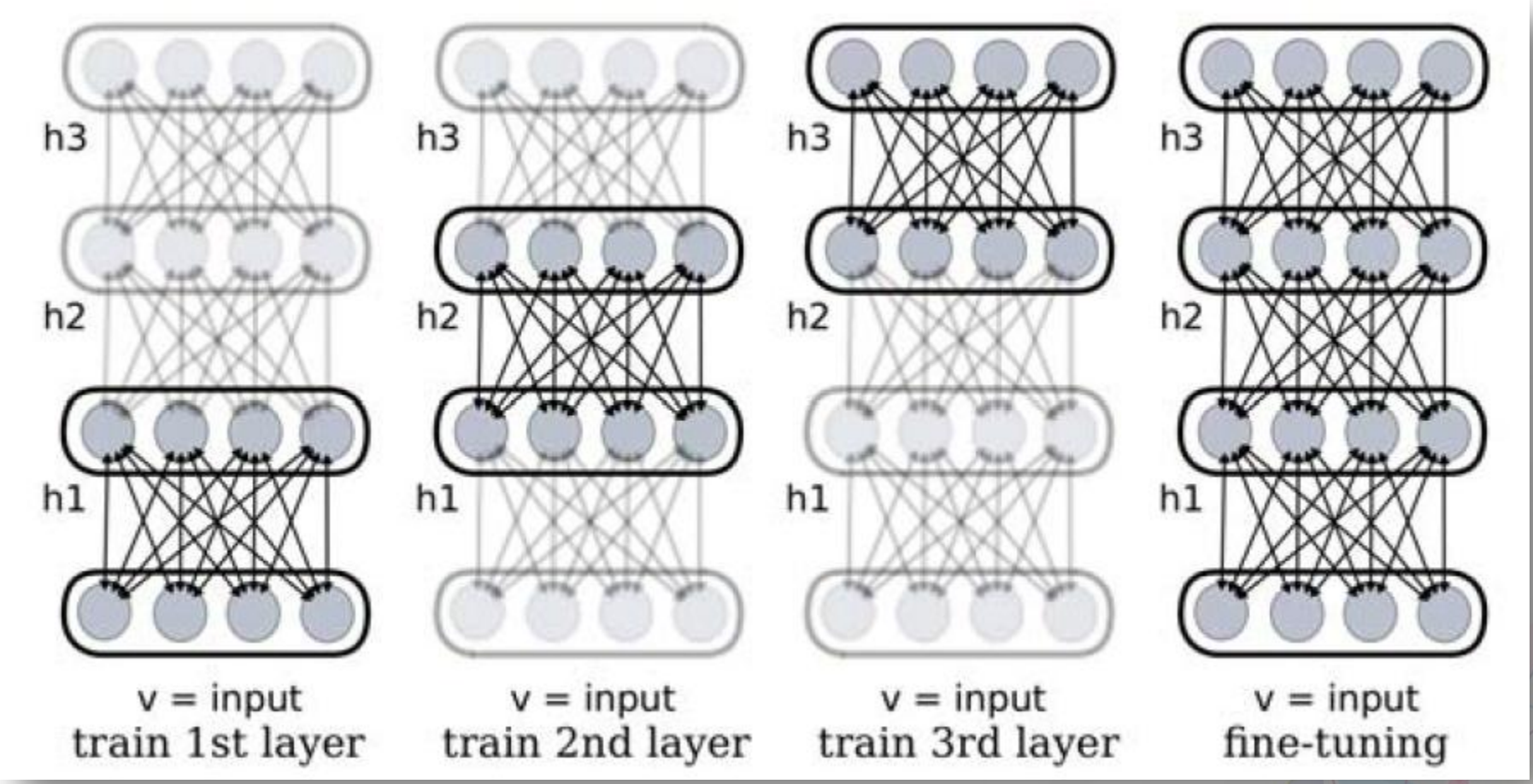
一！层！一！层！来！

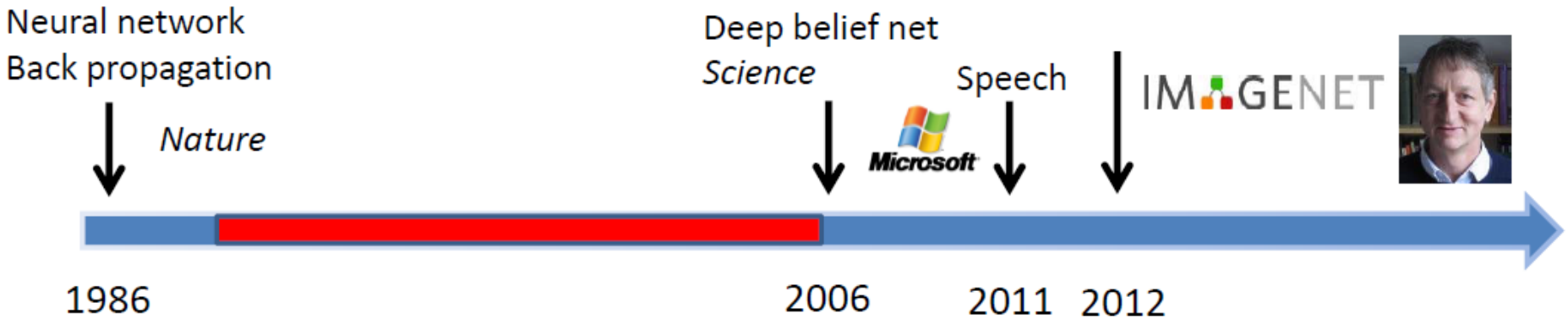
SACC 2016

68 ChinaUnix ITPUB

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





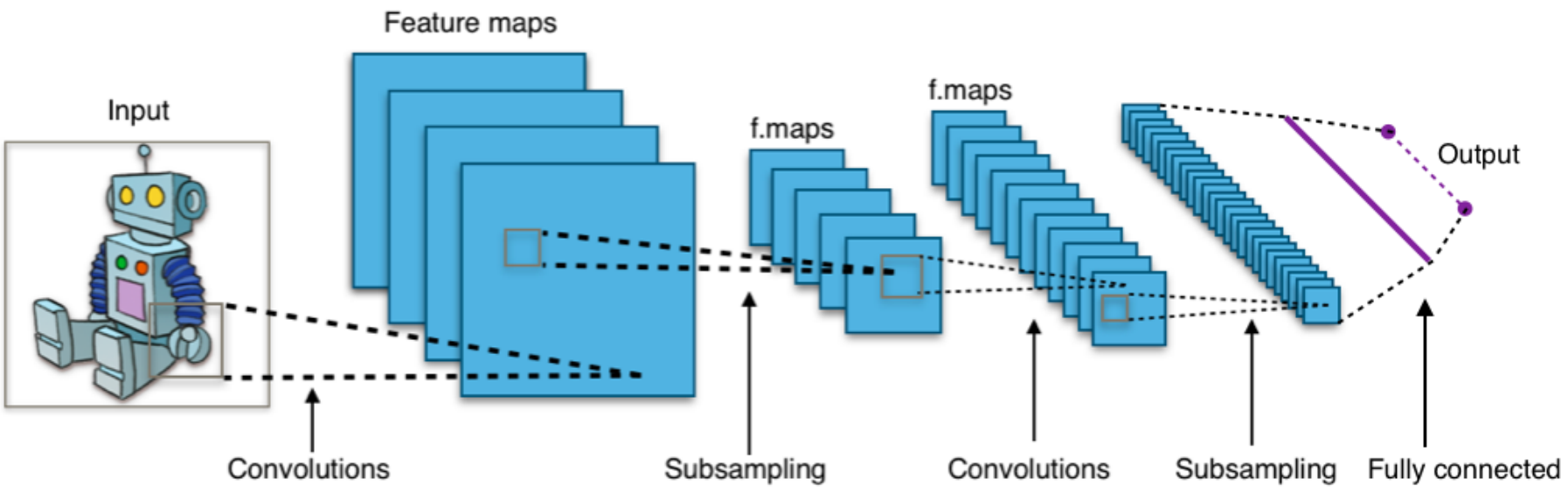
减少训练权重的规模

试试我乐康卷 ☺

（实为LeCun卷积网络）

Credit: Xiaogang Wang, The Chinese University of Hong Kong

乐康卷：大体架构

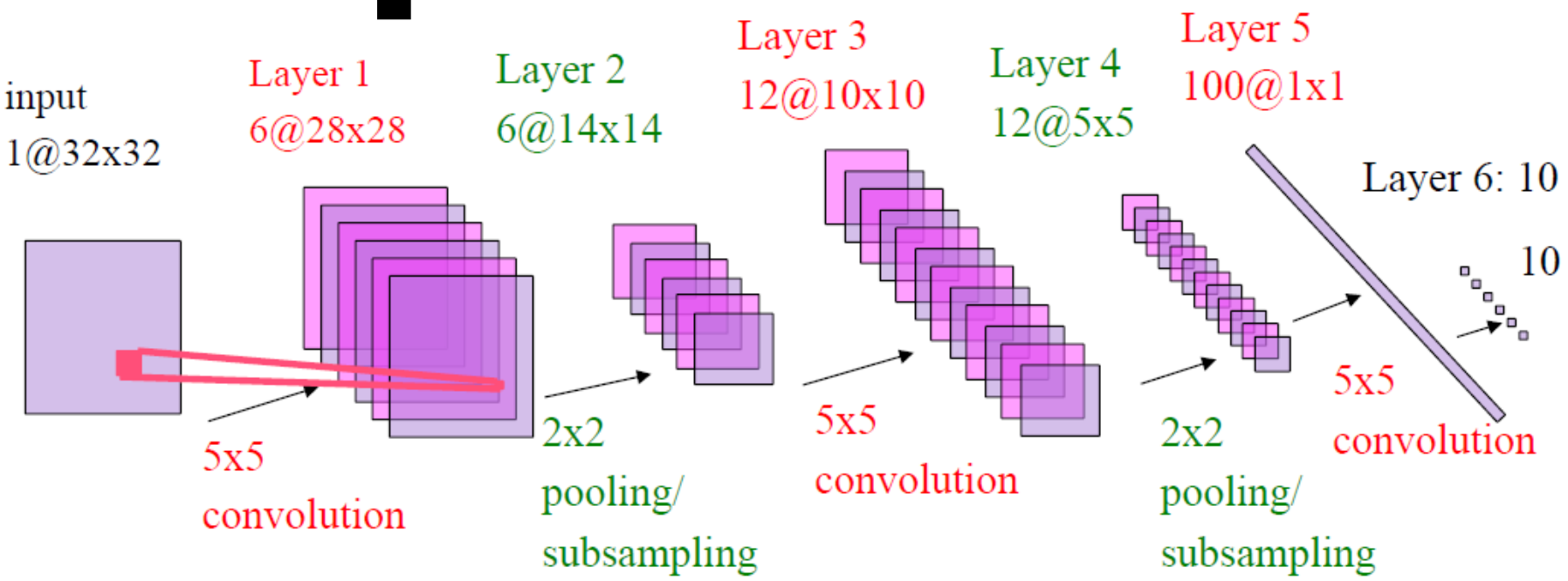


