CNN近期进展与实用技巧

刘昕 博士研究生
中科院计算所VIPL研究组
vipl.ict.ac.cn

中国科学院计算技术研究所
Institute of Computing Technology,Chinese Academy of Sciences

Outline

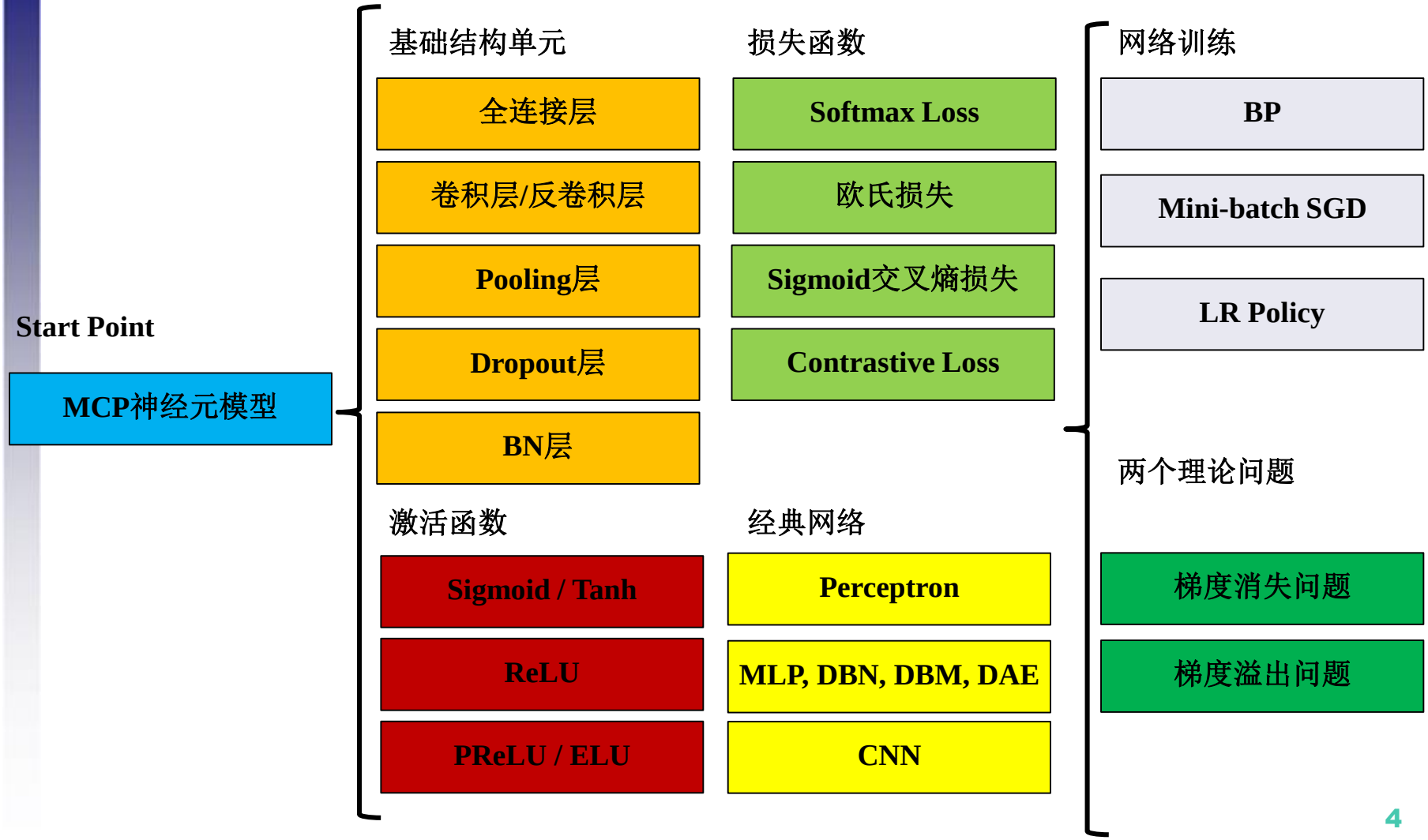
- DL基础理论
- CNN结构演化
- CNN实用技巧：以Caffe为例
- CNN实战：ICCV2015年龄估计竞赛
- 前沿话题讨论

Outline

- **DL基础理论**
- CNN结构演化
- CNN实用技巧：以Caffe为例
- CNN实战：ICCV2015年龄估计竞赛
- 前沿话题讨论

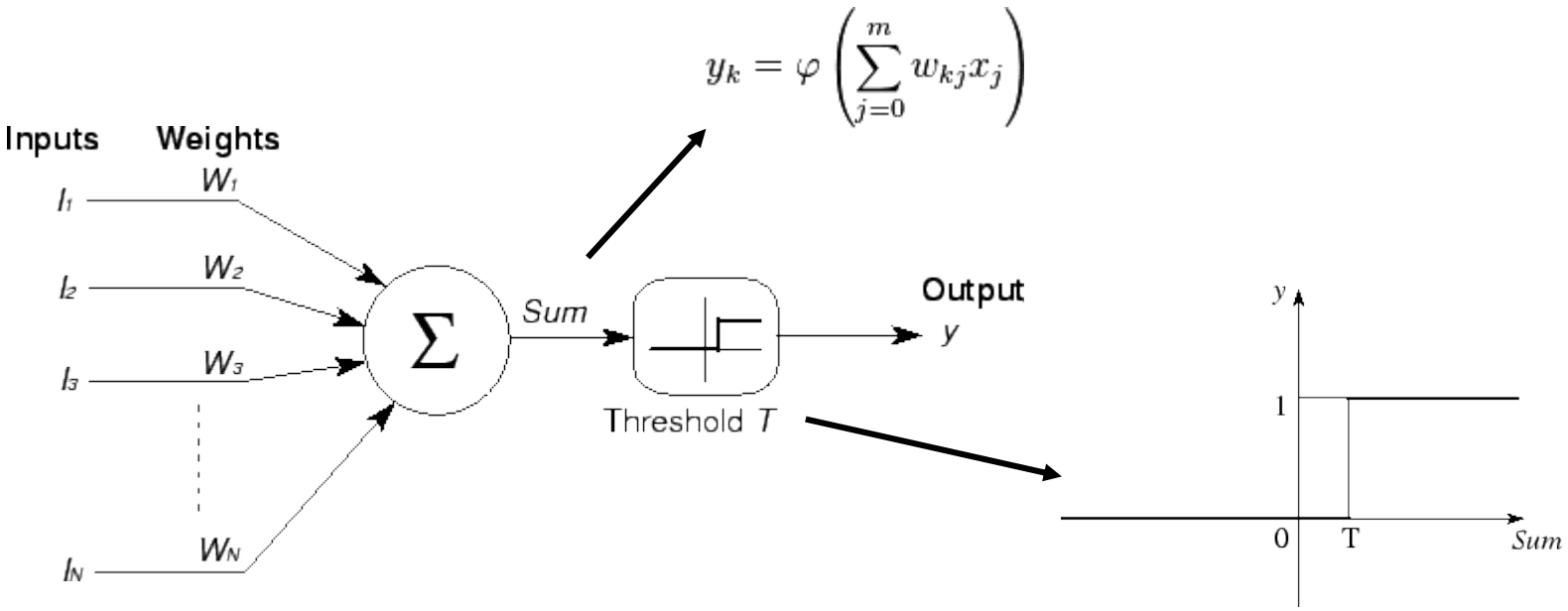
DL基础理论

本节内容概要



DL基础理论—基础结构单元

- **MCP神经元模型（1943）**

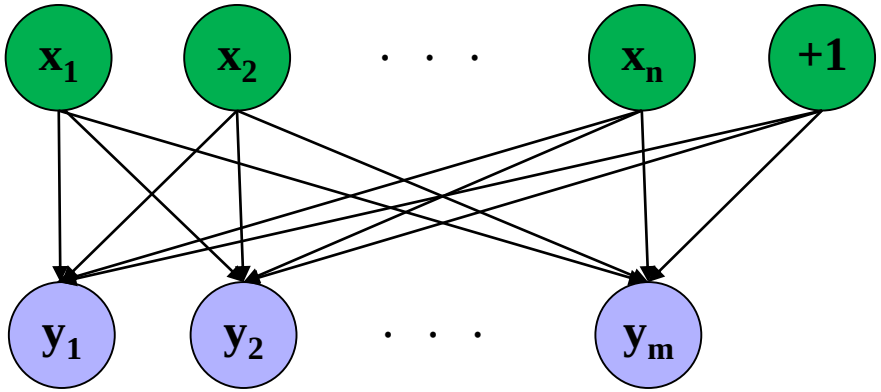


MCP人工神经元模型：输出 = 多个输入加权和 + 二值激活函数

McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics* **5** (4): 115–133

DL基础理论—基础结构单元

- 全连接层
  - 相当于内积运算，图中“+1”表示偏置项b
  - 输出层的神经元和输入层的每个神经元都相连：得名“全”连接



  - Forward运算：$y = W^T x + b$，其中$y \in R^{m\times 1}, x \in R^{n\times 1}, W \in R^{n\times m}$
  - Backward运算：$\frac{\partial L}{\partial x} = W * \frac{\partial L}{\partial y}$，$\frac{\partial L}{\partial W} = x * \left(\frac{\partial L}{\partial y}\right)^T$

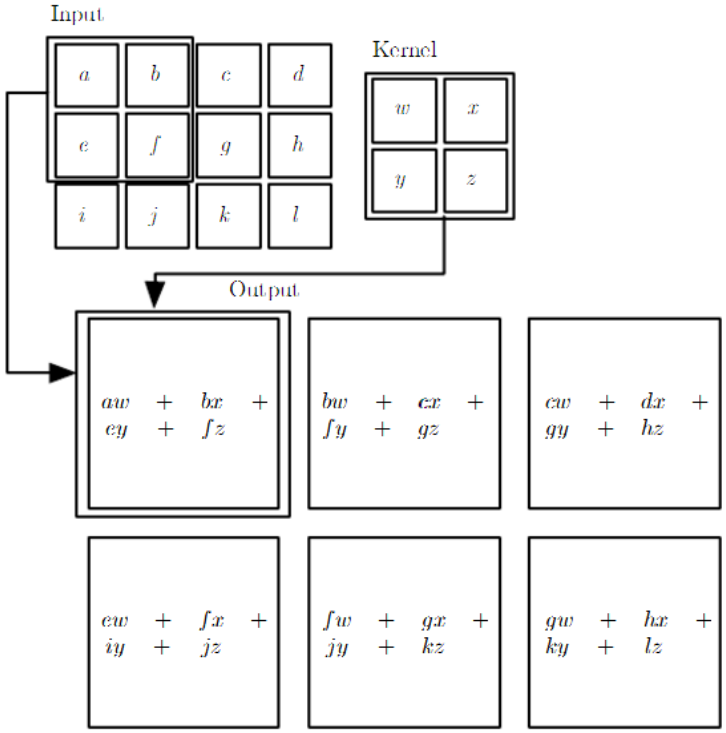
DL基础理论—基础结构单元

- 卷积层
  - 2D卷积的数学形式

连续卷积：$h(x,y) = i * k(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} i(u,v)k(x-u,y-v)dudv$

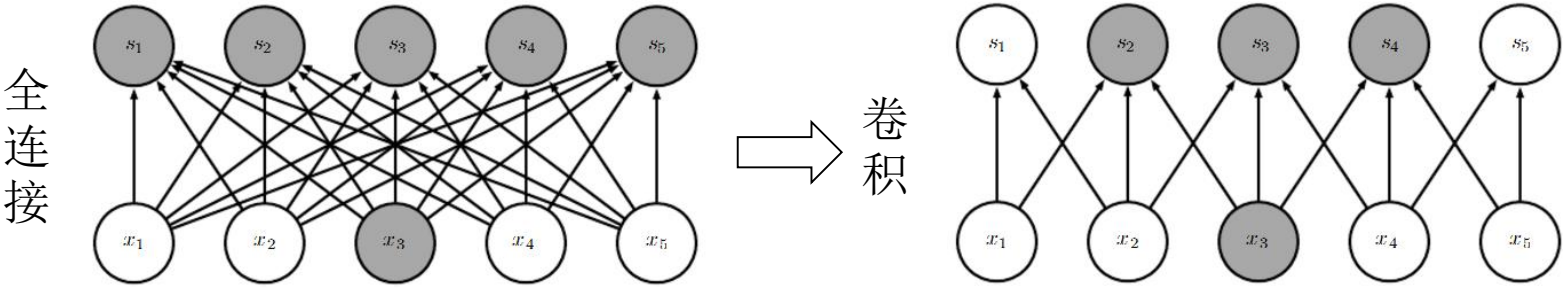
离散卷积：$H(x,y) = I * K(x,y) = \sum_m \sum_n I(m,n)K(x-m,y-n)$

Caffe实现：$H(x,y) = I * K(x,y) = \sum_m \sum_n I(x+m,y+n)K(m,n)$



DL基础理论—基础结构单元

- 卷积层
  - 稀疏连接: 输出层神经元只和部分输入层神经元相连

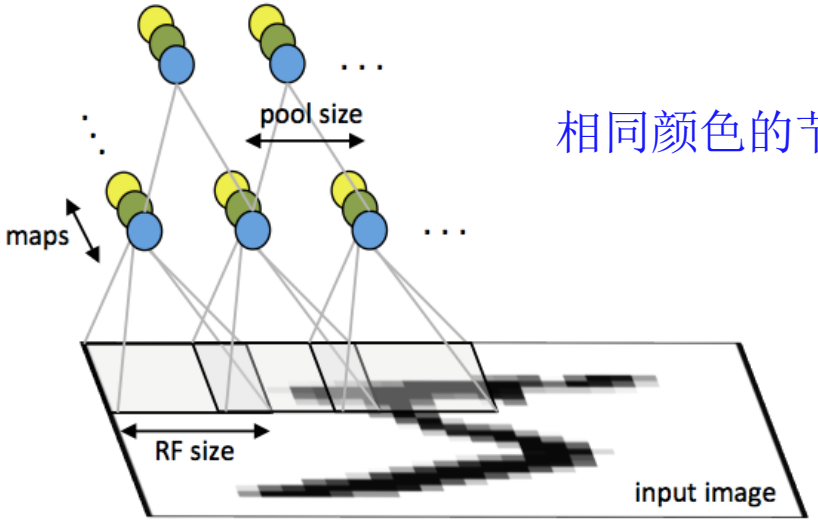


  - 权值共享

Pooling层： 3x22x22

Conv层: 3x24x24
Kernel Size: 5x5
Stride: 1

Data层: 1x28x28



相同颜色的节点共享权值

DL基础理论—基础结构单元

- 卷积层
  - 多通道卷积的快速实现（以Caffe为例）

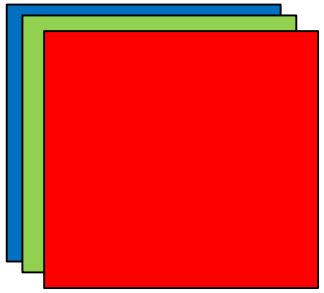
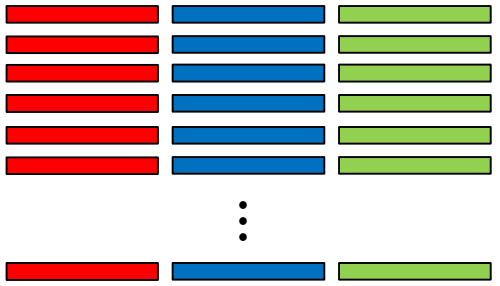


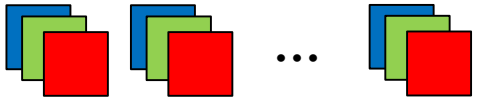
Image: $C \times H \times W$

Im2Col



Feature Matrix X: $(H_{out} \times W_{out}) \times (\mathrm{C} \times \mathrm{K} \times \mathrm{K})$

$WX^T + b$



Kernel:$C_{out} \times C \times K \times K$

$\boldsymbol{W}$:$C_{out} \times (C \times K \times K)$

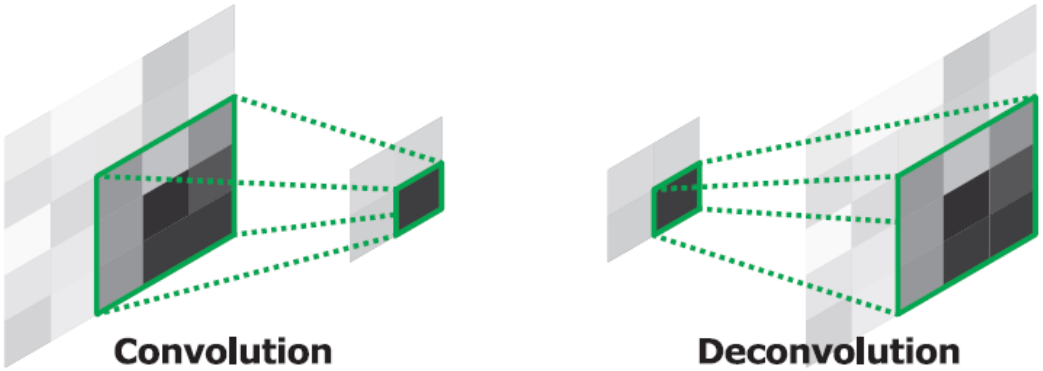
bias:$C_{out} \times 1 \times 1 \times 1$

$\boldsymbol{b}$: $C_{out} \times (H_{out} \times W_{out})$

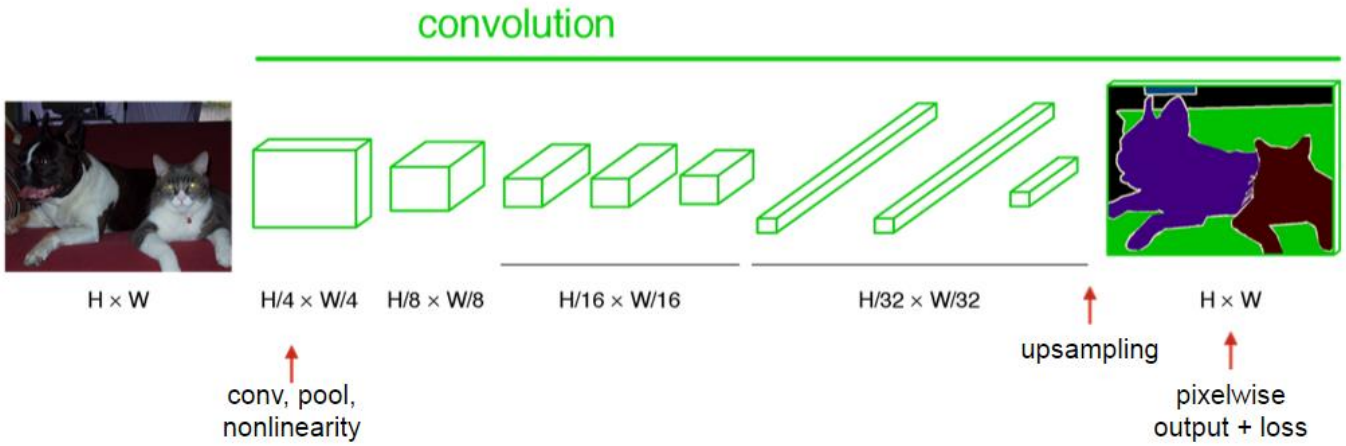
Output Matrix：$C_{out} \times (H_{out} \times W_{out})$

