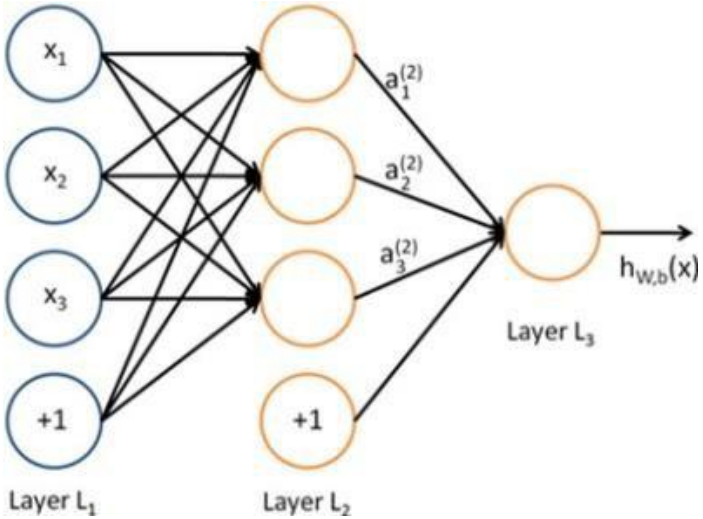
Python 深度学习

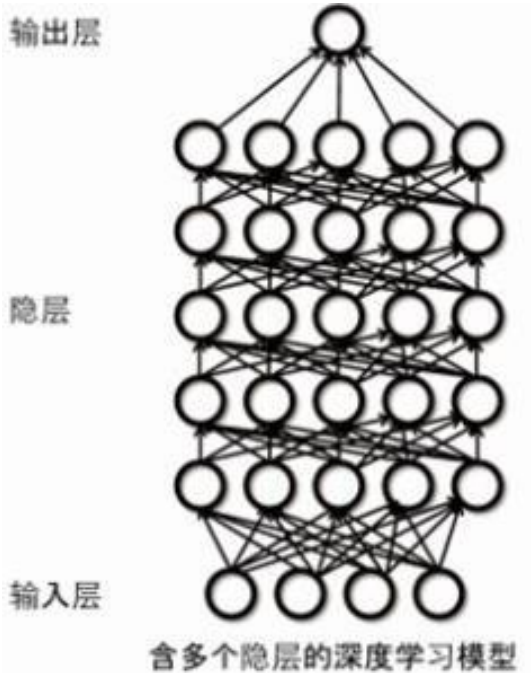
目录

深度学习 vs. 神经网络

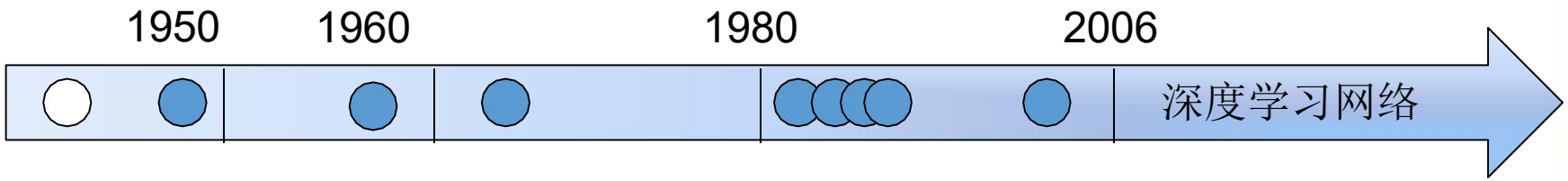
神经网络 （浅层）：



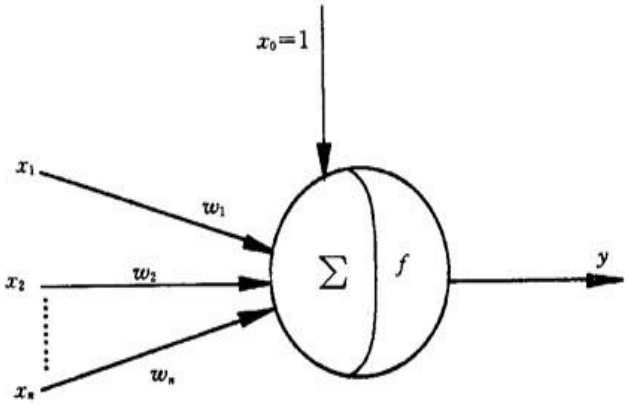
深度学习：

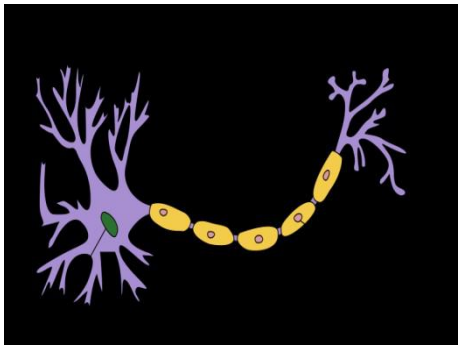


神经网络的发展



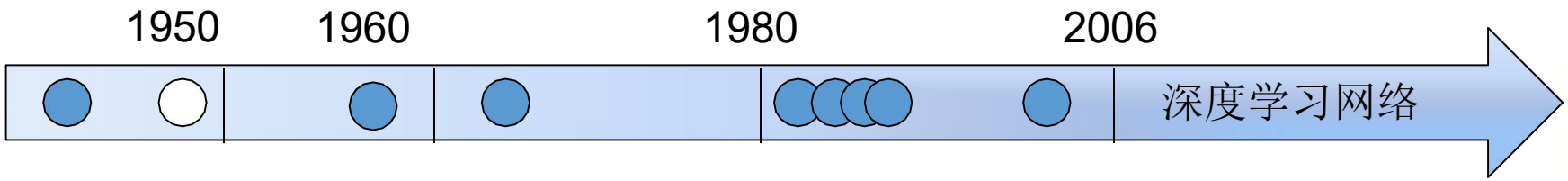
1943：生理学家McCulloch和数学家Pitts提出神经元MP模型，
该模型将神经元当作一个功能逻辑器件来对待，从而开创了神经网络模型的研究





弱点：不具有学习能力

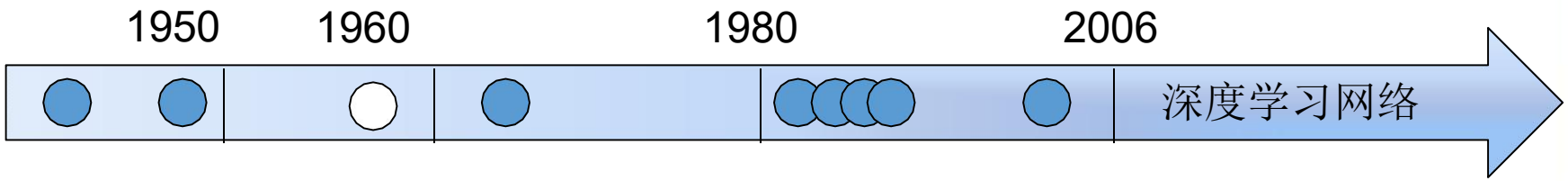
神经网络的发展



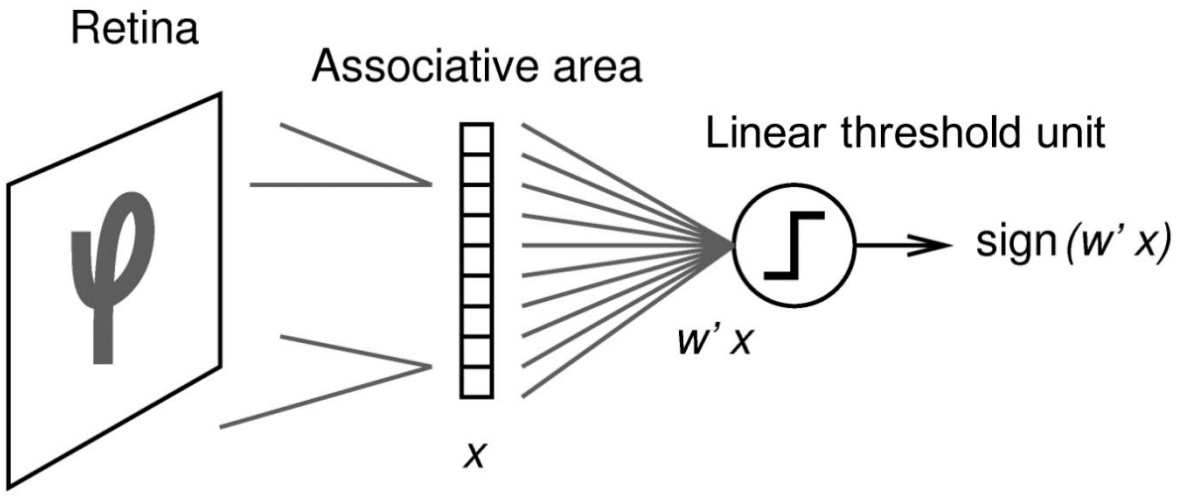
1949：Hebb提出的权值修改Hebb学习规则：
一种通过神经元的激活和抑制来调整权重的非监督学习机制