https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

乐康卷：参数配置

LeNet5

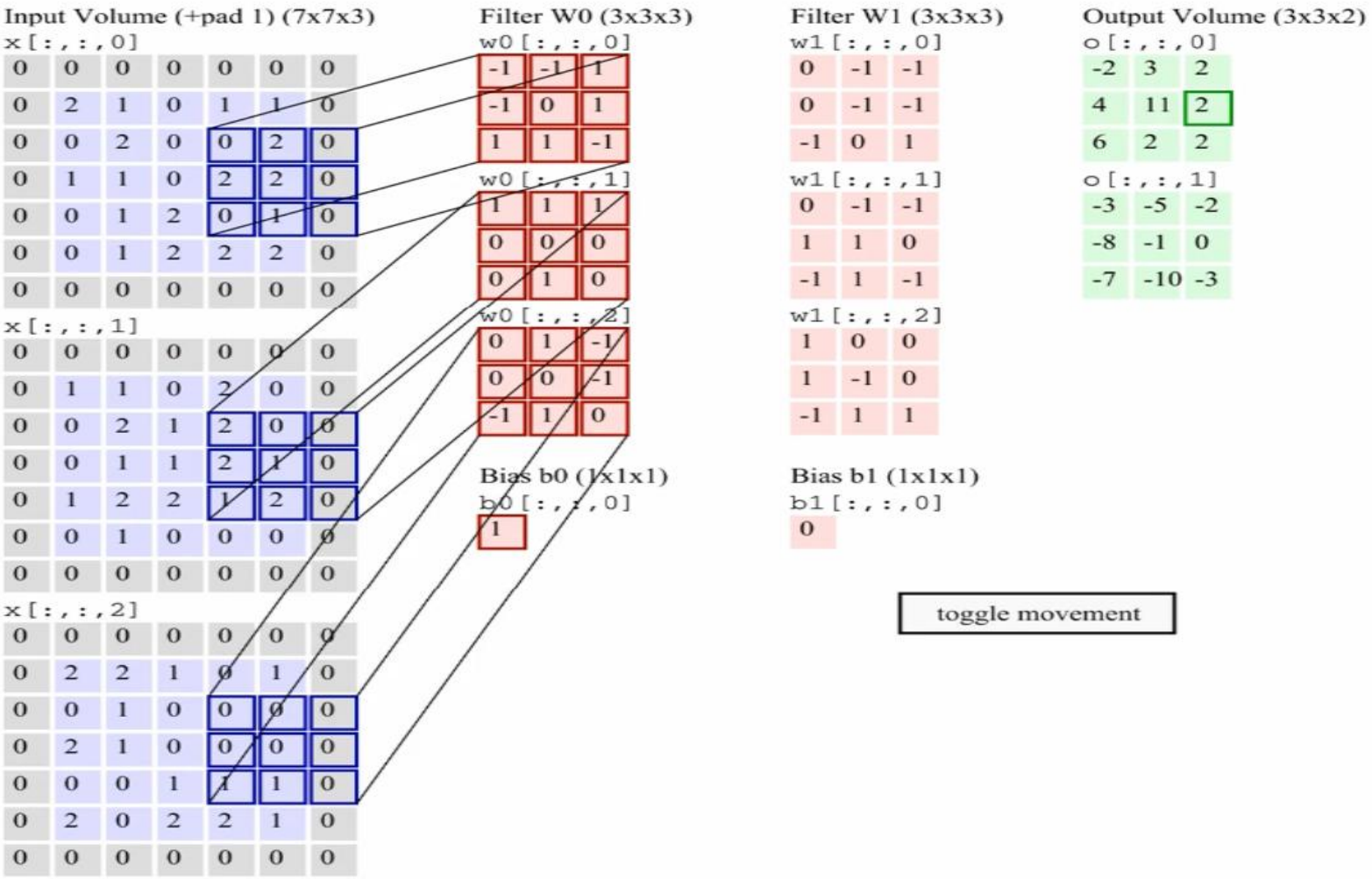
针对MNIST MNIST稍后介绍



乐康卷：具体实例（来自LeCun大神）

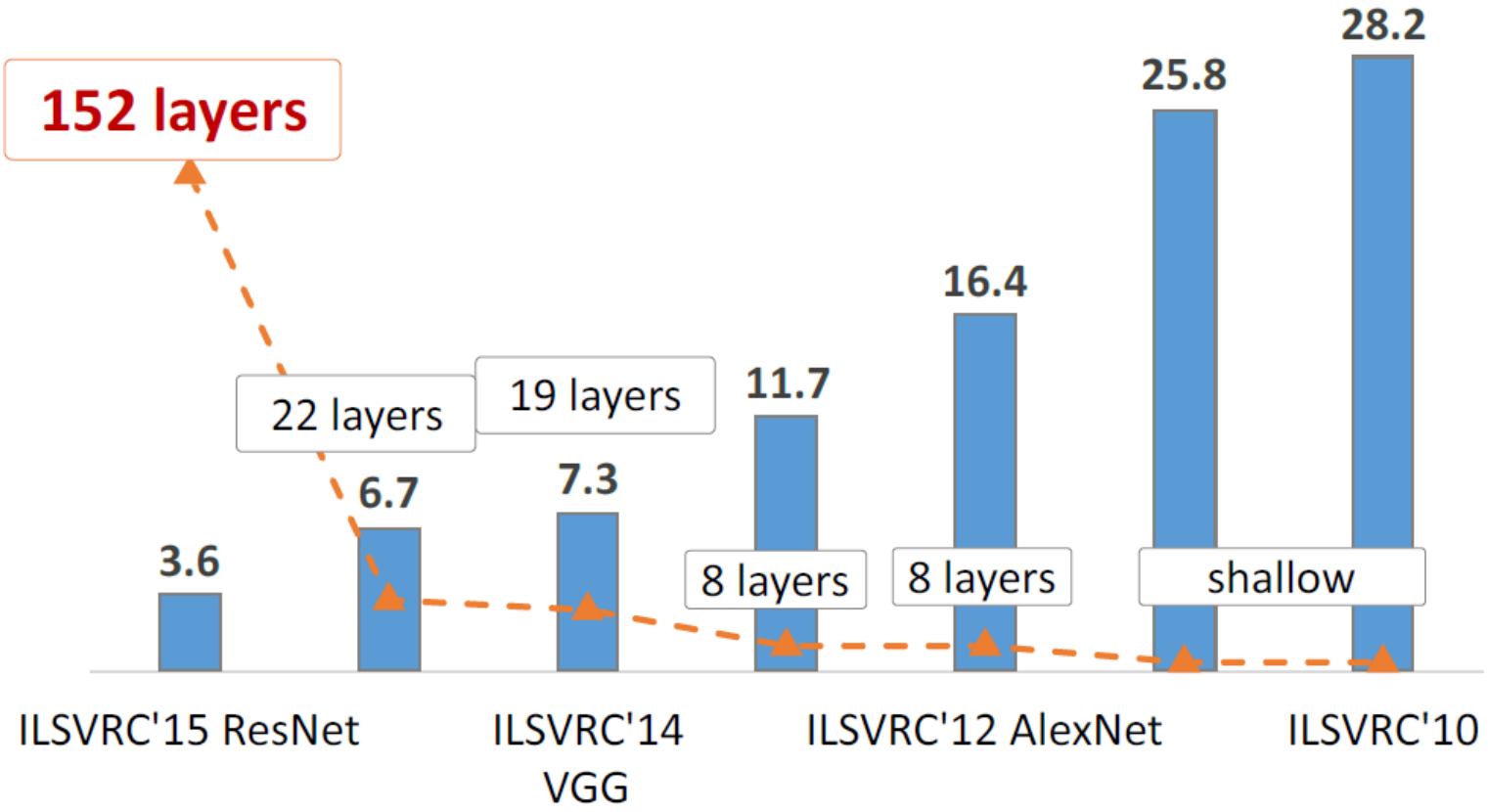
Multiple Convolutions

Y LeCun



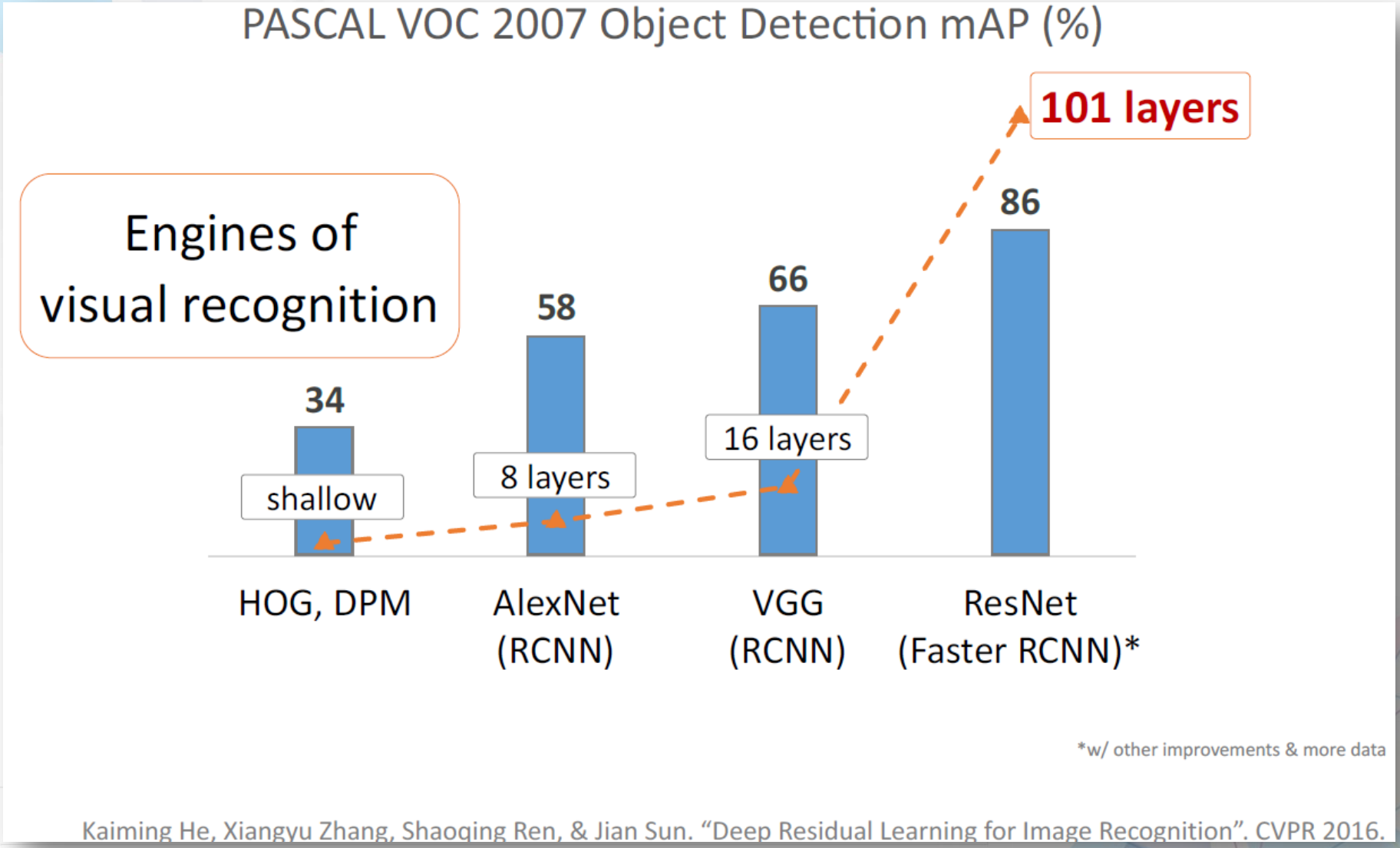
Animation: Andrej Karpathy http://cs231n.github.io/convolutional-networks/