DL基础理论—基础结构单元

- 反卷积层
  - 卷积的逆过程，实现信号复原

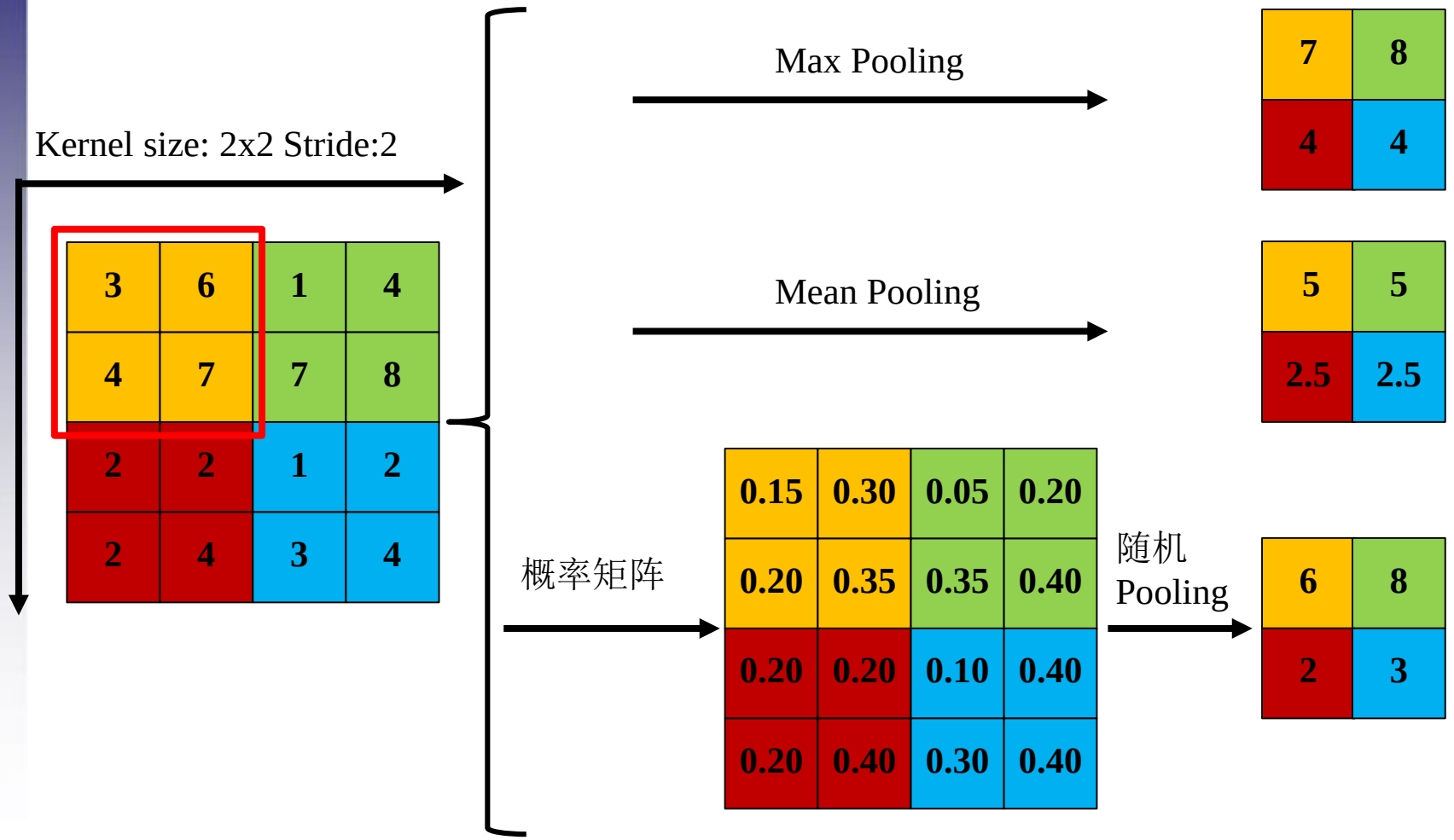


  - 全卷积网络（FCN），反卷积层实现上采样（upsampling）



DL基础理论—基础结构单元

- **Pooling层**
  - 一般配合卷积层使用



DL基础理论—基础结构单元

- **激活函数**
  - 用于卷积层和全连接层之后
  - 网络非线性来源

Sigmoid $S(x) = 1/(1+e^{-x})$

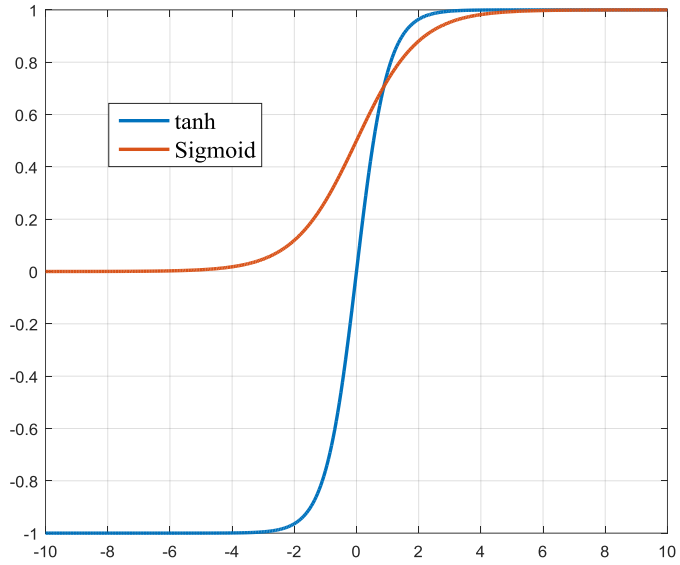
tanh $\tanh(x)$

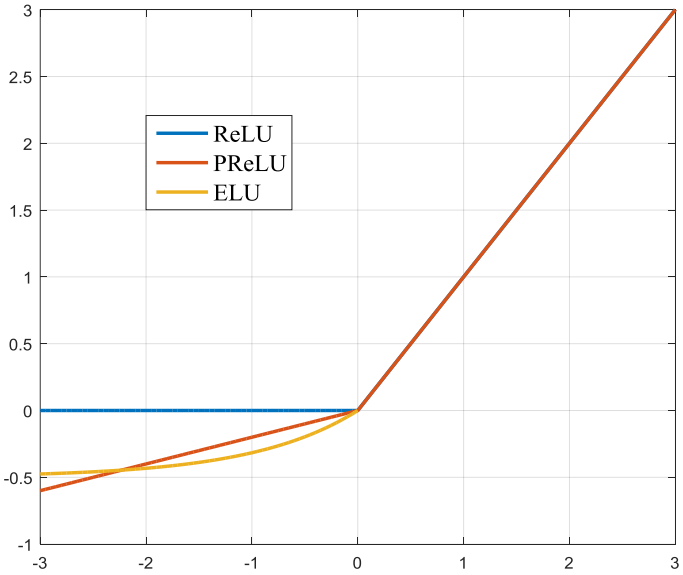
ReLU $\max(0, x)$

PReLU $\max(ax, x)$

ELU $y = \begin{cases} x \text{ if } x > 0 \\ \alpha(e^x - 1) \text{ if } x \le 0 \end{cases}$

Maxout $\max(w_1 x_1 + b_1, w_2 x_2 + b_2)$





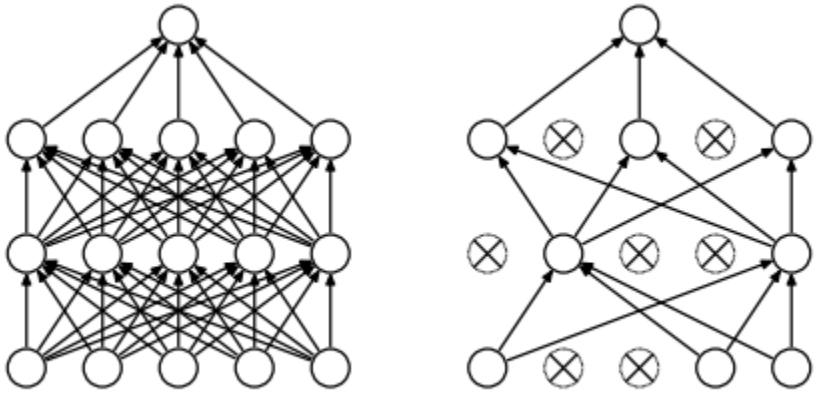
DL基础理论—基础结构单元

- **Dropout（2012）**
  - 引入Bernoulli随机数u, p代表dropout ratio

$$y_{\text{train}} = \begin{cases} \frac{x}{1-p} & \text{if } u > p \\ 0 & \text{otherwise} \end{cases} \qquad \text{Where, } u \sim U(0,1)$$

$$E(y_{\text{train}}) = p \cdot 0 + (1-p)\frac{E(x)}{1-p} = E(x)$$

  - 测试阶段：Do Nothing
  - 正则化手段，提高泛化能力



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

G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

DL基础理论—基础结构单元

- **Batch Normalization（2015）**
  - 逐层尺度归一，避免了梯度消失和梯度溢出
  - 加速收敛5x~20x, 同时作为一种正则化技术也提高了泛化能力

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.

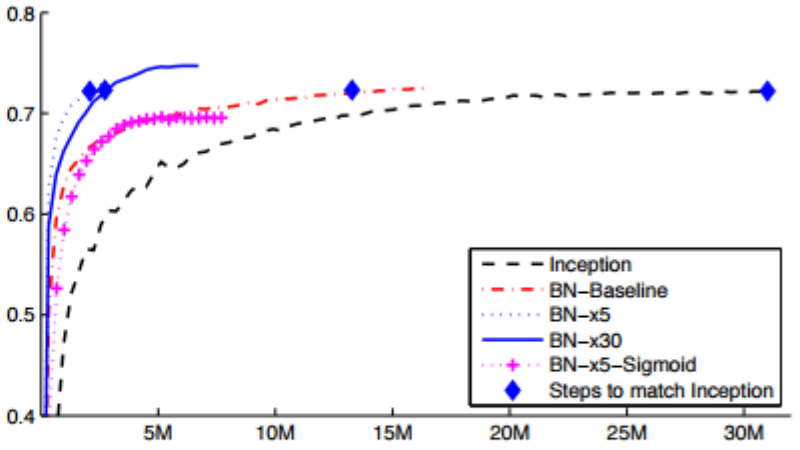
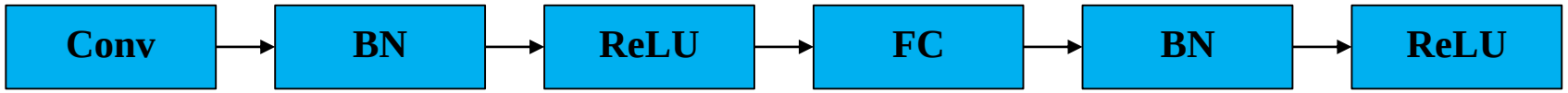


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*



Sergey Ioffe, Christian Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICML15

DL基础理论—损失函数

- 损失函数
  - **Softmax Loss**

    损失函数：$E=\frac{-1}{N}\sum_{n=1}^{N}\log(\hat{p}_{nl_n}),l_n\in[0,1,...,K-1]$

    其中 $\hat{p}_{nl_n}$ 由softmax函数计算 $\hat{p}_{nk}=\frac{e^{x_{nk}}}{\sum_{l=0}^{K-1}e^{x_{nl}}}$

    适用场景：单标签分类问题
  - **Euclidean Loss**

    损失函数：$E=\frac{-1}{N}\sum_{n=1}^{N}\left\|\hat{y}_n-y_n\right\|_2^2$

    适用场景：实数值回归问题
  - **Sigmoid Cross Entropy Loss**

    损失函数：$E=\frac{-1}{N}\sum_{n=1}^{N}[p_n\log\hat{p}_n+(1-p_n)\log(1-\hat{p}_n)]$

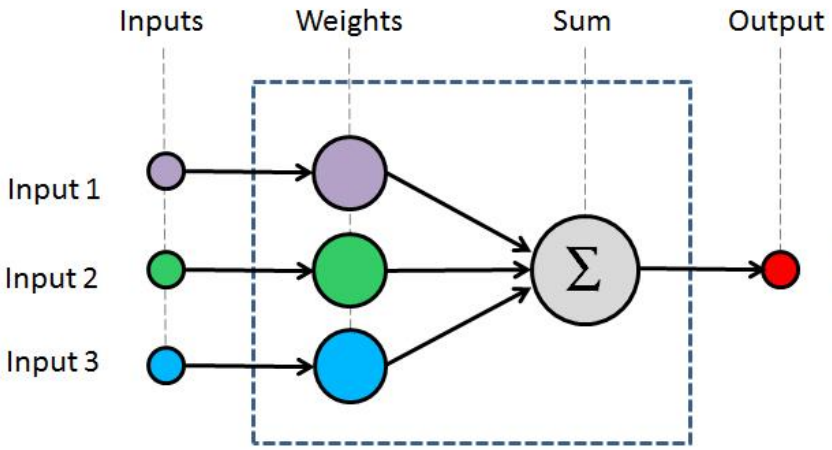
    适用场景：预测目标概率分布，可用于多标签学习
  - **Contrastive Loss**

    损失函数：$E=\frac{-1}{N}\sum_{n=1}^{N}[(y)d^2+(1-y)\max(\text{margin}-d,0)^2,d=\left\|a_n-b_n\right\|_2$

    适用场景：深度度量学习

DL基础理论—经典网络

- **Rosenblatt 1957年感知机模型 (1/2)**
  - 最早用于二类分类问题



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

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386-408, 1958

DL基础理论—经典网络

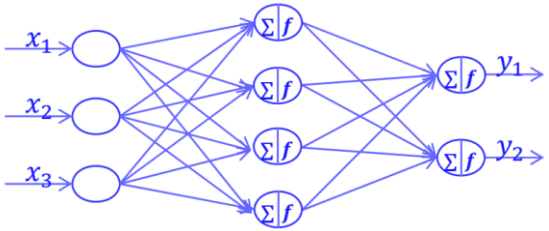
- **单层感知机模型的缺陷 (2/2)**
  - 1962年，单层感知机模型的收敛性得到理论证明[1]
  - Minsky & Papert的专著*Perceptron*(1969)
  - 解决不了XOR(异或)问题，本质上是线性模型
  - 几乎宣判了单层感知机甚至人工神经网络的死刑，导致了后来NN研究多年的寒冬

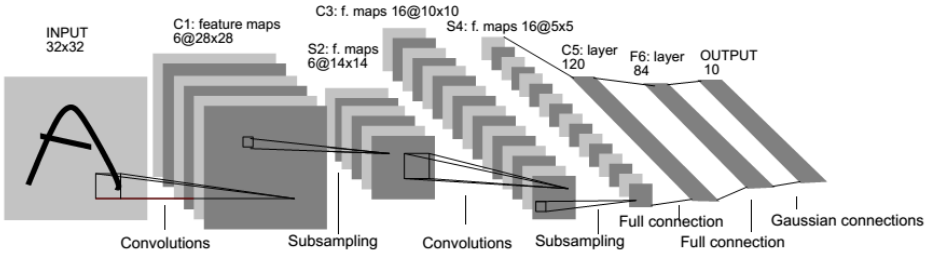
Novikoff, A. B. J. (1962). On convergence proofs on perceptrons. Proceedings of the Symposium on the Mathematical Theory of Automata (pp. 615–622)
XOR problem theory: http://home.agh.edu.pl/~vlsi/AI/xor_t/en/main.htm

Credited to Prof. Shiguang Shan

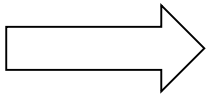
DL基础理论—经典网络

- **1980s~2006**
  - 多层感知机（MLP）
  - 卷积网络（LeNet）
  - **Error梯度反向传播**
  - 非线性激活函数





- **缺陷**
  - 优化困难：梯度消失
  - 训练数据不足
  - 计算资源有限



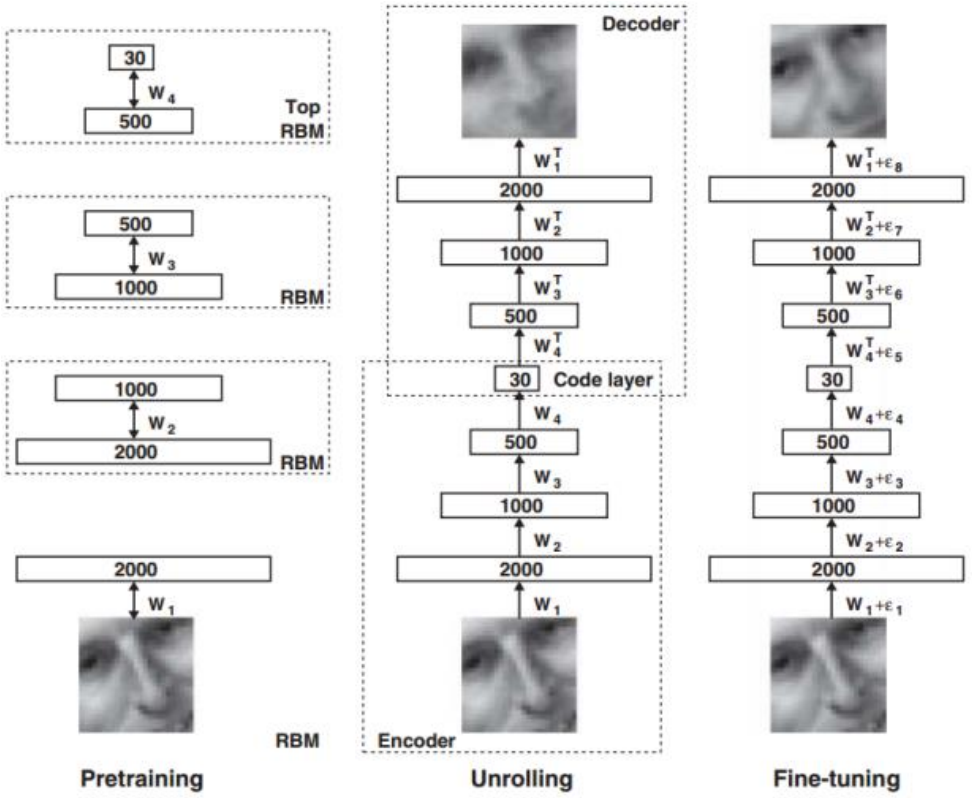
- Decision Tree
- SVM
- Boosting
- Sparse Coding
- Graph Model

Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". *Nature* **323** (6088): 533–536.

Credited to Prof. Shiguang Shan

DL基础理论—经典网络

- **2000s**
  - 无监督学习
  - 分层预训练
  - 新的网络结构
    - DBM, DBN, **DAE**
  - 得名“深度”学习



Hinton, G. E., Osindero, S. and Teh, Y., A fast learning algorithm for *deep belief nets*. Neural Computation 18:1527-1554, 2006

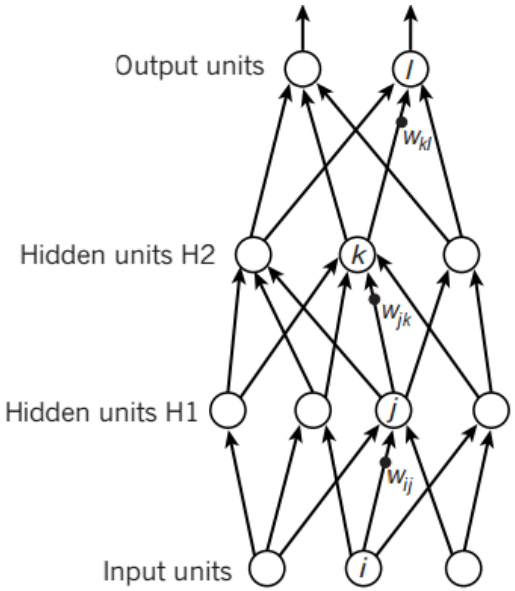
Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. Science, Vol.313. no. 5786, pp. 504 - 507, 28 July 2006

Yoshua Bengio, Pascal Lamblin, Dan Popovici and Hugo Larochelle, Greedy *Layer-Wise Training of Deep Networks*, NIPS2006

DL基础理论—训练方法

- **Back Propagation**
  - 1974年Webos在博士论文中首次提出BP算法，但未引发关注
  - 目前广泛使用的BP算法诞生于1986年
  - 以全连接层为例：链式求导，梯度反向传导

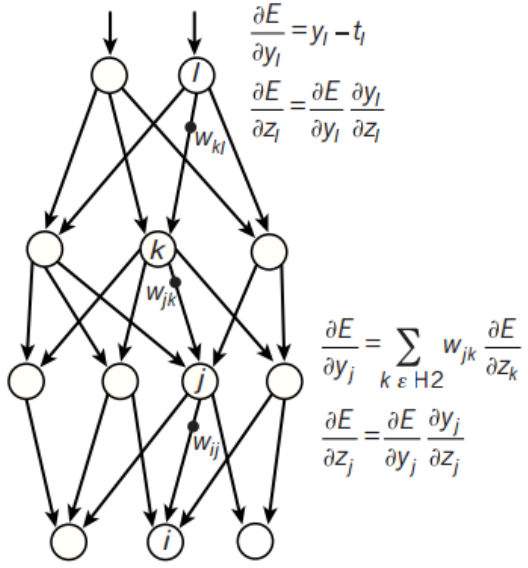
网络前向计算



误差反向传播

$$\frac{\partial E}{\partial y_k} = \sum_{l \varepsilon out} w_{kl} \frac{\partial E}{\partial z_l}$$

$$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k}$$



Werbos, P.. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, 1974

Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". *Nature* **323** (6088): 533–536.