Donald Olding Hebb

神经网络的发展

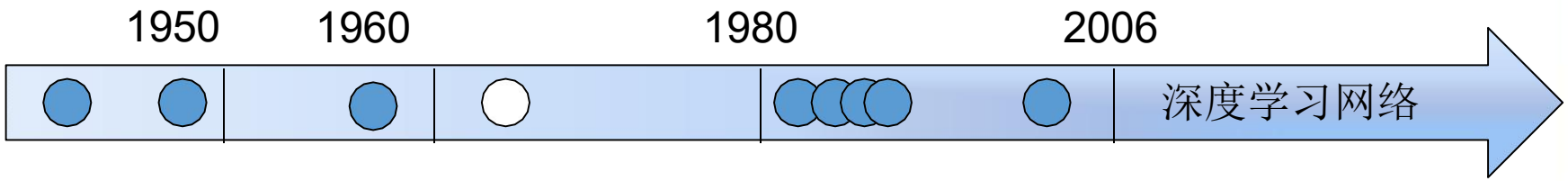


1958：Rossenblatt提出感知机模型



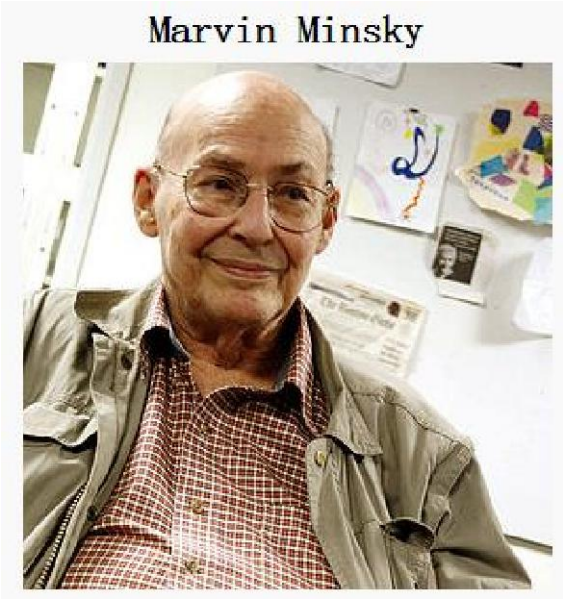
采用MP模型的结构，但第一次引入学习的概念，即采用感知算法监督学习权重矩阵，引起了广泛的关注

神经网络的发展

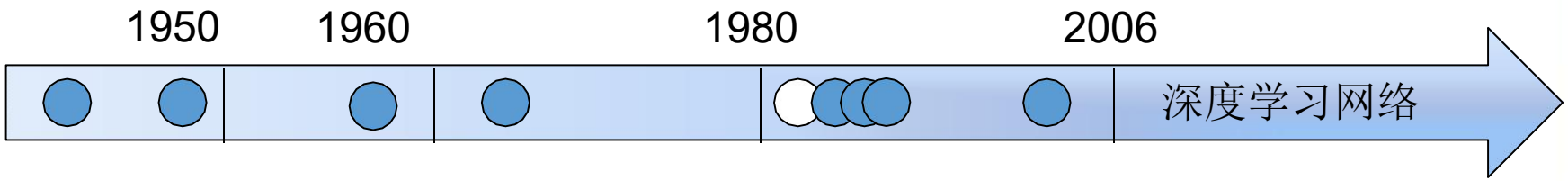


1969：Minsky和Papert的著作：《Perceptions》

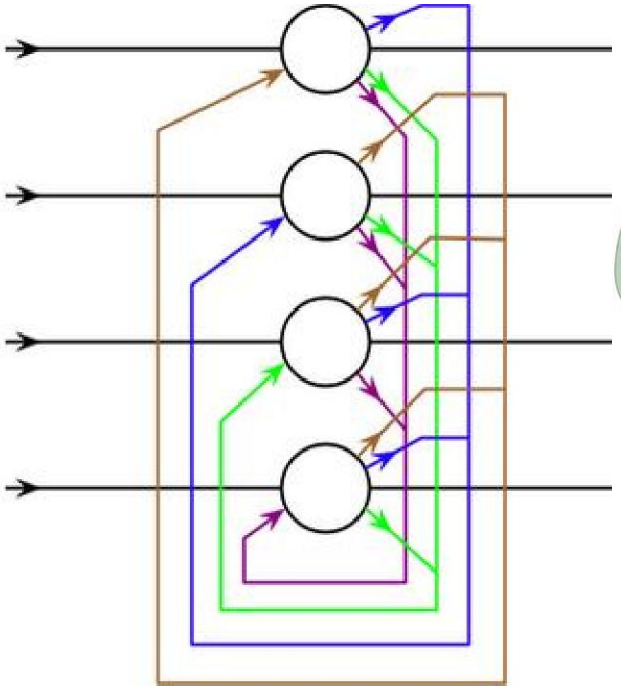
数学上证明了单层神经网络功能有限，分类能力、学习能力不足，甚至不能解决“亦或”这样的简单逻辑运算