算法和模型都完备后，PhD们的心开始躁动起来



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

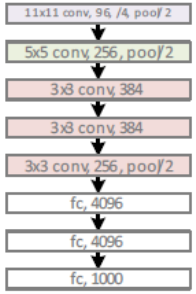
算法和模型都完备后，PhD们的心开始躁动起来



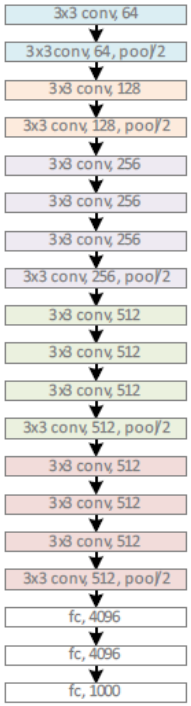
算法和模型都完备后，PhD们的心开始躁动起来

算法和模型都完备后，PhD们的心开始躁动起来

AlexNet, 8 layers (ILSVRC 2012)



VGG, 19 layers (ILSVRC 2014)

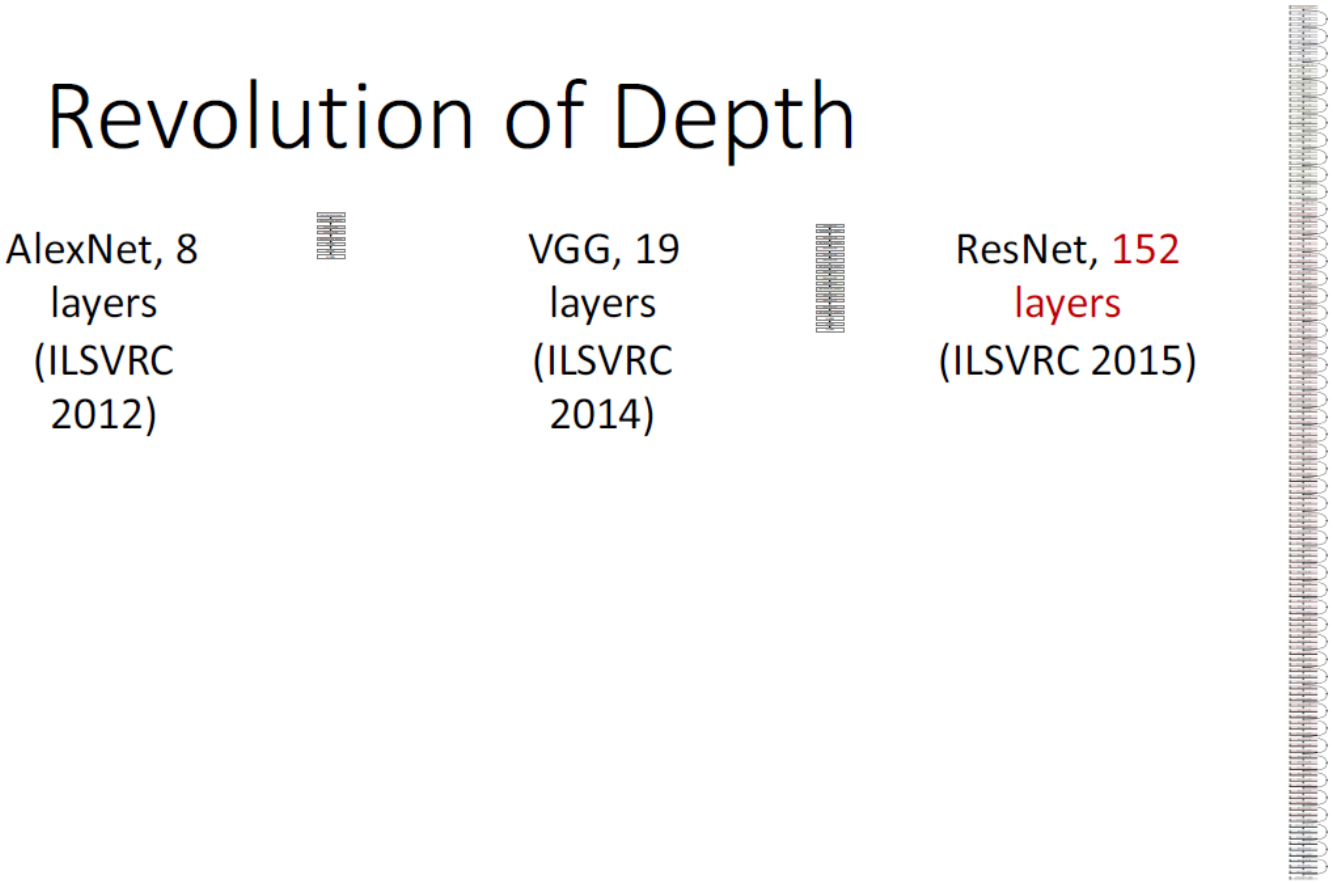


GoogleNet, 22 layers (ILSVRC 2014)

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



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

README.md

Deep Residual Networks with 1K Layers

1001层DRN

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

Microsoft Research Asia (MSRA).

Table of Contents

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 Univeristy for the ease of release.

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

https://github.com/KaimingHe/resnet-1k-layers

故事背景

张总看到这里很激动：之所以预测不好，

一定是因为层数不够。直接加到两千层试试！

小李：……

小结与思考

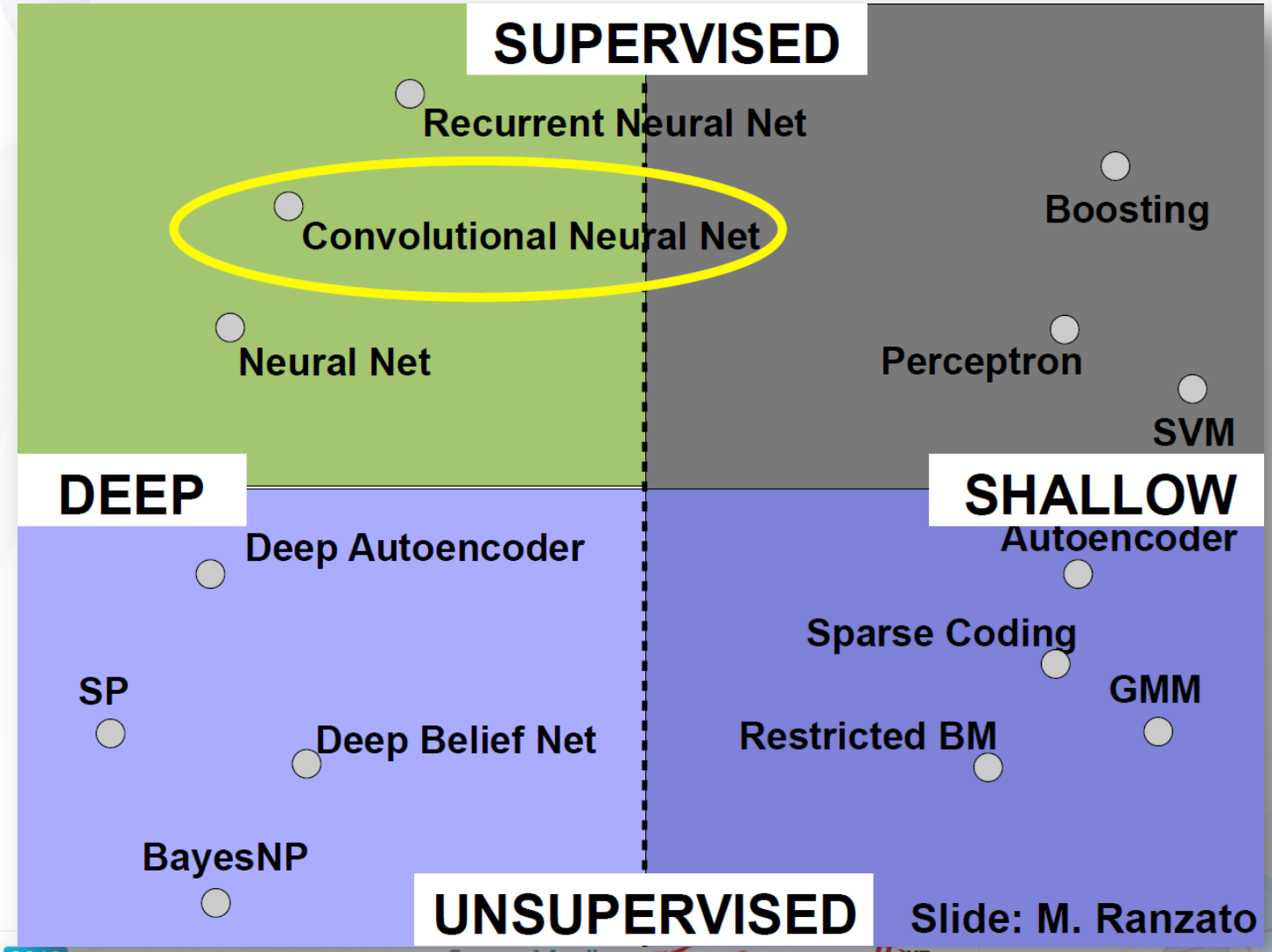
为什么深度学习会在图像、音频、NLP领域成功？

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？

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

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

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

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

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

$$p(v_i = x|\mathbf{h}) = \frac{1}{\sigma_i\sqrt{2\pi}} \cdot e^{-\frac{\left(x - a_i - \sigma_i \sum_j w_{ij} h_j\right)^2}{2\sigma_i^2}},$$

$$p(h_j = 1|\mathbf{v}) = \text{sigm}\left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij}\right). \tag{4}$$

An Introduction to Deep Learning Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

本阶段讨论的目标

> 一直从事底层软件/嵌入式的工作，
> 数学和英语功底都不太好，最近学了
> 一些深度学习的基础，有了理论基础，
> **想实际操刀练习，该如何上手？**

为什么说前页PPT上那个问题不是好问题？

深度学习怎么玩，取决于你多有钱

本阶段讨论的目标

> 一直从事底层软件/嵌入式的工作，
> 数学和英语功底都不太好，最近学了
> 一些深度学习的基础，有了理论基础，
> **想实际操刀练习，目前没有什么预算，**
> **该如何上手？**

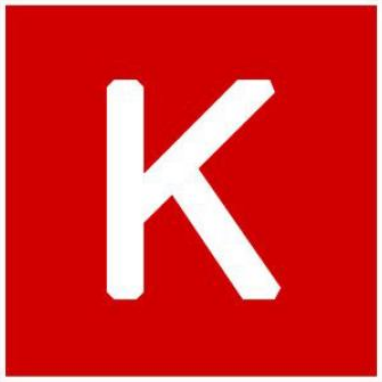
本阶段讨论的目标

> 一直从事底层软件/嵌入式的工作，
> 数学和英语功底都不太好，最近学了
> 一些深度学习的基础，有了理论基础，
> **想实际操刀练习，目前没有什么预算，**
> **该如何上手？**

在2016年上手学DL是非常容易的

Deep-Learning Package Zoo

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



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.

选取目前最常用的两个包进行介绍

Caffe

So what is Caffe?

- Pure C++ / CUDA architecture for deep learning
  - command line, Python, MATLAB interfaces
- Fast, well-tested code
- Tools, reference models, demos, and recipes
- Seamless switch between CPU and GPU
  - `Caffe::set_mode(Caffe::GPU);`

Prototype

Training

Deployment

All with essentially the same code!

from the tutorial by Evan Shelhamer , Evan Shelhamer, Jeff Donahue, Yangqing Jia, and Ross Girshick







Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev
Jonathan Long, Ross Girshick, Sergio Guadarrama

https://github.com/UCB-ICSI-Vision-Group/caffe-paper/raw/arxiv/caffe.pdf

from the tutorial by Evan Shelhamer , Evan Shelhamer, Jeff Donahue, Yangqing Jia, and Ross Girshick

Why Caffe? In one sip…

- **Expression**: models + optimizations are plaintext schemas, not code.
- **Speed**: for state-of-the-art models and massive data.
- **Modularity**: to extend to new tasks and architectures.
- **Openness**: common code and reference models for reproducibility.
- **Community**: joint discussion, development, and modeling.