DL基础理论—训练方法

- **Gradient Descent and its Variant**
  - Gradient Descend：利用所有样本计算梯度

$$w_{t+1} = w_t - \eta_t \nabla_w L(w)$$

速度慢！大数据内存不足！

  - Stochastic Gradient Descend：随机选择单个样本

$$w_{t+1} = w_t - \eta_t \nabla_w L(w, x_i, y_i)$$

方差大！损失函数震荡严重！

  - Mini-batch SGD: 对随机mini-batch计算梯度，进行参数更新

$$w_{t+1} = w_t - \frac{1}{N}\eta_t \sum_{i=1}^{N} \nabla_w L(w_t, x_i, y_i)$$

Mini-batch SGD:
for i = 1 : Num_Iterations
1. 随机采样一个Mini-batch的样本
2. 前向操作：计算损失
3. 后向操作：通过BP算法计算梯度
4. 网络更新：利用步骤3）计算的梯度更新网络参数

end

http://sebastianruder.com/optimizing-gradient-descent/index.html

DL基础理论—训练方法

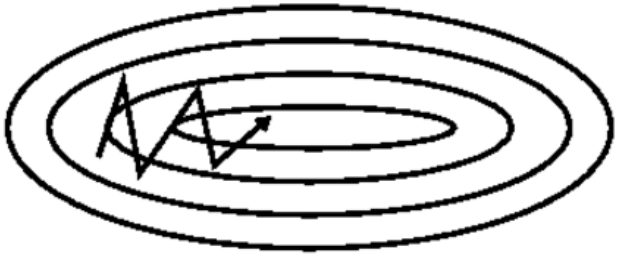
- **Mini-batch SGD**
  - Weight Decay
    - 避免过拟合，二范数较常用，一般 λ 设置的较小，例如 0.0005

$$\tilde{L}(w) = L(w) + \frac{\lambda}{2}\|w\|_2^2$$

  - Momentum
    - 使随机梯度下降跳出局部最优，一般 $\mu = 0.9$

$$v_{t+1} = \mu v_t - \eta_t \nabla L(w_t)$$
$$w_{t+1} = w_t + v_{t+1}$$

SGD without Momentum



SGD with Momentum

DL基础理论—训练方法

- **Learning Rate Policy（以Caffe为例）**
  - Fixed
    - Learning Rate固定不变
  - Step
    - Learning Rate在每隔stepsize轮迭代后减少gamma倍
  - Polynomial
    - Learning Rate依多项式曲线下降

$$LR(t) = \text{base\_lr} \times (1\text{-}\frac{t}{T})^{power}$$

  - Inv
    - Learning Rate随迭代次数增加而下降

$$LR(t) = \text{base\_lr} \times (1+\text{gamma*iter})^{-power}$$

Solver.prototxt

```
base_lr: 0.01
lr_policy: “fixed”
```

```
base_lr: 0.01
lr_policy: “step”
gamma: 0.1
stepsize: 100000
```

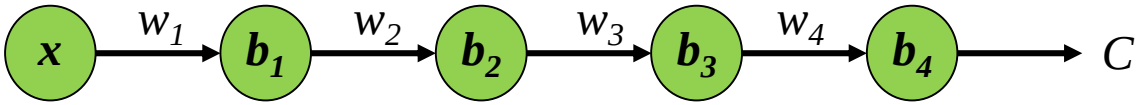
```
base_lr: 0.01
lr_policy: “poly”
power: 0.5
```

```
base_lr: 0.01
lr_policy: “inv”
gamma: 0.0001
power: 0.75
```

DL基础理论—两个理论问题

- 梯度消失问题
  - Sigmoid激活函数导致多层网络收敛困难

多层网络示例：



利用BP算法计算 b_1 的梯度:

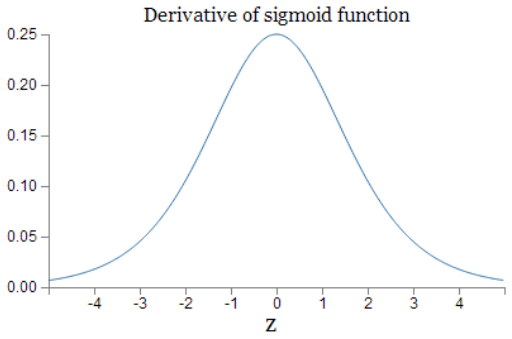
$$\frac{\partial C}{\partial b_1} = \sigma'(b_1) w_2 \sigma'(b_2) w_3 \sigma'(b_3) w_4 \sigma'(b_4) \frac{\partial C}{\partial b_4}$$

Sigmoid激活函数梯度：

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \in (0, \frac{1}{4}]$$

由此可得：

$$\frac{\partial C}{\partial b_1} \leq \boxed{(\frac{1}{4})^4} w_2 w_3 w_4 \frac{\partial C}{\partial b_4}$$

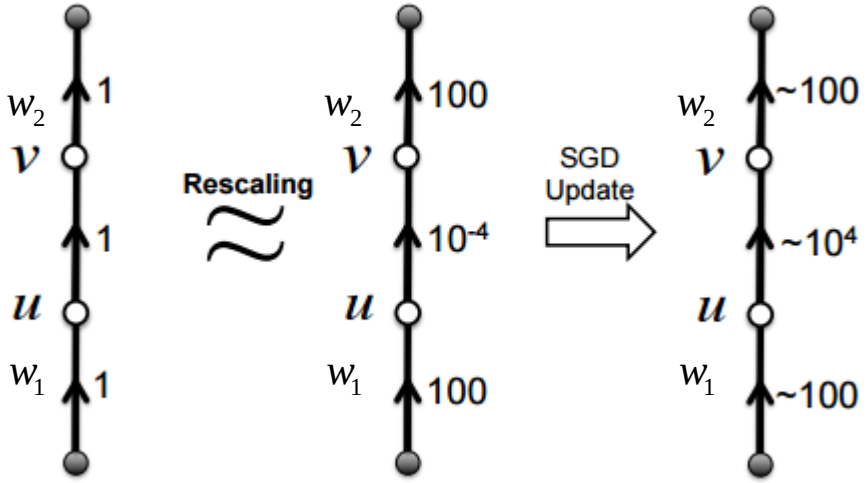


DL基础理论—两个理论问题

- **解决梯度消失问题的策略**
  - **LSTM**: 通过选择记忆和遗忘机制克服RNN的梯度消失问题，从而可以建模长时序列
  - **Layer-wise Pre-train**: 2006年Hinton等提出的训练深度网路的方法，用无监督数据作分层预训练，再用有监督数据fine-tune
  - **ReLU**: 新的激活函数解析性质更好，克服了sigmoid函数和tanh函数的梯度消失问题
  - **辅助损失函数**：e.g. GoogLeNet中的两个辅助损失函数，对浅层神经元直接传递梯度
  - **Batch Normalization:** 逐层的尺度归一

DL基础理论—两个理论问题

- 梯度溢出问题
  - 尺度不平衡的初始化容易导致网络NAN



最底层为输入节点，最上层为输出节点, 假设输入层节点（1，1），前向操作之后隐层节点u和v分别为100和0.01, 输出层梯度（-1，-1）。
按BP算法:w_1 梯度：$-1 * v = -0.01$, 节点v的梯度$-1 * 100 = -100$
节点u的梯度-0.01,w_2 梯度 -10^4

Neyshabur B, Salakhutdinov R R, Srebro N. Path-SGD: Path-normalized optimization in deep neural networks, NIPS15

DL基础理论—两个理论问题

- **梯度溢出问题**
  - **Xavier方差不变准则保持网络节点尺度不变**

卷积和全连接层单节点抽象：

$$y = \sum_{i=1}^{n} w_i x_i + b$$

$$\mathrm{Var}(w_i x_i) = E(x_i)^2 \mathrm{Var}(x_i) + E(w_i)^2 \mathrm{Var}(x_i) + \mathrm{Var}(w_i)\mathrm{Var}(x_i)$$

$$\mathrm{Var}(Y) = \mathrm{Var}(\sum_{i=1}^{n} w_i x_i) = n\mathrm{Var}(w_i)\mathrm{Var}(x_i)$$

方差不变准则 $\mathrm{var}(y) = \mathrm{var}(x)$

$$\mathrm{Var}(w_i) = \frac{1}{n}$$

$$\mathrm{Var}(w_i) = \frac{2}{n_{in} + n_{out}}$$

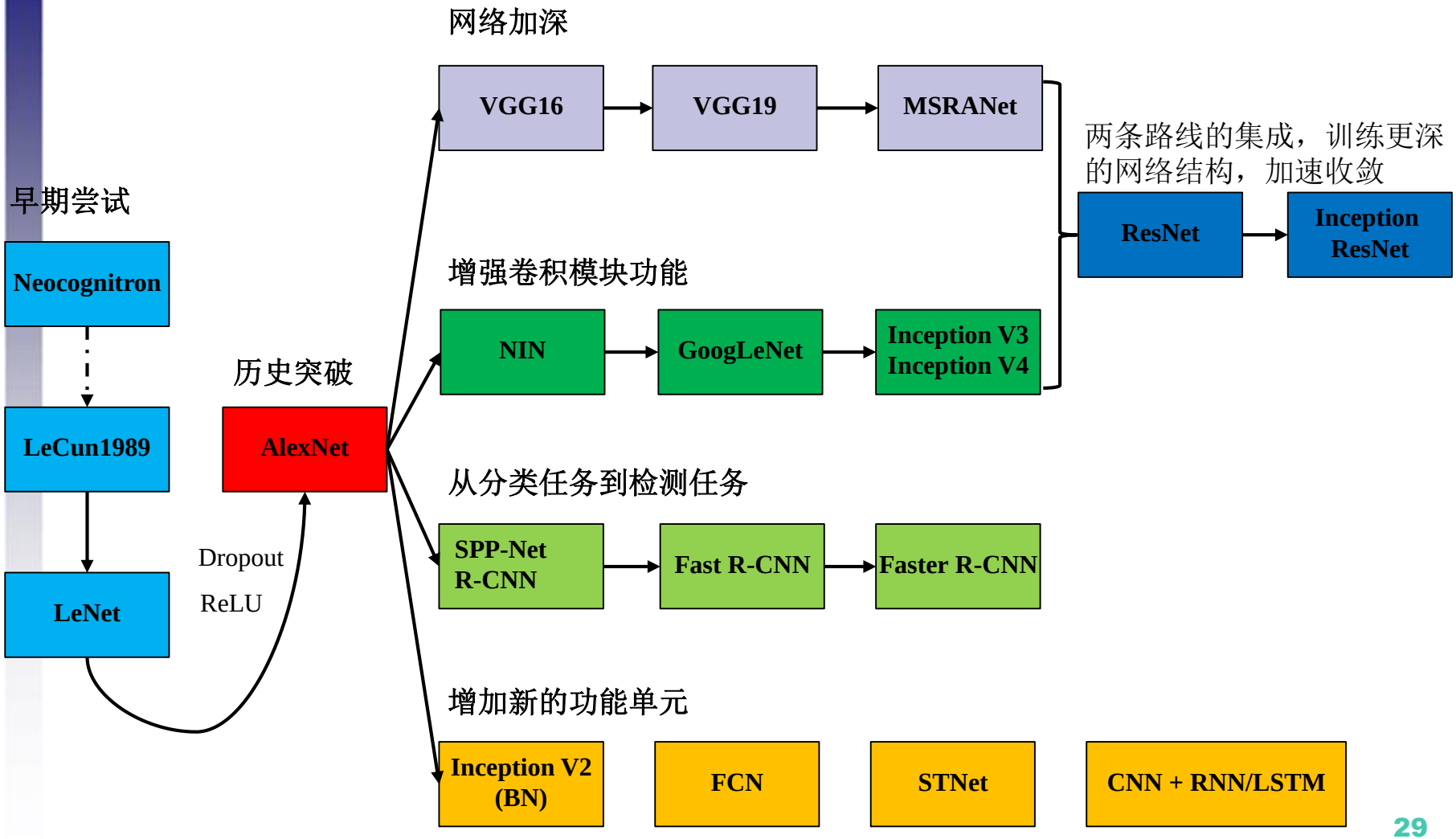
Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks[C], International conference on artificial intelligence and statistics. 2010: 249-256.

Outline

- CNN基础理论
- **CNN结构演化**
- CNN实用技巧：以Caffe为例
- CNN实战：ICCV2015年龄估计竞赛
- 前沿话题讨论

CNN结构演化

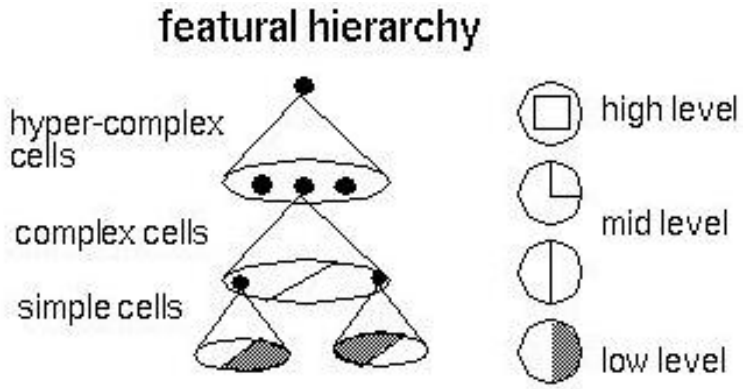
- 演化脉络



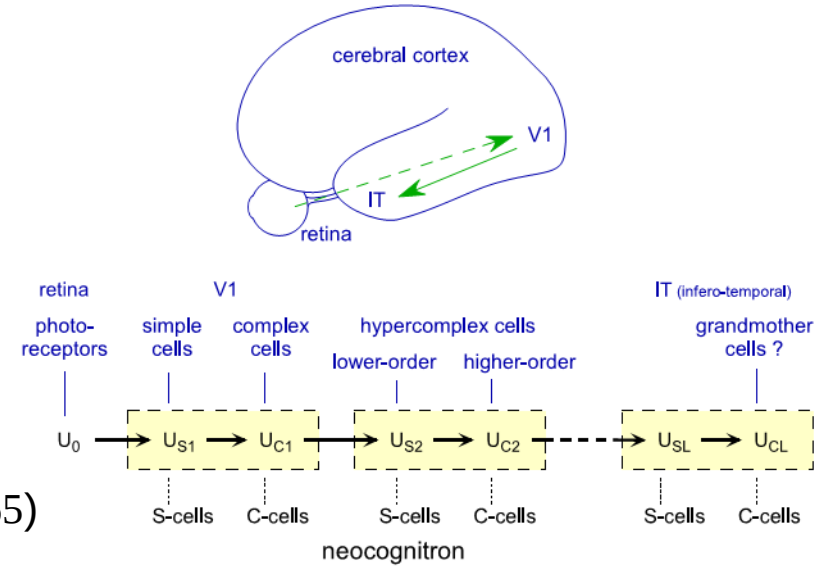
CNN结构演化—早期尝试

Kunihiko Fukushima

- **1980-Neocognitron 1980 (1/2)**



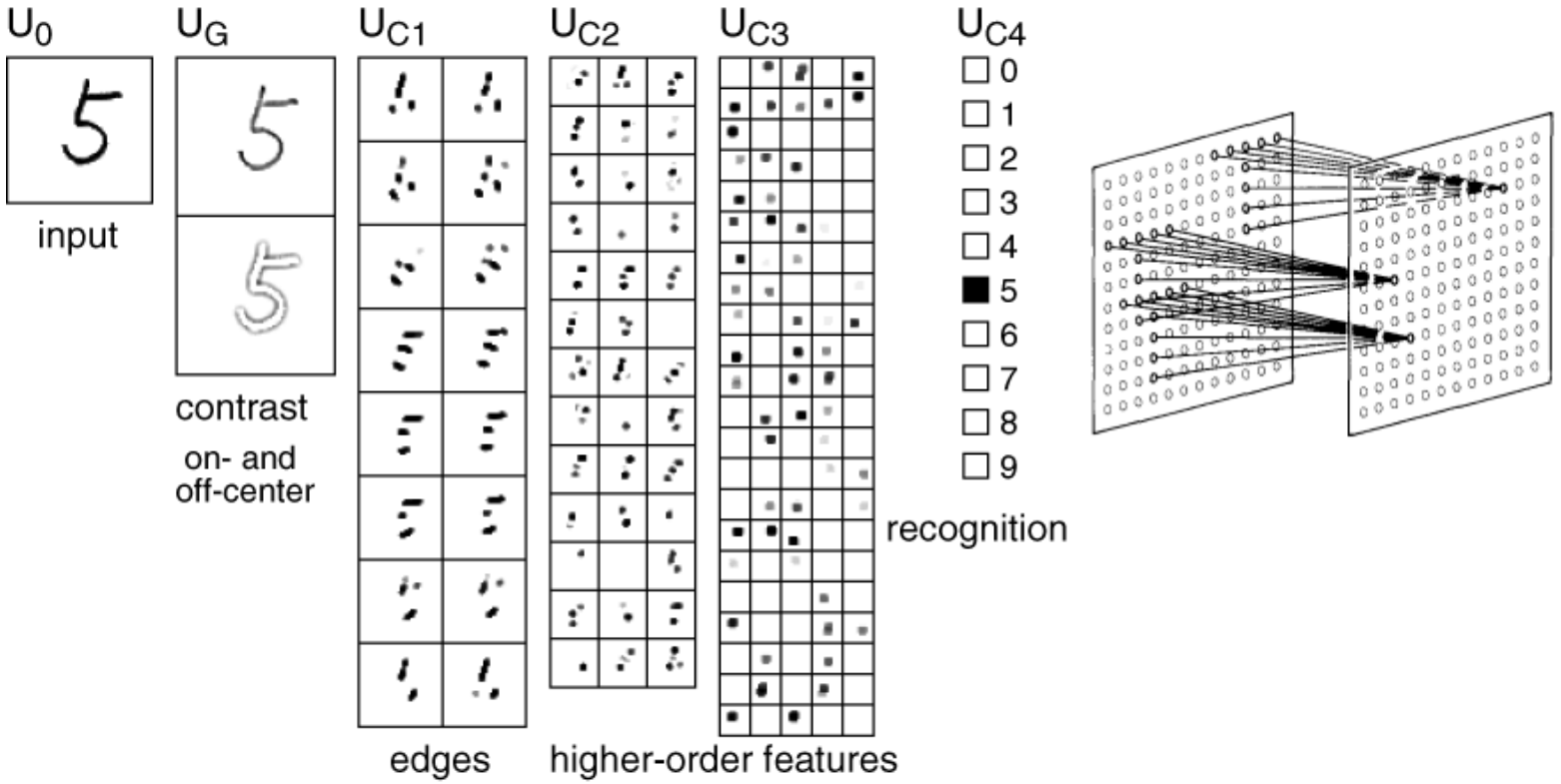
Hubel & Wiesel对视觉皮层的功能划分
Hubel, D. H. & Wiesel, T. N. (1959, 1962,1965)



K. Fukushima: "**Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position**", *Biological Cybernetics*, **36**[4] (April 1980).