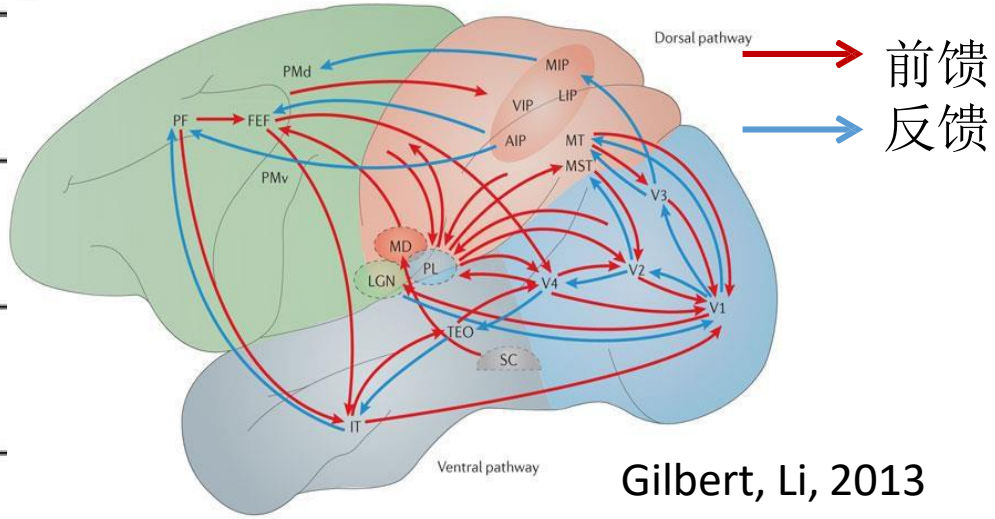


神经网络的发展



1982：Hopfiled提出反馈神经网络HNN



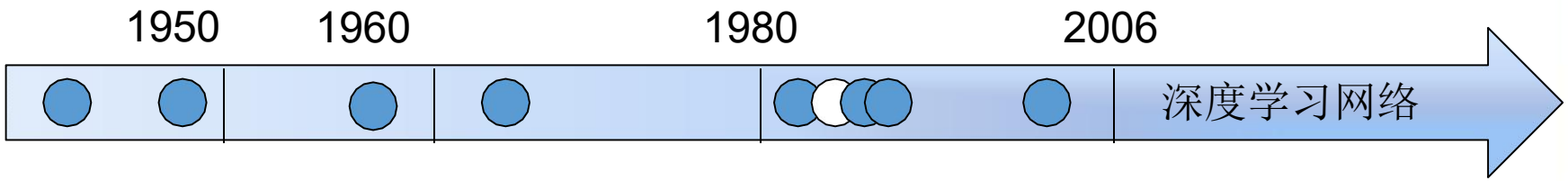


Gilbert, Li, 2013

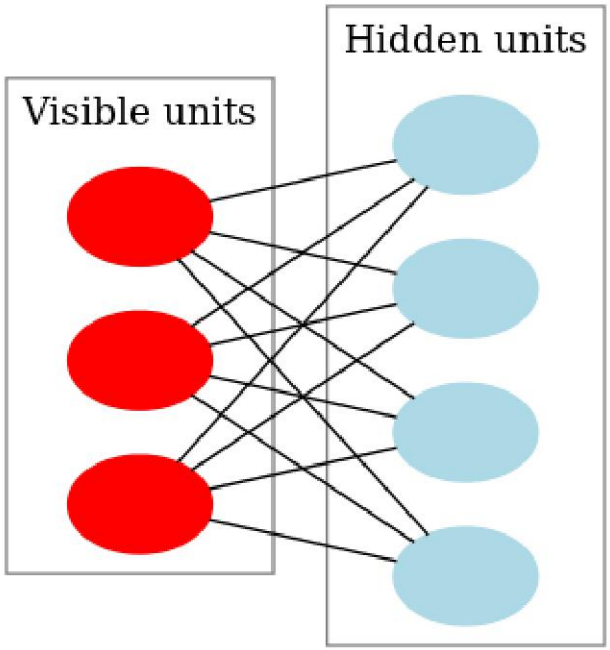
激起了神经网络研究第二次浪潮

Hopfield netwok 要求静态输入，它能保证系统收敛，其连接权重可采用Hebb学习准则学习。

神经网络的发展

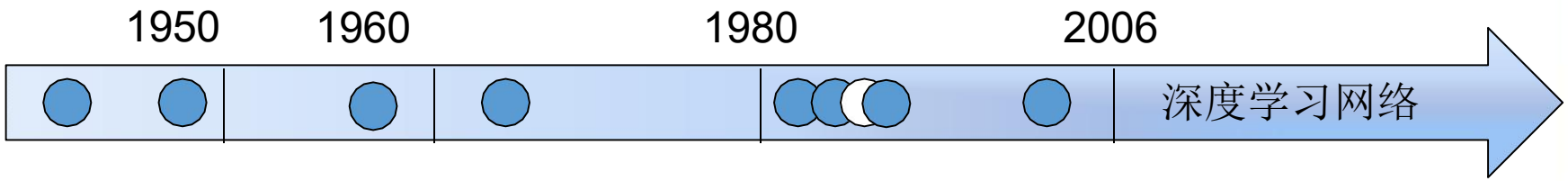


1985：Ackley, Hinton和Sejnowskj提出Boltzmann机

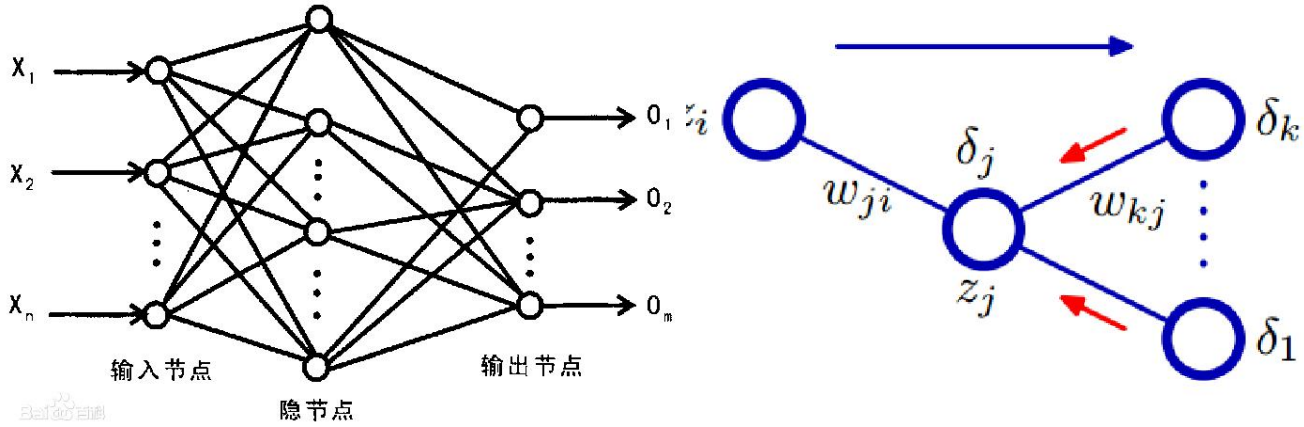


受限的Boltzmann机为2006年Hinton 在Science 论文提出的深层信念网络的基本模块

神经网络的发展

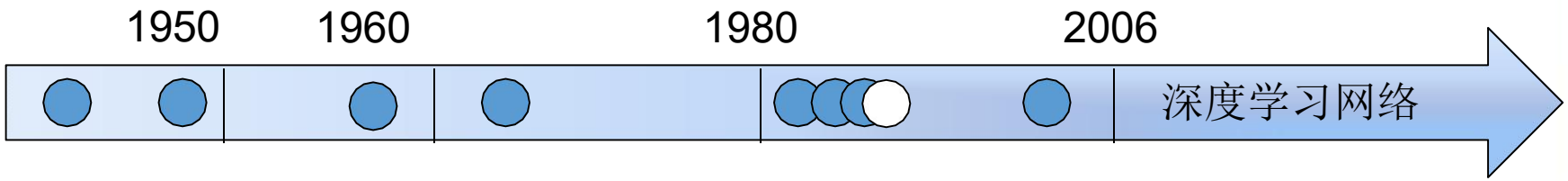


1986：Rumelhart, Hinton 和McClelland提出误差反向传播（Error Back Proragation，简称BP）算法

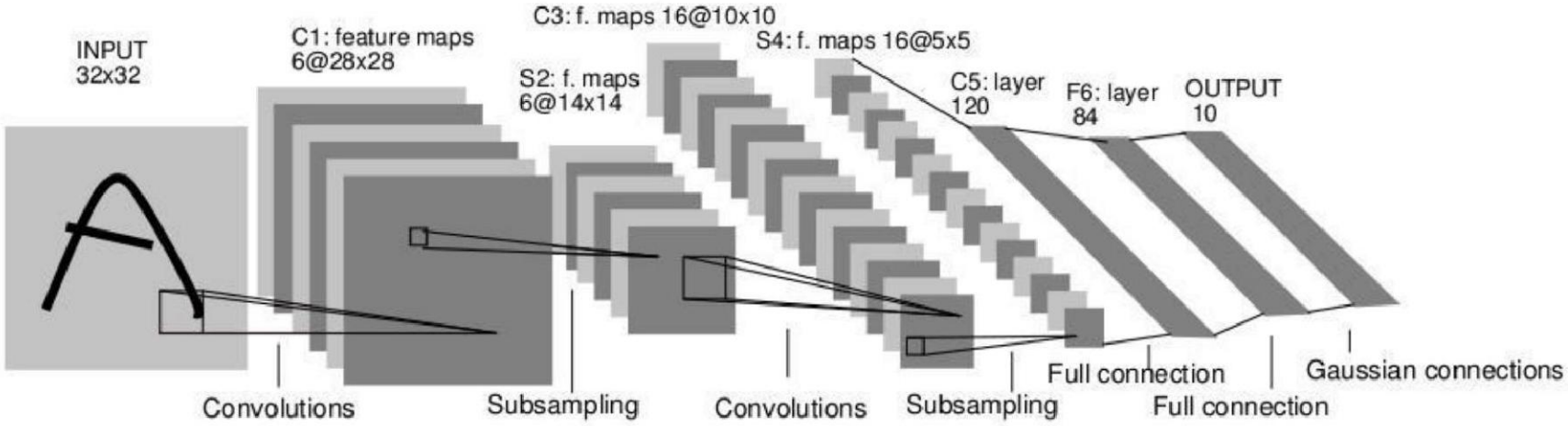


网络的训练过程是通过信号正向传播与误差反向传播来调整的各层权值

神经网络的发展

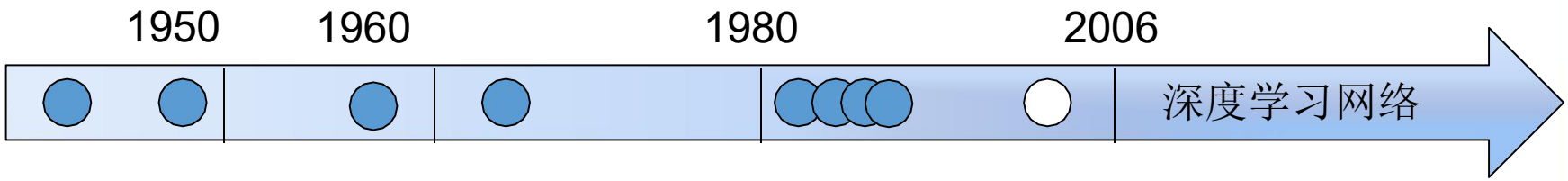


1989：LeCun和Bengio提出的卷积网络

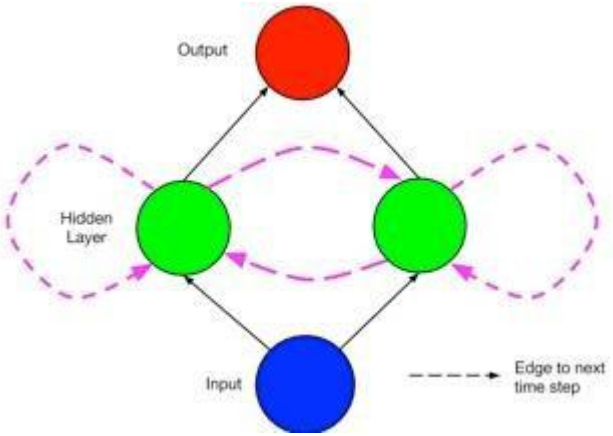


当前的深层卷积网络依然沿用这一结构

神经网络的发展



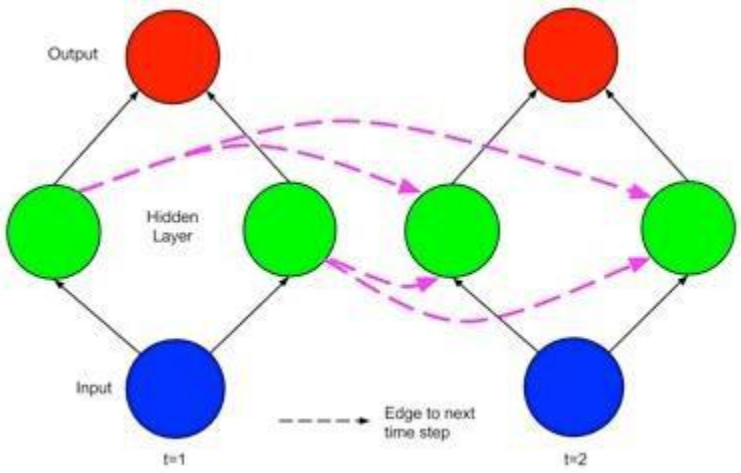
递归神经网络(Recurrent Neural Networks)



特点：隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出

用来处理序列数据，如自然语言处理等

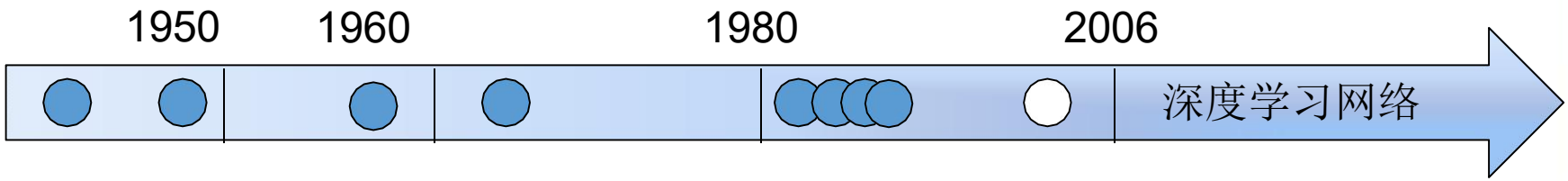
展开



展开：该网络可以通过通过时间步来展开，变成无环的形式

展开后，便可以使用反向传播算法训练。这种跨时间步的反向传播算法被称为Backpropagation Through Time

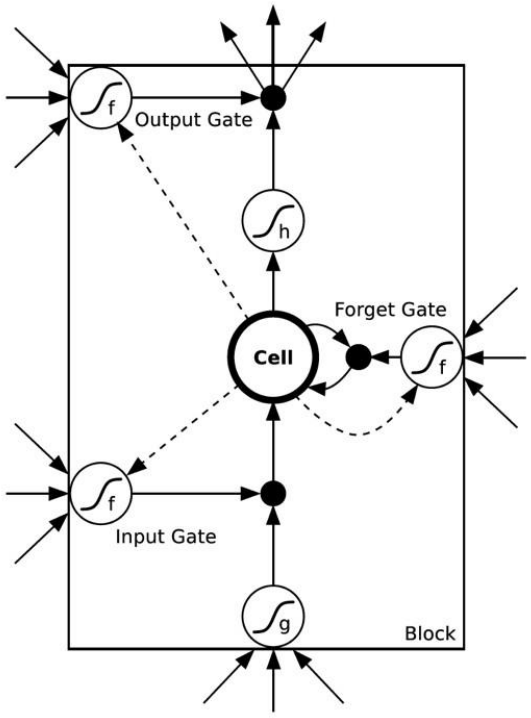
神经网络的发展



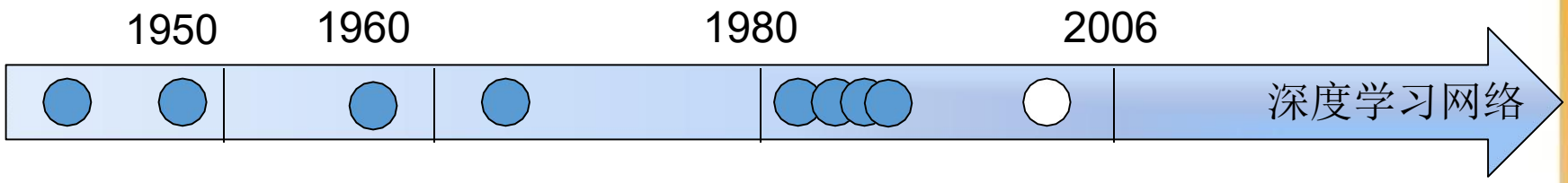
1997: Hochreiter和Schmidhuber提出了**LSTMs(Long Short-Term Memory，长短时记忆模型) 模型**，成为目前使用最广泛最成功的RNNs模型