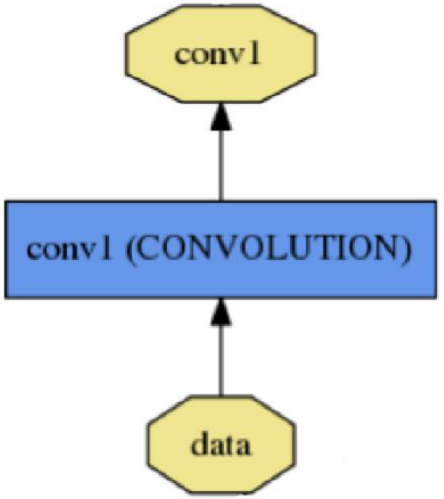
from the tutorial by Evan Shelhamer , Evan Shelhamer, Jeff Donahue, Yangqing Jia, and Ross Girshick

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



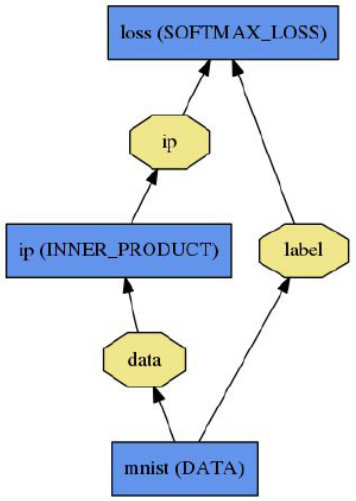
*Link to the Google Protobuffer Documentation

from the tutorial by Evan Shelhamer , Evan Shelhamer, Jeff Donahue, Yangqing Jia, and Ross Girshick

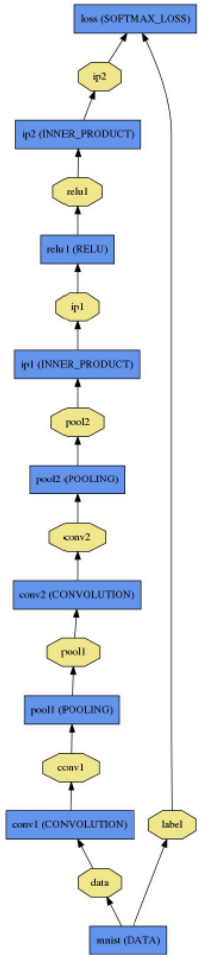
A Caffe Network

- 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" …}
```



LogReg ↑



LeNet →

from the tutorial by Evan Shelhamer , Evan Shelhamer, Jeff Donahue, Yangqing Jia, and Ross Girshick

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.

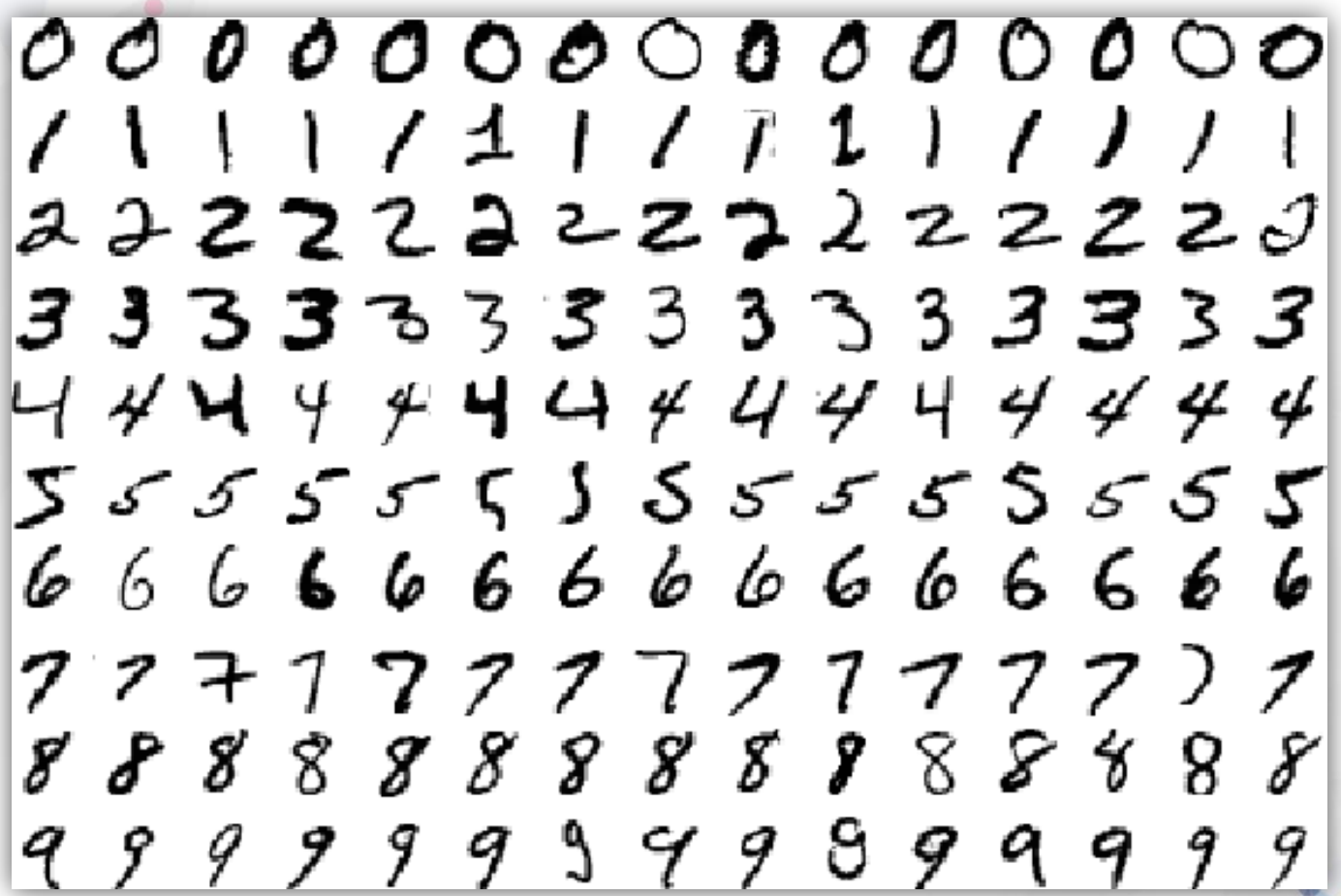
It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

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)
```


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

MNIST：深度学习领域的 Hello World



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

Misclassified examples

Credit: Barnabás Póczos & Aarti Singh, CMU-10701

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

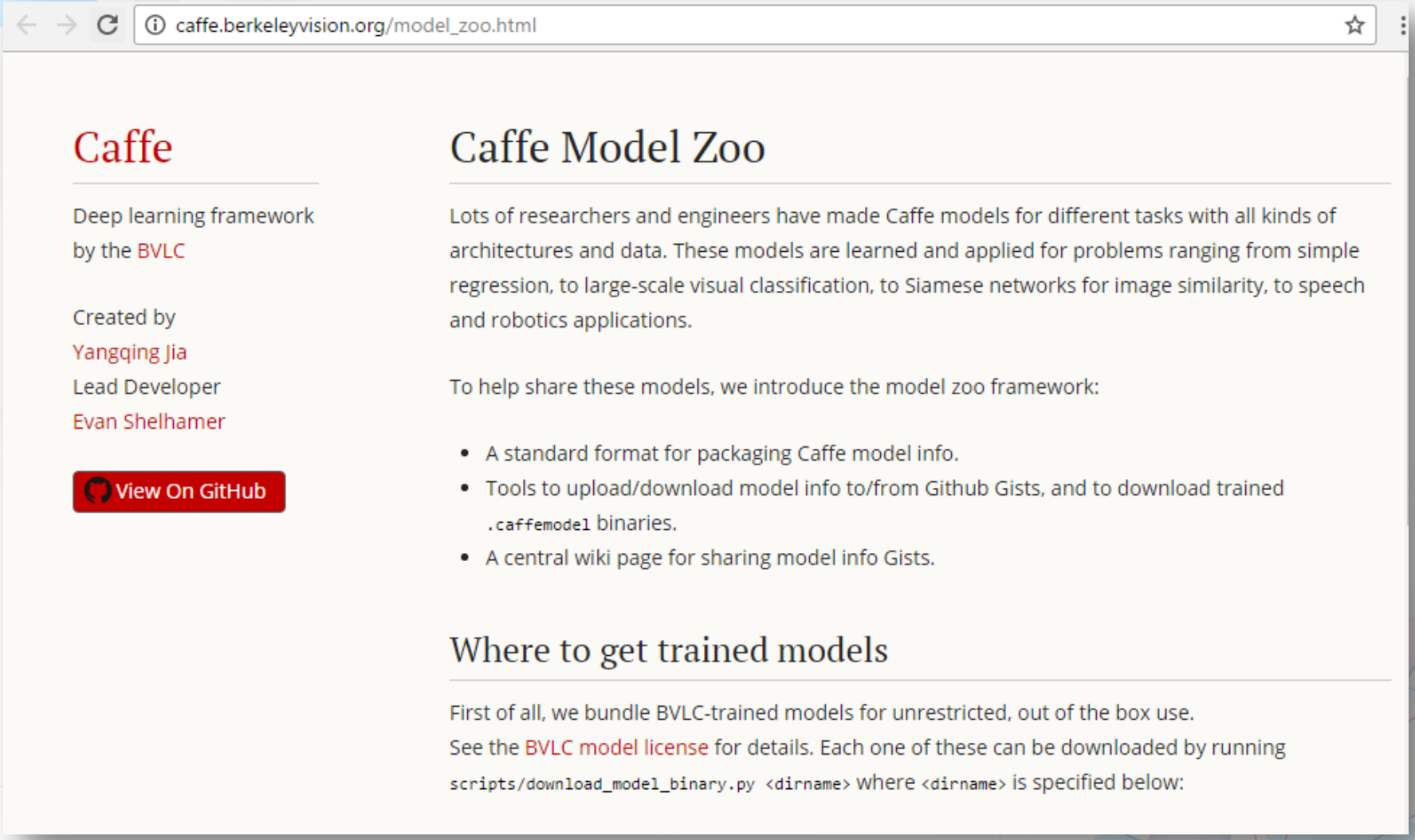
Before we actually run the training program, let's explain what will happen. We will use the LeNet network, which is known to work well on digit classification tasks. We will use a slightly different version from the original LeNet implementation, replacing the sigmoid activations with Rectified Linear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

