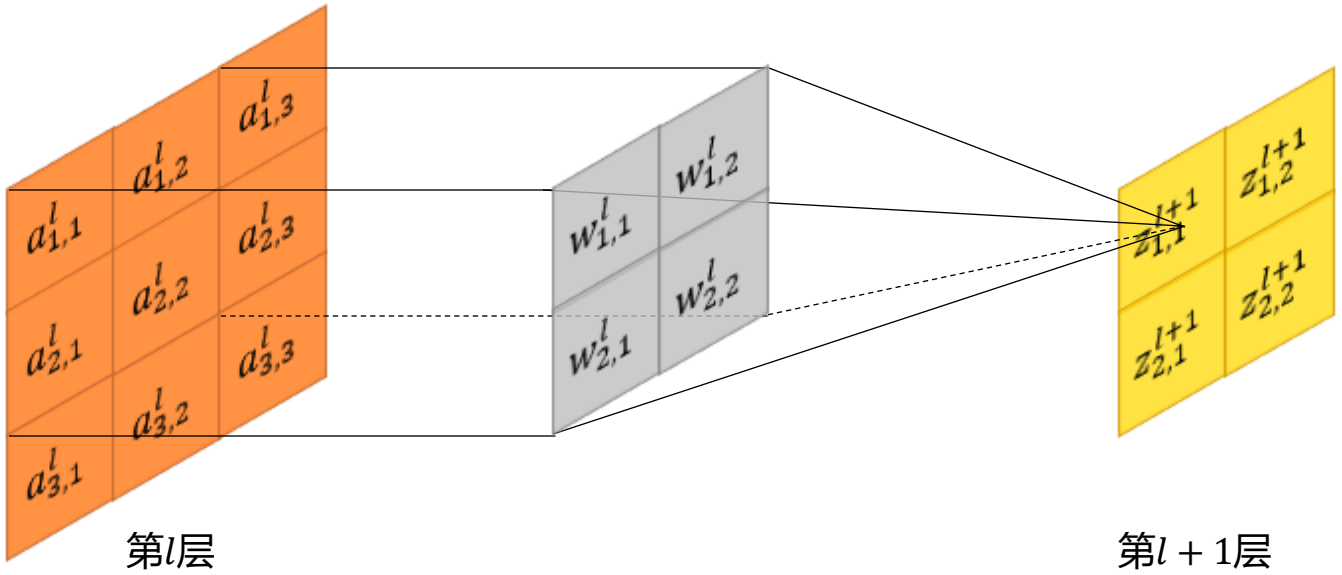
四川大學
SICHUAN UNIVERSITY

机器学习-第十章 深度残差神经网络

教师：胡俊杰 副教授

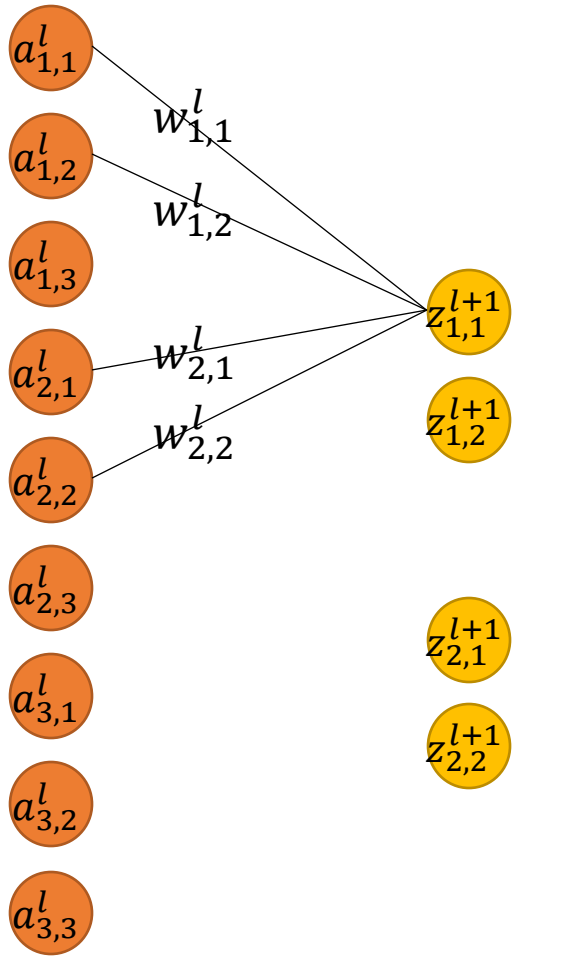
邮箱：hujunjie@scu.edu.cn

CNNs的组件



第l层　　第$l+1$层

局部连接

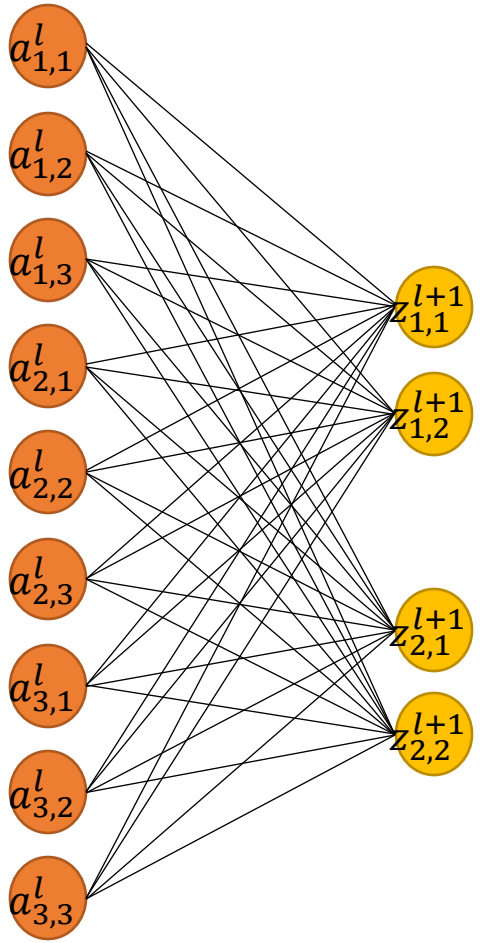


第l层　　第$l+1$层

$$z^{l+1}_{n,m} = \sum_{i=1}^{I}\sum_{j=1}^{J} a^l_{n+j-1,m+i-1} \cdot w^l_{j,i}$$