相对于一般的RNNs，该模型能够对**长序列依赖**进行更好的表达，较好地解决了**“消失的梯度”问题**(即BP算法对深层结构训练的致命弱点问题)

由门电路控制的线性通路，其导数为常数，因此，回传时有效避免了梯度消失



深度学习模型的里程碑：DBNs

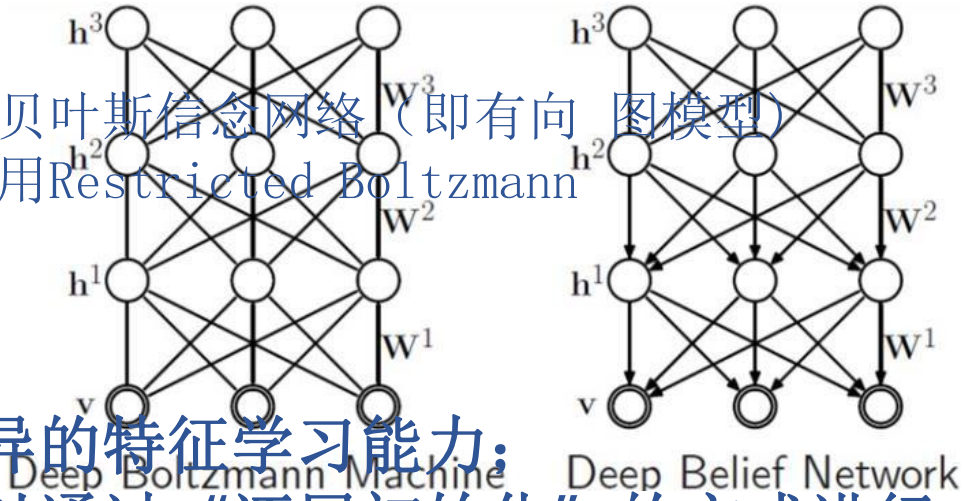


2006: 多伦多大学的***Hinton*** 等在***Science*** 发表了论文：

Reducing the dimensionality of data with neural networks, Science, 2016.

DeepBelief Networks:

在靠近可视层的部分使用 贝叶斯信念网络（即有向 图模型）
在最远离可视层的部分使 用Restricted Boltzmann Machine的模型



(1)深层模型具有更优异的特征学习能力；
(2)深度网络的训练可以通过“逐层初始化”的方式进行

深度学习应用的分水岭

- ImageNet 2012挑战赛

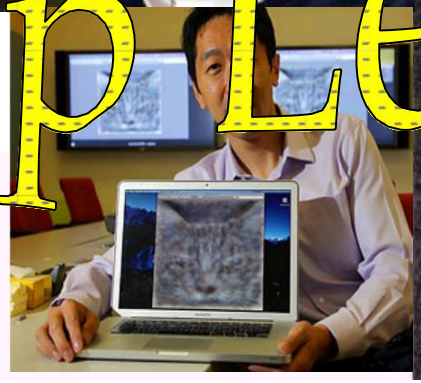
Rank	Name	Error rate	Description
1	**U. Toronto**	0.15315	Deep learning
2	U. Tokyo	0.26172	Hand-crafted features and learning models. Bottleneck.
3	U. Oxford	0.26979	
4	Xerox/INRIA	0.27058	

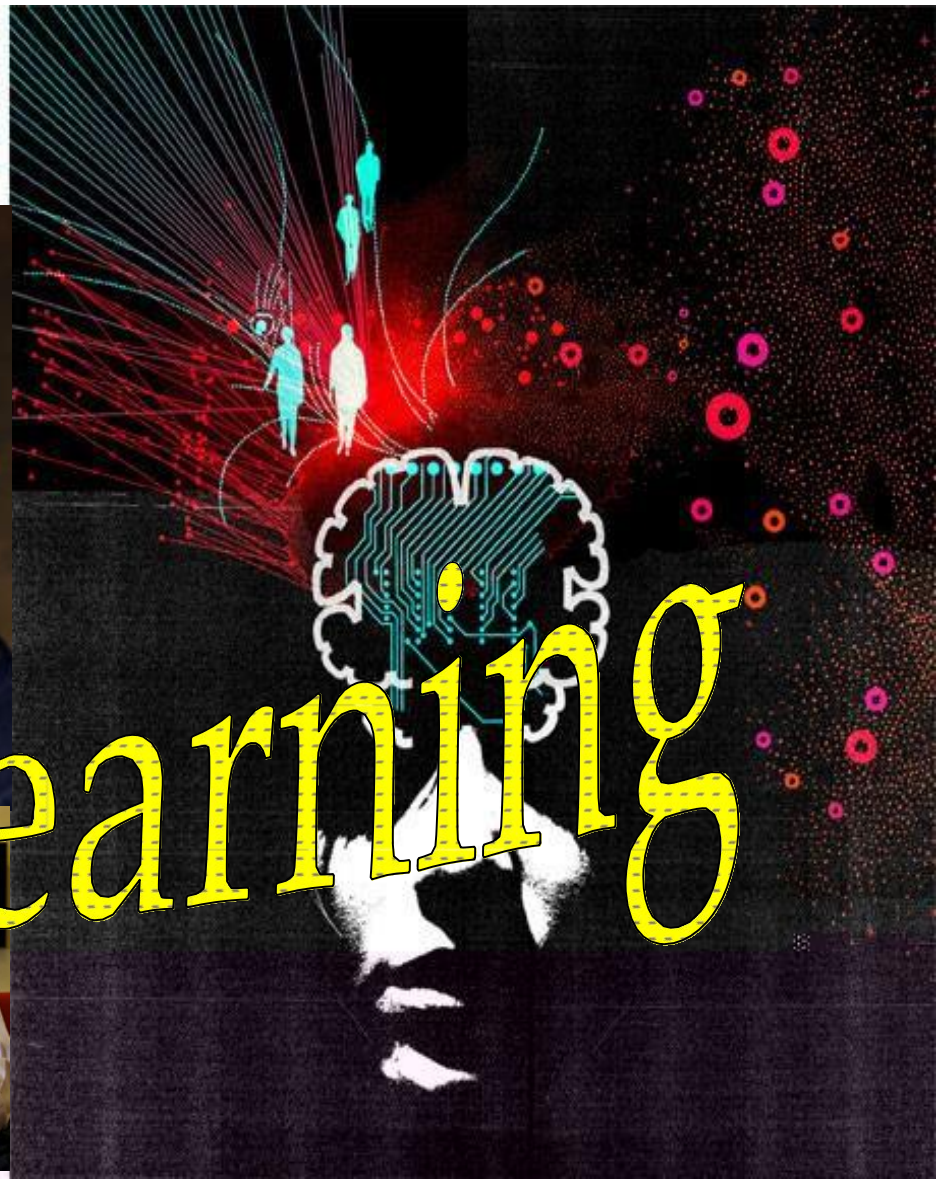
Object recognition over 1,000,000 images and 1,000 categories (2 GPU)

Alex Krizhevsky, L. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NIPS, 2012.