http://caffe.berkeleyvision.org/gathered/examples/mnist.html

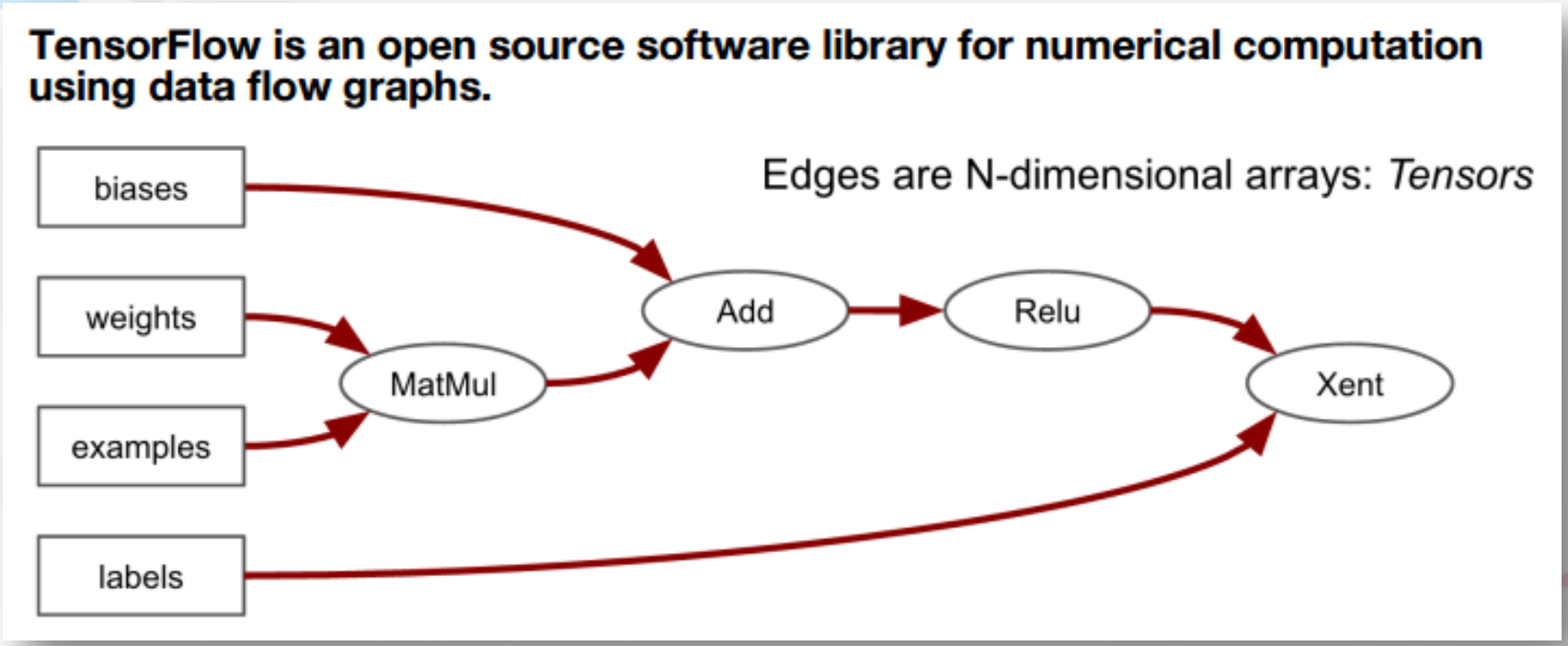
Demo：用Caffe构建LeNet，用MNIST训练

（切换到TK1命令行界面）

Caffe Model Zoo：拿来主义好去处



TensorFlow：搞清楚Tensor和Flow就行了

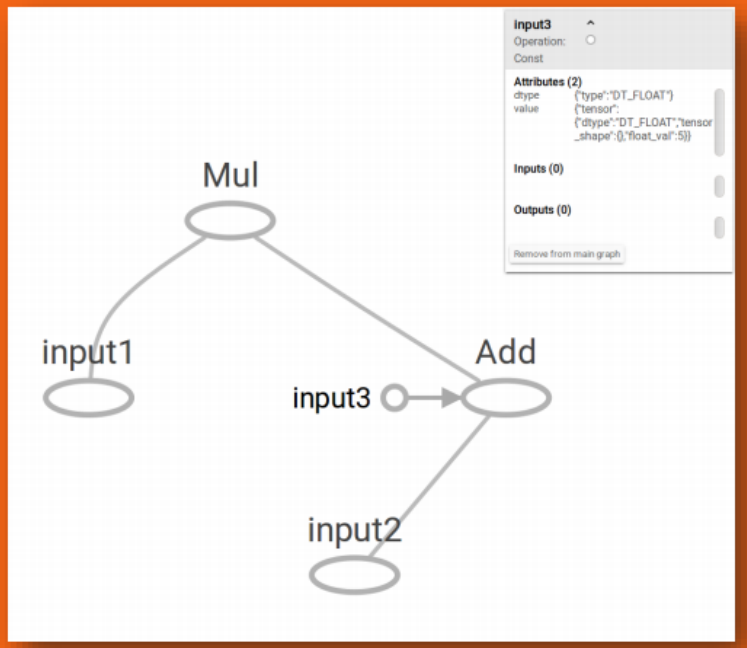


https://indico.cern.ch/event/433556/contributions/1930588/attachments/1231746/1806025/tensorflow_introduction.pdf