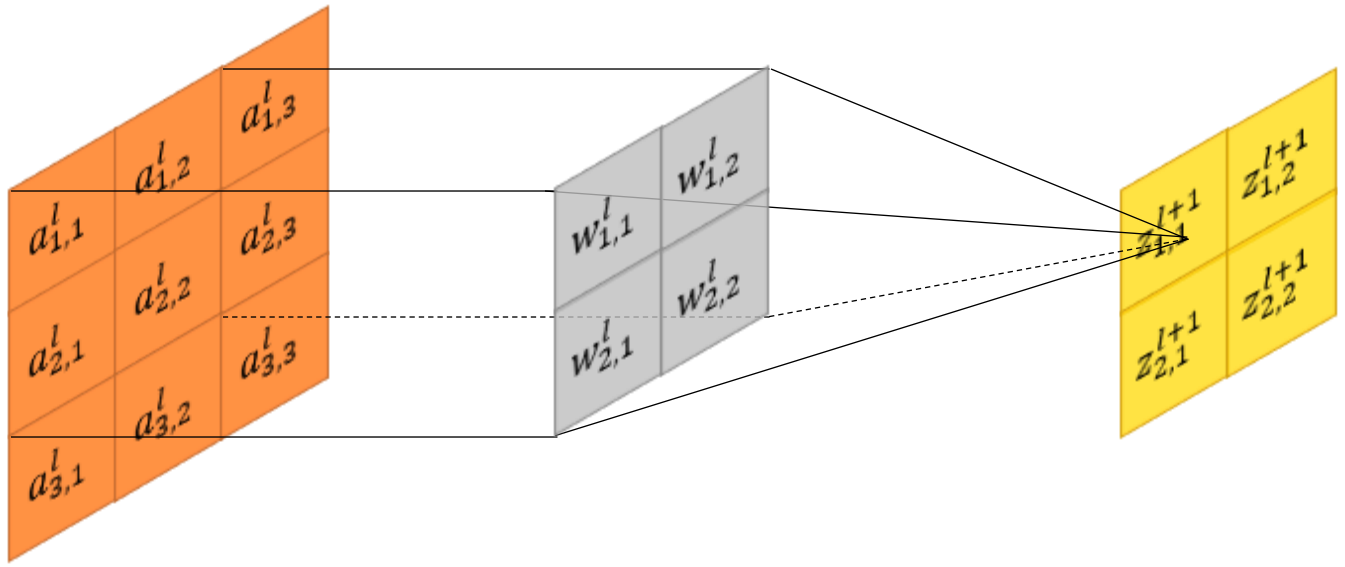
- 对位元素相乘并求和等价于全连接中的局部连接
- 每次运算共享卷积核

CNNs的组件



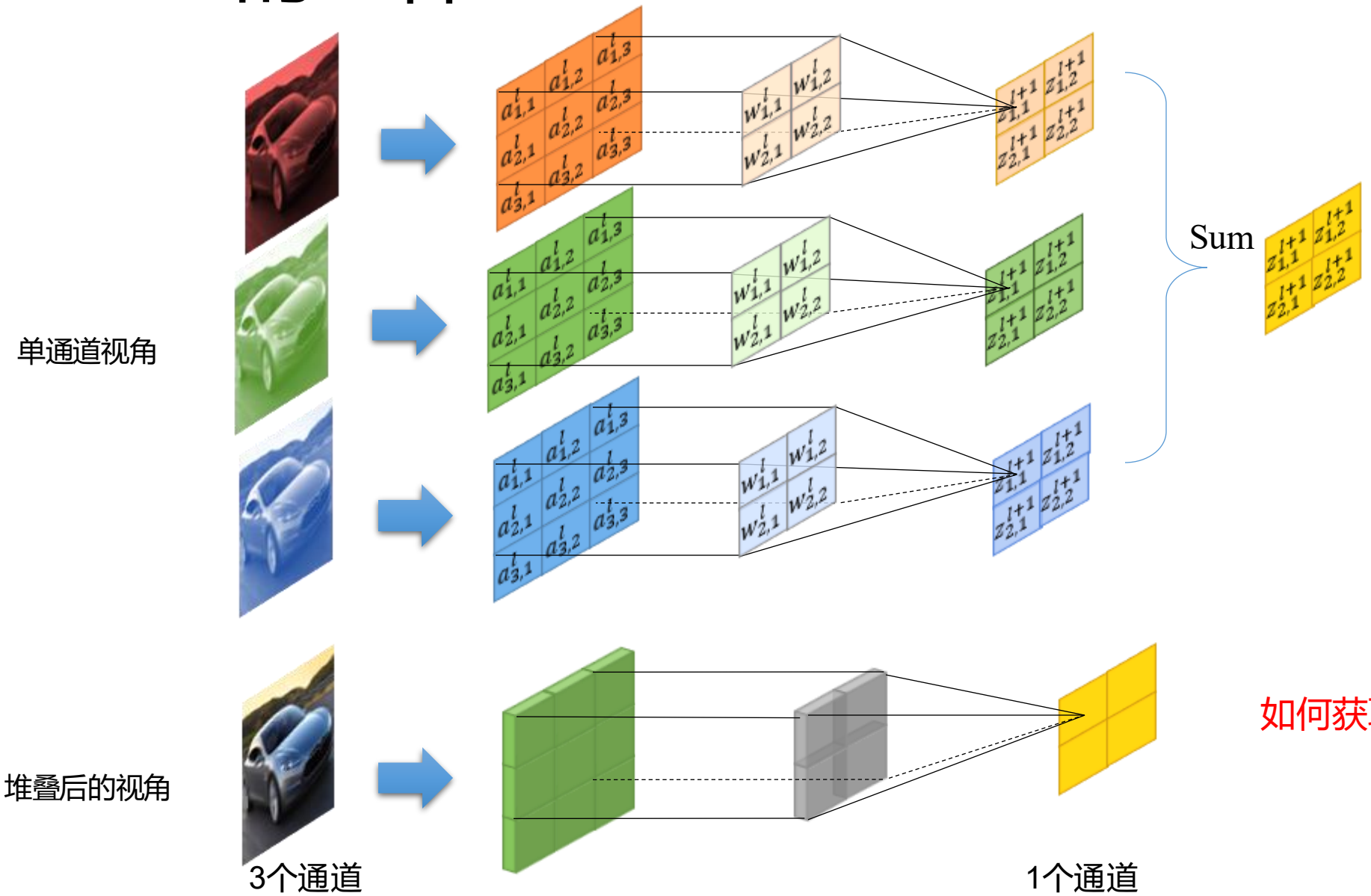
$a^l \in R^9$ $z^{l+1} \in R^4$

- 全连接中的参数为$9 \times 4 = 36$



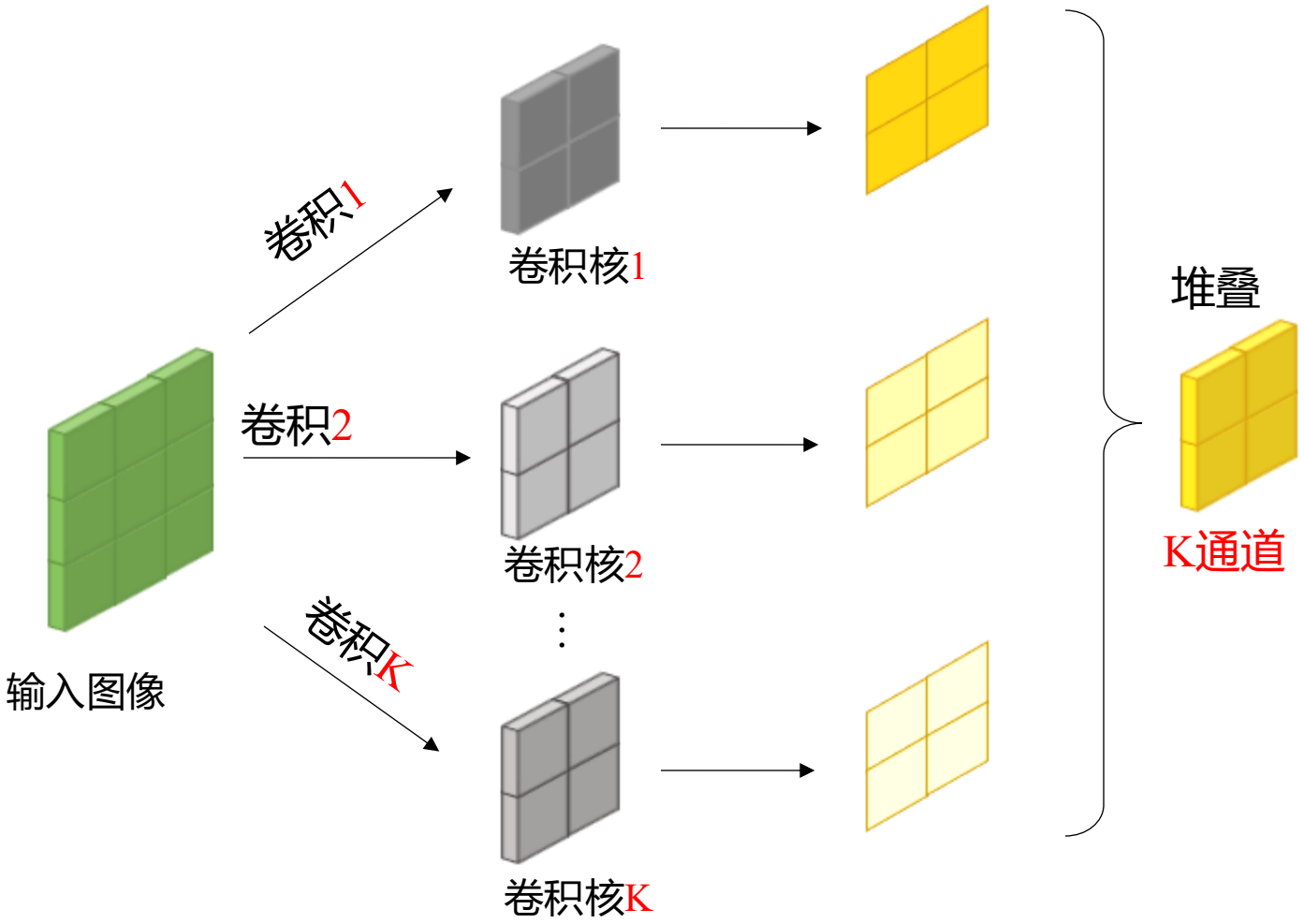
- 卷积操作中的参数量等于卷积核的大小，即2×2，远小于全连接中的参数量

CNNs的组件



如何获取具有多个通道的输出？

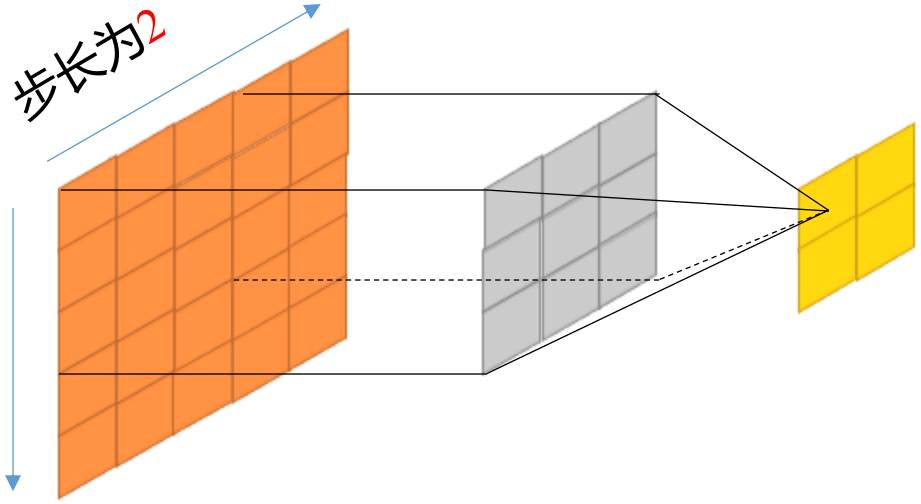
CNNs的组件

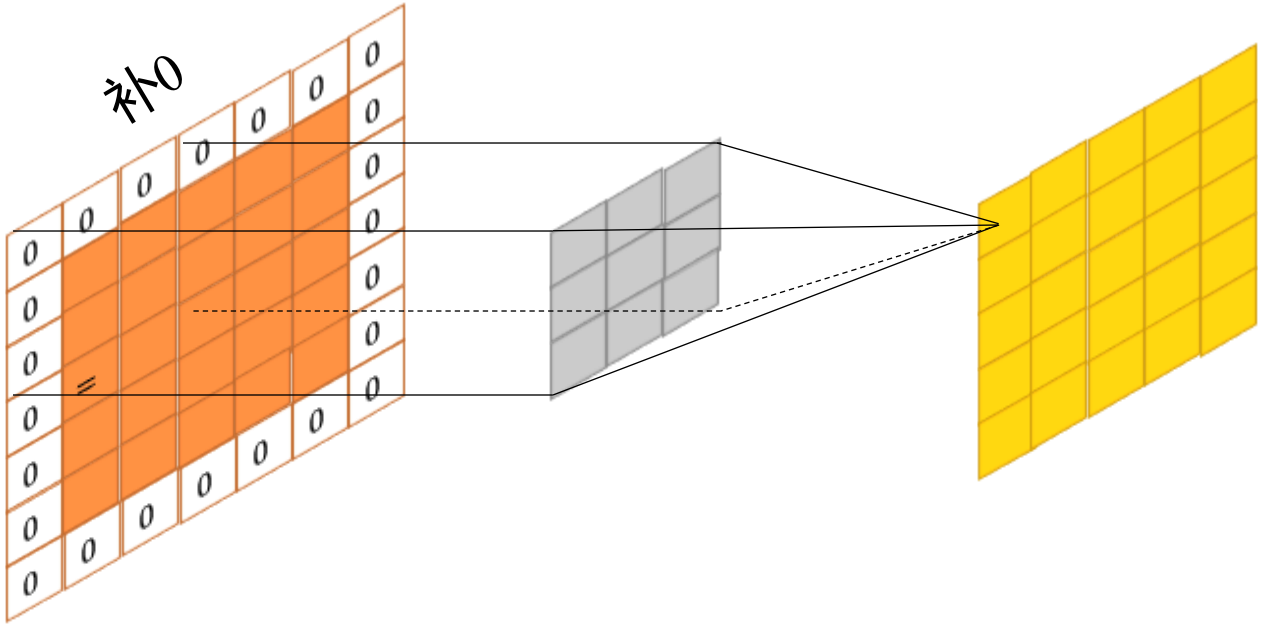


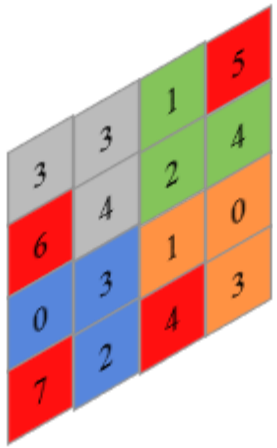
第1步: 使用多套卷积核对输入进行卷积

第2步: 堆叠每个卷积的输出，其中输出的通道数等于卷积核的数目

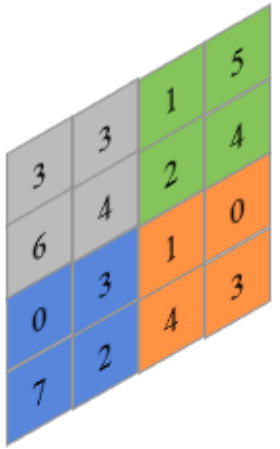
CNNs的组件





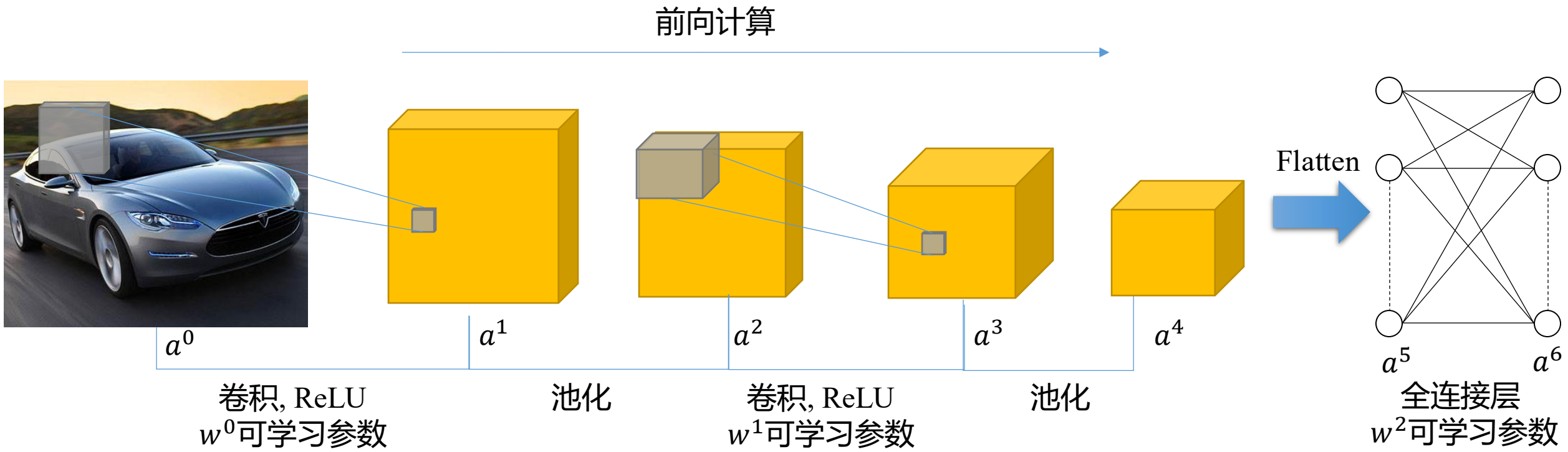


最大值池化
Max pooling

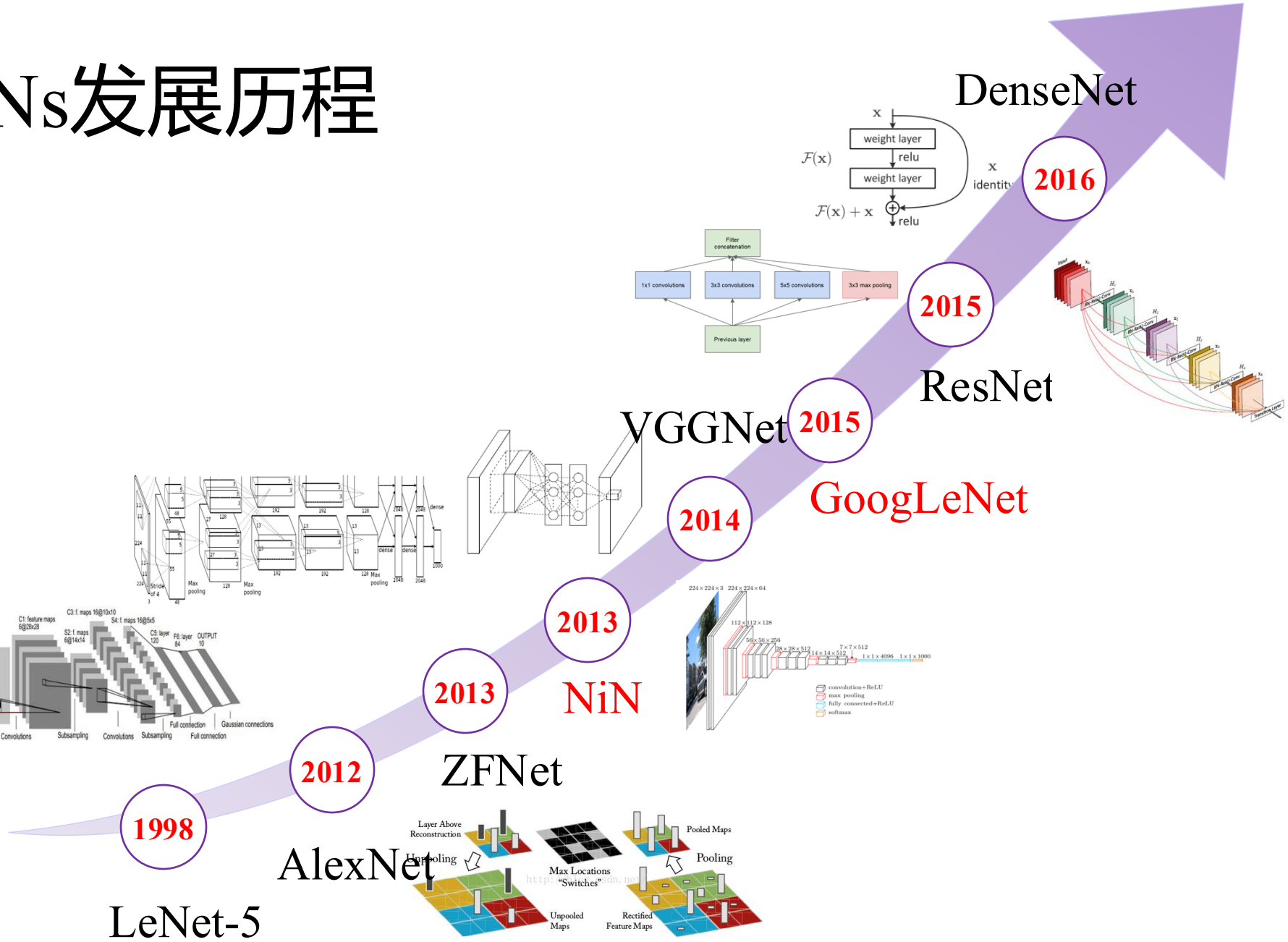


均值池化
Average pooling

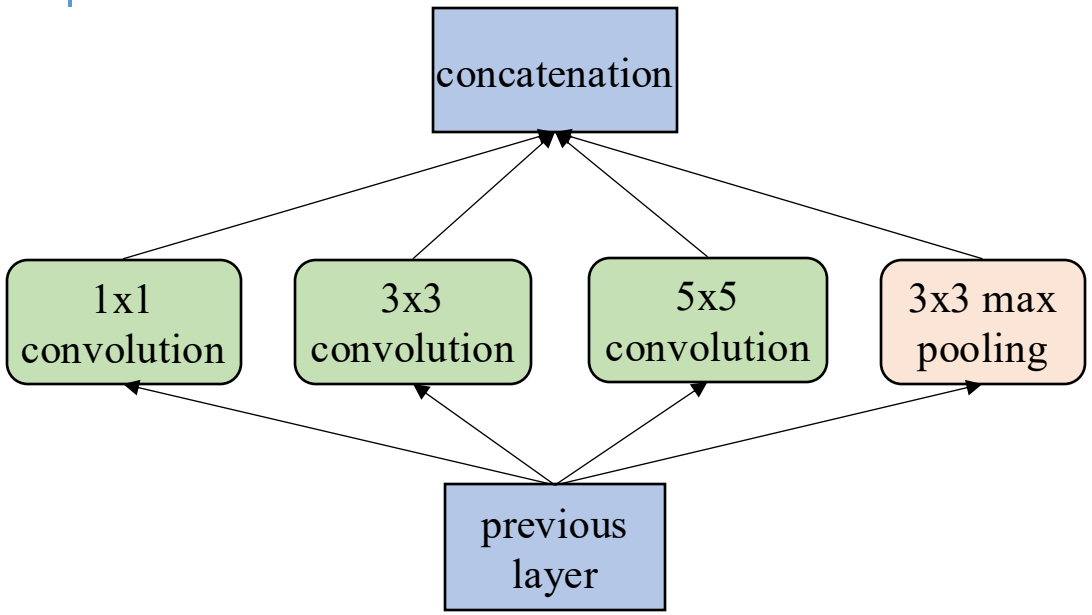
CNNs的结构


前向计算
Flatten
a^0
a^1
a^2