深度学习时代的到

深度学习被评为2013
MIT 技术评论之首





Deep Learning

Google

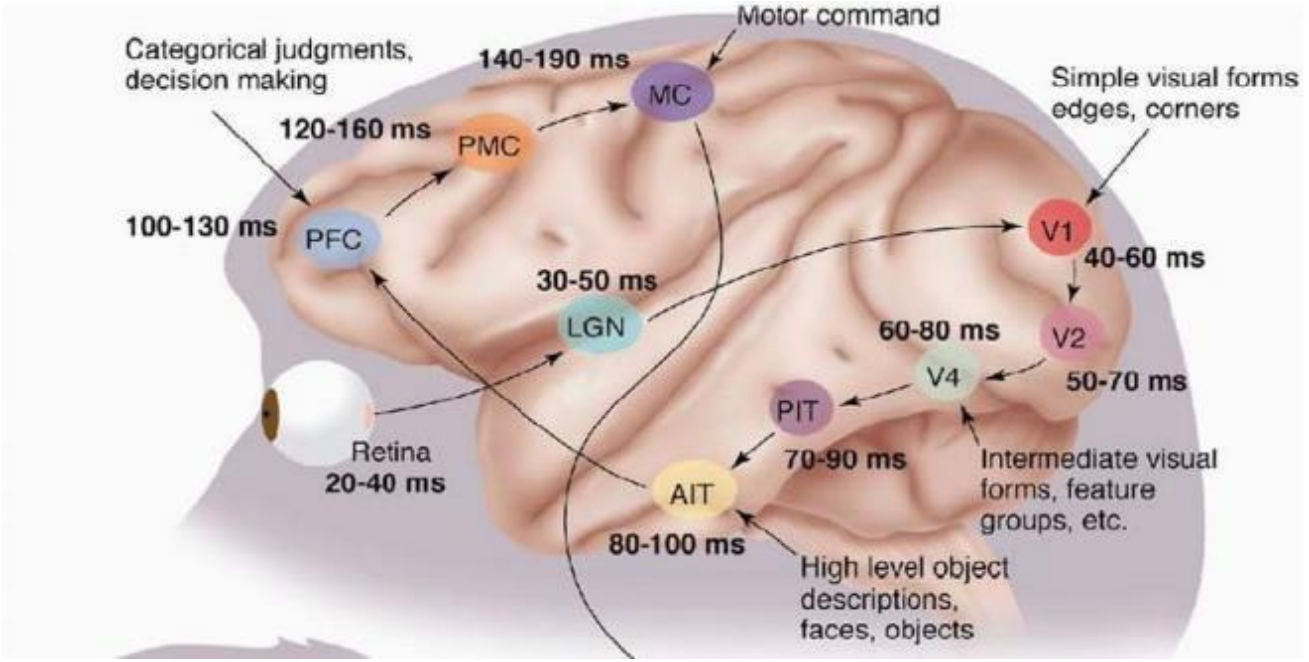
DEEPMIND

目录

为什么需要深层网络？（认知科学）

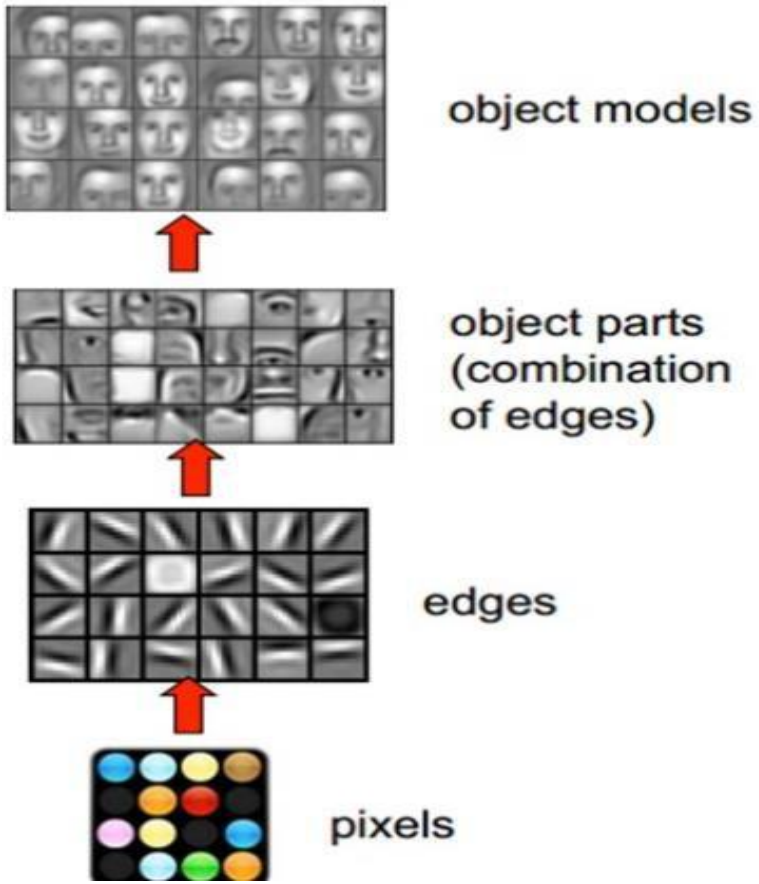
- 视觉系统是一个深度架构 1981年的诺贝尔医学奖，颁发给了D. Hubel和 T. Wiesel。两位 的主要贡献是: 发现了人的视觉系统的信息处理是分级的。

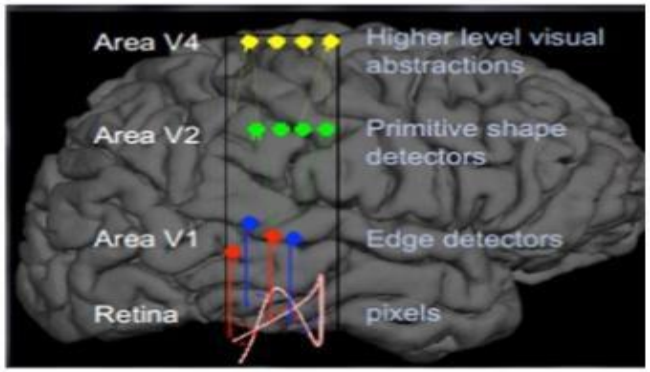
  如图所示，从视网膜（Retina）出发，经过低级的V1区提取边缘特征，到V2区的基本形状或目标的局部，再到高层的整个目标（如判定为一张人脸），以及到更高层的PFC（前额叶皮层）进行分类判断等。



为什么需要深层网络？（模型可视化）

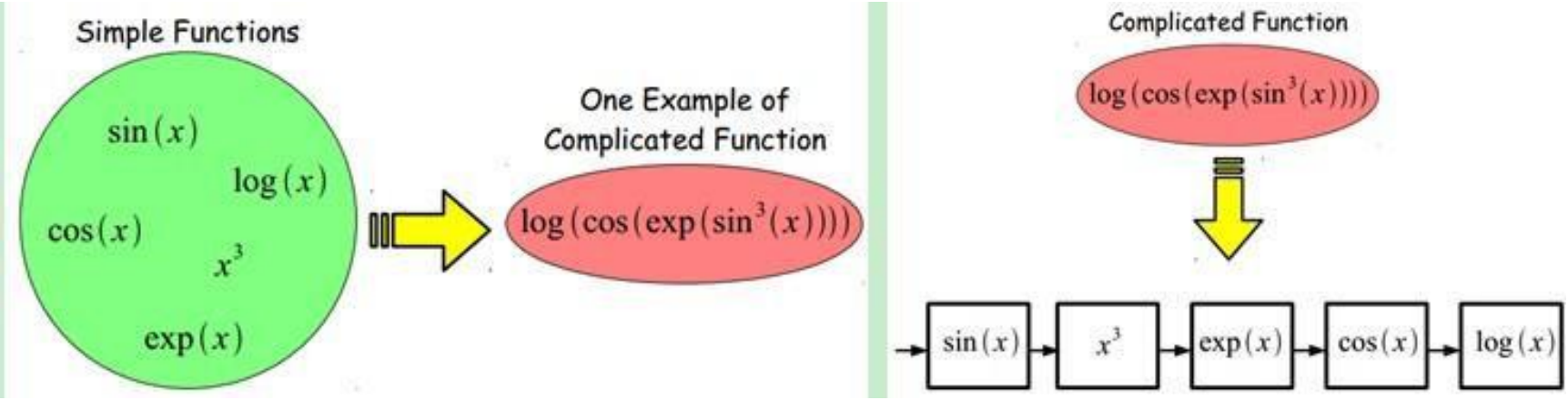
- 深度网络的优势
  - 发现数据中复杂的层次结构
  - 高层的特征是低层特征的组合，从低层到高层的特征表达越来越抽象和概念化，也即越来越能表现语义或者意图。





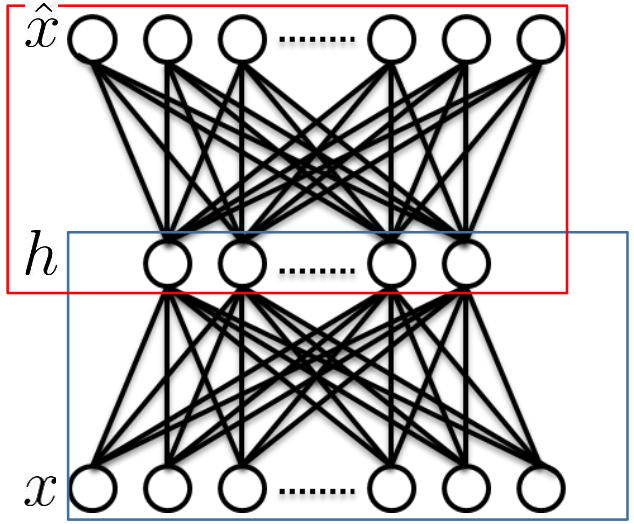
为什么需要深层网络？（表示代价）

理论上讲，浅层的网络可逼近任意复杂的函数（不考虑计算代价）
多层的好处是可以用较少的参数表示复杂的函数



深度学习基本模块--自动编码器

- 自动编码器（Auto-Encoders, AEs）



Decoder

Encoder

编码器: $\mathbf{h} = s(\mathbf{W}\mathbf{x} + \mathbf{b}_e)$

解码器: $\hat{\mathbf{x}} = \mathbf{W}^T\mathbf{h} + \mathbf{b}_d$

目标函数:

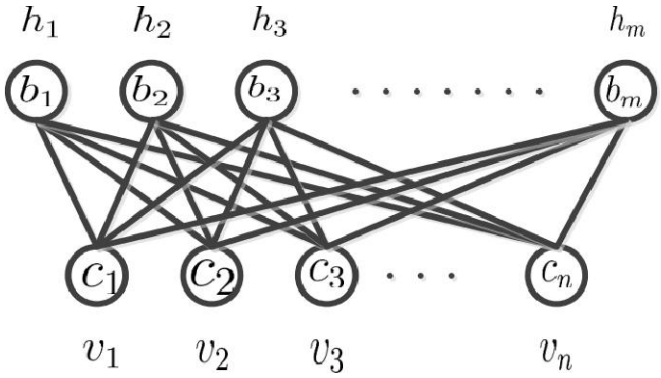
$$J = \sum_{i=1}^{N} |\hat{x}^{(i)} - x^{(i)}|_2 + \lambda g(h)$$