CNN结构演化—早期尝试

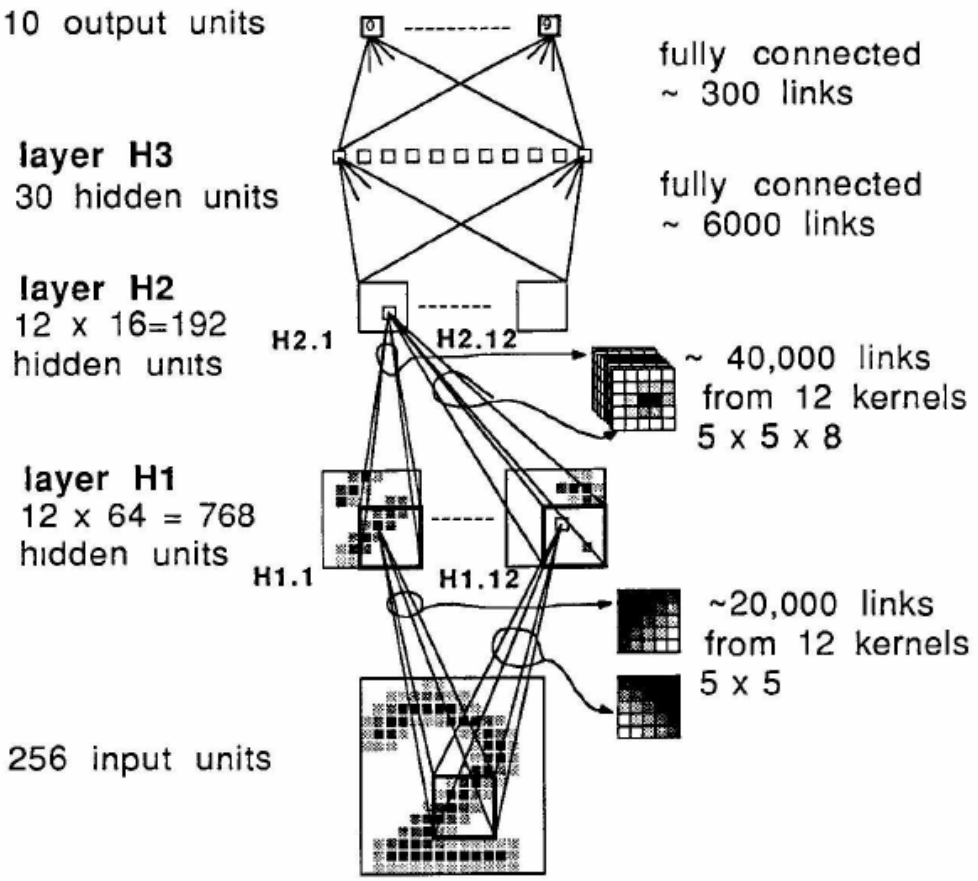
- **1980-Neocognitron (2/2)**



(a) **Handwritten digit recognition by a neocognitron.**

CNN结构演化—早期尝试

- **First Baby: LeCun1989 (1/2)**



Highlights:

1) 3个隐层，已经可以称为是deep network

2) Response map之间也共享卷积核权重

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989

CNN结构演化—早期尝试

- **First Baby: LeCun1989 (2/2)**

All simulations were performed using the backpropagation simulator SN (Bottou and LeCun 1988) running on a SUN-4/260.

The nonlinear function used at each node was a scaled hyperbolic tangent. Symmetric functions of that kind are believed to yield faster convergence, although the learning can be extremely slow if some weights are too small (LeCun 1987). The target values for the output units were chosen within the quasilinear range of the sigmoid. This prevents the weights from growing indefinitely and prevents the output units from operating in the flat spot of the sigmoid. The output cost function was the mean squared error.

Before training, the weights were initialized with random values using a uniform distribution between $-2.4/F_i$ and $2.4/F_i$ where F_i is the number of inputs (fan-in) of the unit to which the connection belongs. This technique tends to keep the total inputs within the operating range of the sigmoid.

From empirical study (supported by theoretical arguments), the stochastic gradient was found to converge much faster than the true gradient,

Highlights:

1) 正切激活函数收敛更快！

2) Sigmoid归一 + 欧氏损失！

3) 网络参数初始化！

4) SGD!

CNN结构演化—早期尝试

- **1998-LeNet**

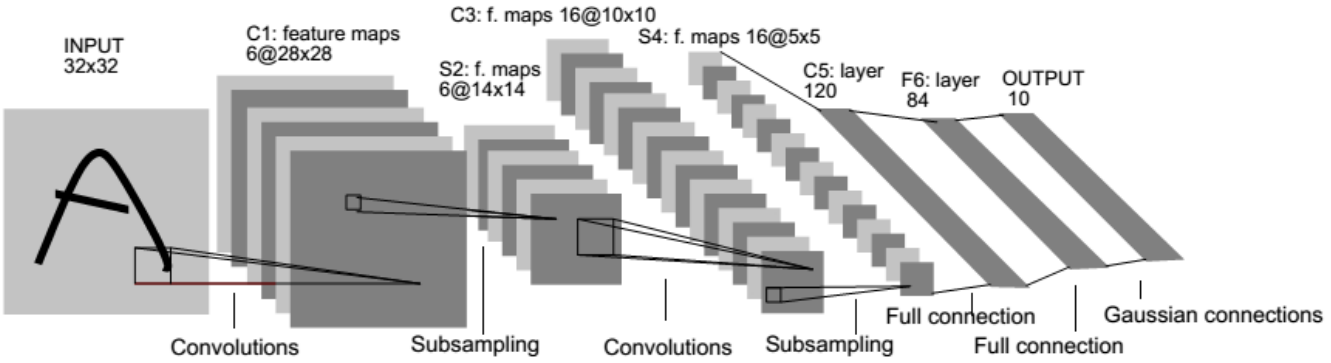


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

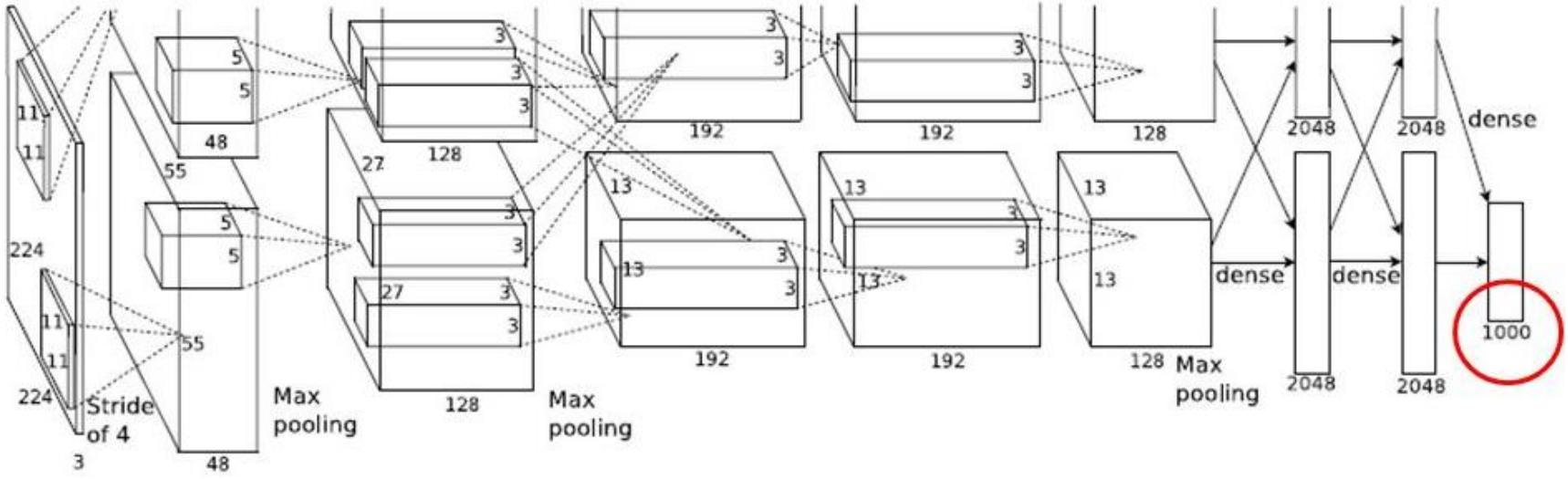
TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998

CNN结构演化—历史突破

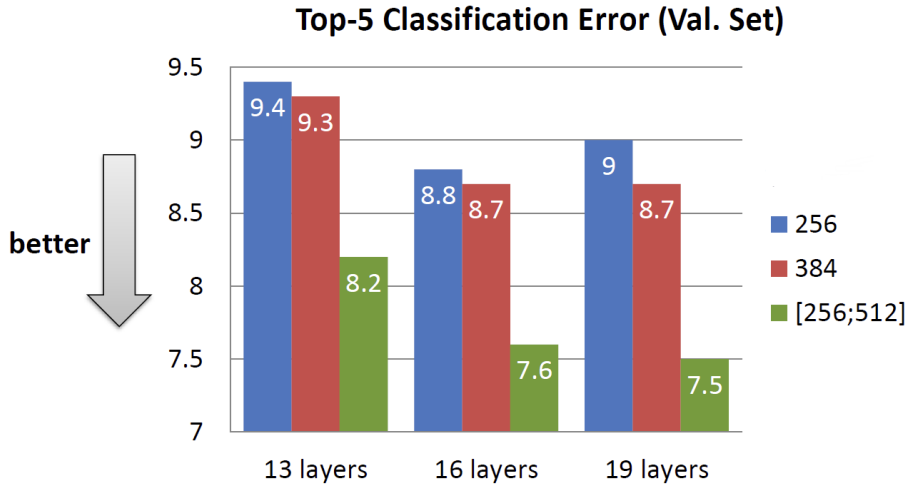
- **2012-AlexNet**
  - 非线性激活函数：ReLU
  - 防止过拟合：Dropout, 数据增广
  - 大数据训练：百万级ImageNet图像数据
  - 其他：分Group实现双GPU并行，LRN归一化层



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

CNN结构演化—网络加深

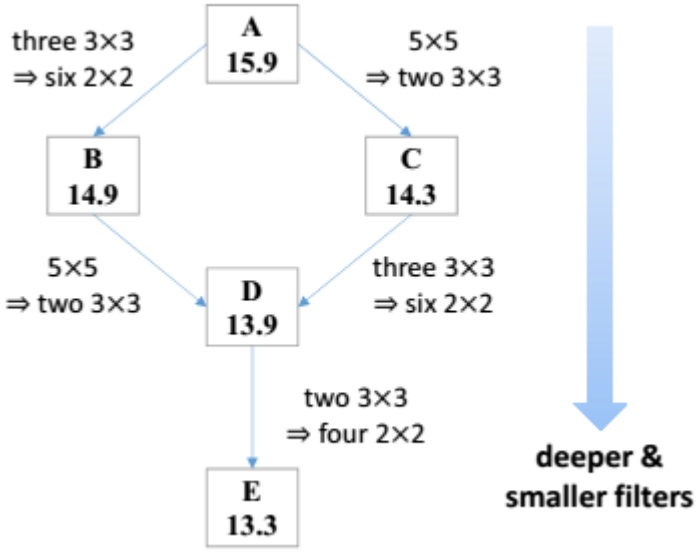
- **2014-VGG Net**
  - Slower to be deeper
    - 卷积核大小3x3
    - 卷积采样间隔1x1
    - Max Pooling间隔2x2
  - 多尺度融合



Simonyan, Karen, and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

CNN结构演化—网络加深

- **2015-MSRA-Net(1/2)**
  - 5x5拆成两个3x3层可以降低计算量，深度增加从而提高性能



Similarly, we replace the 5×5 filters by two 3×3 filters. The complexity involved is:

$$64 \cdot 5^2 \cdot 128 \tag{4}$$
$$\Rightarrow \quad 64 \cdot 3^2 \cdot 128 + 128 \cdot 3^2 \cdot 128.$$

He, Kaiming, et al, Convolutional Neural Networks at Constrained Time Cost. CVPR2015

CNN结构演化—网络加深

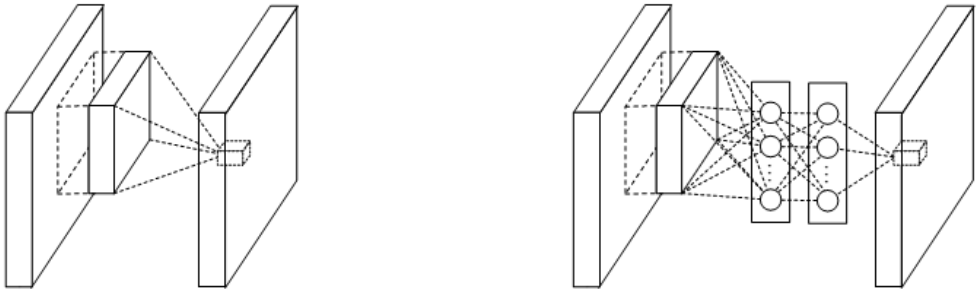
- **2015-MSRA-Net(2/2)**

input size	VGG-19 [25]	model A	model B	model C
224	3×3, 64 3×3, 64 2×2 maxpool, /2	7×7, 96, /2	7×7, 96, /2	7×7, 96, /2
112	3×3, 128 3×3, 128 2×2 maxpool, /2	2×2 maxpool, /2	2×2 maxpool, /2	2×2 maxpool, /2
56	3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 maxpool, /2	3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 maxpool, /2	3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 maxpool, /2	3×3, 384 3×3, 384 3×3, 384 3×3, 384 3×3, 384 3×3, 384 2×2 maxpool, /2
28	3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 768 3×3, 768 3×3, 768 3×3, 768 3×3, 768 3×3, 768 2×2 maxpool, /2
14	3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 maxpool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 spp, {7, 3, 2, 1}	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 spp, {7, 3, 2, 1}	3×3, 896 3×3, 896 3×3, 896 3×3, 896 3×3, 896 3×3, 896 spp, {7, 3, 2, 1}
fc_1 fc_2 fc_3	4096 4096 1000			
depth (conv+fc)	19	19	22	22
complexity (ops., $\times 10^{10}$)	1.96	1.90	2.32	5.30

Table 3. Architectures of large models. Here "/2" denotes a stride of 2.

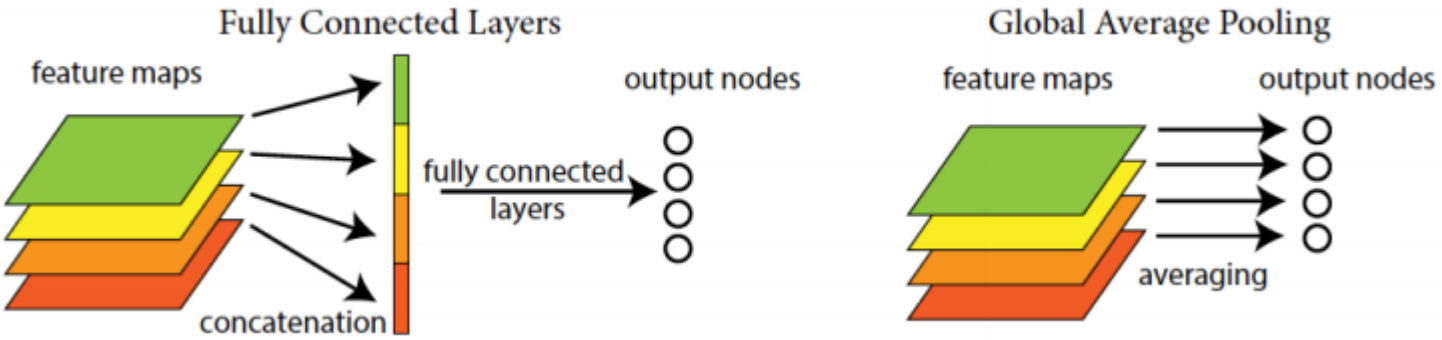
CNN结构演化—增强卷积模块功能

- **2013-Network in Network(1/2)**
  - **MLP网络替换线性卷积**



(a) Linear convolution layer　　(b) Mlpconv layer

  - **去掉全连接层，使用Global Average Pooling**

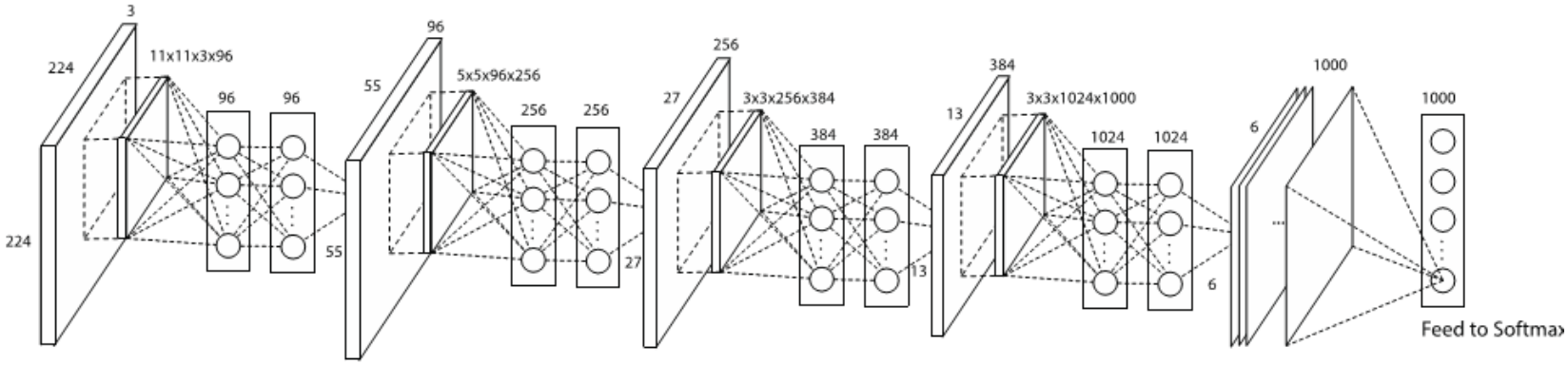


Min Lin et al, Network in Network, Arxiv 2013.