a^3
a^4
a^5
a^6
卷积, ReLU
w^0可学习参数
池化
卷积, ReLU
w^1可学习参数
池化
全连接层
w^2可学习参数

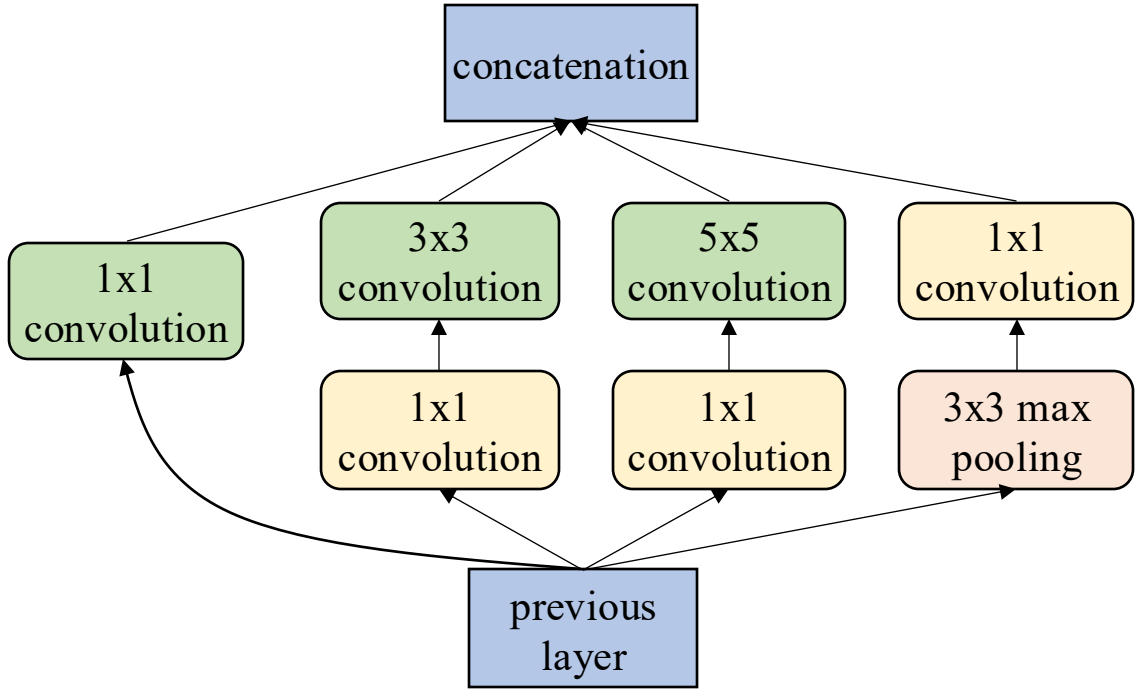
CNNs发展历程


1998
LeNet-5
2012
AlexNet
2013
ZFNet
2013
NiN
2014
VGGNet
2015
GoogLeNet
2015
ResNet
2016
DenseNet
weight layer
relu
weight layer
identity
1x1 convolutions
3x3 convolutions
5x5 convolutions
3x3 max pooling
Previous layer
Convolutions
Subsampling
Full connection
Gaussian connections
Max Locations "Switches"
Pooled Maps
Unpooled Maps
Rectified Feature Maps
Pooling

Inception-V1



Inception-V1普通版本

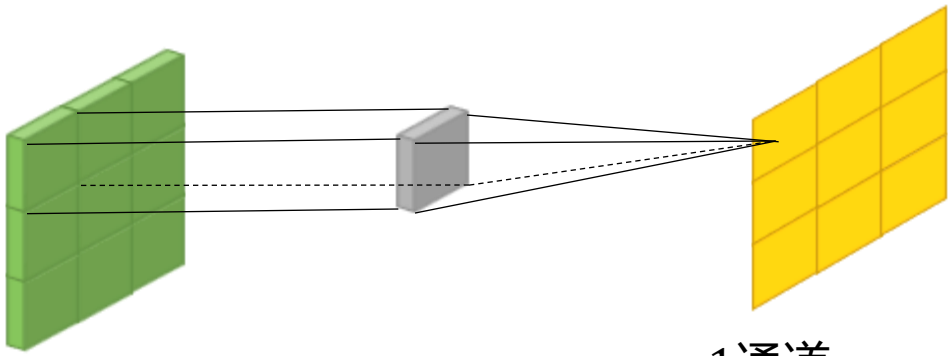


Inception-V1

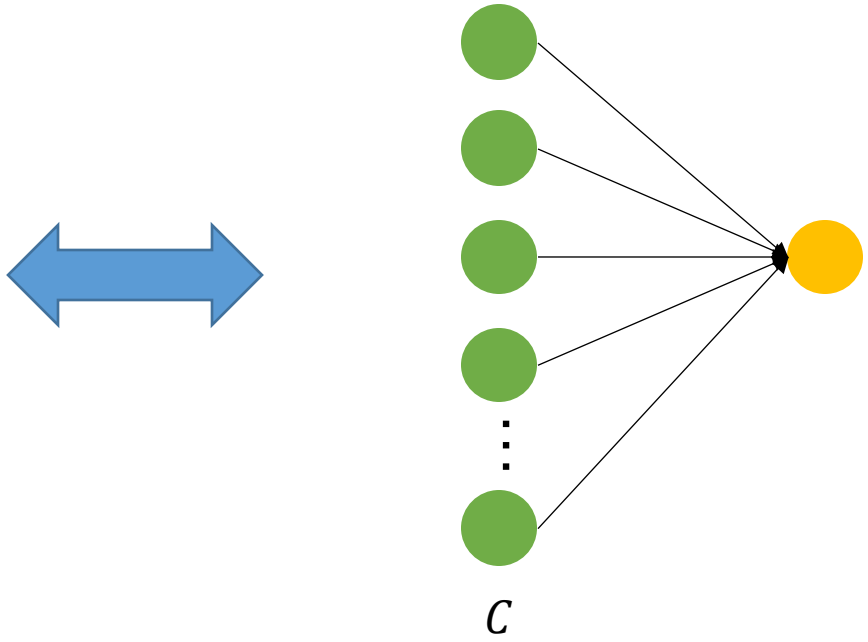
- 对3x3 、 5x5和最大值池化使用1x1卷积减少通道数目

[*] C. Szegedy, W. Liu, Y. Jia, et al. Going Deeper with Convolutions[C]. CVPR, 2015.