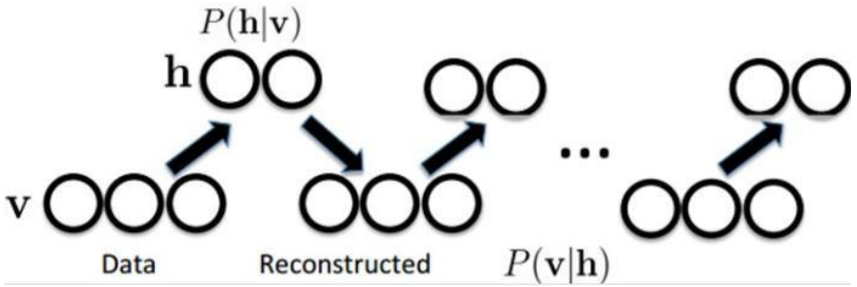
$$= \sum_{i=1}^{N} |W^T\mathbf{h} - x^{(i)}|_2 + \lambda g(\mathbf{h})$$

Sparse AE, Denoising AE, Contractive AE

深度学习基本模块--受限玻尔兹曼

- 受限玻尔兹曼机 (Restricted Boltzmann Machines, RBMs)





目标函数：

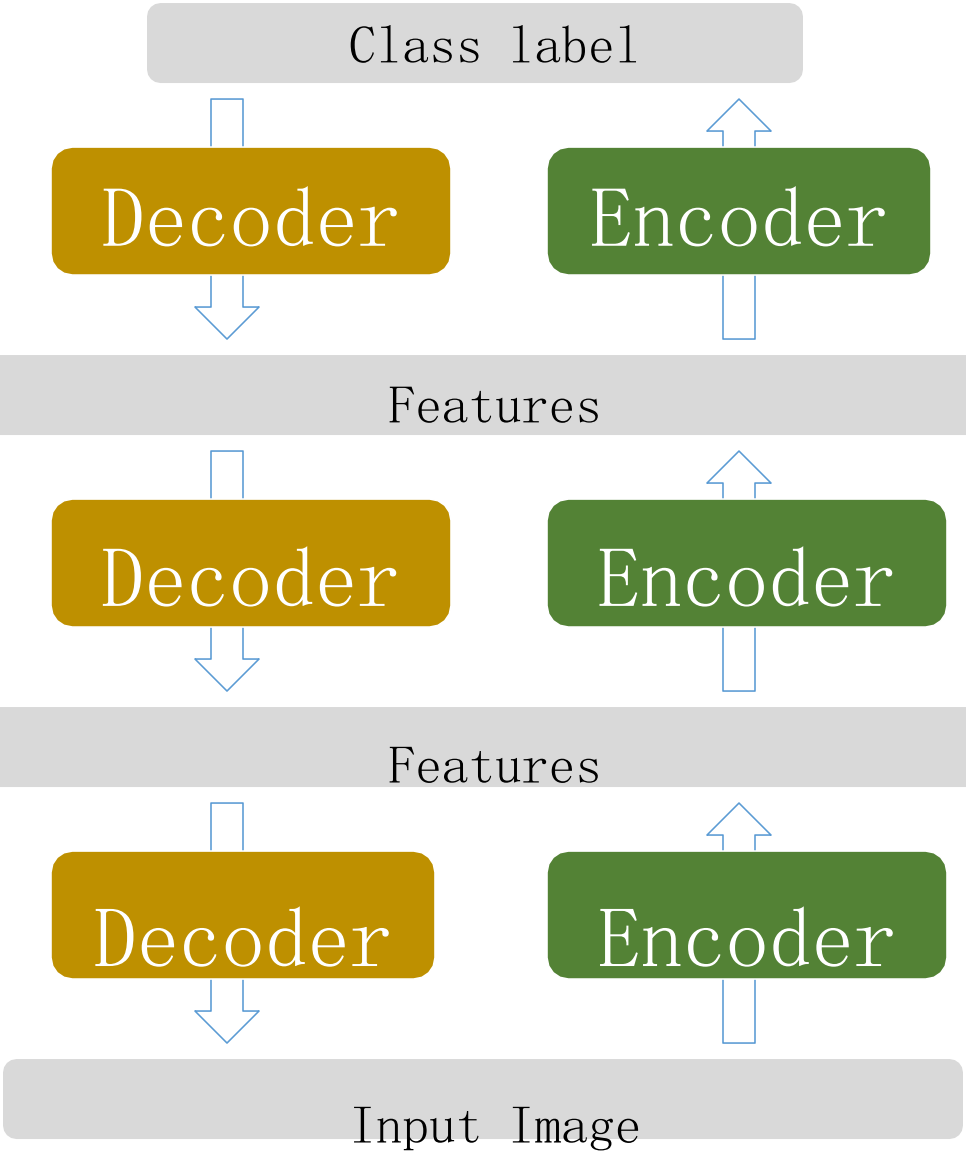
$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}h_j v_i - \sum_{i=1}^{n} c_i v_i - \sum_{j=1}^{m} b_j h_j \quad \Leftrightarrow \quad p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))$$

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = < v_i h_j >_{\text{data}} - < \hat{v}_i \hat{h}_j >_{\text{model}}$$

RBMs：固定可见单元，隐单元相互独立；固定隐单元，可见单元相互独立。

$$p(H_j = 1|\mathbf{v}) = \sigma(\sum_{i=1}^{n} w_{ij}v_i + b_j) \quad \text{和} \quad p(V_i = 1|\mathbf{h}) = \sigma(\sum_{j=1}^{m} w_{ij}h_j + c_i)$$

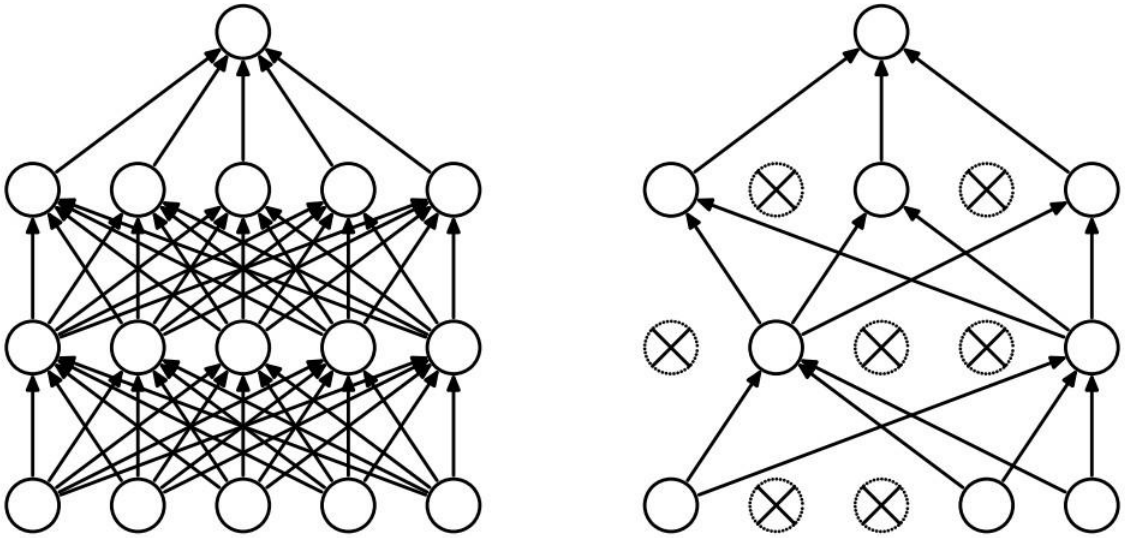
深度学习训练过程



- 第一步：采用自下而上的无监督学习（Pretraining）
- 第二步：自顶向下的监督学习

深度学习的关键技术： Dropout

- Dropout正则化技术



(a) Standard Neural Net (b) After applying dropout.

Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Nitish Srivastava, Geoffrey Hinton, et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15 (2014) 1929-1958

深度学习的关键技术：Batch Normalization

- Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

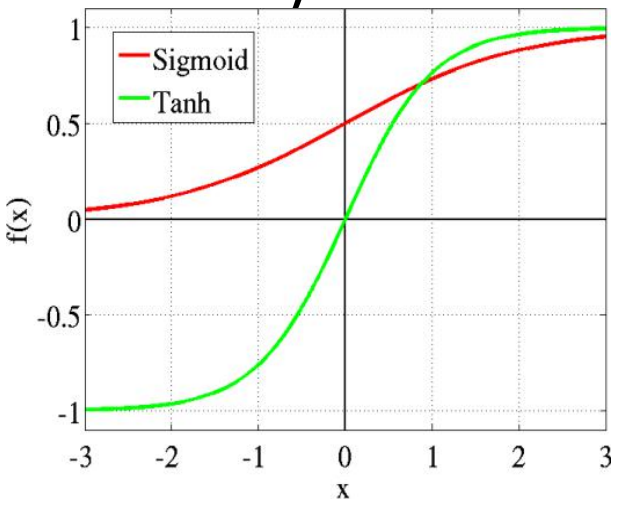
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

((1))可视为一种正则化技术，某些情况下不再需要Dropout;
((2))有效降低初始化的影响，允许使用higher learning rates，大幅度加快训练 速度

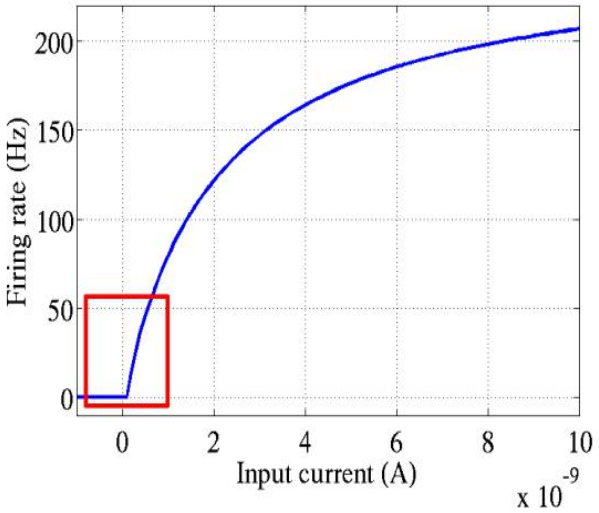
S. Ioffe, C. Szegedy, et al. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv:1502.03167v3