CNN结构演化—增强卷积模块功能

- **2013-Network in Network(2/2)**

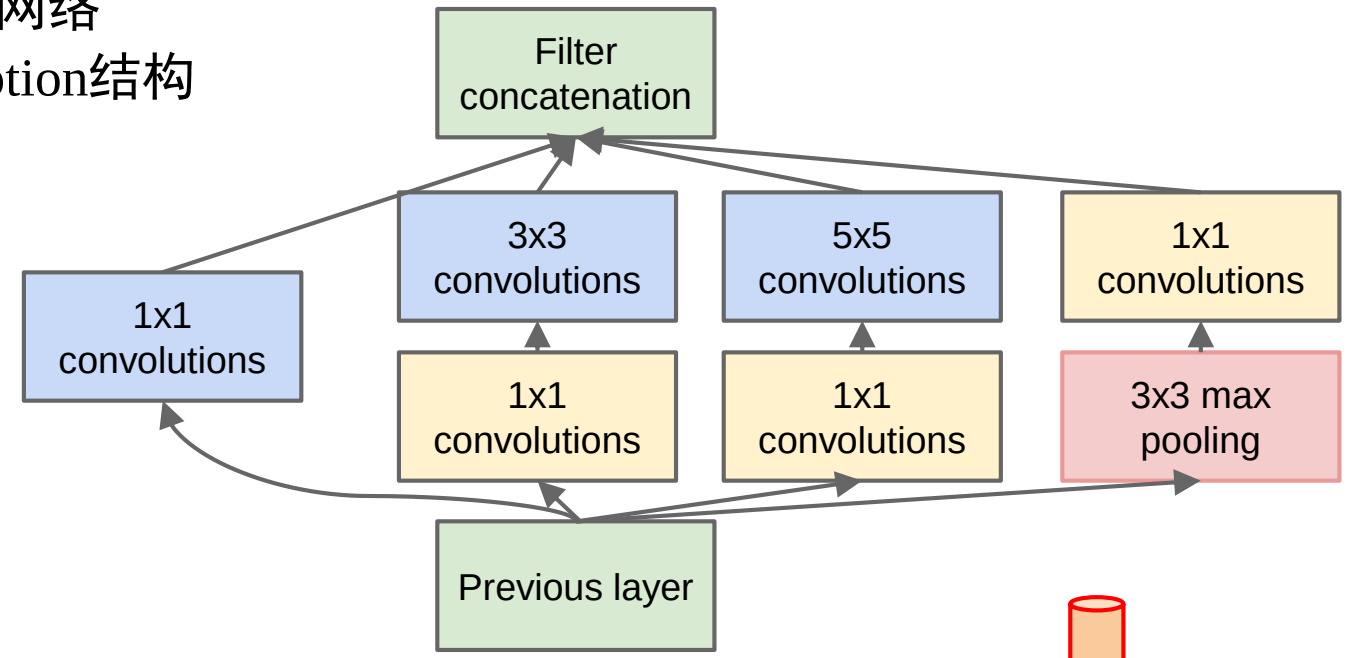
A simple 4 layer NIN + Global Average Pooling:

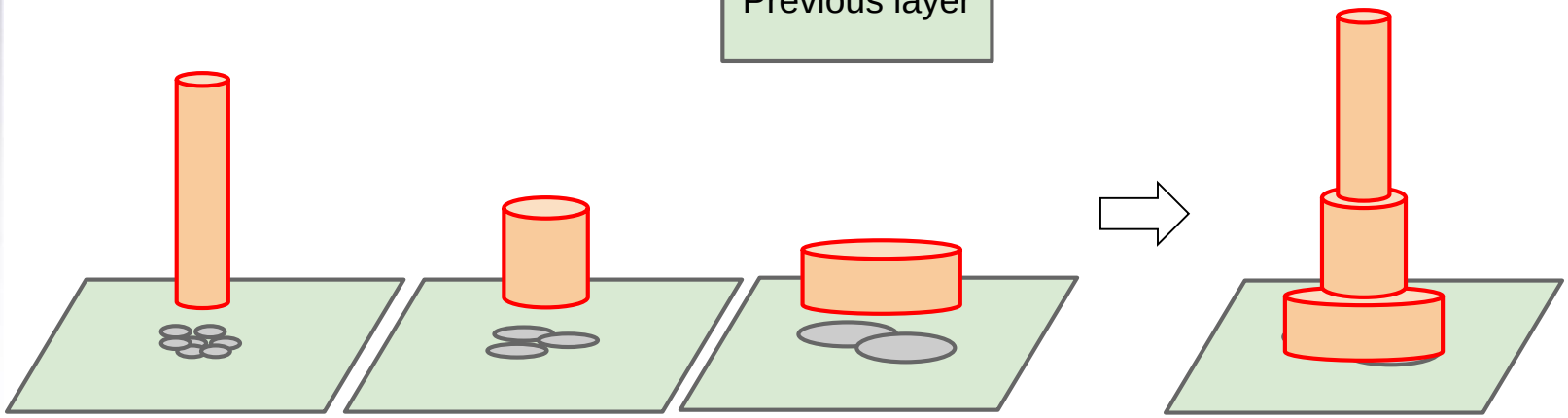


	Parameter Number	Performance	Time to train (GTX Titan)
AlexNet	60 Million (230 Megabytes)	40.7% (Top 1)	8 days
NIN	7.5 Million (29 Megabytes)	39.2% (Top 1)	4 days

CNN结构演化—增强卷积模块功能

- **2014-GoogLeNet(1/2)**
  - 24层网络
  - Inception结构



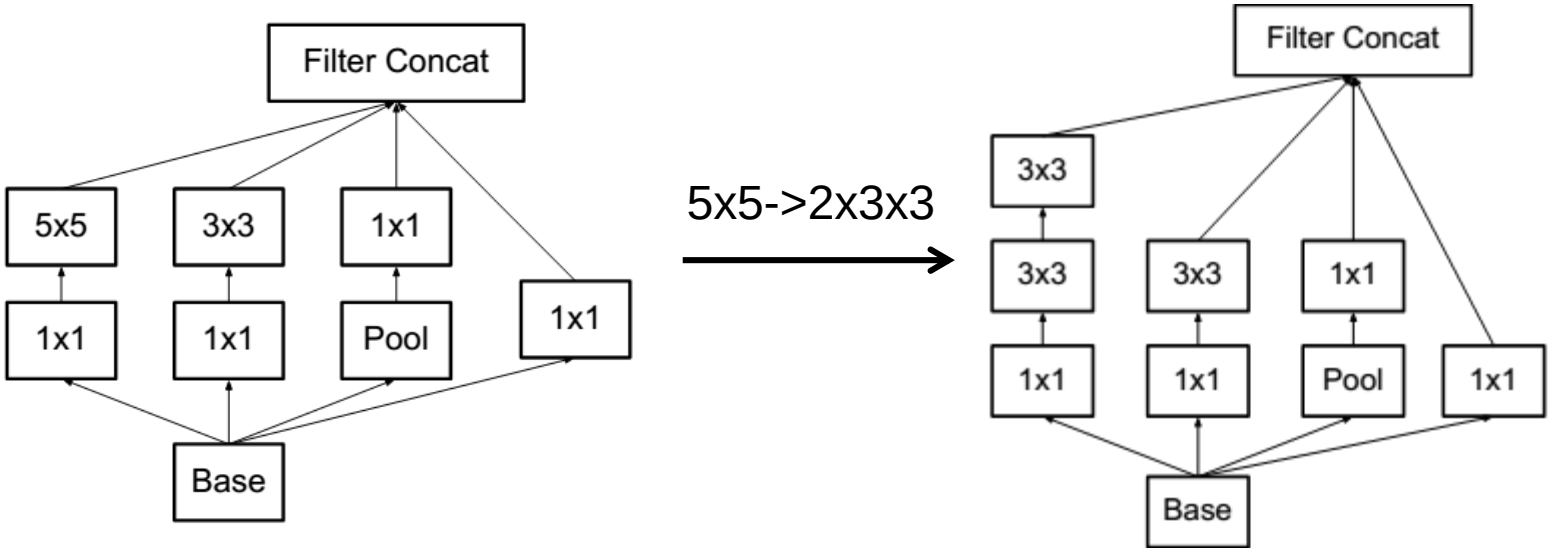


CNN结构演化—增强卷积模块功能

- **2014-GoogLeNet(2/2)**
  - 超大规模24层网络
  - 多个Inception结构串联
  - 两个辅助损失层

CNN结构演化—增强卷积模块功能

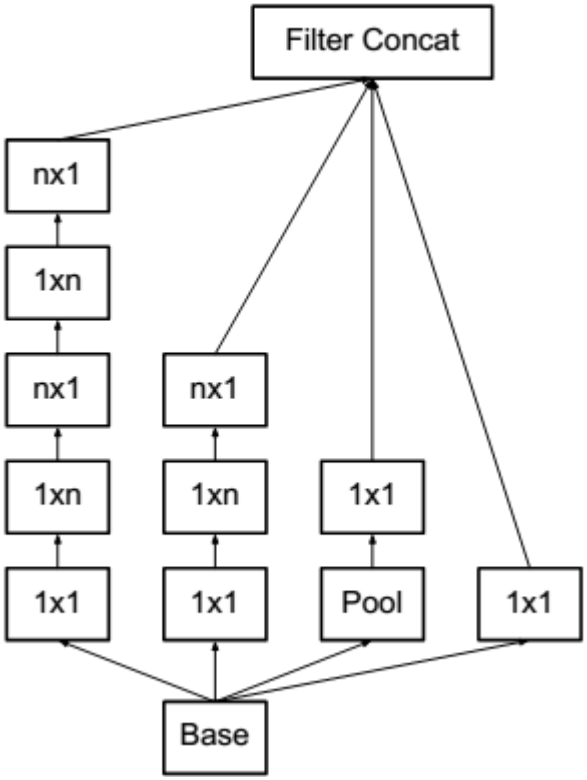
- **2015-Inception V3(1/2)**
  - 更深的Inception结构
  - 5x5卷积核拆成两层3x3卷积核

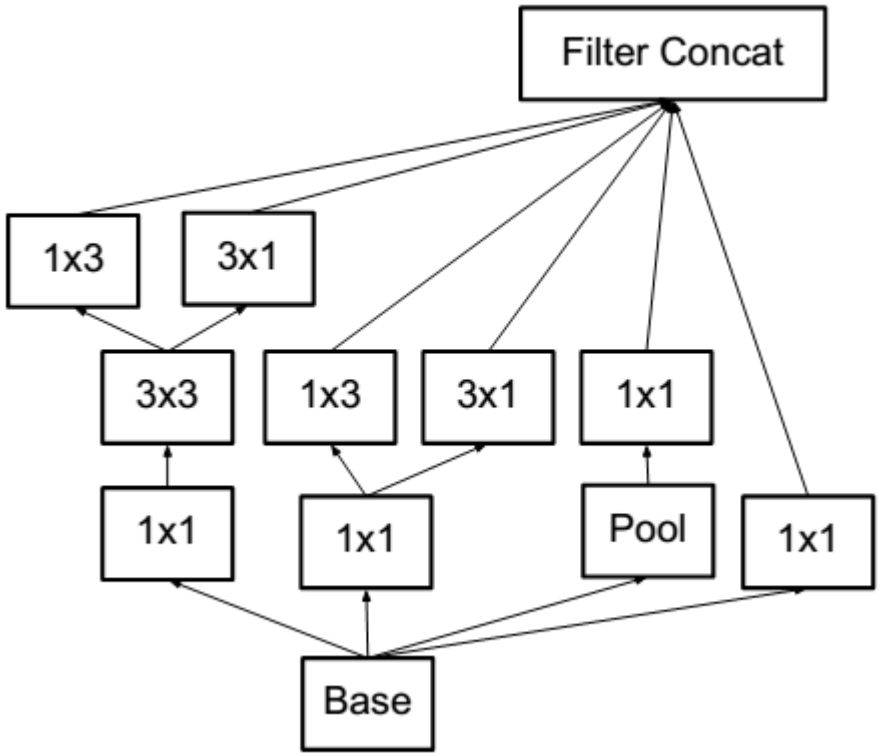


Szegedy, Christian, et al. " Rethinking the Inception Architecture for Computer Vision." *arXiv preprint 2015*

CNN结构演化—增强卷积模块功能

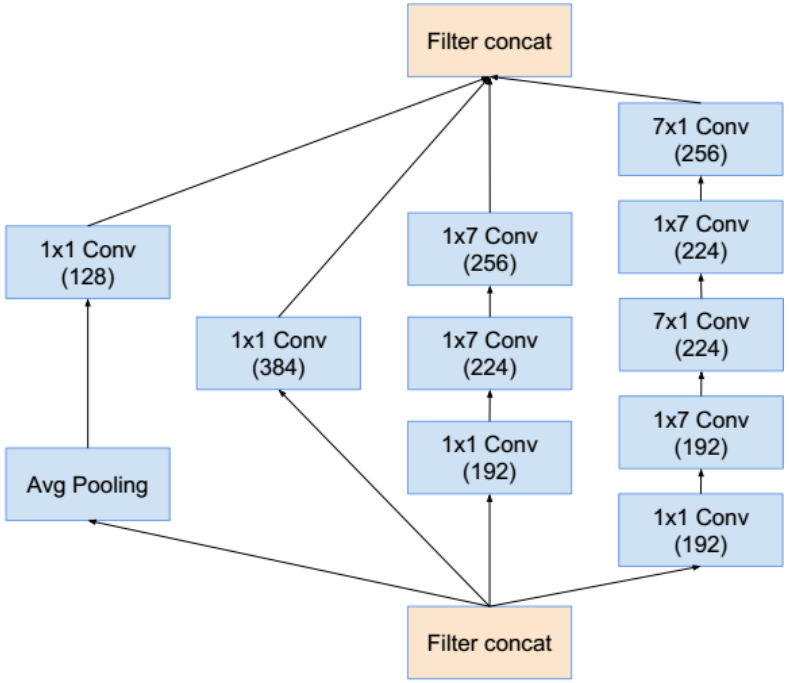
- **2015-Inception V3(2/2)**
  - 加入1 x n和 n x 1的卷积核

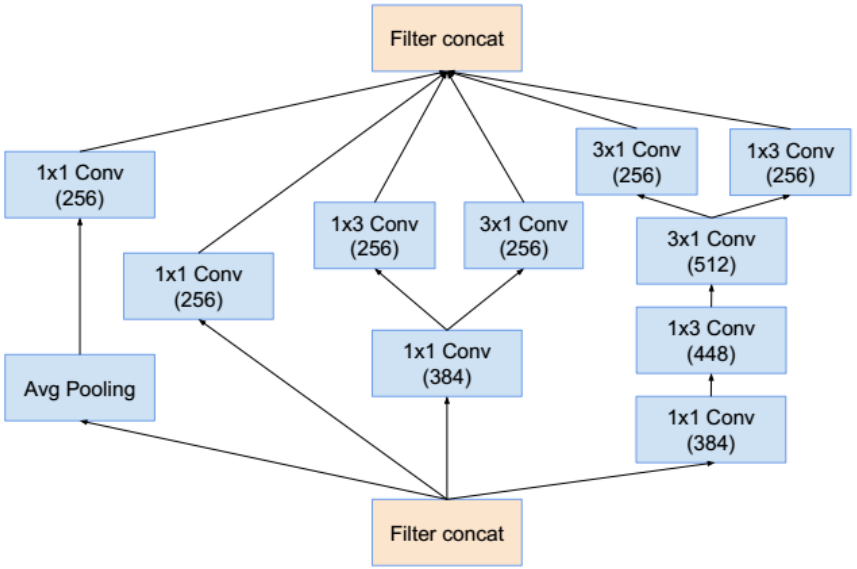




CNN结构演化—增强卷积模块功能

- **2015-Inception V4**
  - 更深更复杂的Inception结构



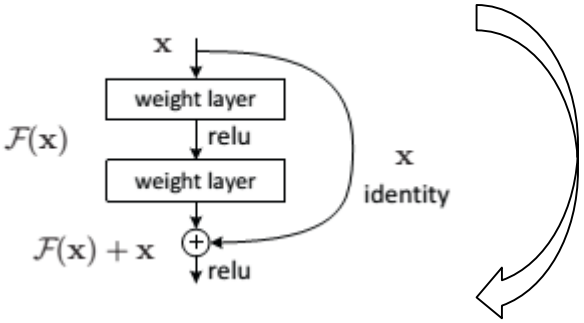


Inception V4

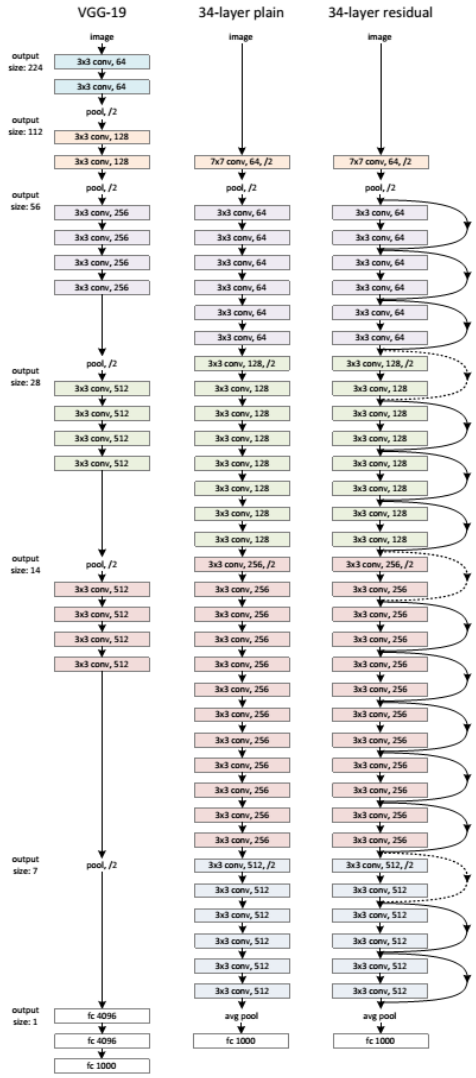
Szegedy C, Ioffe S, Vanhoucke V. Inception-v4, inception-resnet and the impact of residual connections on learning[J]. arXiv preprint arXiv:1602.07261, 2016.

CNN结构演化—集成路线

- **2015-Deep Residual Net**



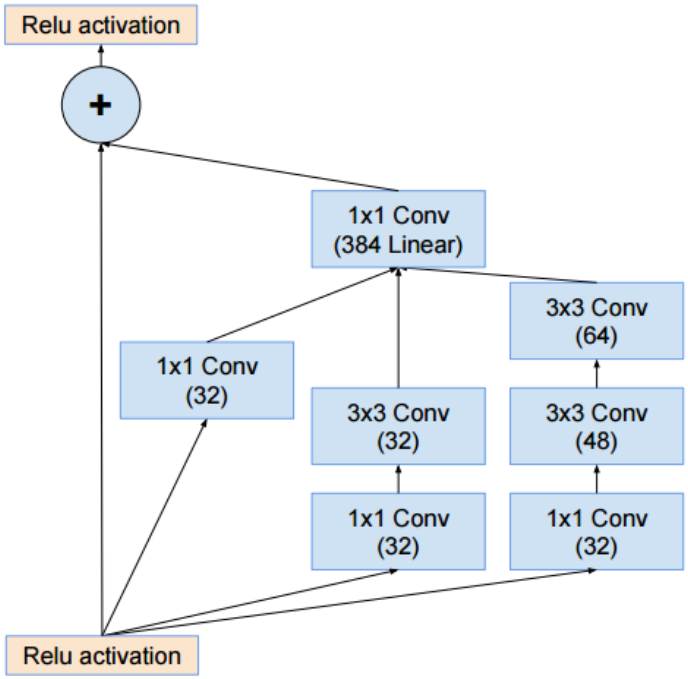
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

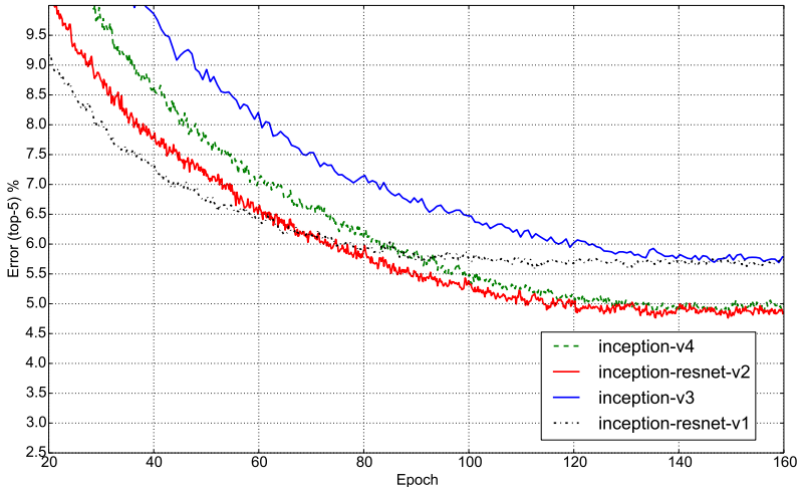


He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. arXiv preprint arXiv:1512.03385, 2015.