Inception-V1

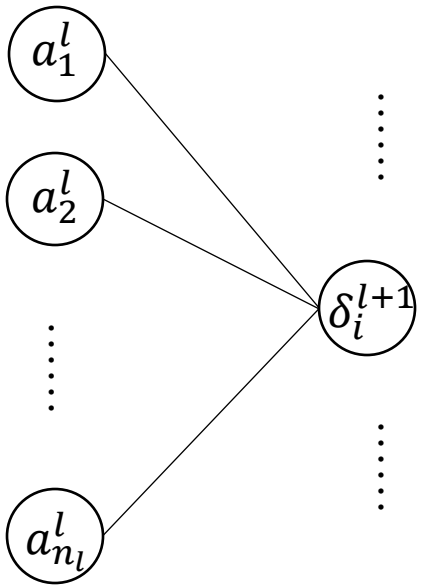


1x1卷积



[*] C. Szegedy, W. Liu, Y. Jia, et al. Going Deeper with Convolutions[C]. CVPR, 2015.

Normalization

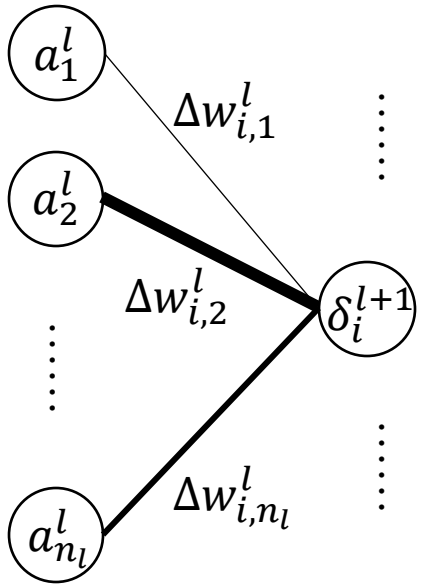


$$\frac{\partial J}{\partial w_{i,*}^l} = \delta_i^{l+1} a_*^l$$

- **Zero-centered**

  当a_*^l均为正或均为负时，$\text{sign}(\frac{\partial J}{\partial w_{i,*}^l})$将相同，且只取决于 $\text{sign}(\delta_i^{l+1})$

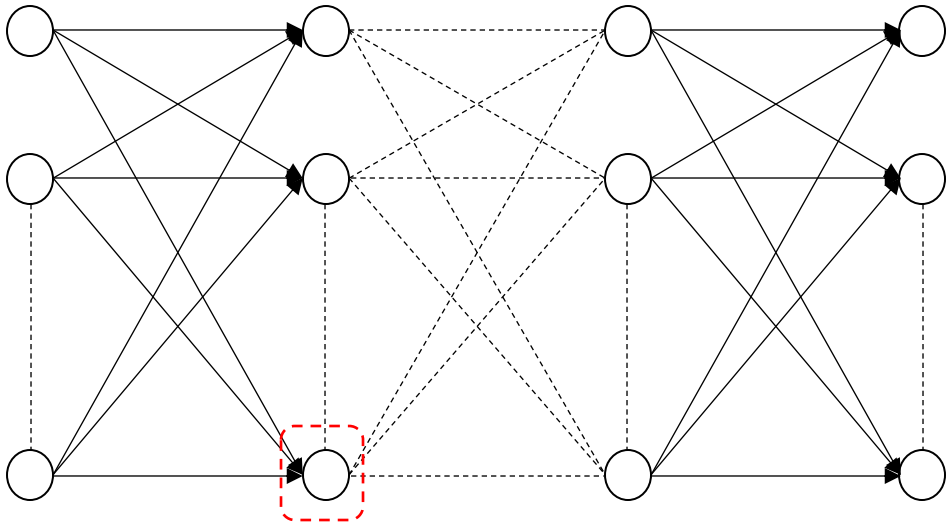
- **Scaled**



[*] LeCun Y A, Bottou L, Orr G B, et al. Efficient backprop[M]. Neural networks: Tricks of the trade, 1998.

Batch Normalization (BN)

训练阶段
全连接神经网络



$$\mu_i = \frac{1}{B}\sum_{b=1}^{B} z_{b,i}^{l+1}$$

$$\sigma_i^2 = \frac{1}{B}\sum_{b=1}^{B} (z_{b,i}^{l+1} - \mu_i)^2$$

B:训练阶段的batch大小

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l \quad \Rightarrow \quad \hat{z}_i^{l+1} = \frac{z_i^{l+1} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad \Rightarrow \quad \tilde{z}_i^{l+1} = \gamma_i \hat{z}_i^{l+1} + \beta_i$$

- μ_i 和 σ_i^2为第i个神经元batch激活值的均值和方差
- ϵ 是一个很小的常量，避免分母为0

- γ_i 和 β_i 为第i个神经元的可学习参数
- 当$\gamma_i = \sigma_i$且$\beta_i = \mu_i$时，$\tilde{z}_i^{l+1} \approx z_i^{l+1}$
- Batch Normalization可让神经元学习调整激活值的分布

[*] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C], ICML. 2015.

Batch Normalization (BN)

测试阶段

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l \quad \Rightarrow \quad \hat{z}_i^{l+1} = \frac{z_i^{l+1} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad \Rightarrow \quad \tilde{z}_i^{l+1} = \gamma_i \hat{z}_i^{l+1} + \beta_i$$

- 测试阶段（推理）不受batch的影响
- 神经网络假设训练集与测试集独立同分布（Independent and identically distributed），因此针对第i个神经元，测试阶段应使用其在全体训练集上计算得到的μ和σ
- 如何使用局部估计全局?

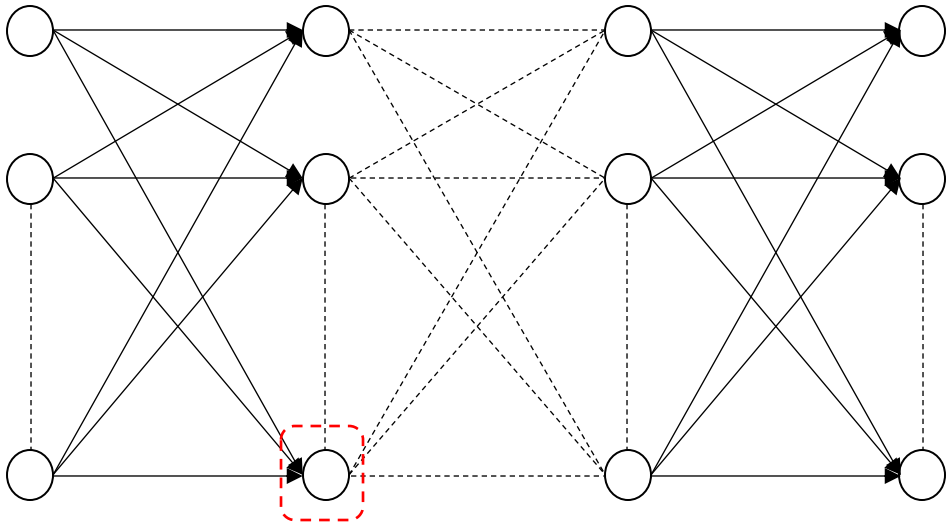
- 在训练时，BN使用指数滑动平均（Exponential moving average）方法估计全体训练集的μ和σ

$$v_t := decay \cdot v_{t-1} + (1 - decay) \cdot y_t$$

  - v_t: t时刻的估计值，y_t: t时刻的观测值
  - 推荐的decay取值： 0.999, 0.99, 0.9…
  - decay越大，v_{t-1} 影响更大。decay越小，y_t影响更大

Batch Normalization (BN)

全连接神经网络



$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l \, a_j^l \quad \Rightarrow \quad \hat{z}_i^{l+1} = \frac{z_i^{l+1} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad \Rightarrow \quad \tilde{z}_i^{l+1} = \gamma_i \hat{z}_i^{l+1} + \beta_i$$

卷积神经网络

- 卷积中的通道可类比于全连接中的神经元
  - 全连接中，μ和σ，γ和β针对单个神经元
  - 卷积中， μ和σ，γ和β针对各通道（channel）

Batch Normalization (BN)

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l \, a_j^l \quad \Rightarrow \quad \hat{z}_i^{l+1} = \frac{z_i^{l+1} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad \Rightarrow \quad \tilde{z}_i^{l+1} = \gamma_i \hat{z}_i^{l+1} + \beta_i$$

- 卷积中的特征维度为：$[B, C, H, W]$
  - Batch Normalization针对各通道计算μ和σ
  - Layer Normalization针对各样本计算μ和σ
  - Instance Normalization针对各样本和通道计算μ和σ
  - Group Normalization对各样本中的通道分组，并计算各组的μ和σ

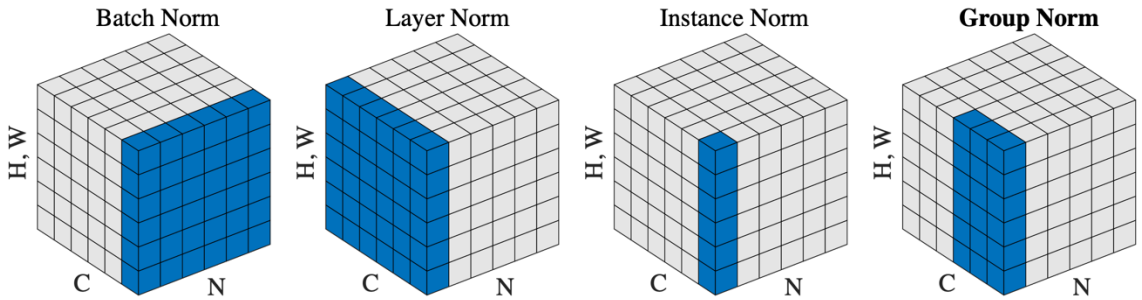
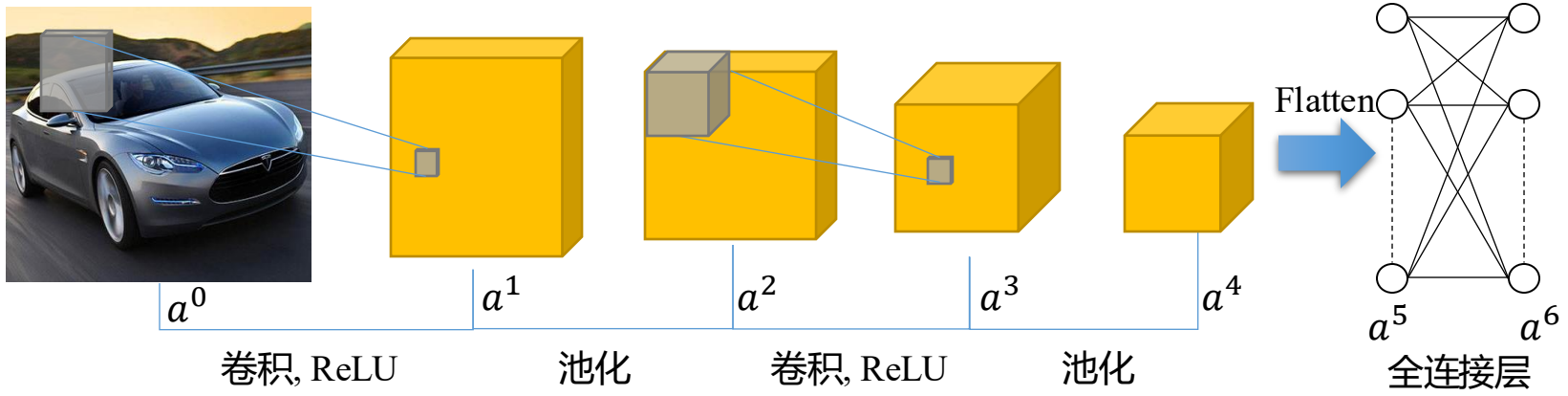