深度学习的关键技术： ReLU激活函数

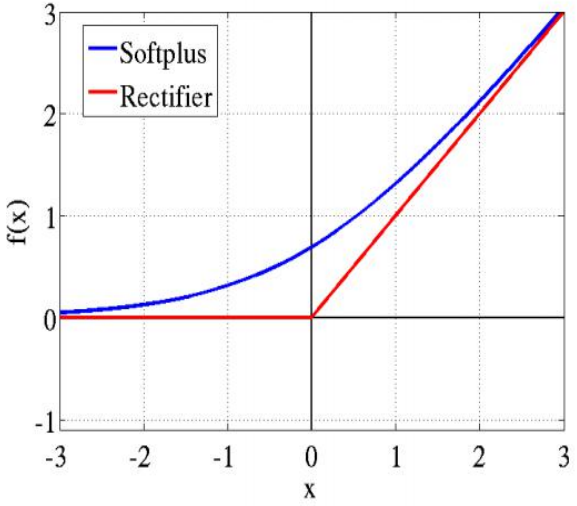
- ReLU (Rectified Linear Units)激活函数: $\sigma(x) = \max(0, x)$



传统神经网络中最常用的两个激活函数，Sigmoid系（Logistic-Sigmoid、Tanh-Sigmoid）



2001年，神经科学家Dayan、Abott从生物学角度，模拟出了脑神经元接受信号的激活模型



ReLu(Rectified Linear Units)激活函数

Xavier Glorot, Antoine Bordes, Yoshua Bengio, Deep Sparse Rectier Neural Networks

ReLU对深度学习的贡献

- 引入稀疏性

与脑神经元的激活机制类似

某种程度上等效于无监督学习的预训练

Neuron	MNIST	CIFAR10	NISTP	NORB
With unsupervised pre-training				
Rectifier	**1.20%**	**49.96%**	**32.86%**	**16.46%**
Tanh	**1.16%**	**50.79%**	35.89%	17.66%
Softplus	**1.17%**	**49.52%**	**33.27%**	19.19%
Without unsupervised pre-training				
Rectifier	**1.43%**	**50.86%**	**32.64%**	**16.40%**
Tanh	1.57%	52.62%	36.46%	19.29%
Softplus	1.77%	53.20%	35.48%	17.68%

没做预训练情况下，ReLu激活网络遥遥领先其它激活函数。

在预训练后，ReLu仍然有提升空间。

- 减轻梯度消失问题

Grad=Error·Sigmoid′(x)·xGrad=Error·Sigmoid′(x)·x使用双端饱和(即值域被限制)Sigmoid系函数会有两个问题：

①Sigmoid'(x)∈(0,1) 导数缩放

②x∈(0,1)或x∈(-1,1) 饱和值缩放

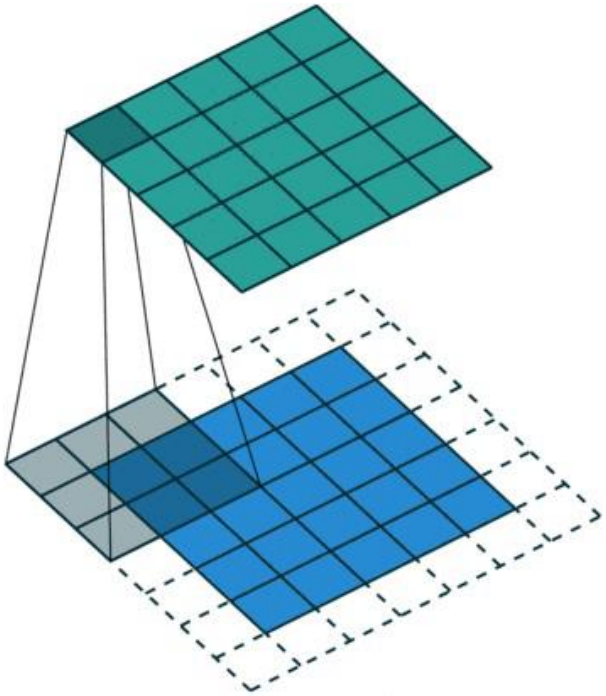
这样，经过每一层时，Error都是成倍的衰减，一旦进行递推式的多层的反向传播，梯度就会不停的衰减，消失

而ReLU的梯度是1，从而梯度很好的在回传，训练速度得到了很大的提高

这与LSTM的Gate有异曲同工之妙

代表性模型：CNNs

- Convolutional Neural Networks (CNNs)



内核大小：内核大小定义了卷积的视野。二维的常见选择是3——即3x3像素。

•步幅：步幅定义了遍历图像时内核的 步长。虽然它的默认值通常为1，但我 们可以使用2的步长，类似于最大池化 对图像进行下采样。

•padding：padding定义样本的边框如何处理。一（半）个padding卷积将保持 空间输出尺寸等于输入尺寸，而如果内 核大于1，则不加卷积将消除一些边界。

图1 二维卷积、内核大小为3、步幅为1

这12张图生动的告诉你，深度学习中的卷积网络是怎么一回事？
http://mp.weixin.qq.com/s/CLFbhWMcat4rN8YS_7q25g

代表性模型：CNNs

常用的**CNN**模型：

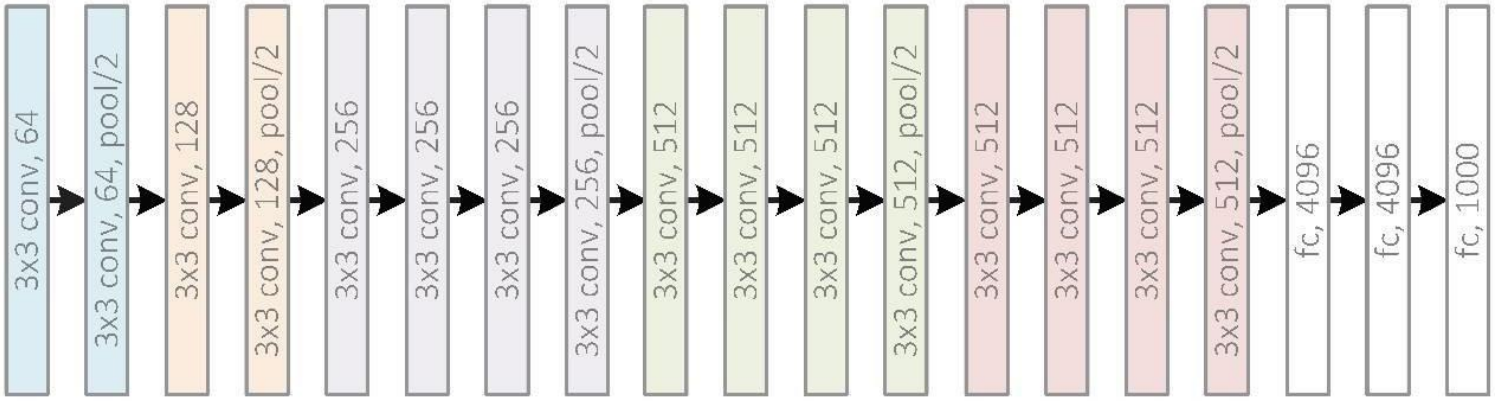
AlexNet

Alex Krizhevsky, L. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” NIPS, 2012.

VGG Net （加深版的**AlexNet**，牛津大学）

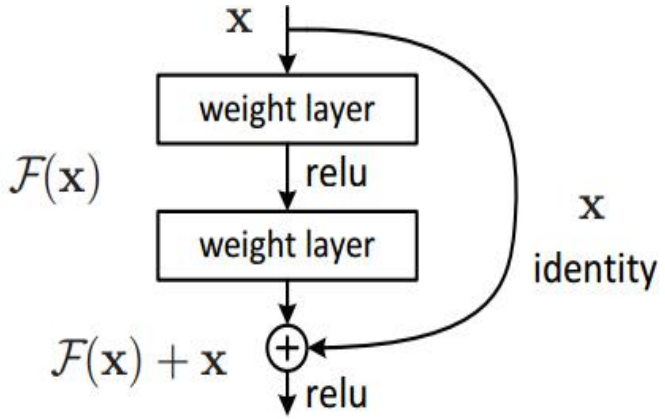
Karen Simonyan, Andrew Zisserman,Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556

遵循基本卷积网络的原型布局：一系列卷积层、最大池化层和激活层，最后还有一些全连接的分类层。



代表性模型: Residual Network

- Deep Residual Network



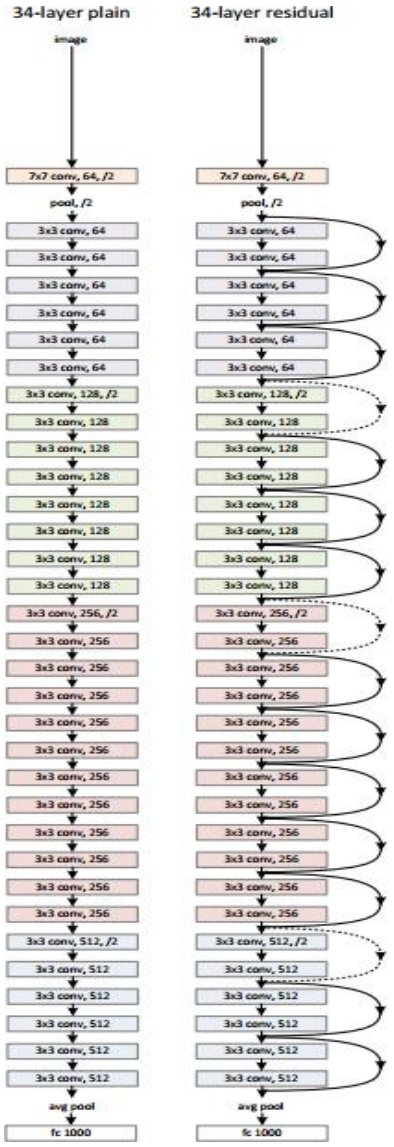
$$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}.$$

Figure 2. Residual learning: a building block.