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)
```



http://flarecast.eu/wp-content/uploads/2015/04/TensoFlow.pdf

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.

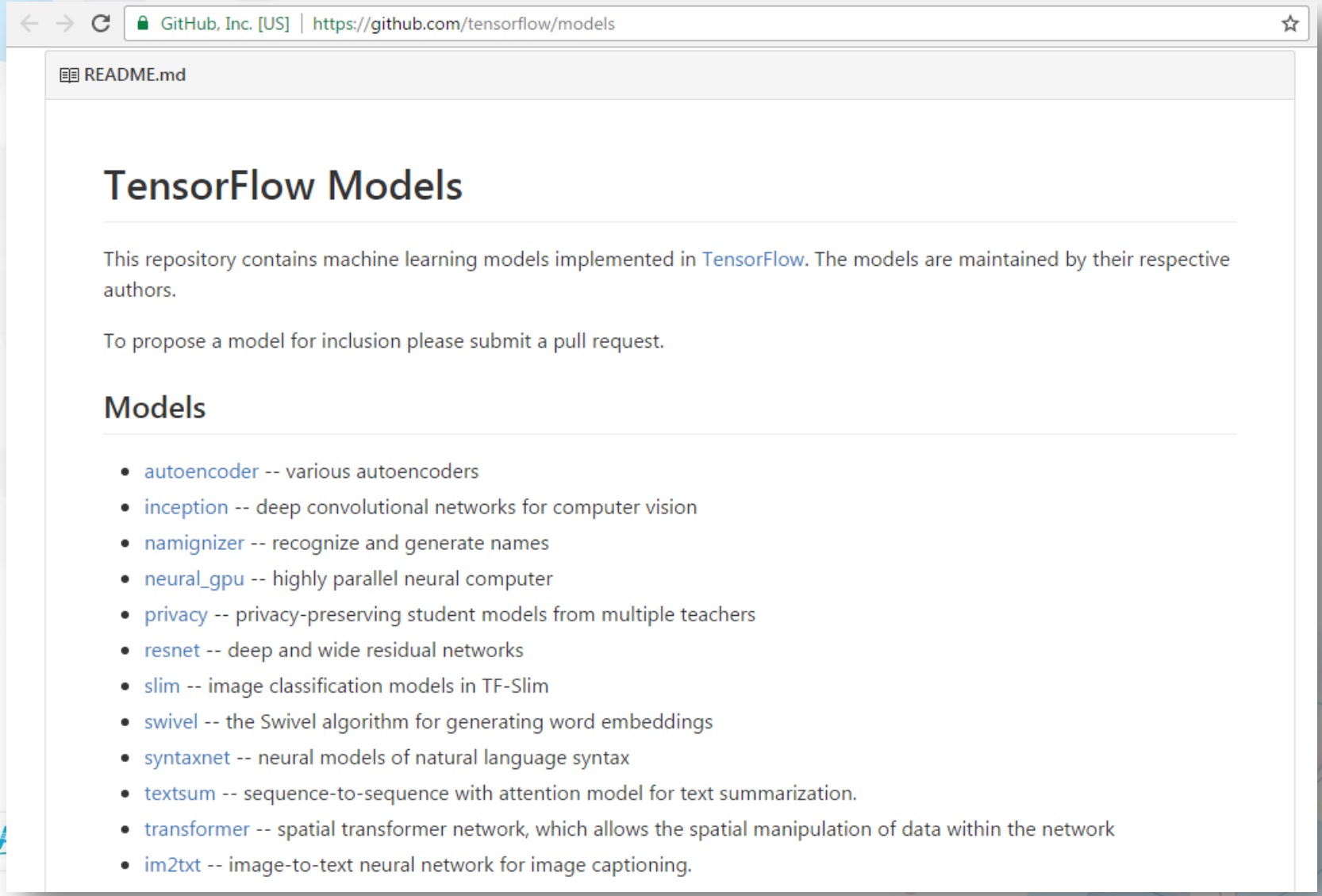
The actual code for this tutorial is very short, and all the interesting stuff happens in just three lines. However, it is very important to understand the ideas behind it: both how TensorFlow works and the core machine learning concepts. Because of this, we are going to very carefully work through the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

https://www.tensorflow.org/versions/r0.11/tutorials/mnist/beginners/index.html

Demo：用TensorFlow构建LeNet

（切换到TK1命令行界面）

TensorFlow Models：拿来主义好去处



为什么说GPU是深度学习的标配？

深度学习怎么玩，取决于你多有钱

GPU Acceleration

`-gpu N` flag tells `caffe` which gpu to use

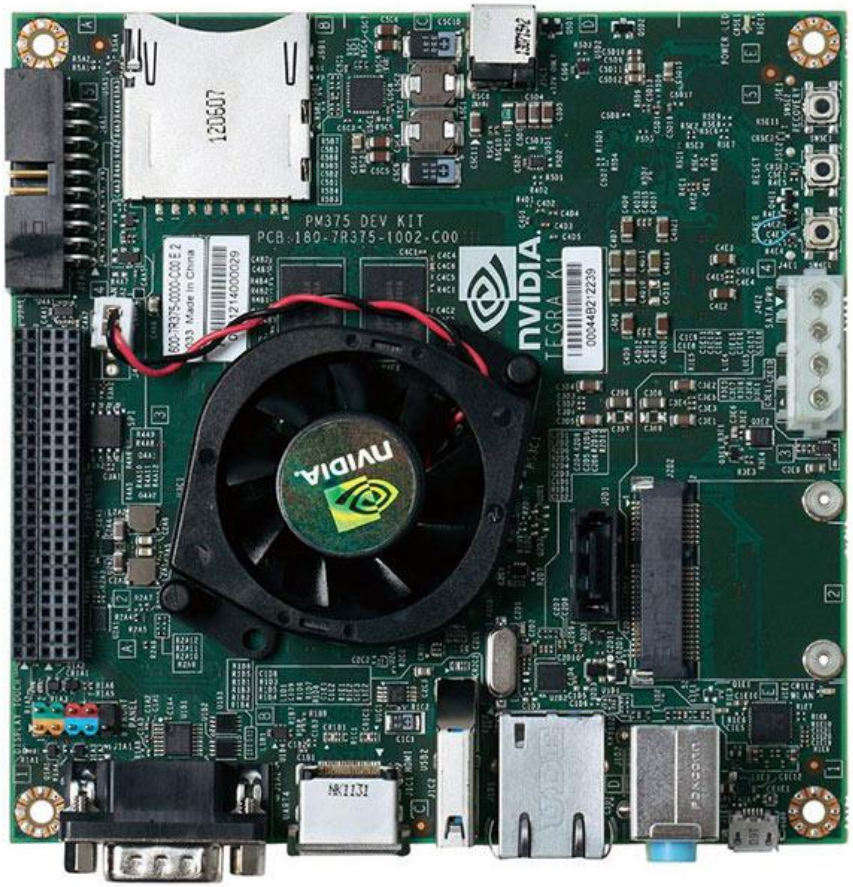
Alternatively, specify `solver_mode: GPU` in `solver.prototxt`

Tesla K40m, cuDNN v3-RC
ECC off, autoboost 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

NVIDIA TK1（Tegra K1 是芯片）



192 NVIDIA CUDA Cores

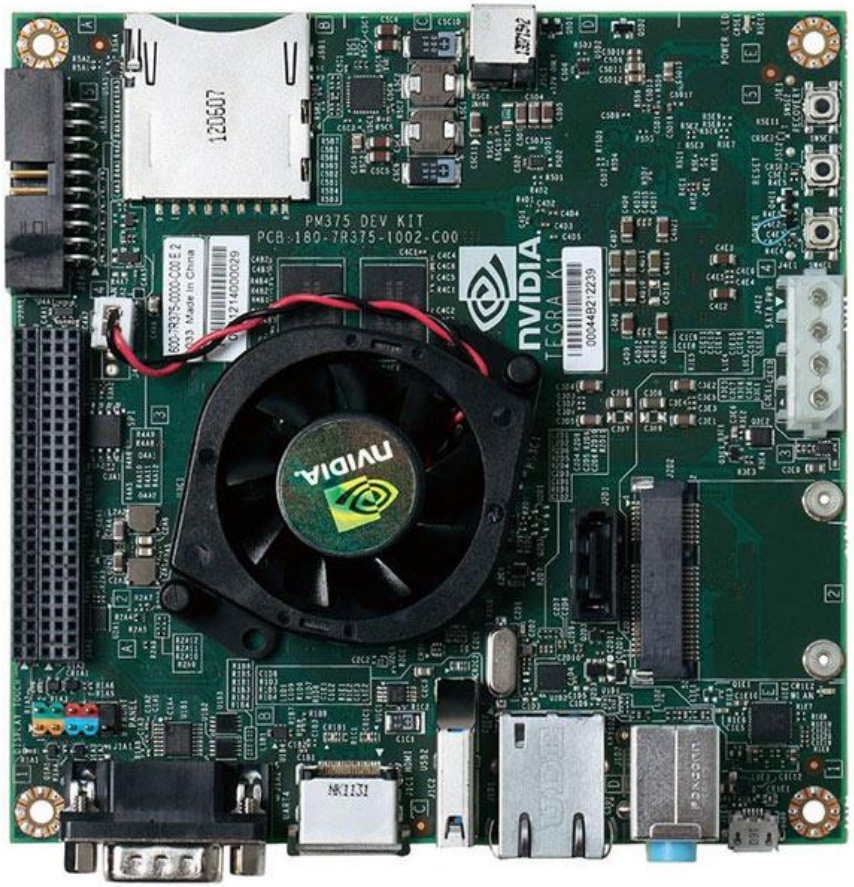
Quad-Core Cortex A15

2.3 Ghz

2GB Memory

16GB eMMC

NVIDIA TK1 (Tegra K1 是芯片)



192 NVIDIA CUDA Cores

Quad-Core Cortex A15

2.3 Ghz

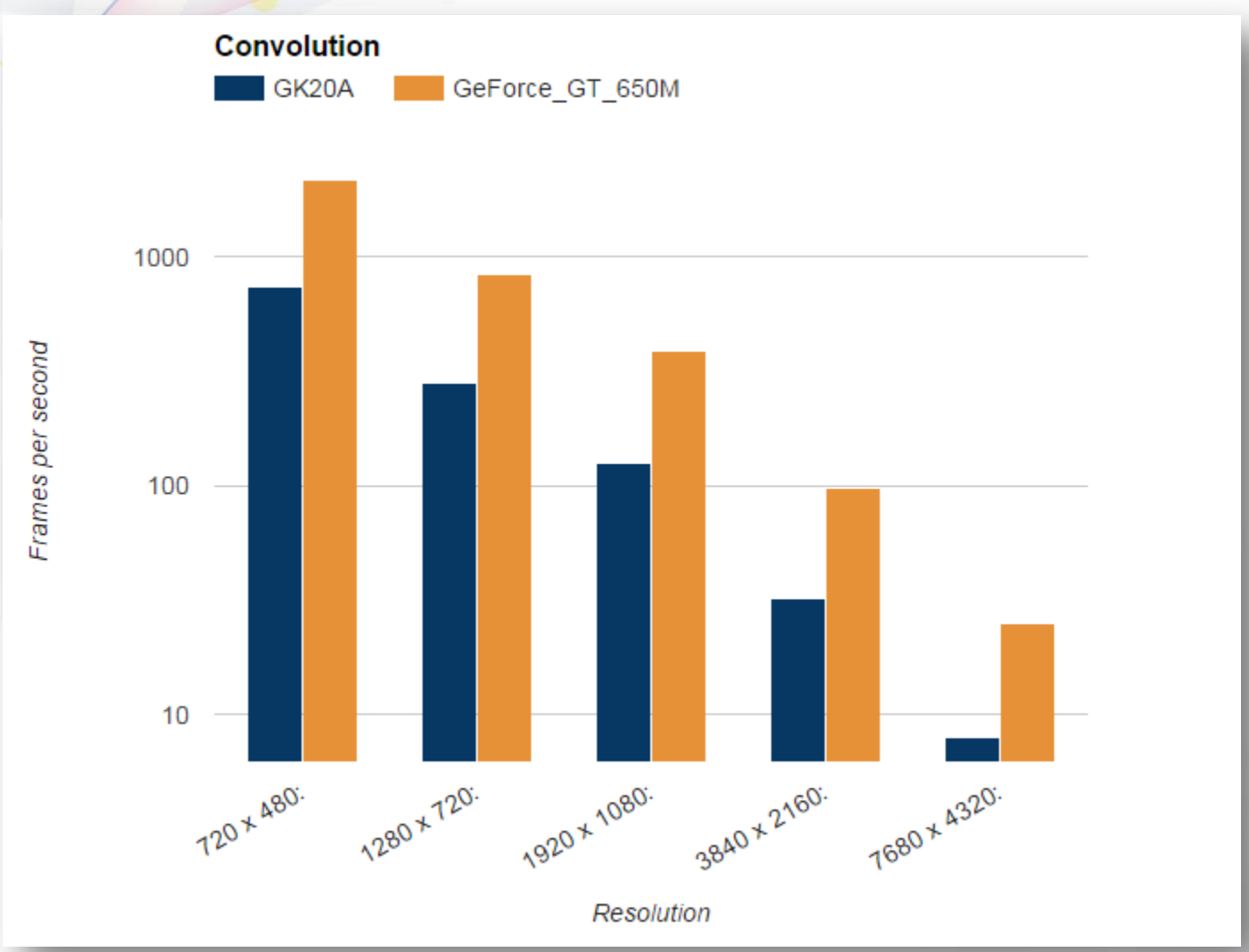
2GB Memory

16GB eMMC

注意是TK1是32位ARMv7, 科技树被锁定 →_→

Property Name / Device Name	Jetson TK1 GK20A	GT 650M
Compute	3.2	3.0
Number of multiprocessors	1	2
Cores	192	384
Base clock rate	852 MHz	950 MHz
Total global memory	1746 MB	2048 MB
Total shared memory per block	48 KB	48 KB
Total constant memory	64 KB	64KB
Memory clock rate	924 MHz	900 MHz
Memory bus width	64-bit	128-bit
Total registers per block	32768	65536
Warp size	32	32

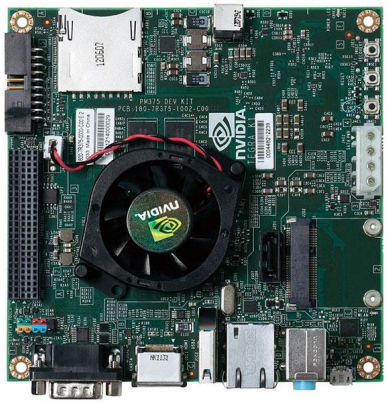
http://arrayfire.com/arrayfire-on-nvidia-tegra-tk1/



http://arrayfire.com/arrayfire-on-nvidia-tegra-tk1/

TK1 → TX1

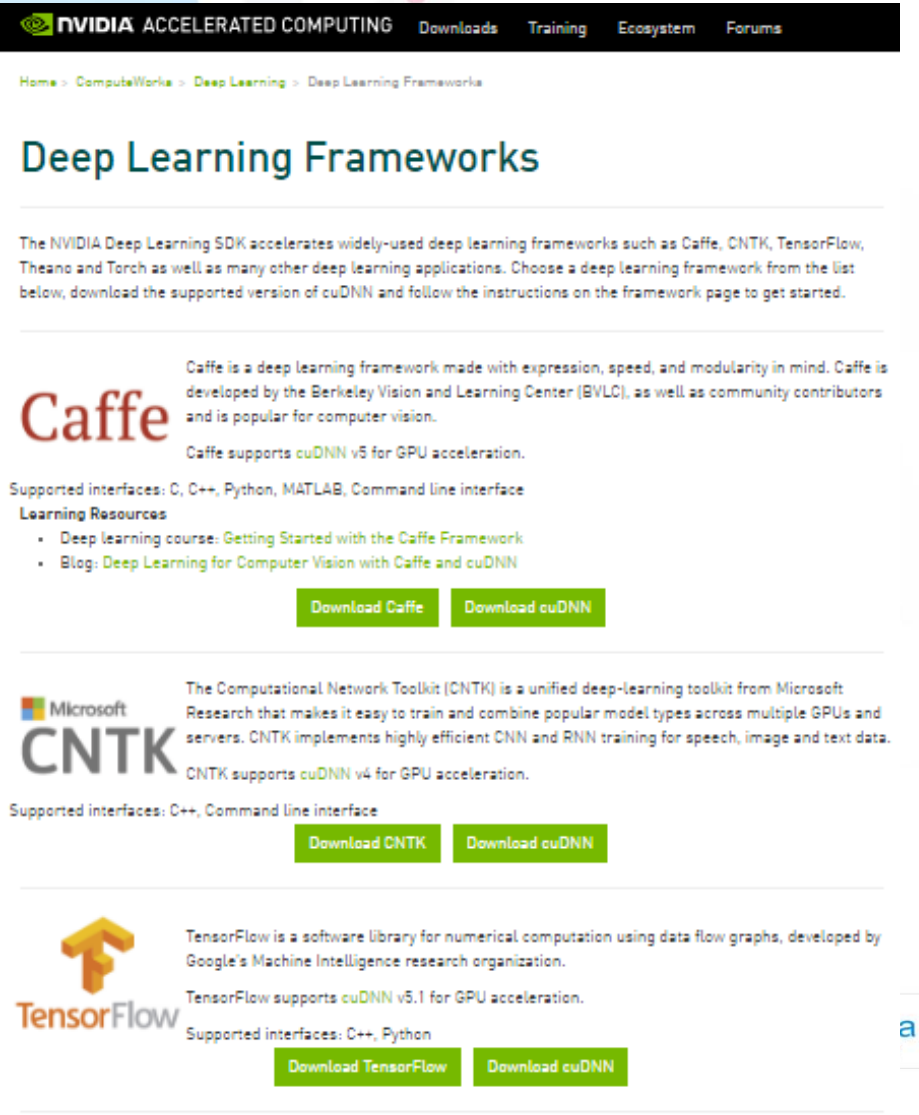
- 第一代 → 第二代
- ARM32 → ARM64
- 192 Kepler → 256 Maxwell
- 2GB DDR3 → 4GB DDR4
- $192 → $599



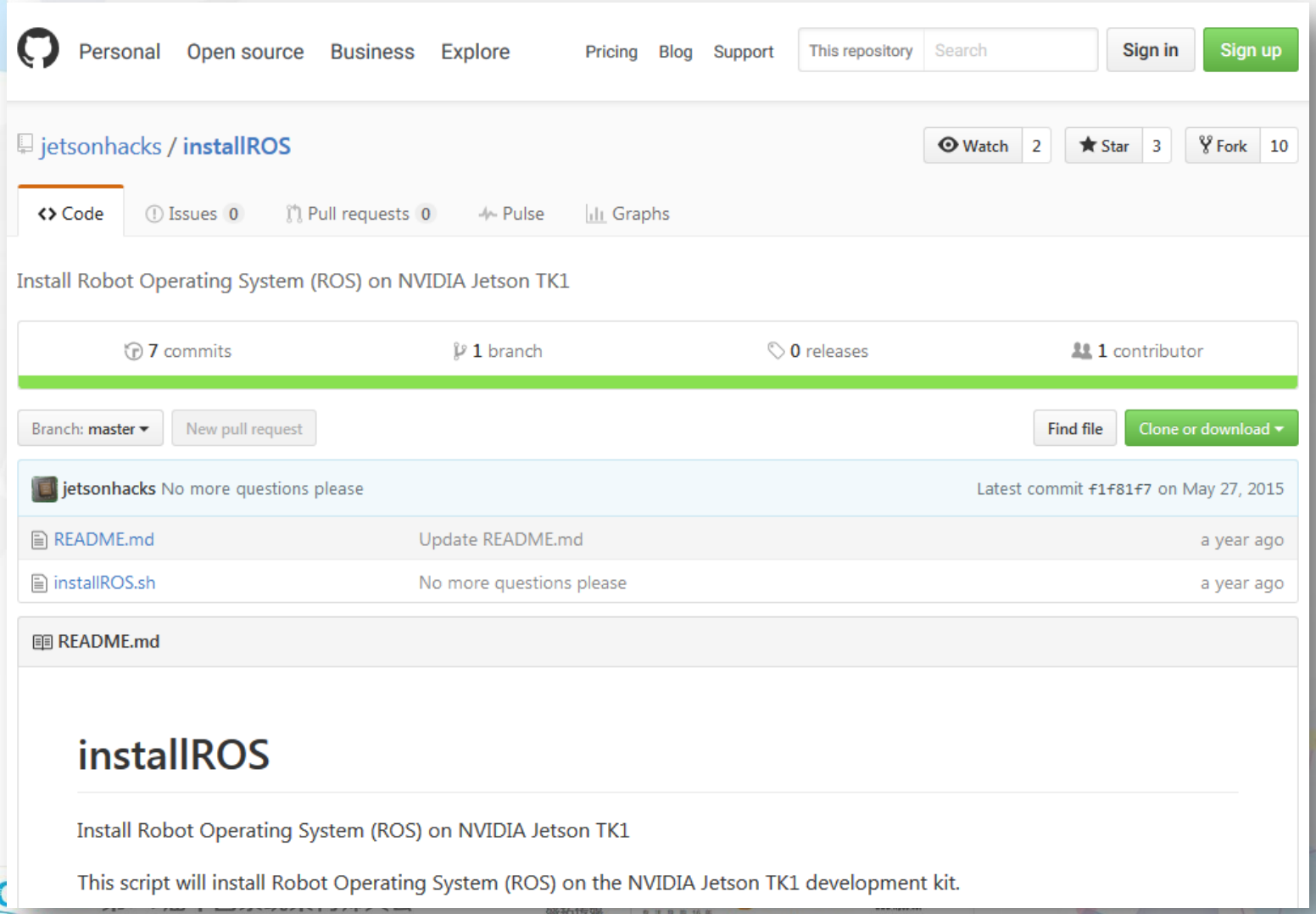
踩过的坑

- TK1和TX1的资料非常容易混淆，但是两者不兼容
- 网上搜脚本经常不兼容，NVIDIA对TK1投入有限(便宜)
- cuDNN v2 以后的版本无法在TK1上(TX1可以)
- Caffe等都不支持低版本cuDNNv2(最新是cuDNNv5.1)
- 一定量的backporting苦力活，需要回滚相关API
  - 因为想要使用最新的软件包版本

NVIDIA的支持清单，TK1多数都支持

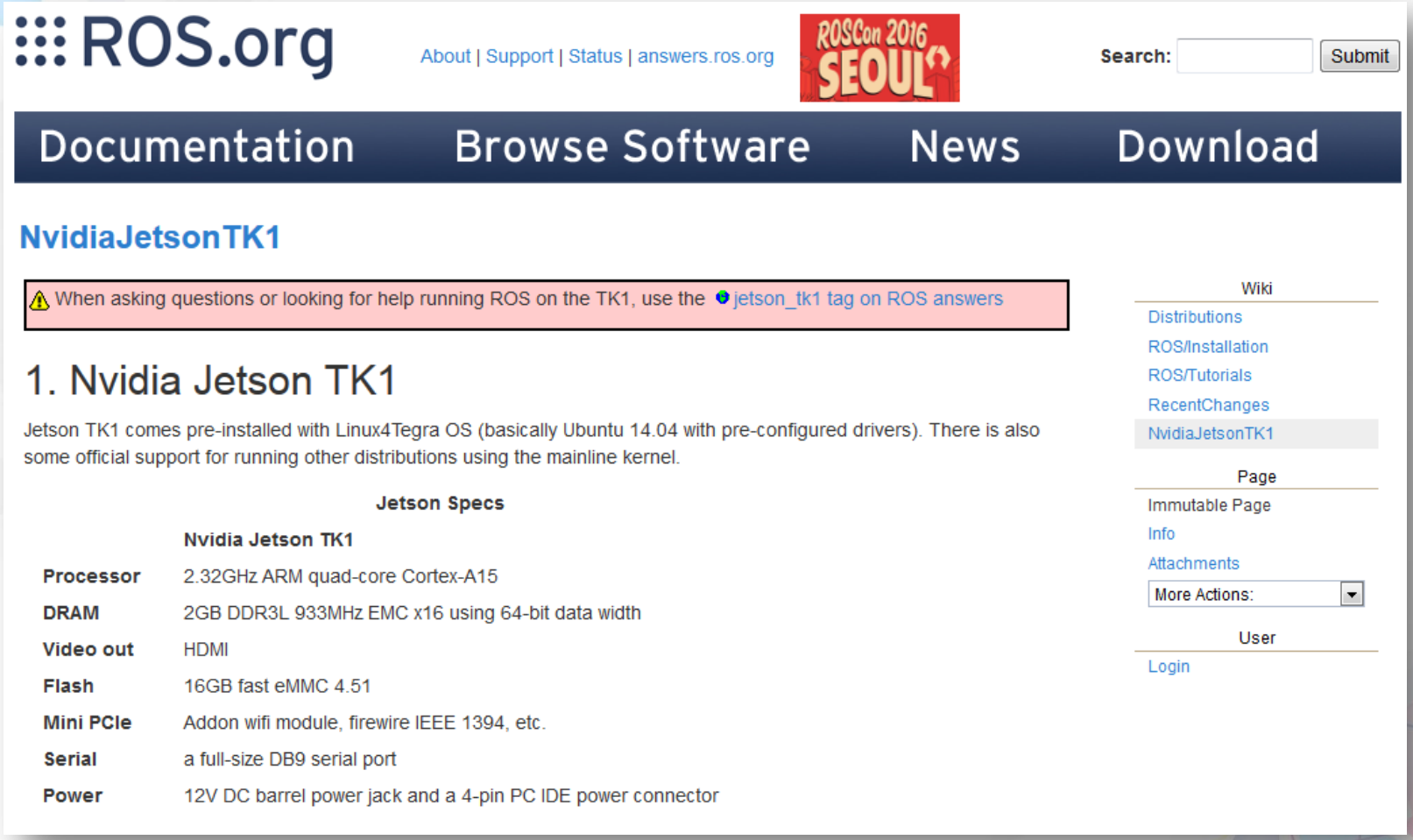


TK1可以完美使用ROS



https://github.com/jetsonhacks/installROS

TK1可以完美使用ROS（cont.）



ROS.org

About | Support | Status | answers.ros.org

ROSCon 2016 SEOUL

Search: Submit

Documentation | Browse Software | News | Download

NvidiaJetsonTK1

⚠ When asking questions or looking for help running ROS on the TK1, use the jetson_tk1 tag on ROS answers

1. Nvidia Jetson TK1

Jetson TK1 comes pre-installed with Linux4Tegra OS (basically Ubuntu 14.04 with pre-configured drivers). There is also some official support for running other distributions using the mainline kernel.

Jetson Specs

	Nvidia Jetson TK1
Processor	2.32GHz ARM quad-core Cortex-A15
DRAM	2GB DDR3L 933MHz EMC x16 using 64-bit data width
Video out	HDMI
Flash	16GB fast eMMC 4.51
Mini PCIe	Addon wifi module, firewire IEEE 1394, etc.
Serial	a full-size DB9 serial port
Power	12V DC barrel power jack and a 4-pin PC IDE power connector

Wiki
- Distributions
- ROS/Installation
- ROS/Tutorials
- RecentChanges
- NvidiaJetsonTK1

Page
- Immutable Page
- Info
- Attachments
- More Actions:

User
- Login

http://wiki.ros.org/NvidiaJetsonTK1

Demo：在TK1上演示ROS及DL内容

（切换到TK1命令行界面）

本阶段讨论的目标

一直从事底层软件/嵌入式的工作，最近学了一些深度学习的基础，有了理论基础，也实际动手感受了一下。

想要开始在自己的产品中使用起来，该如何入手？

Are Very Deep Neural Networks Feasible on Mobile Devices?

S. Rallapalli, H. Qiu, A. J. Bency, S. Karthikeyan, R. Govindan, B.S.Manjunath, R. Urgaonkar