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

CNNs的学习



网络输出 $a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$

标签 $y = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_L} \end{bmatrix}$

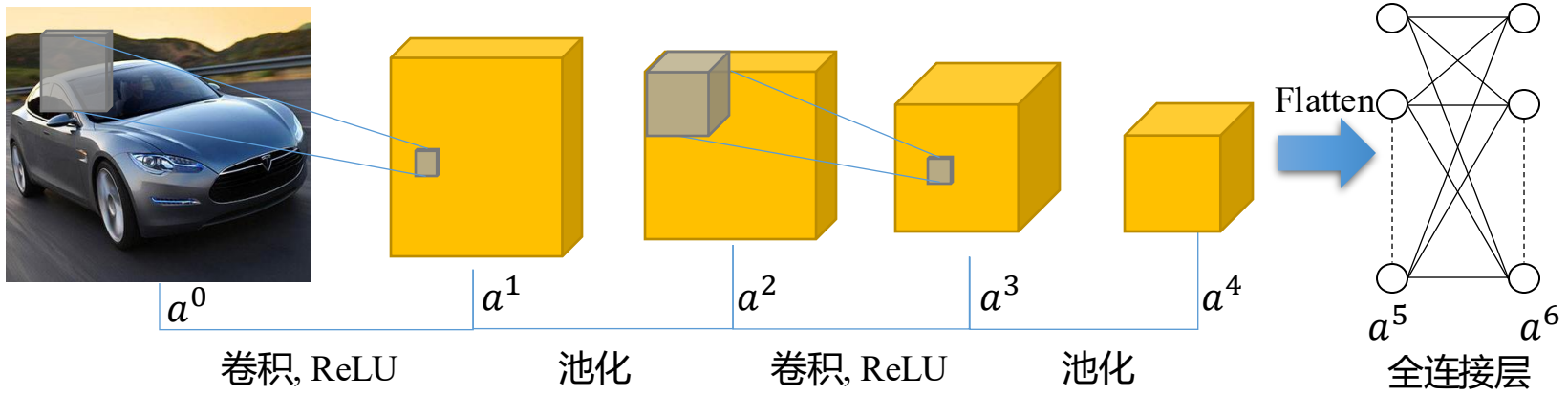
- 使用MSE代价函数

$$J = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j\right)^2$$

- 假设最后一层激活函数为f，即$a_i^L = f(z_i^L)$

求$\delta_i^L = \frac{\partial J}{\partial z_i^L}$

CNNs的学习



网络输出 $a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$

标签 $y = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_L} \end{bmatrix}$

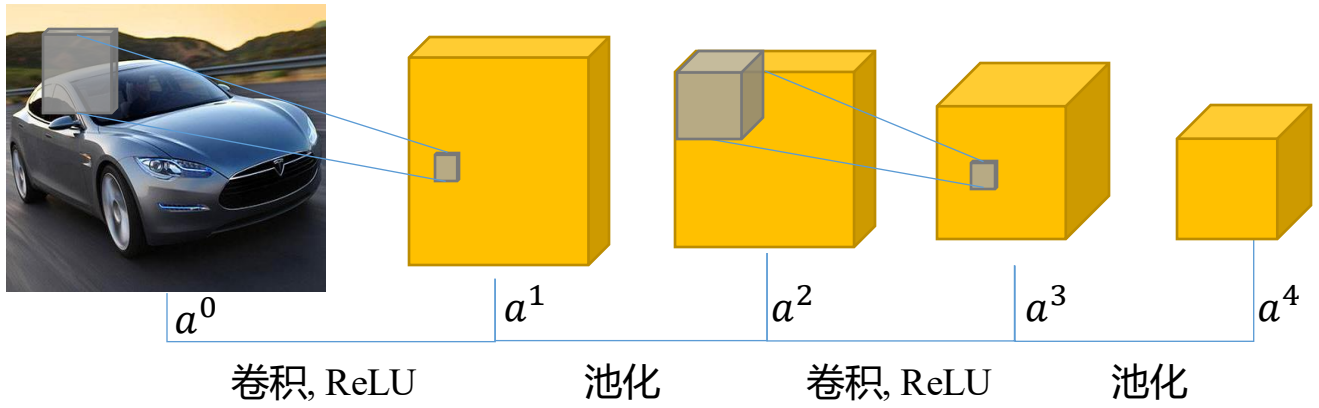
- 使用MSE代价函数

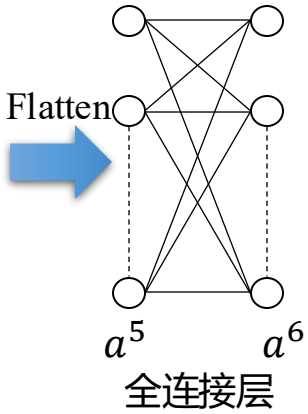
$$J = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j\right)^2$$

- 假设最后一层激活函数为f，即$a_i^L = f(z_i^L)$

$$\delta_i^L = \frac{\partial J}{\partial z_i^L} = \left(a_i^L - y_i\right)\dot{f}\left(z_i^L\right)$$

CNNs的学习





网络输出 $a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$

标签 $y = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_L} \end{bmatrix}$

- 最后一层作用softmax激活函数

$$a_i^L = Softmax(z_i^L) = \frac{e^{z_i^L}}{\sum_{j=1}^{n_L} e^{z_j^L}}$$

  - 将最后一层每个神经元的输出压缩至(0,1)区间
  - $\sum_{i=1}^{n_L} Softmax(x_i) = 1$

- Cross Entropy (交叉熵) 代价函数

$$J = -\sum_{j=1}^{n_L} y_j \ln(a_j^L)$$

标签为OneHot

$$J = -\ln(a_i^L)$$

假设i是真实类别下标

- 求$\delta_i^L = \frac{\partial J}{\partial z_i^L}$

CNNs的学习

$$a_i^L = \frac{e^{z_i^L}}{\sum_{j=1}^{n_L} e^{z_j^L}}$$

$$J = -\ln(a_i^L)$$

$$\delta_i^L = \frac{\partial J}{\partial z_i^L} = \frac{\partial J}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L} = -\frac{1}{a_i^L}\frac{e^{z_i^L}\sum e^{z_j^L} - \left(e^{z_i^L}\right)^2}{\left(\sum e^{z_j^L}\right)^2}$$

$$= -\frac{\sum e^{z_j^L}}{e^{z_i^L}}\ \frac{e^{z_i^L}\sum e^{z_j^L} - \left(e^{z_i^L}\right)^2}{\left(\sum e^{z_j^L}\right)^2}$$

$$= -\frac{\sum e^{z_j^L} - e^{z_i^L}}{\sum e^{z_j^L}} = -\left(1 - \frac{e^{z_i^L}}{\sum e^{z_j^L}}\right) = a_i^L - 1$$