CNN结构演化—集成路线

- **2016-Inception Residual Net**
  - 主要结论：Residual Net可以加速收敛，用Inception结构也可以达到可比的性能





ImageNet Top-5 Error收敛曲线

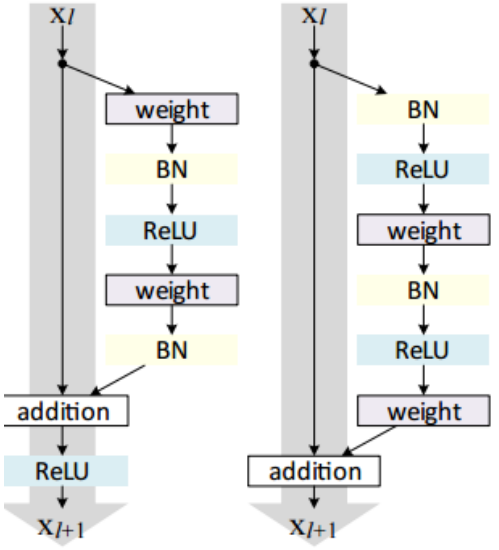
ImageNet Top-5 Error对比

Network	Top-1 Error	Top-5 Error
BN-Inception [6]	25.2%	7.8%
Inception-v3 [15]	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

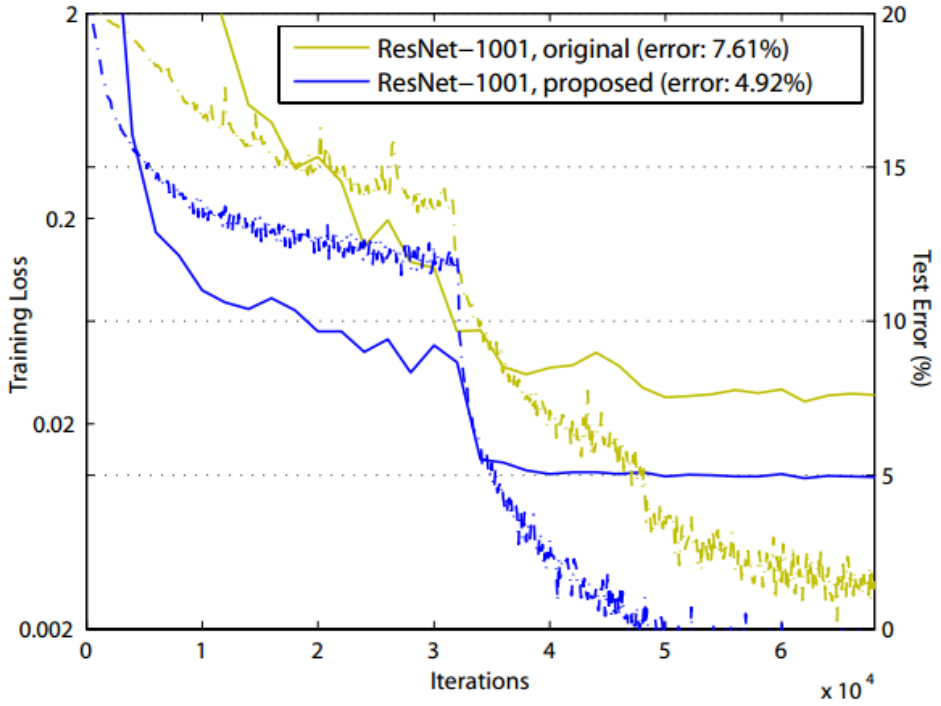
Szegedy C, Ioffe S, Vanhoucke V. Inception-v4, inception-resnet and the impact of residual connections on learning[J]. arXiv preprint arXiv:1602.07261, 2016.

CNN结构演化—集成路线

- **2016-Identity Mapping Residual Net**
  - 去掉Addition后面的ReLU操作
  - 残差项跨层可加



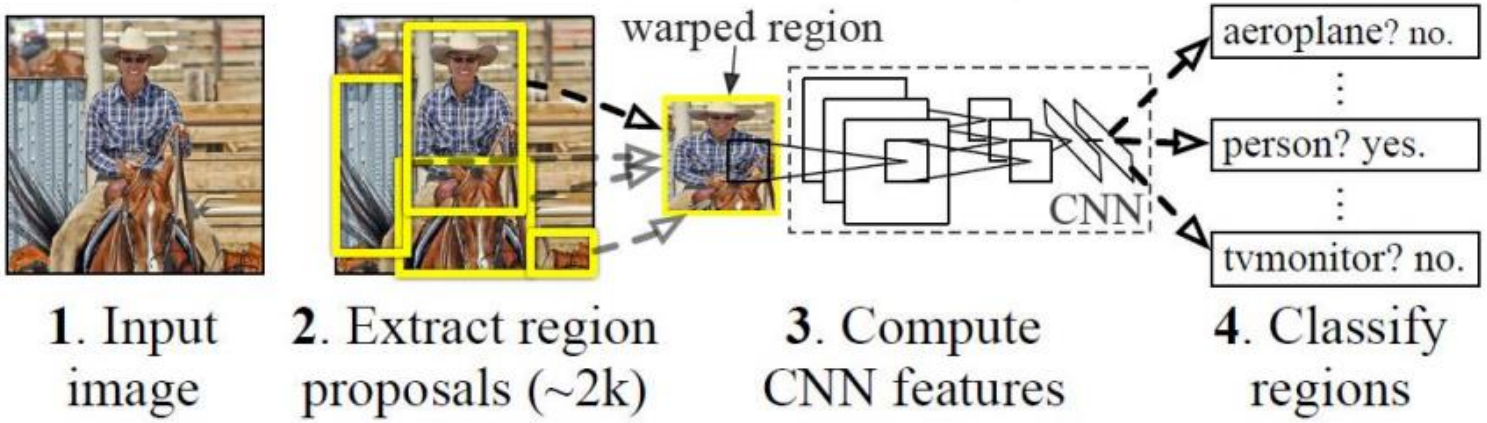
(a) original (b) proposed



He K, Zhang X, Ren S, et al. Identity Mappings in Deep Residual Networks[J]. arXiv preprint arXiv:1603.05027, 2016.

CNN结构演化—从分类到检测

- **R-CNN** Ross Girshick' CVPR14



RCNN用于目标检测的流程：

1）输入图像

2）利用Selective Search (fast mode)提取大约2000个左右的region proposal

3）对于每一个region proposal我们先将其resize成227x227输入ImageNet图像分类网络

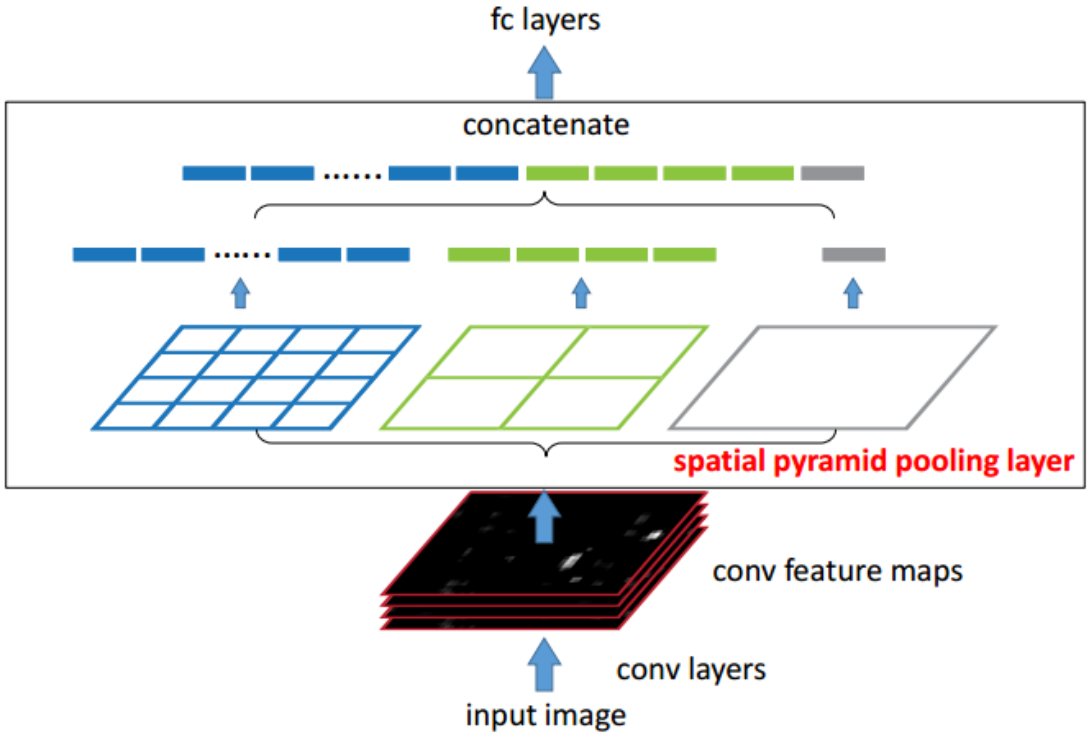
4）将提取到的每个region proposal的特征输入SVM分类器进行分类

缺点1：需要保留大量的中间特征， 5K图像数百G特征文件

缺点2：每个region proposal单独处理，速度慢，GPU上47s/张图像

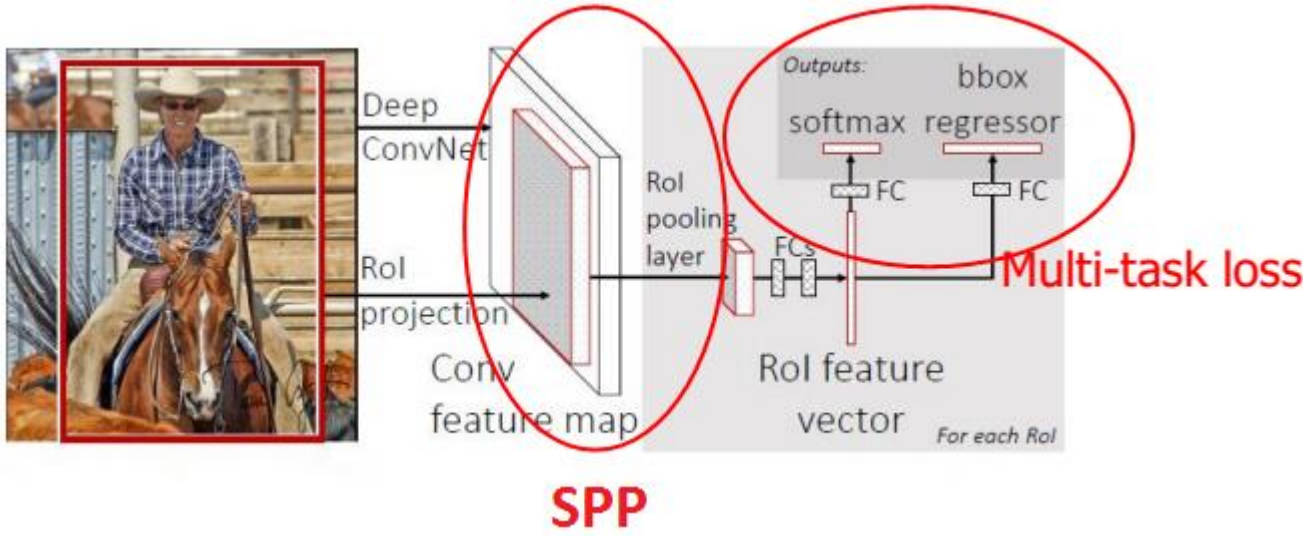
CNN结构演化—从分类到检测

- **SPP-Net** Kaiming He's ECCV14
  - CNN经典结构要求固定尺寸的输入图像，可能损失信息
  - 利用Spatial Pyramid Pooling （SPP），任意尺寸的响应图可以得到等长的特征
  - 多层次Pooling: 对形变的鲁棒性



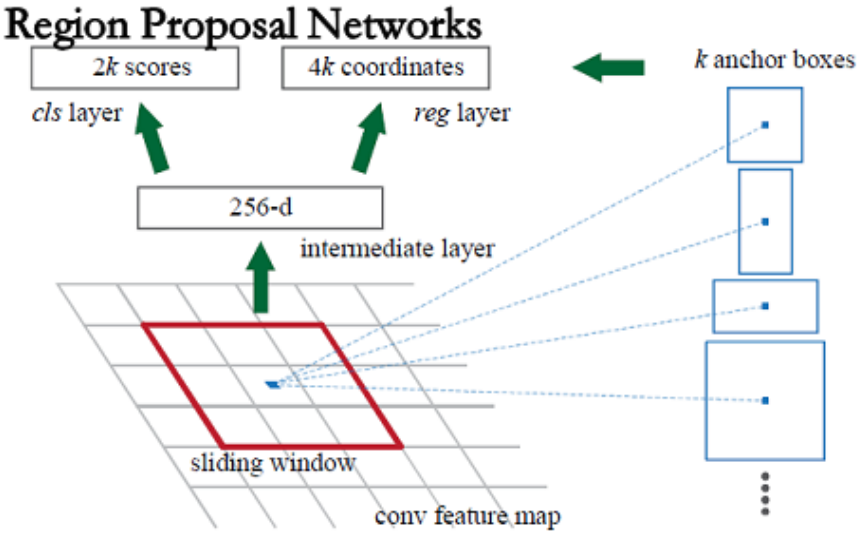
CNN结构演化—从分类到检测

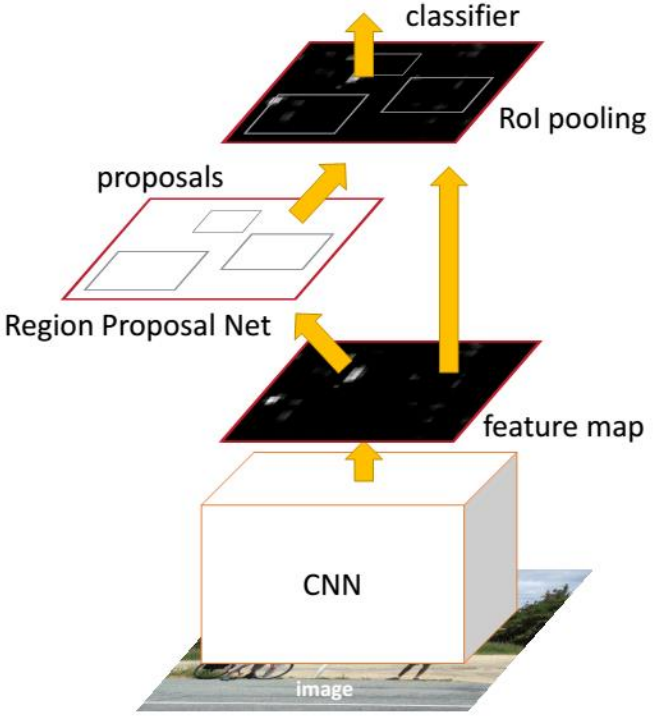
- **Fast R-CNN** **Ross Girshick ICCV15**
  - 利用Spatial Pyramid Pooling （SPP）进行加速，整张图像只需要通过卷积网络一次
  - 分类+窗口回归在统一框架中训练，不需要保留中间结果
  - 速度从RCNN的46s加速到2~3s（主要时间花在region proposal)



CNN结构演化—从分类到检测

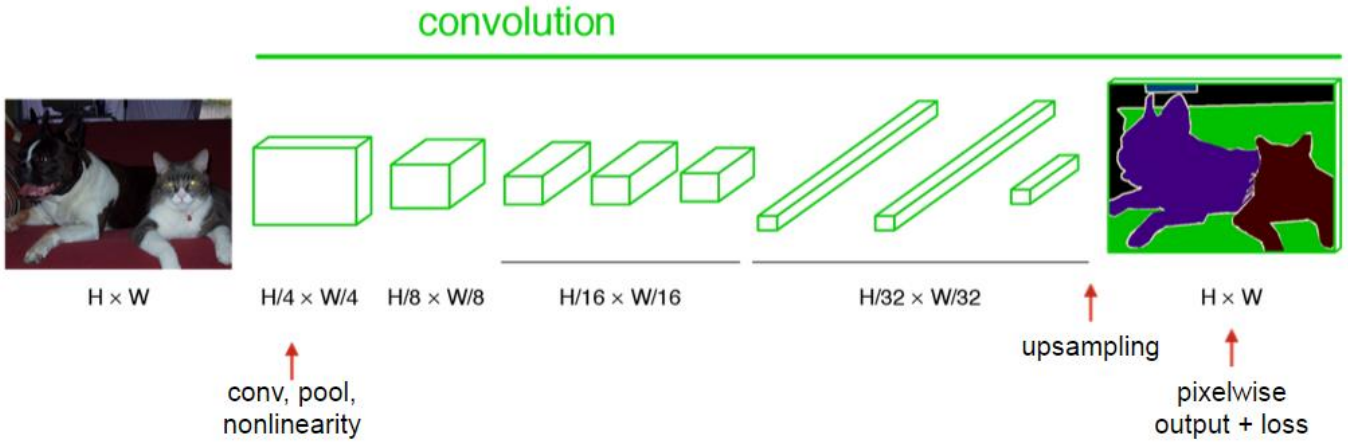
- **Faster R-CNN** Shaoqing Ren' NIPS15
  - **Faster R-CNN = RPN + Fast R-CNN**
  - 用RPN网络代替Edge box等Region Proposal 方法，共享了RPN和Fast R-CNN卷积层参数，支持端到端训练
  - 速度显著提高，GPU 200ms /张图像





CNN结构演化—新的功能模块

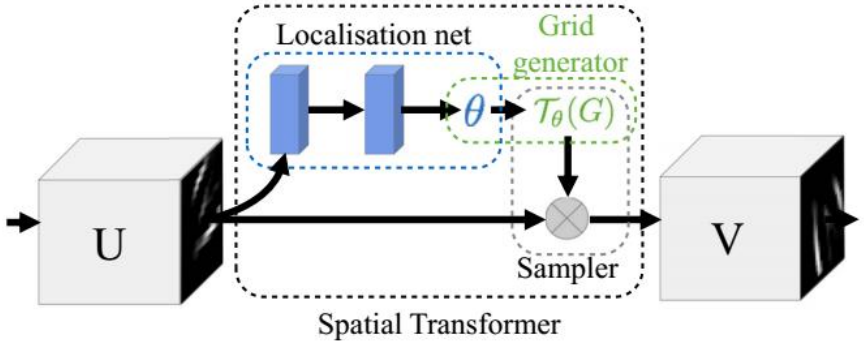
- **FCN** Jon Long's CVPR15
  - **结构特点**：
  1）支持任意长宽比图像，使用1x1卷积核替代全连接保持空间关系
  2）利用反卷积层进行上采样
  3）对输入图像的每个像素进行分类（Softmax Loss)

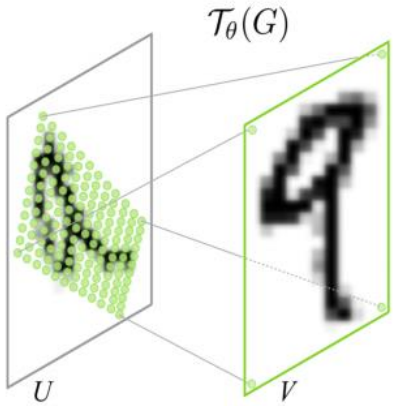


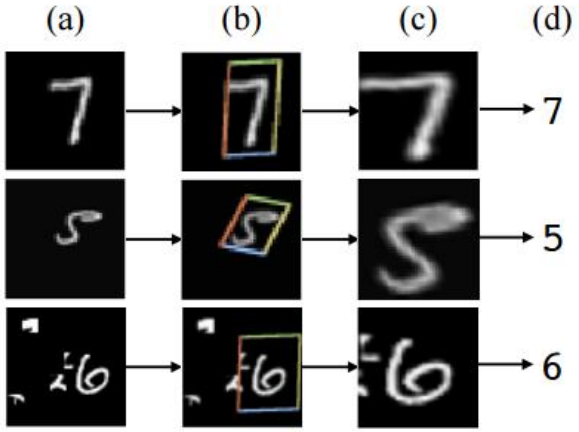
  - **训练机制**：基于ImageNet图像分类模型fine-tune, base_lr一般需较小，例如10^{-6}