(1)直接映射是难以学习的,不再学习从 x 到 H(x) 的基本映射关系,而是学习这两者之间的差异，也就是残差 F(x)=H(x)-x

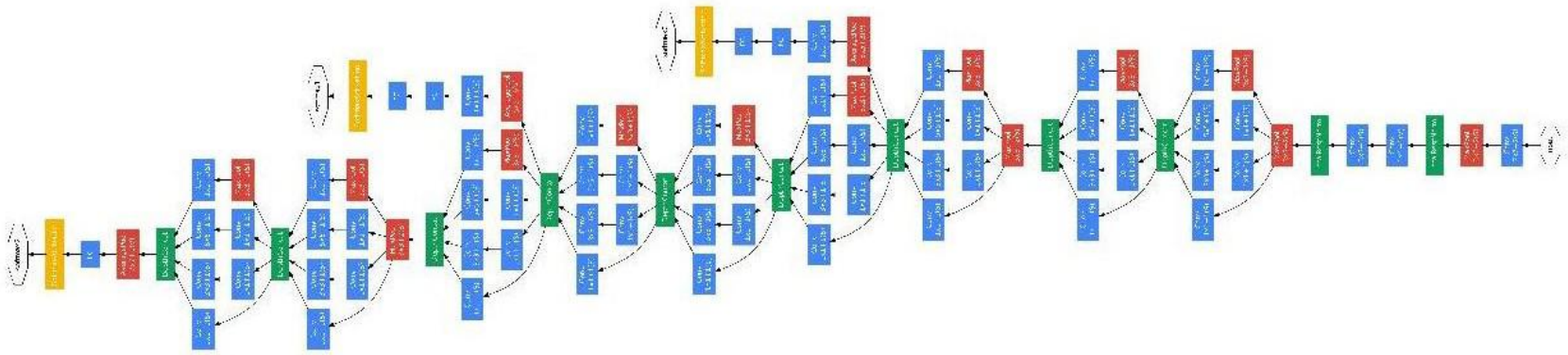
(2)梯度信号可以直接通过捷径连接回到更早的层, 较好地解决了梯度消失问题

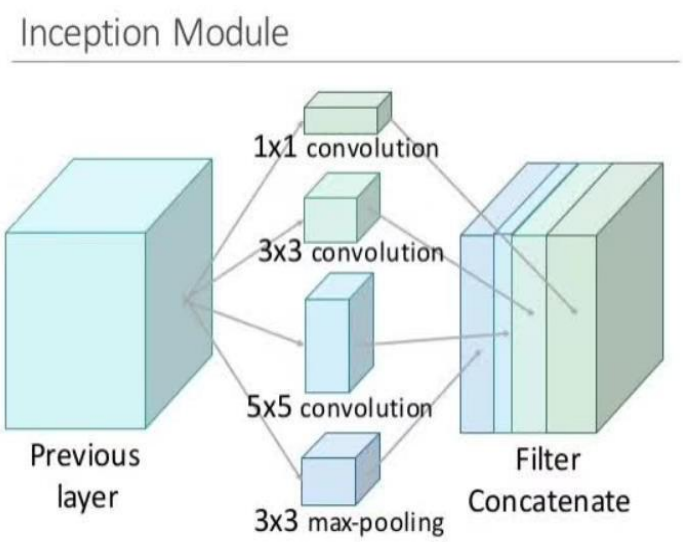
He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)



代表性模型：GoogleNet

- Google Net (Network in network)

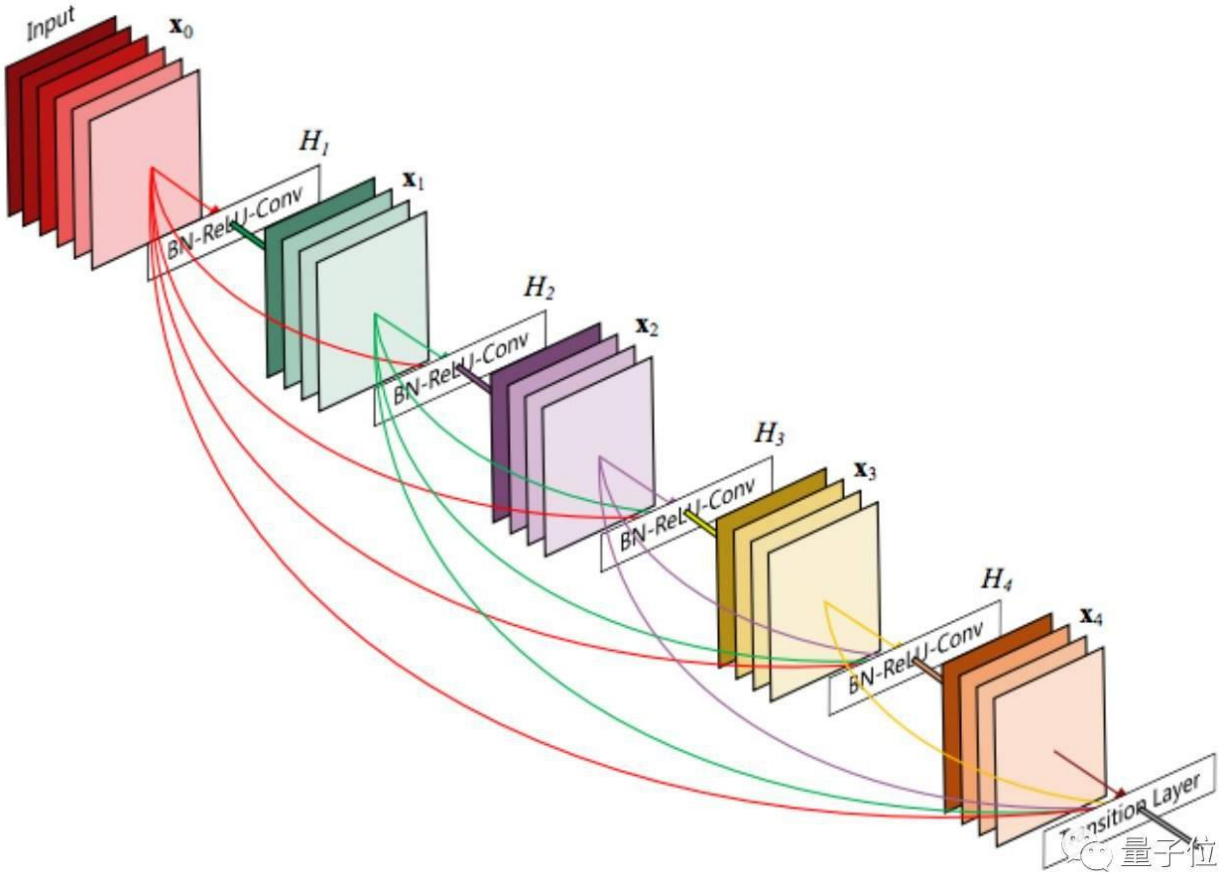




其它代表性模型：LSTMs等

- Recurrent Neural Networks (RNNs)，LSTMs

- DenseNet (CVPR2017 Best paper)



目录

Keras

- Keras:基于Python的深度学习库

- Keras是一个高层神经网络API，Keras由纯Python编写而成并基Tensorflow、Theano以及CNTK后端。Keras为支持快速实验而生，能够把你的idea迅速转换为结果，如果你有如下需求，请选择Keras：

- 简易和快速的原型设计（keras具有高度模块化，极简，和可扩充特性）
- 支持CNN和RNN，或二者的结合
- 无缝CPU和GPU切换

Keras 安装

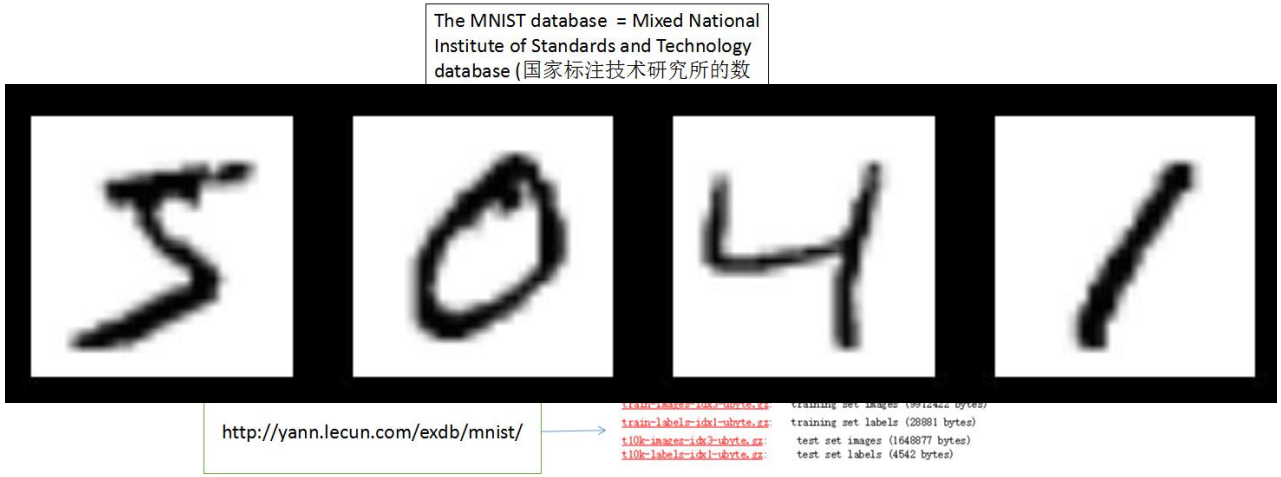
Keras适用的Python版本是： Python 2.7-3.6

```
# GPU 版本
>>> pip install --upgrade tensorflow-gpu

# CPU 版本
>>> pip install --upgrade tensorflow

# Keras 安装
>>> pip install keras -U --pre
```

Mnist

这个MNIST数据库是一个手写数字的数据库，它提供了六万的训练集和一万的测试集。

它的图片是被规范处理过的，是一张被放在中间部位的28px*28px的灰度图




```
>>> conda install git
>>> git clone https://github.com/fchollet/keras.git
>>> cd keras/examples/
>>> python mnist_mlp.py
```