$$= a_i^L - y_i$$

CNNs的学习

- Cross Entropy代价函数

$$J = -\ln(a_i^L) \qquad a_i^L = \frac{e^{z_i^L}}{\sum_{j=1}^{n_L} e^{z_j^L}}$$

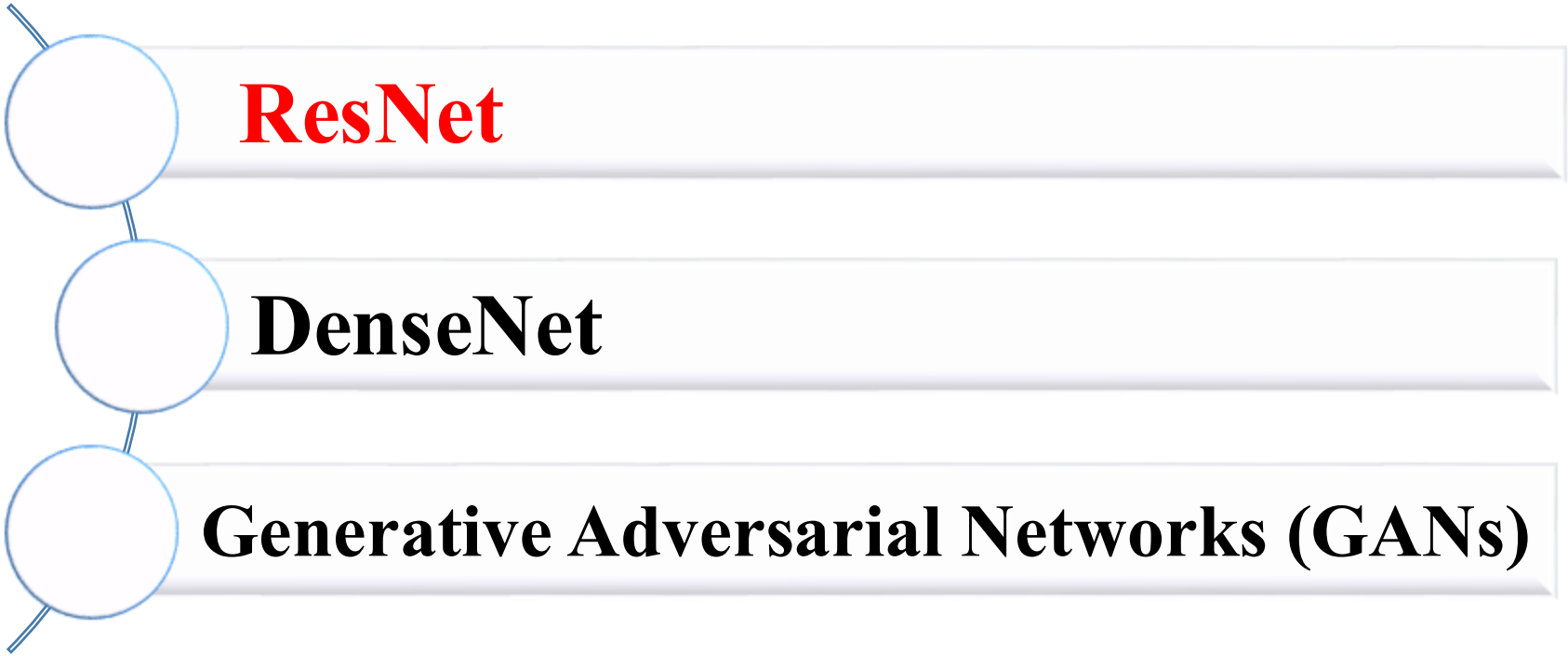
$$\delta_i^L = a_i^L - y_i$$

- MSE代价函数

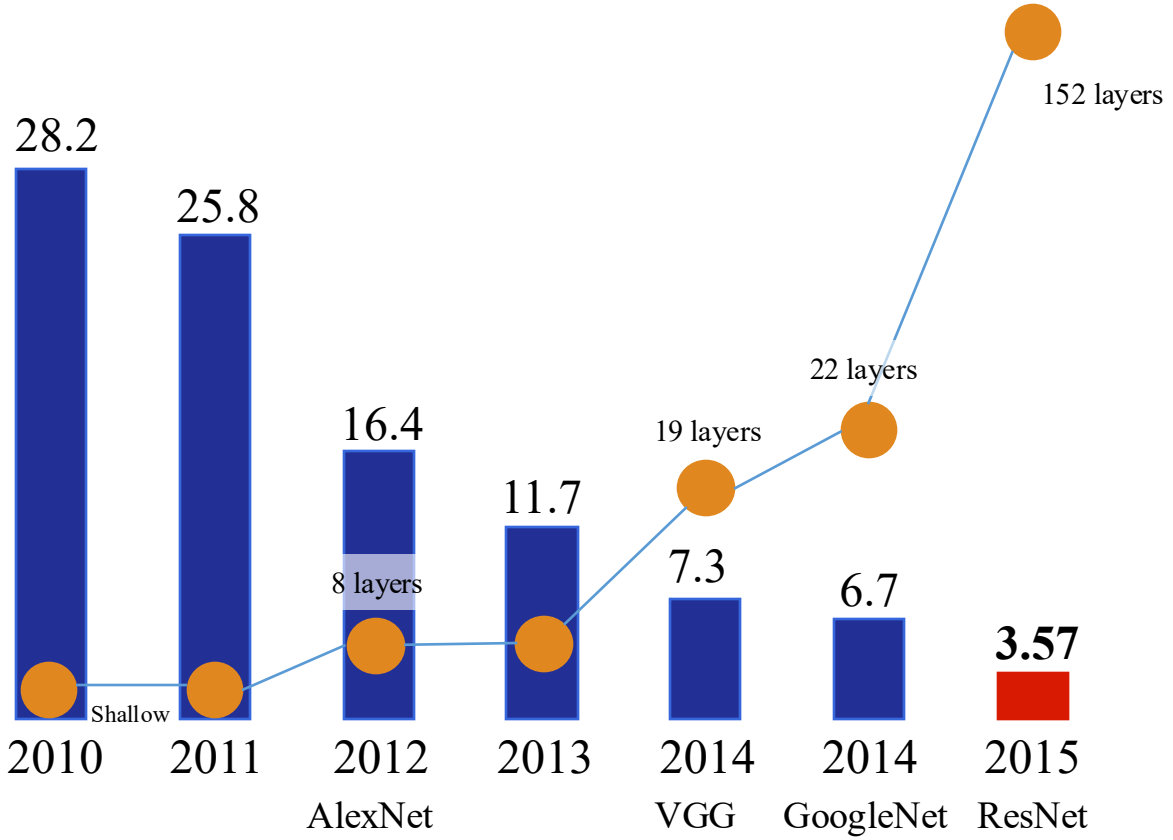
$$J = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j\right)^2$$

$$\delta_i^L = \left(a_i^L - y_i\right)\dot{f}\left(z_i^L\right)$$

大纲



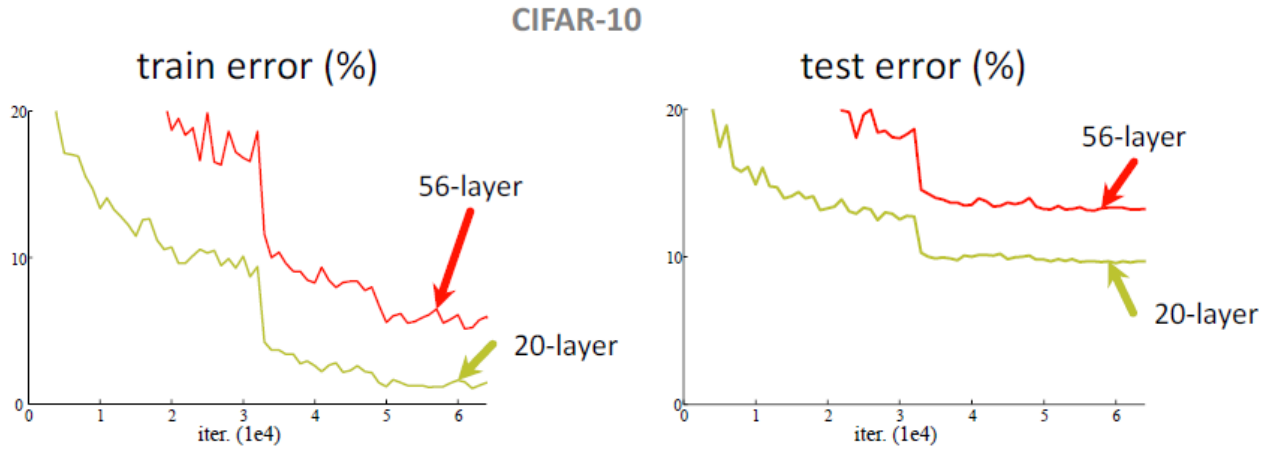
ImageNet Classification Error Rate



ImageNet Top-5分类错误率 (%)

Degradation Problem

- 神经网络是否越深越好?



- ResNet是如何解决degradation问题的?

Residual Blocks

在CNNs中加入跨层连接（shortcut connection）

- 普通的CNNs

$$y = F(x, \{W_i\})$$

  - 其中F代表由卷积、BN、ReLU组合而成的非线性映射
  - W_i代表可学习参数

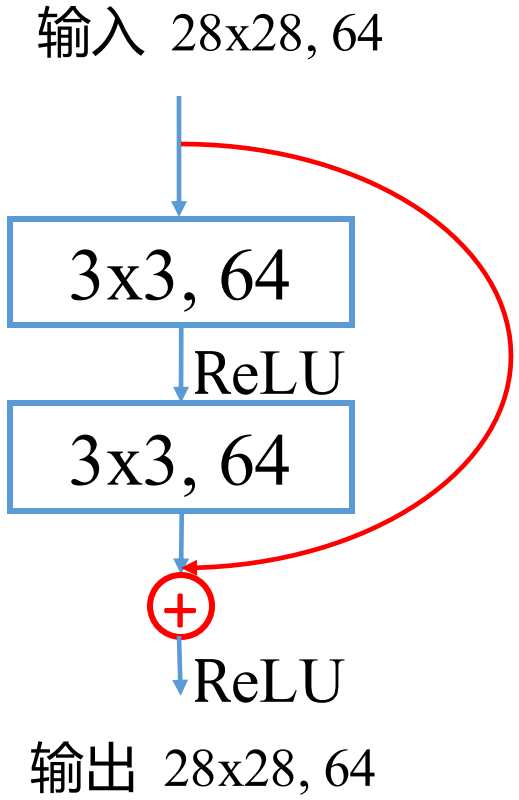
- ResNet

$$y = F(x, \{W_i\}) + x$$

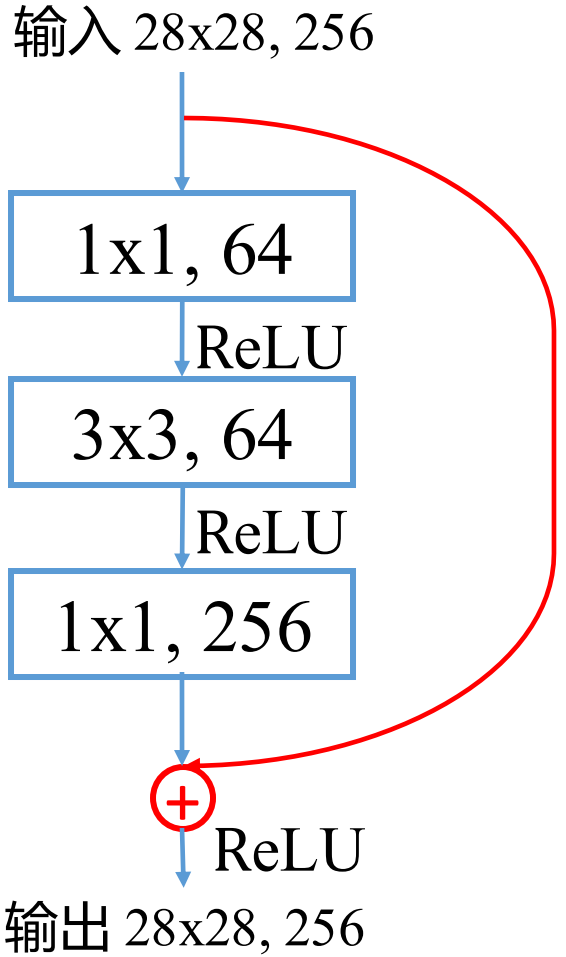
  - 左侧的x代表跨层连接

Residual Blocks with Identity

- 普通版本
- $y = W_2 f(W_1 x) + x$



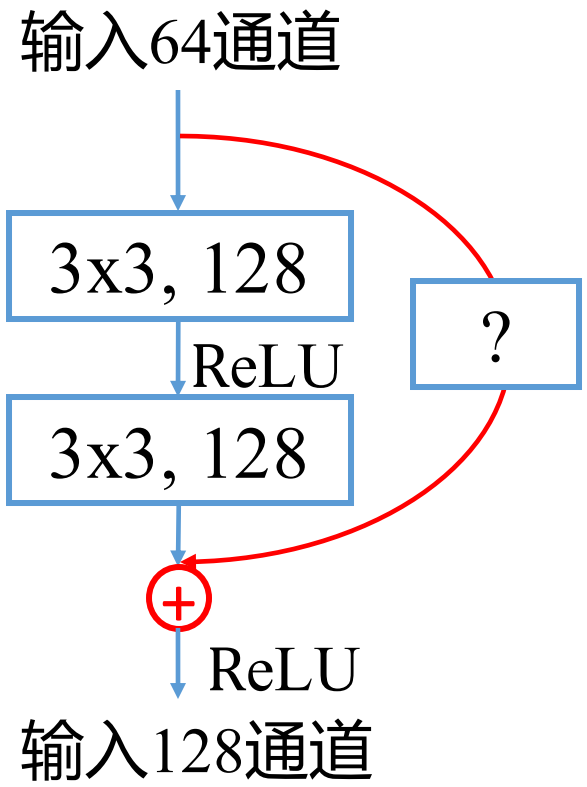
- bottleneck版本
- $y = W_3 f\left(W_2 f\left((W_1 x)\right)\right) + x$



Residual Blocks with W_s

If $\dim(x) \neq \dim(F)$,

$$y = F(x, \{W_i\}) + W_s x$$

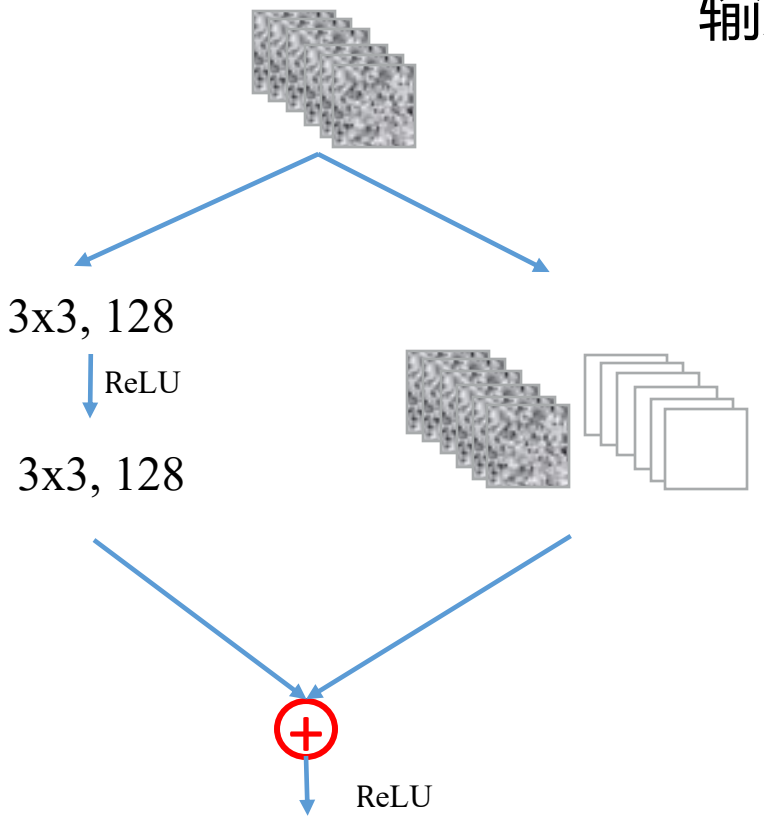


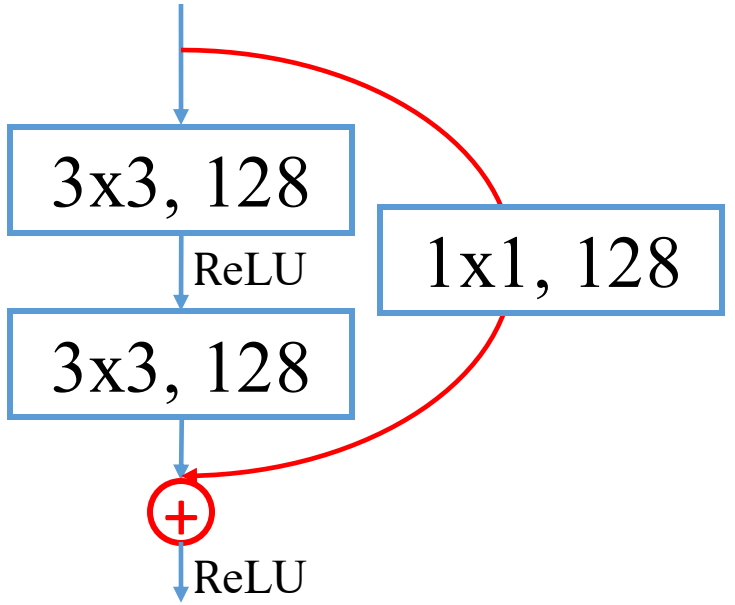
- W_s的两种形式
  - Zero-padding
  - Projection shortcuts