CNN结构演化—新的功能模块

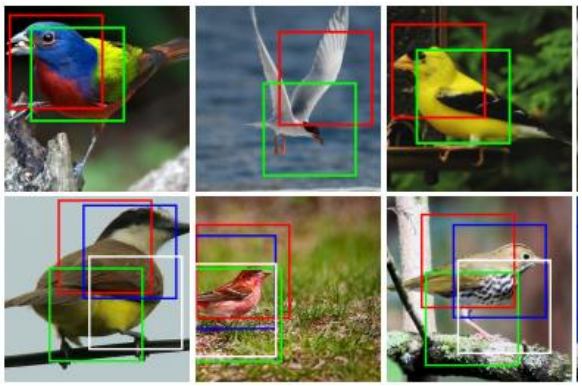
- **End to End图像对齐网络ST-Net**
  - 通过子网络估计图像仿射变换参数，端到端的完成图像校正







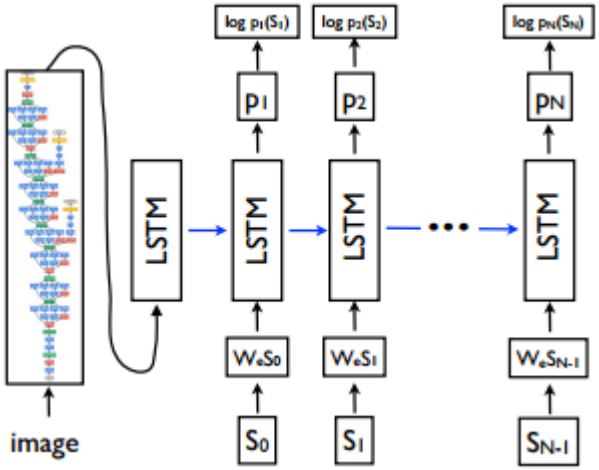
Model	
Cimpoi '15 [5]	66.7
Zhang '14 [40]	74.9
Branson '14 [3]	75.7
Lin '15 [23]	80.9
Simon '15 [30]	81.0
CNN (ours) 224px	82.3
2×ST-CNN 224px	83.1
2×ST-CNN 448px	83.9
4×ST-CNN 448px	**84.1**

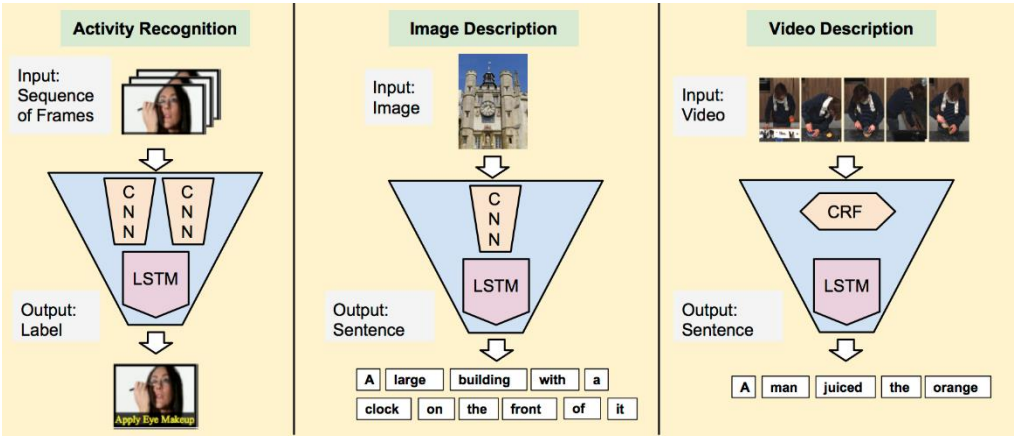


Jaderberg M, Simonyan K, Zisserman A. Spatial transformer networks, NIPS15

CNN结构演化—新的功能模块

- **CNN + RNN/LSTM**
  - CNN学表示，RNN/LSTM建模时序信号
  - 应用：Image Caption, Image QA, Video Representation





Show and Tell: A Neural Image Caption Generator (a work from Google)

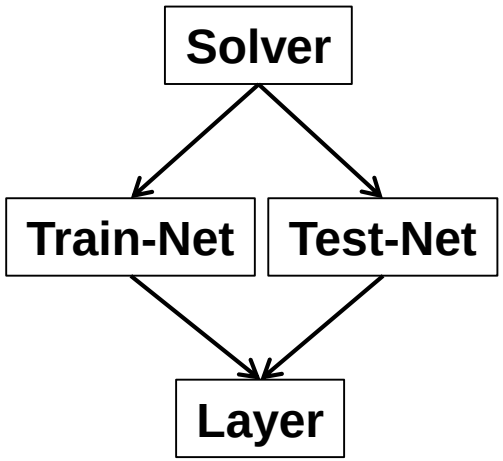
Long-term Recurrent Convolutional Networks for Visual Recognition and Description (A work from UTA/UCB/UML)

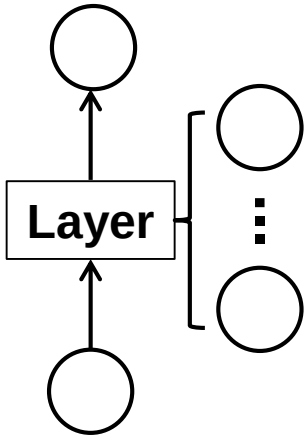
Outline

- CNN基础理论
- CNN结构演化
- **CNN实用技巧：以Caffe为例**
- CNN实战：ICCV2015年龄估计竞赛
- 前沿话题讨论

CNN实用技巧—Caffe整体架构

- **核心概念**
  - caffe.proto 公共数据结构定义
  - Solver：求解器：负责网络参数更新
  - Net：网络结构
  - Layer：结构单元
  - Blob：物理数据结构





四维矩阵：

$N \times C \times H \times W$

$N: Number$

$C: Channel$

$H: Height$

$W: Width$

CNN实用技巧—PyCaffe

- Python定义网络结构
  - 定义LeNet

```
def lenet(batch_size):
    n = caffe.NetSpec()
    n.data, n.label = L.DummyData(shape=[dict(dim=[batch_size, 1, 28, 28]),
                                         dict(dim=[batch_size, 1, 1, 1])],
                                  transform_param=dict(scale=1./255), ntop=2)
    n.conv1 = L.Convolution(n.data, kernel_size=5, num_output=20,
        weight_filler=dict(type='xavier'))
    n.pool1 = L.Pooling(n.conv1, kernel_size=2, stride=2, pool=P.Pooling.MAX)
    n.conv2 = L.Convolution(n.pool1, kernel_size=5, num_output=50,
        weight_filler=dict(type='xavier'))
    n.pool2 = L.Pooling(n.conv2, kernel_size=2, stride=2, pool=P.Pooling.MAX)
    n.ip1 = L.InnerProduct(n.pool2, num_output=500,
        weight_filler=dict(type='xavier'))
    n.relu1 = L.ReLU(n.ip1, in_place=True)
    n.ip2 = L.InnerProduct(n.relu1, num_output=10,
        weight_filler=dict(type='xavier'))
    n.loss = L.SoftmaxWithLoss(n.ip2, n.label)
    return n.to_proto()
```

配置文件: caffe.proto

```
message InnerProductParameter {
  optional uint32 num_output = 1; // The number
  optional bool bias_term = 2 [default = true];
  optional FillerParameter weight_filler = 3; //
  optional FillerParameter bias_filler = 4; // Th

  // The first axis to be lumped into a single in
  // all preceding axes are retained in the outpu
  // May be negative to index from the end (e.g.
  optional int32 axis = 5 [default = 1];
  // Specify whether to transpose the weight matr
  // If transpose == true, any operations will be
  // of the weight matrix. The weight matrix itse
  // but rather the transfer flag of operations w
  optional bool transpose = 6 [default = false];
}
```

入口参数

https://github.com/BVLC/caffe/blob/master/python/caffe/test/test_net_spec.py

CNN实用技巧—PyCaffe

中科院计算所

- Python定义网络结构
  - 定义CaffeNet的building block

```
def conv_relu(bottom, ks, nout, stride=1, pad=0, group=1):
    conv = L.Convolution(bottom, kernel_size=ks, stride=stride,
                                num_output=nout, pad=pad, group=group)
    return conv, L.ReLU(conv, in_place=True)

def fc_relu(bottom, nout):
    fc = L.InnerProduct(bottom, num_output=nout)
    return fc, L.ReLU(fc, in_place=True)
```

  - 定义Residual Net的building block

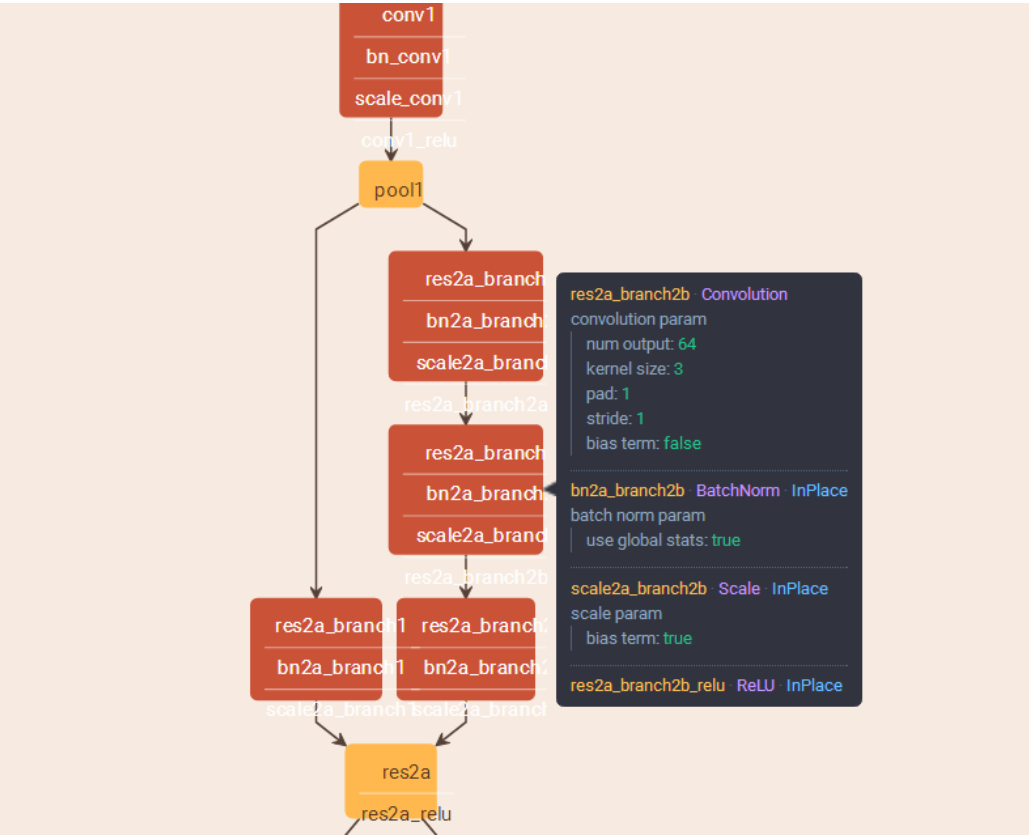
```
# for resnet v2 unit
def bn_scale_relu_conv(bottom, ks, nout, stride=1, pad=0):
    bn = L.BatchNorm(bottom, in_place=True)
    scale = L.Scale(bn, in_place=True)
    relu = L.ReLU(scale, in_place=True)
    conv = L.Convolution(relu, kernel_size=ks, stride=stride, num_output=nout, pad=pad,
                weight_filler=dict(type='xavier'), bias_filler=dict(type='constant', value=0))
    return conv

def res_v2_unit(bottom, ks, nout, stride=1, pad=1):
    conv1 = bn_scale_relu_conv(bottom, ks, nout, stride=stride, pad=pad)
    conv2 = bn_scale_relu_conv(conv1 , ks, nout, stride=stride, pad=pad)
    sum = L.Eltwise(conv2, bottom, operation=P.Eltwise.SUM)
    return sum
```

https://github.com/BVLC/caffe/blob/master/examples/pycaffe/caffenet.py

CNN实用技巧—PyCaffe

- Python定义网络结构
  - 一个可视化网络结构的在线工具

http://ethereon.github.io/netscope/#/editor



CNN实用技巧—特殊功能层

- **特殊功能层**
  - Split层

功能：当一个Blob作为两个或以上Layer的bottom时对梯度进行相加

使用场景：Caffe框架自动控制

  - Slice层

功能：Blob按维度切片

使用场景：Contrastive Loss中对Pairwise数据进行拆分

  - Concat层

功能：多个Blob在指定维度连接

使用场景：Inception结构中不同尺度卷积核结果的串联

  - Eltwise层

功能: 逐元素计算

使用场景：Residual Net中实现short-cut连接的相加操作

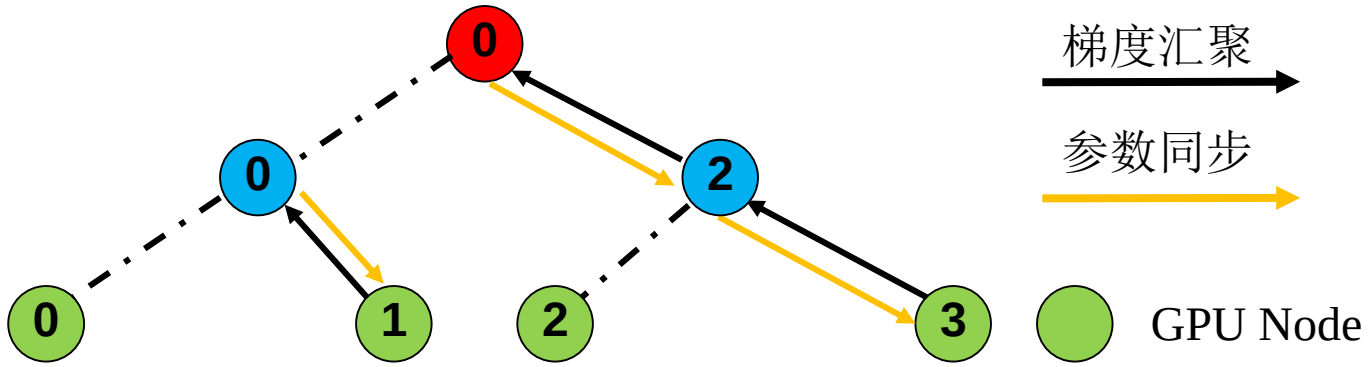
  - Scale层

功能: BN中的scale与shift操作

适用场景：在BatchNorm层之后实用，实现完整的BN操作

CNN实用技巧—Caffe单机多卡实现

- **Multi-GPU扩展**
  - 实现方式：同步模式，每轮迭代按树形结构交换梯度



  - 多GPU服务器性能瓶颈
    - GPU之间通讯方式：P2P / SOC
    - 如果不支持P2P，则需要依赖于socket连接，存在I/O瓶颈
    - 可以用nvidia-smi topo –m命令查看

	GPU0	GPU1	GPU2	GPU3
GPU0	X	PIX	SOC	SOC
GPU1	PIX	X	SOC	SOC
GPU2	SOC	SOC	X	PIX
GPU3	SOC	SOC	PIX	X

```
X   = Self
SOC = Path traverses a socket-level link (e.g. QPI)
PHB = Path traverses a PCIe host bridge
PXB = Path traverses multiple PCIe internal switches
PIX = Path traverses a PCIe internal switch
```

Outline

- CNN基础理论
- CNN结构演化
- CNN实用技巧：以Caffe为例
- **CNN实战：ICCV2015年龄估计竞赛**
- 前沿话题讨论

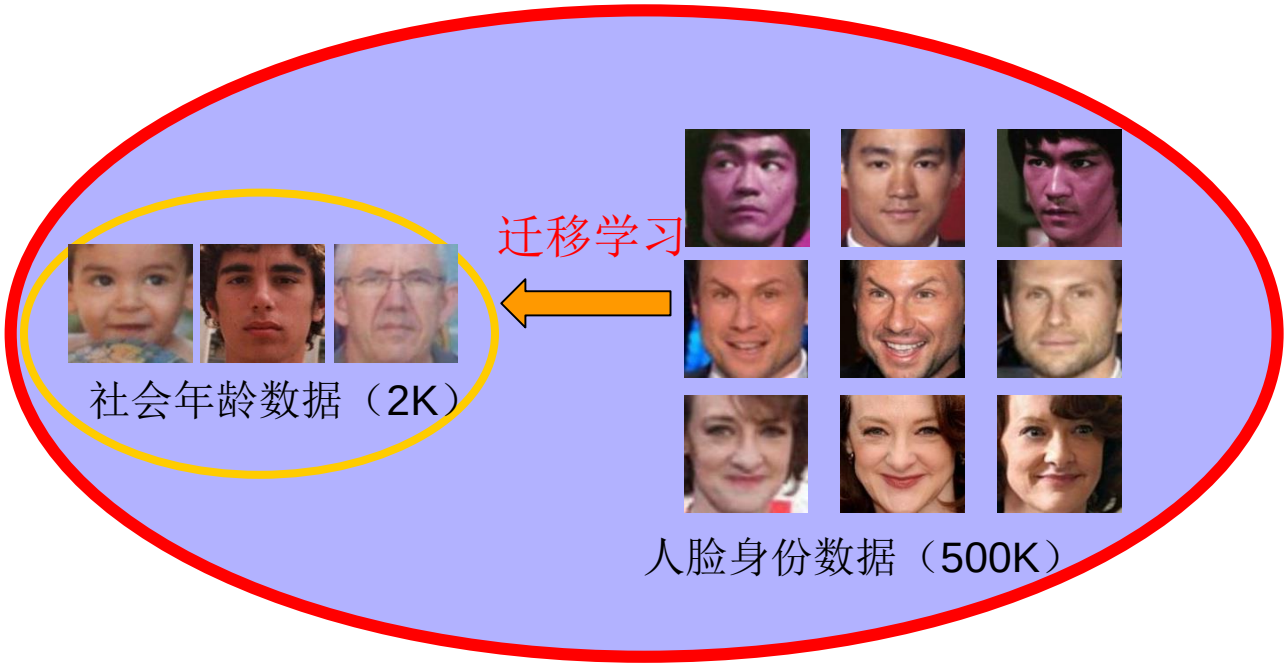
人脸身份->年龄的表示学习

- **ICCV2015社会年龄估计竞赛**
  - 4, 699张图像, 分为训练、验证和测试三个子集合
  - 开发阶段：用训练集训练，在验证集上汇报性能
  - 测试阶段：用训练集加测试集训练，在测试集上汇报性能