In the recent years, the computing power of mobile devices has increased tremendously, a trend that is expected to continue in the future. With high-quality onboard cameras, these devices are capable of collecting large volumes of visual information. Motivated by the observation that processing this video on the mobile device can enable many new applications, we explore the feasibility of running very deep Convolutional Neural Networks (CNNs) for video processing tasks on an emerging class of mobile platforms with embedded GPUs. We find that the memory available in these mobile GPUs is significantly less than necessary to execute very deep CNNs. We then quantify the performance of several deep CNN-specific memory management techniques, some of which leverage the observation that these CNNs have a small number of layers that require most of the memory. We find that a particularly novel approach that offloads these bottleneck layers to the mobile device's CPU and pipelines frame processing is a promising approach that does not impact the accuracy of these tasks. We conclude by arguing that such techniques will likely be necessary for the foreseeable future despite technological improvements.

1. INTRODUCTION

Smart mobile devices, e.g., smartphones, tablets, wearables, are being adopted by users at an unprecedented rate. Moreover, the capabilities of cameras on these devices has also been increasing, to the point where these devices can generate high volumes of visual data in the form of video and images. Video and image data is semantically rich and being able to process this data in near real-time on the mobile device (i.e., without incurring the latency of off-loading processing to the cloud) could be immensely useful for several applications: in a military scenario to catch suspicious activities, or in a civilian scenario like localized advertising, or personal assistants.

In the last few years, computer vision techniques have been mov-

object but also draws bounding box around it. This is a useful primitive in tracking objects in a video (i.e., detecting objects across successive frames): once an object's bounding box has been determined, lightweight trackers can be used to achieve this.

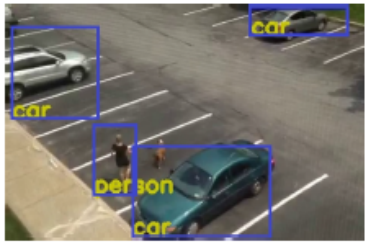


Figure 1: Output of YOLO object detection: classification + localization (Accurate, except the very small dog is not detected).

We find that a primary bottleneck in running very deep CNNs is *memory* (Section 3). The mobile GPU we use, the TK1 [29], has 2 GB of memory, but YOLO fails to run on this device for lack of memory. In part, this is because the mobile GPU is physically addressed and lacks memory management hardware. We also find that one of the fully-connected (FC) layers (these last layers perform the detection and localization) requires almost 80% of the memory required by the CNN.

We then explore a range of memory optimizations (Section 3), some of which leverage CNN structure and properties of YOLO, and quantify their performance (Section 4). These optimizations include: pruning unused variables in the neural network, using "managed" memory in the GPUs, splitting the FC layer into sub-parts each of which is loaded onto memory and executed sequentially, trading-off accuracy by reducing the size of the neural network, and, in a novel twist, offloading the FC layer to the CPU while

Caffe Framework on the Jetson TK1: Using Deep Learning for Real Time Object Detection

Christopher Alicea-Nieves (University of Puerto Rico at Mayaguez, Computer Engineering, *SUNFEST Fellow*)

Dr. Camillo J. Taylor, Computer and Information Science, University of Pennsylvania