Residual Blocks with W_s

- Zero-padding (parameter-free)
- Projection shortcuts

输入：28x28, 64





输出：28x28, 128

Deep Residual Networks

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

Deep Residual Networks

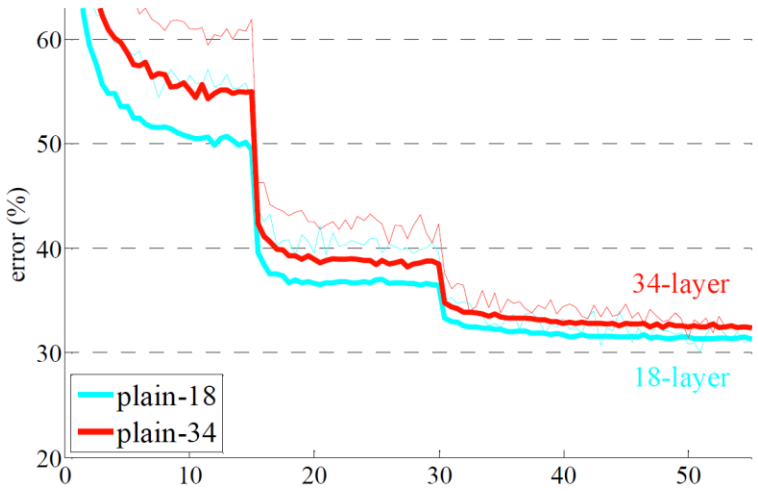
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

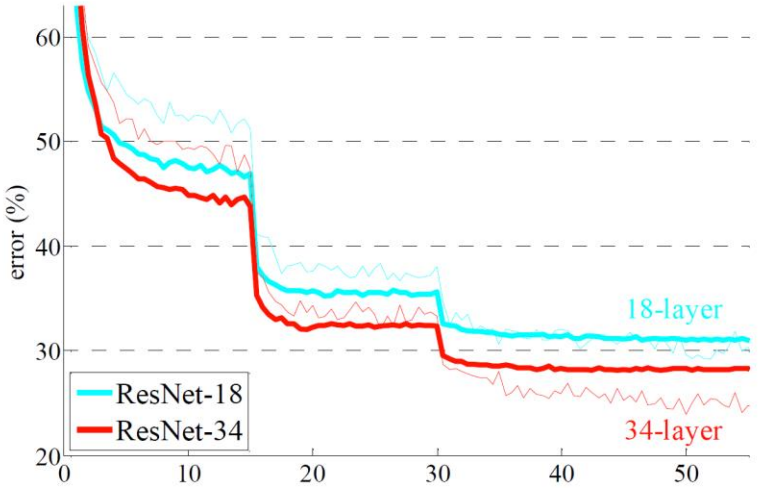
- conv3_1, conv4_1, conv5_1的步长为2，实现降采样

- Shortcut connection的三种方式
  - A. Zero-padding
  - B. 需要增加shortcut connection通道数时使用projection，其余保持identity connection
  - C. 所有shortcut connection均作用projection

Deep Residual Networks

ImageNet分类错误率（%）





- Plain Network: 与ResNets一样的网络结构，但无跨层连接
- ResNet-18与ResNet-34

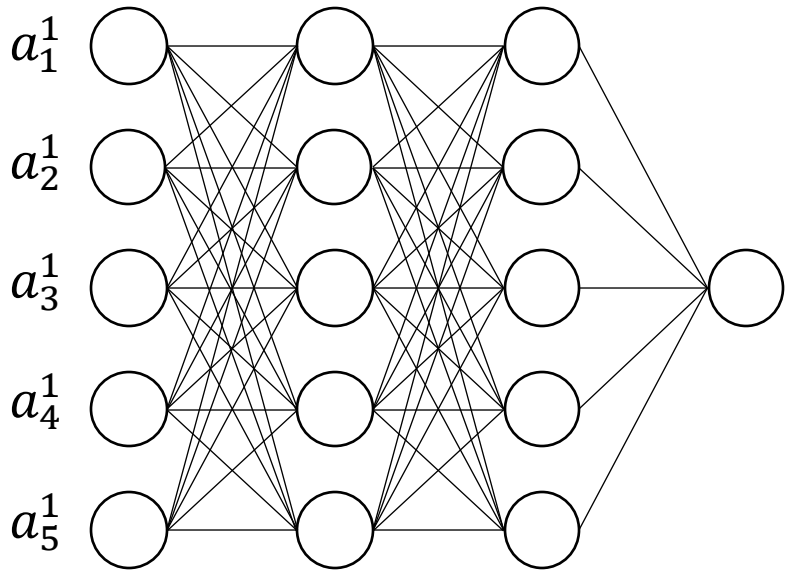
Deep Residual Networks

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

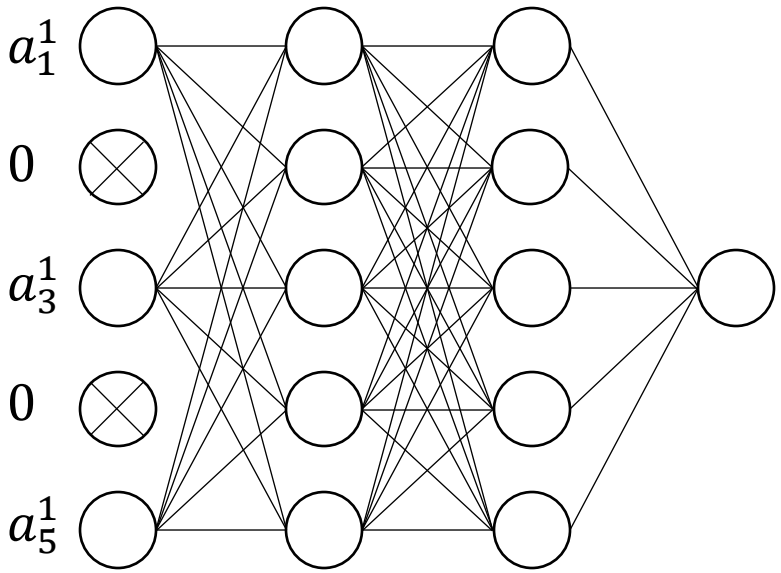
- ImageNet 验证集错误率
- Ablation study (消融实验)

- Shortcut connection的三种方式
  - A. Zero-padding
  - B. 需要增加shortcut connection通道数时使用projection，其余保持identity connection
  - C. 所有shortcut connection均作用projection

Dropout

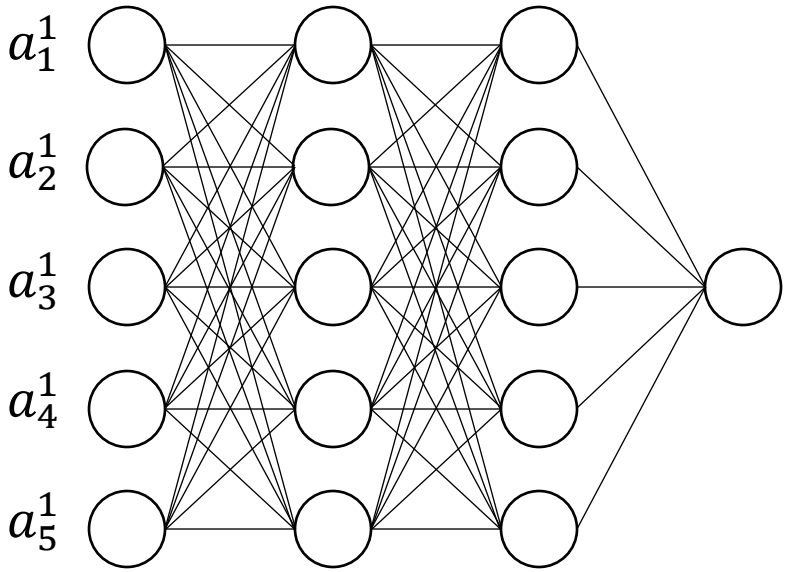


标准的全连接网络

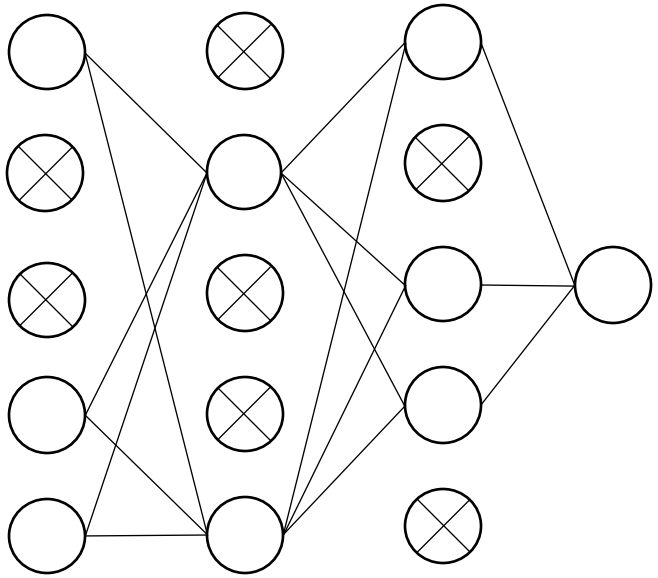


将a_2^1和a_4^1置为0

Dropout



标准的全连接网络

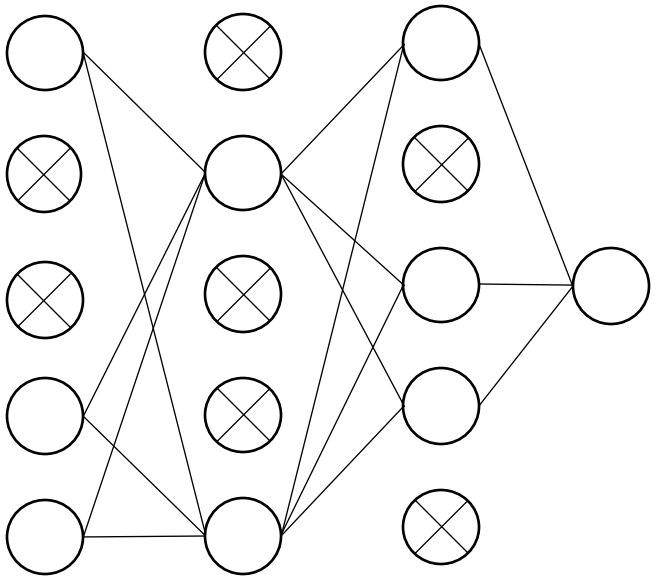


Dropout: 训练阶段以概率p^l随机地激活/禁止各神经元

训练阶段：$\hat{a}_i^l = a_i^l \circ Bernoulli(p^l)$

- 假设有n个神经元作用了Dropout，相当于训练了2^n个神经网络

Dropout



训练阶段：$\hat{a}_i^l = a_i^l \circ Bernoulli(p^l)$

- 测试阶段策略I：针对作用了Dropout神经元单独激活/禁止，即预测2^n回，对预测结果取平均，计算代价过高
- 测试阶段策略II： $\hat{a}_i^l = a_i^l \circ p^l$