子集合	图像数量
训练集	2,476
验证集	1,136
测试集	1,087

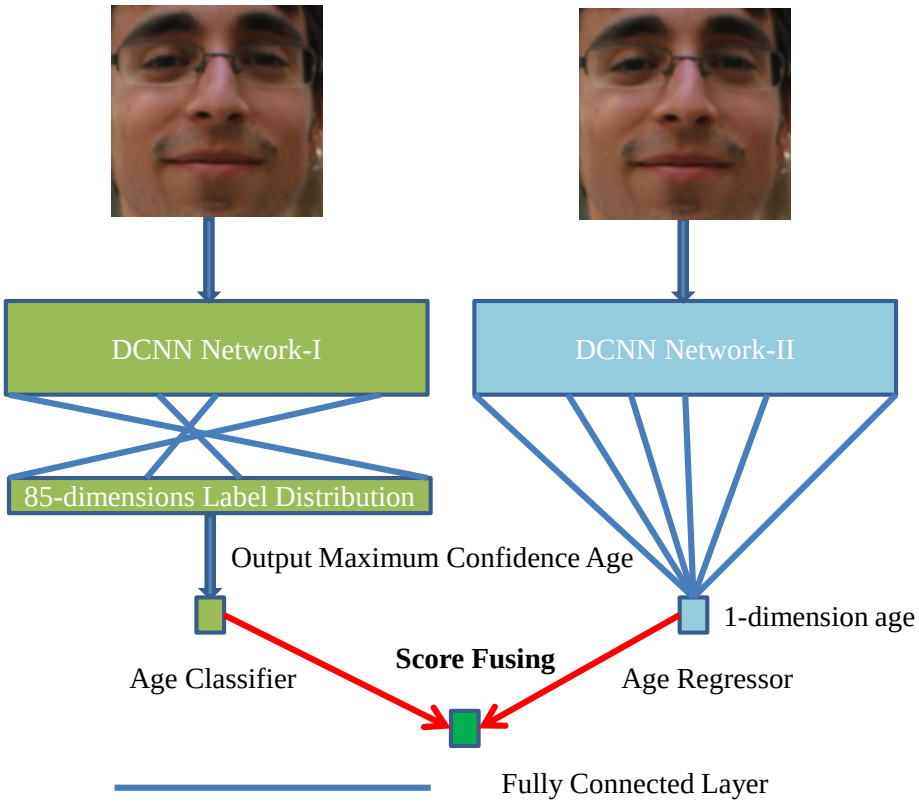
人脸身份->年龄的表示学习—研究动机

- **动机：用海量人脸身份数据来辅助年龄表示学习**
  - 大量的人脸身份数据（500K）
  - 特定场景的领域小数据：社会年龄数据集（2K）
  - 问题的提出：小数据条件下的深度迁移学习



人脸身份->年龄的表示学习—AgeNet

- **AgeNet**
  - 子网络1:基于分布标记年龄编码的年龄分类网络
  - 子网络2:基于一维实值年龄编码的年龄回归网络
  - 两个子网络的年龄输出进行融合



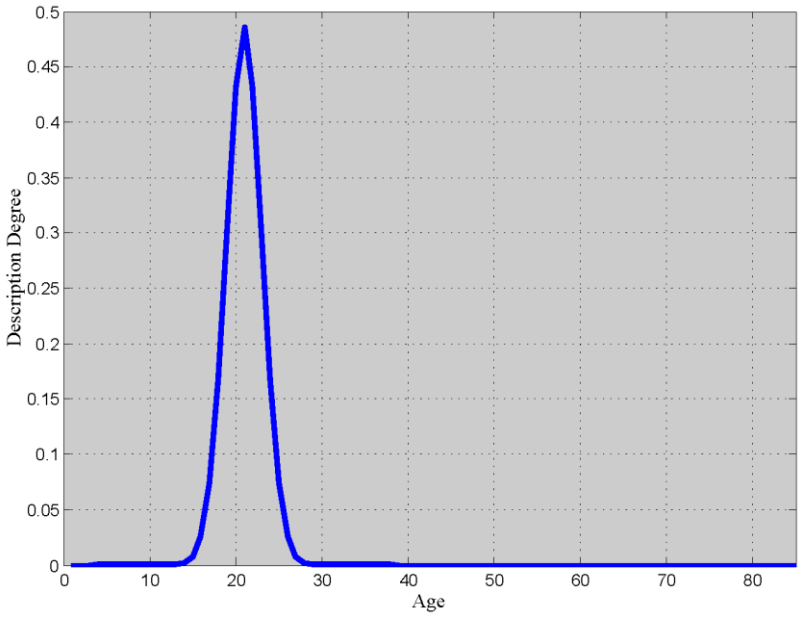
人脸身份->年龄的表示学习—AgeNet

- **年龄编码方式与损失函数**
  - **AgeNet年龄回归器: 一维实值编码+ 欧式损失**

将年龄归一化到[0,1] 并且用欧式损失学习年龄回归器

  - **AgeNet年龄分类器: LDL+交叉熵损失**

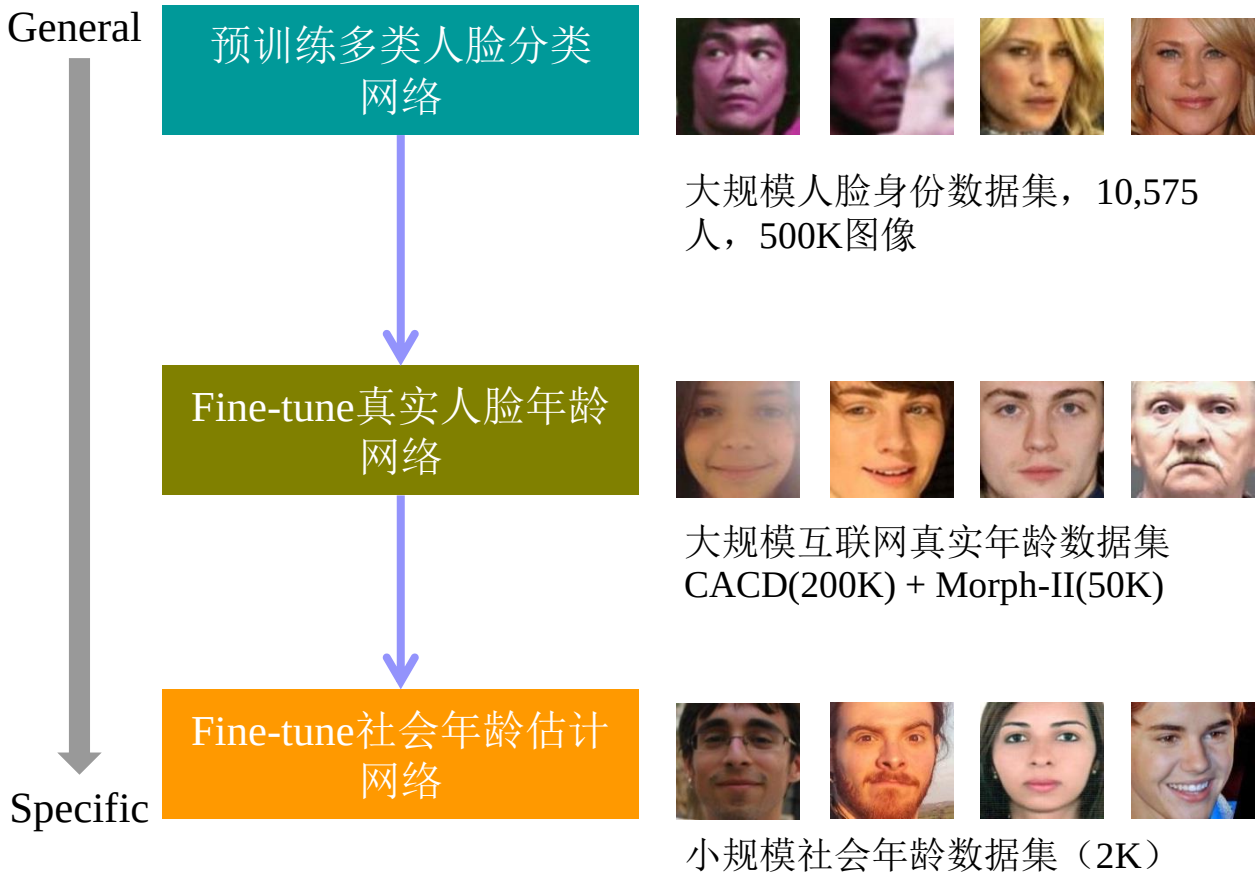
使用 (LDL Geng' TPAMI2013) 来编码年龄并学习年龄分类器



$$\text{label}(j)=\exp\left(\frac{-(j-y)^2}{2*\text{cvar}^2}\right)/\text{cvar},\ j=1,2,\ldots,MaxAge$$

人脸身份->年龄的表示学习—AgeNet

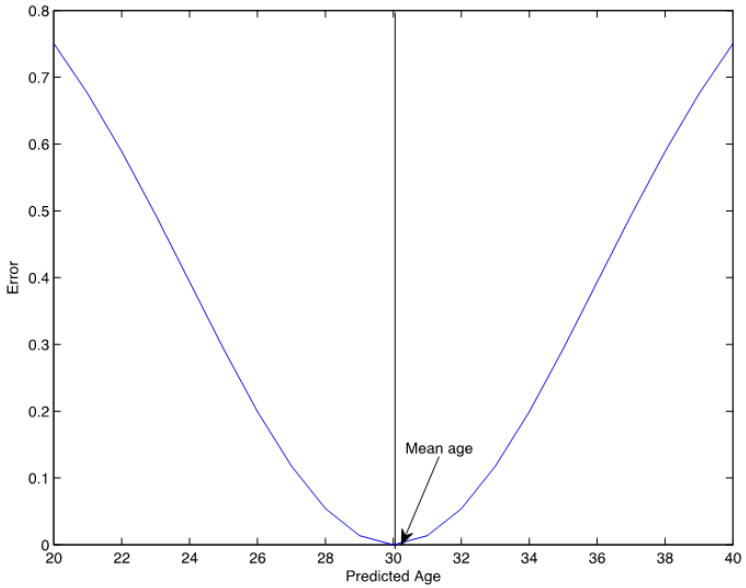
- 从General到Specific的深度迁移学习策略
  - 从人脸身份网络->真实年龄网络->社会年龄网络的迁移学习



人脸身份->年龄的表示学习—实验验证

- **性能评价指标**
  - 平均归一误差，y_i 表示ground-truth, $\hat{y}_i$表示估计的年龄

$$\bar{\varepsilon} = \frac{1}{N}\sum_{i=1}^{N}[1-\exp(\frac{-(y_i-\hat{y}_i)^2}{2\sigma_i^2})]$$



  - 平均绝对误差

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i-\hat{y}_i|$$

人脸身份->年龄的表示学习—实验验证

- **深度迁移学习策略的比较**
  - 由于在小数据上的过拟合，随机初始化性能差
  - 人脸身份->社会年龄的深度迁移学习优于真实年龄->社会年龄的迁移学习
  - 人脸身份->真实年龄->社会年龄的深度迁移学习性能最优

迁移学习策略	平均归一误差	平均误差
随机初始化	0.5381	7.0924
真实年龄->社会年龄	0.3994	4.7526
人脸身份->社会年龄	0.3504	4.2095
人脸身份->真实年龄->社会年龄	**0.3360**	**3.9489**

人脸身份->年龄的表示学习—实验验证

- **ICCV2015年龄估计竞赛最终评测**
  - 我们是亚军，性能略低于第一名的工作，计算量显著小（8个 **GoogLeNet** vs. 20 **VGGNet**)

排名	队伍	方法介绍	开发阶段	测试阶段
1	CVL_ETH	**20 VGGNet**, 101 reunions softmax normalized output of the last layer	0.295116	**0.264975**
2	**AgeNet**	**8 GoogLeNet**, AgeNet Age Regressor and Classifier	**0.292297**	0.270685
3	Ageseer	VGGNet, Prediction of age codes, fusion of regressors, such as lasso, global and local quadratic regressor, and RF	0.327321	0.287266
3	WVU_CVL	GoogleLet + 10 age groups, RF, SVR, and fusion	0.316289	0.294835
4	SEU_NJU	VGGNet, Fusion of different network setups, softmax loss and KL-divergence	0.380615	0.305763

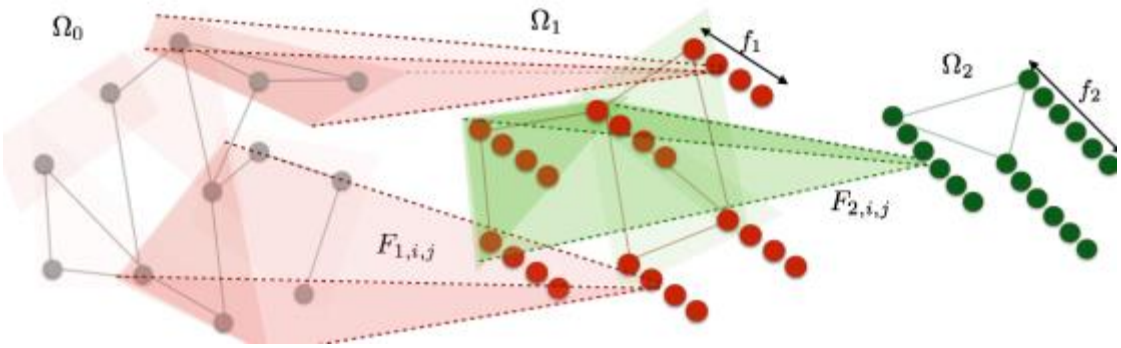
Xin Liu et al.. AgeNet: Deeply Learned Regressor and Classifier for Robust Apparent Age Estimation. International Conference on Computer Vision ChaLearn LaP Workshop (ICCVW), 2015

Outline

- CNN基础理论
- CNN结构演化
- CNN实用技巧：以Caffe为例
- CNN实战：ICCV2015年龄估计竞赛
- **前沿话题讨论**

前沿话题讨论

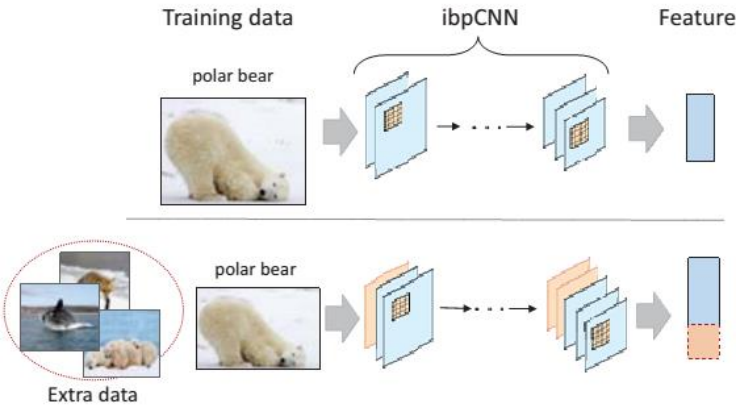
- **Convolution on Graphs**
  - 非2D图像数据如何做使用CNN
    - 3D Mesh Data
    - Time-frequency audio representation
    - Social Network Signals, gene expression
  - CNN on Graphs
    - Hierarchical Clustering of Graph



Joan Bruna, Wojciech Zaremba, Arthur Szlam and Yann LeCun. Spectral networks and locally connected networks on graphs, ICLR2014

前沿话题讨论

- **网络结构学习ibpCNN** Feng's ICCV2015



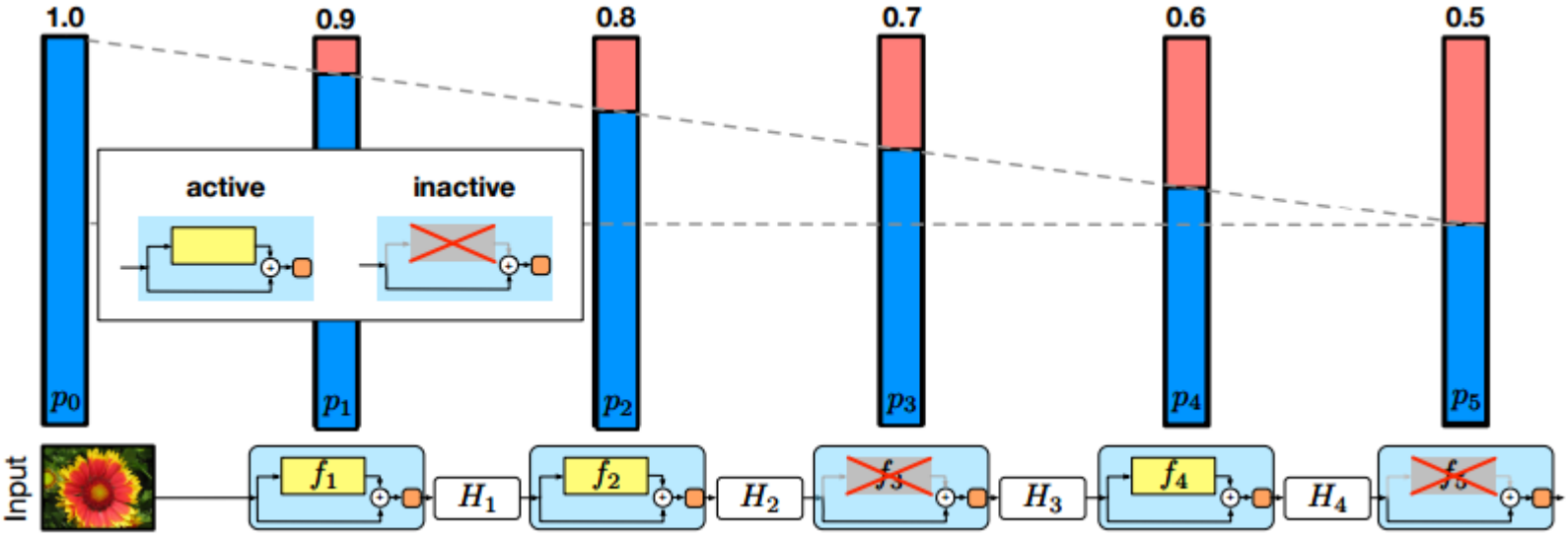
Key idea: 模型复杂度要与样本复杂度相匹配，并可以随着训练动态调整模型复杂度

$$\mathcal{L}_{\text{conv}}(W,Z,K) := \sum_i \|X_i - \sum_{k=1}^{K} w_k * Z_{i,k}\|^2 + \lambda^2 K \quad \sum_{\ell=1}^{L} \mathcal{L}_{\text{task}}(Z^{(\ell)};X,y) + \sum_{\ell=1}^{L_{\text{conv}}} \mathcal{L}_{\text{conv}}(W^{(\ell)},Z^{(\ell)},K^{(\ell)};X^{(\ell-1)})$$
$$+ \sum_{\ell=L_{\text{conv}}+1}^{L} \mathcal{L}_{\text{fc}}(W^{(\ell)},Z^{(\ell)},K^{(\ell)};X^{(\ell-1)}), \qquad (4)$$

# training	1%	5%	10%	50%	100%
DictLearn [7]	14.23	18.68	21.38	29.71	31.24
CNN-AlexNet [6]	5.02	10.80	21.17	**47.12**	**61.7**
CNN-GAP	11.23	15.91	26.21	41.23	54.38
Ours	**17.79**	**23.36**	**31.04**	46.87	58.07
# filters (rel.)	4.2%	14.5%	17.5%	28.2%	37.5%

前沿话题讨论

- **Deep Networks with Stochastic Depth**
  - 训练阶段随机drop掉层，加速收敛，测试阶段使用完整结构
  - 加速收敛，可以训练1200层网络，CIFAR-10上4.91%测试误差



$$H_\ell^{\text{Test}} = \text{ReLU}(p_\ell f_\ell(H_{\ell-1}^{\text{Test}}; W_\ell) + H_{\ell-1}^{\text{Test}})$$

Huang G, Sun Y, Liu Z, et al. Deep Networks with Stochastic Depth[J]. arXiv preprint arXiv:1603.09382, 2016.

前沿话题讨论

- **网络压缩**
  - 思路1：离线的模型低秩逼近[1]或量化[2]
  - 思路2：训练过程中的网络剪枝（Network Pruning）[3,4]
  - 思路3：连接二值化，将乘法运算转为位运算[5]

[1] Zhang X, Zou J, Ming X, et al. **Efficient and accurate approximations of nonlinear convolutional networks**,CVPR2015

[2] Wu J, Leng C, Wang Y, et al**. Quantized Convolutional Neural Networks for Mobile Devices**[J]. arXiv preprint arXiv:1512.06473, 2015.

[3] Han S, Pool J, Tran J, et al. **Learning both Weights and Connections for Efficient Neural Network**, NIPS15

[4] Polyak A, Wolf L. **Channel-Level Acceleration of Deep Face Representations**[**J**]. Access, IEEE, 2015, 3: 2163-2175.

[5] Courbariaux M, Bengio Y. **BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to+ 1 or-1**[J]. arXiv preprint arXiv:1602.02830, 2016.

致谢

中科院计算所

Shiguang Shan

Shaoxin Li

Meina Kan

Shuzhe Wu

Wanglong Wu

欢迎关注深度学习大讲堂微信公众号

dlclass