Abstract— **Deep Neural Networks are an approach to using state-of-the-art machine learning algorithms in order to improve the applications of the computer vision field. These neural networks are complex mathematical models which can be trained to identify certain objects within a frame, and some of them are also able to identify where on the image is the object that they identified. This technical report focuses on presenting the implementation of a deep neural network on a low-power embedded computer system, specifically the NVIDIA Jetson TK1, in order to run convolutions in real-time for object detection. This can be achieved by implementing Fast R-CNN–a state-of-the-art model for convolutions–in the machine learning framework called Caffe.**

Index Terms—caffe; convolutions; Jetson; machine learning; neural networks.

II. Background

2.1 Deep Neural Networks

A Convolutional Neural Network is based on different layers that look for features in an image. Regions with Convolutional Neural Networks (R-CNN) [6] are a variation of the regular CNNs. They first analyze the frame and propose different regions for different objects, and then proceed to run the CNN on each separate region. This model is more accurate in the PASCAL VOC [3], which is another dataset challenge but for object detection, hence the reason why we will be using it. The region proposal process is very expensive; it is a computational bottleneck in running R-CNNs. Fast-Regions with Convolutional Networks (Fast R-CNNs) [7] are designed to avoid this bottleneck and speed up the process of running

Low-Cost Visually Intelligent Robots with EoT

David Moloney[1], Dexmont Pena[1], Aubrey Dunne[1], Alireza Dehghani[1], Gary Baugh[1], Sam Caulfield[1], Kevin Lee[1], Xiaofan Xu[1], Maximilian Müller[1] , Remi Gastaud[1], Ovidiu Vesa[2]
[1] Movidius Ltd., 1st Floor, O'Connell Bridge House, D'Olier St, Dublin D02 RR99, Ireland
[2] Movidius Inc., 1730 S El Camino Real, San Mateo, CA 94402, United States
Email: {firstname.surname@movidius.com}

Oscar Deniz Suarez, José Luis Espinosa Aranda, Noelia Vallez Enano
VISILAB Grupo de Visión y Sistemas Inteligentes, Av. Camilo José Cela, s/n 13071 Ciudad Real, Spain
Email: oscar.deniz@uclm.es

***Abstract*—this paper describes the development of a low-cost, low-power visual-intelligence and robotics platform based on the H2020 EoT (Eyes of Things platform) and the Movidius Myriad2 VPU (Vision Processing Unit), associated machine vision, communications and motor-control libraries and the Movidius Fathom deep-learning framework. The platform allows a variety of low-cost robots to be built using kits and programmed in micropython. The ultimate goal of the EoT robotics project, which is still in development, is to enable the mass-production of advanced and highly programmable robots for developers, students and hobbyists at a sub $100 price-point that includes robot, control electronics, software and an eBook.**

Keywords-component; Vector Processor, Vision Processing Unit, VPU, ISA, Computer Vision, Computational Imaging

Figure 1 Typical Embedded Robotics Compute Platforms

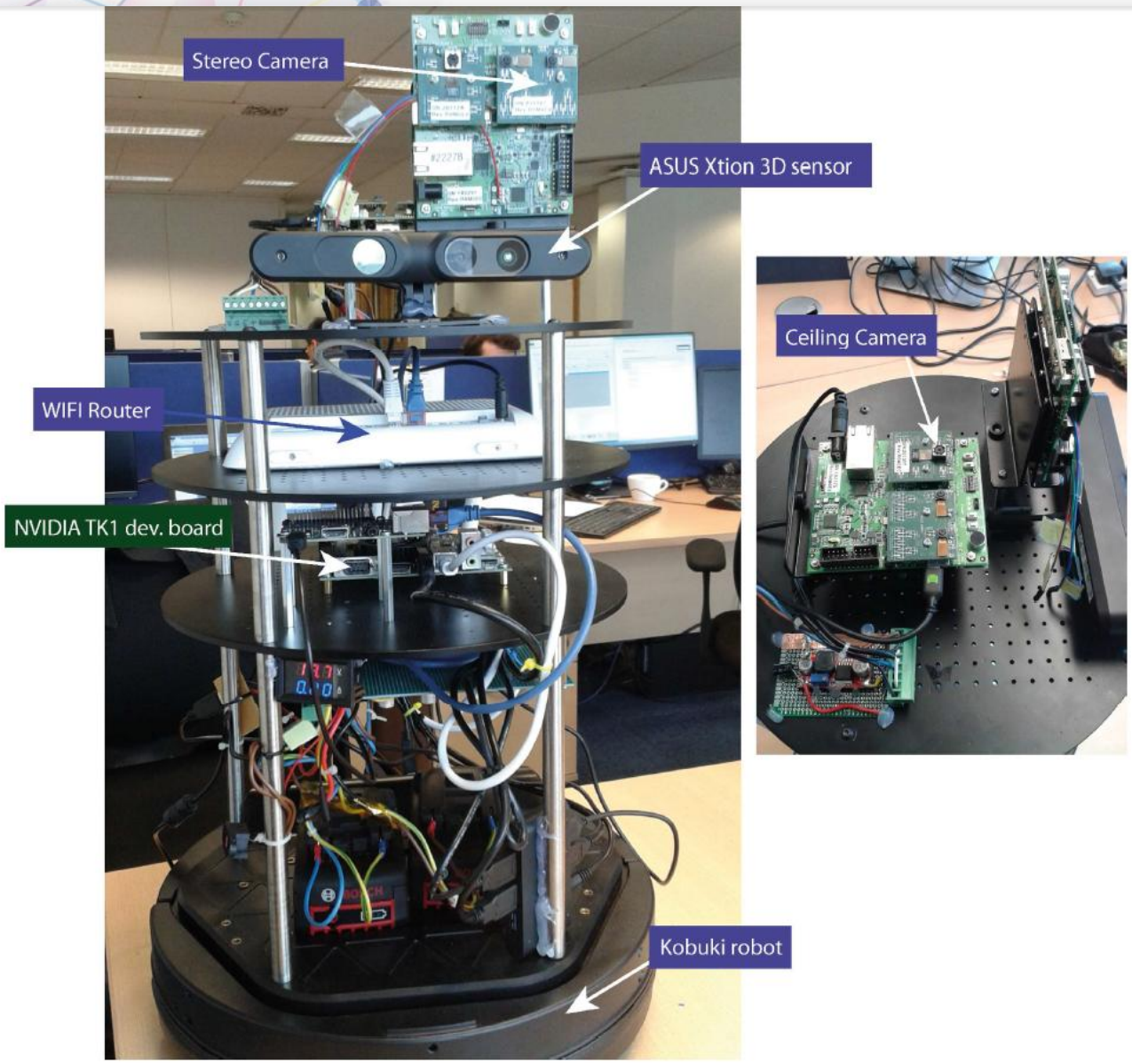
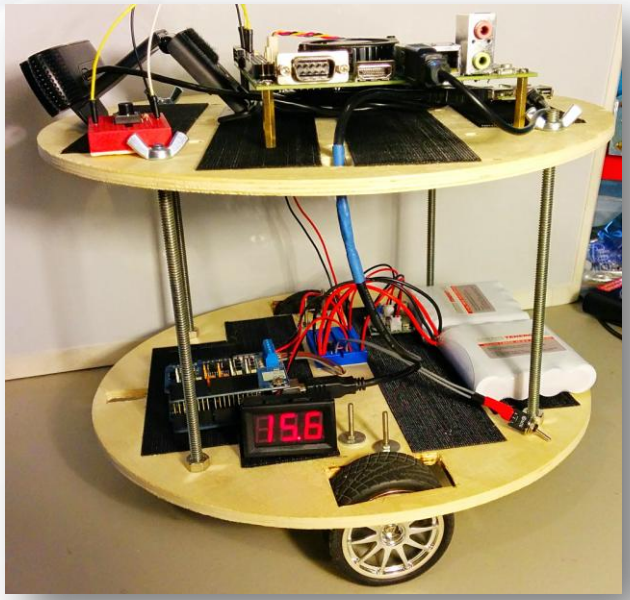
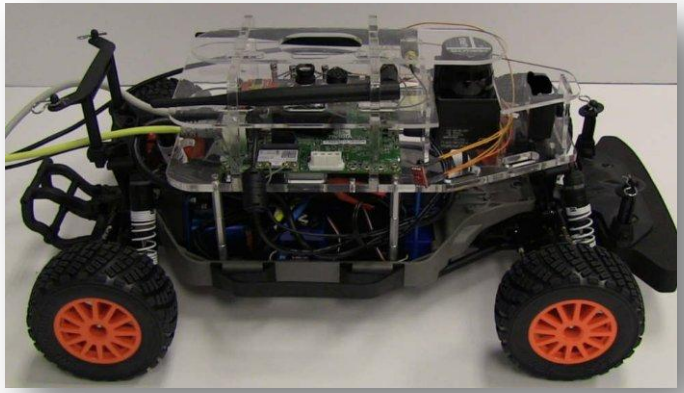
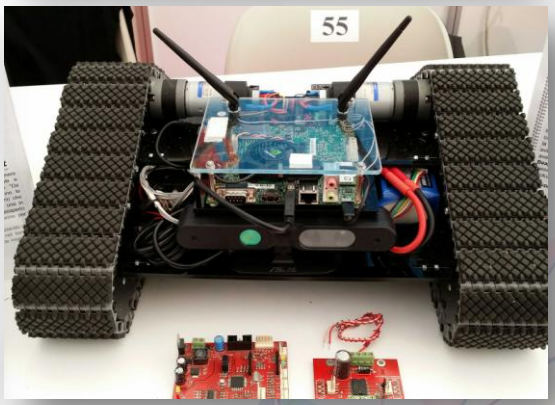


Figure 9 K-Bot robot with Stereo and Volumetric capture

Source : Low-Cost Visually Intelligent Robots with EoT







http://elinux.org/images/thumb/b/bb/NVIDIA_WalkingRobot.jpg/180px-NVIDIA_WalkingRobot.jpg


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


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
NVIDIA
NVIDIA JETSON TX1 系统模组
专为智能设备设计的全功能模组
英伟达携手商汤科技共建人工智能城市
开启万物感知新时代

THANKS

SequeMedia 盛拓传媒 | IT168.com 专注导购16年 | ChinaUnix.net | ITPUB www.itpub.net