Deep Residual Networks with Stochastic Depth

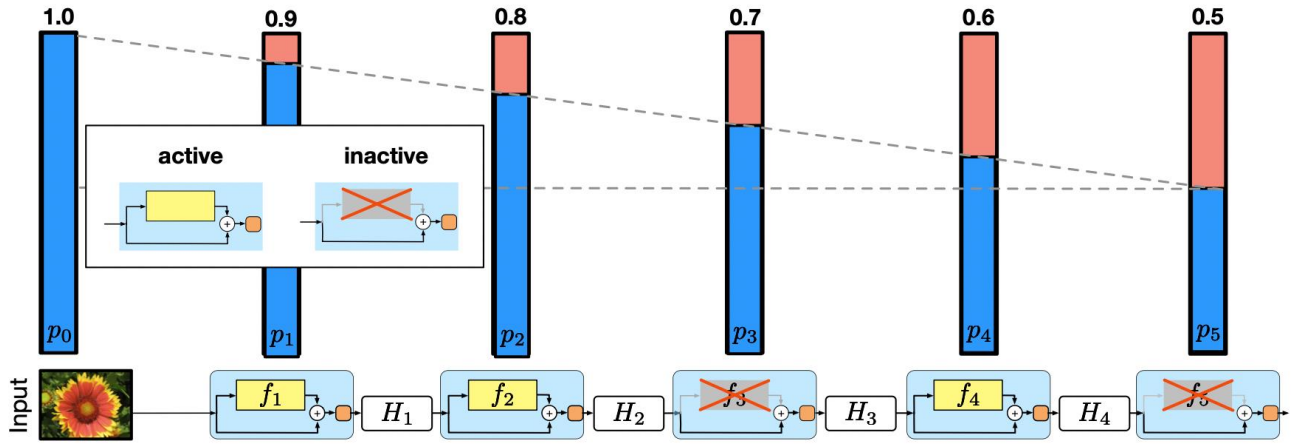


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

Table 1. Test error (%) of ResNets trained with stochastic depth compared to other most competitive methods previously published (whenever available). A "+" in the name denotes standard data augmentation. ResNet with constant depth refers to our reproduction of the experiments by He et al.

	CIFAR10+	CIFAR100+	SVHN	ImageNet
Maxout [21]	9.38	-	2.47	-
DropConnect [20]	9.32	-	1.94	-
Net in Net [24]	8.81	-	2.35	-
Deeply Supervised [13]	7.97	-	1.92	33.70
Frac. Pool [25]	-	27.62	-	-
All-CNN [6]	7.25	-	-	41.20
Learning Activation [26]	7.51	30.83	-	-
R-CNN [27]	7.09	-	1.77	-
Scalable BO [28]	6.37	27.40	1.77	-
Highway Network [29]	7.60	32.24	-	-
Gen. Pool [30]	6.05	-	1.69	28.02
ResNet with constant depth	6.41	27.76	1.80	21.78
ResNet with stochastic depth	5.25	24.98	1.75	21.98

大纲

- ResNet
- DenseNet
- Generative Adversarial Networks (GANs)

Dense Networks

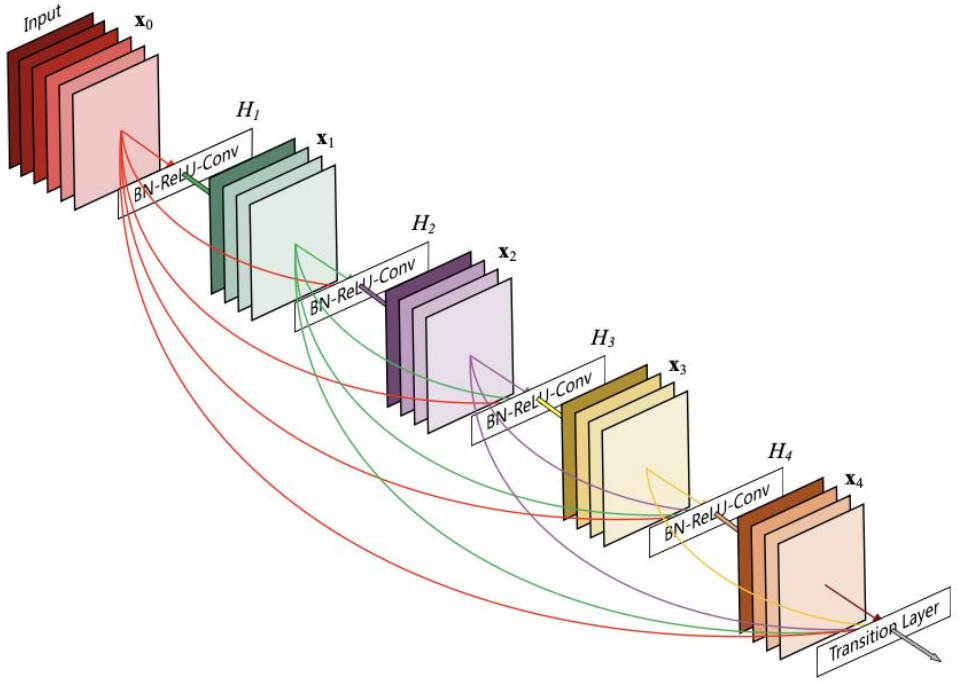


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Dense connectivity. To further improve the information flow between layers we propose a different connectivity pattern: we introduce direct connections from any layer to all subsequent layers. Figure 1 illustrates the layout of the resulting DenseNet schematically. Consequently, the ℓ^{th} layer receives the feature-maps of all preceding layers, $\mathbf{x}_0, \ldots, \mathbf{x}_{\ell-1}$, as input:

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]), \tag{2}$$

where $[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]$ refers to the concatenation of the feature-maps produced in layers $0, \ldots, \ell-1$. Because of its dense connectivity we refer to this network architecture as *Dense Convolutional Network (DenseNet)*. For ease of implementation, we concatenate the multiple inputs of $H_\ell(\cdot)$ in eq. (2) into a single tensor.

Dense Networks

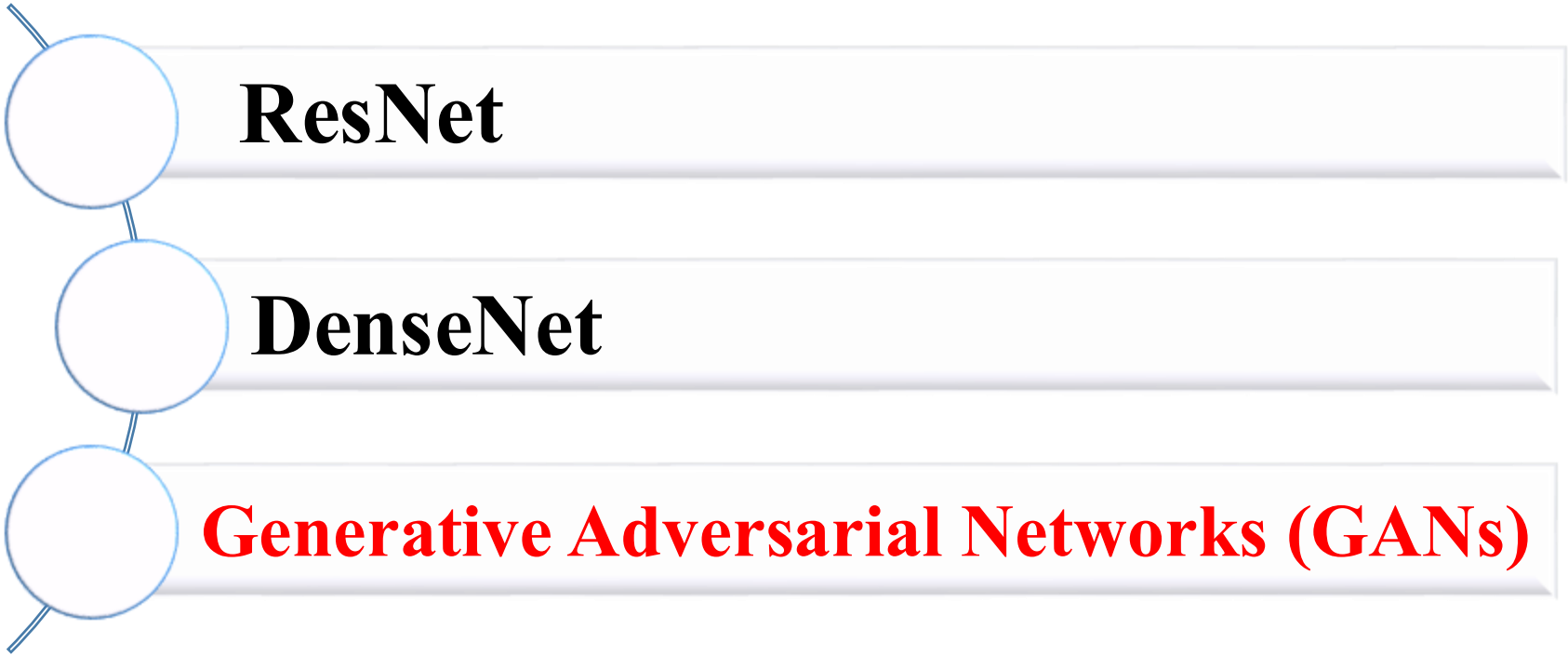
Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Dense Networks

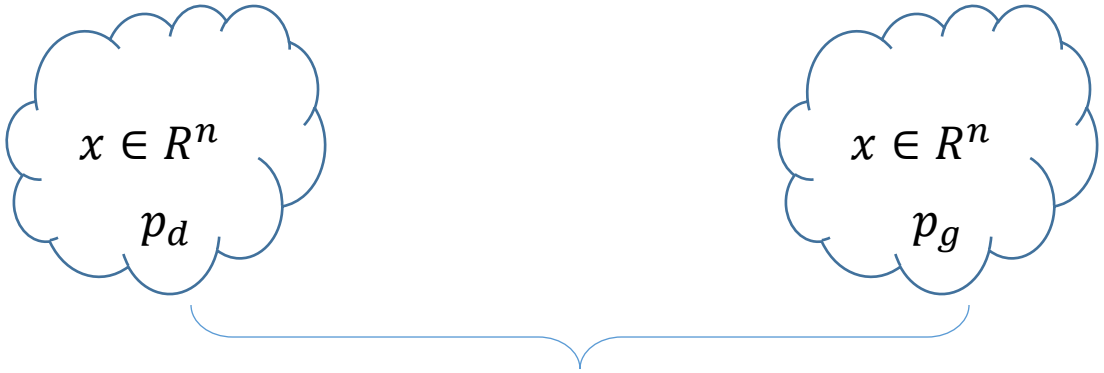
Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [31]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [33]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [41]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Table 2: Error rates (%) on CIFAR and SVHN datasets. k denotes network's growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). $*$ indicates results run by ourselves. All the results of DenseNets without data augmentation (C10, C100, SVHN) are obtained using Dropout. DenseNets achieve lower error rates while using fewer parameters than ResNet. Without data augmentation, DenseNet performs better by a large margin.

大纲



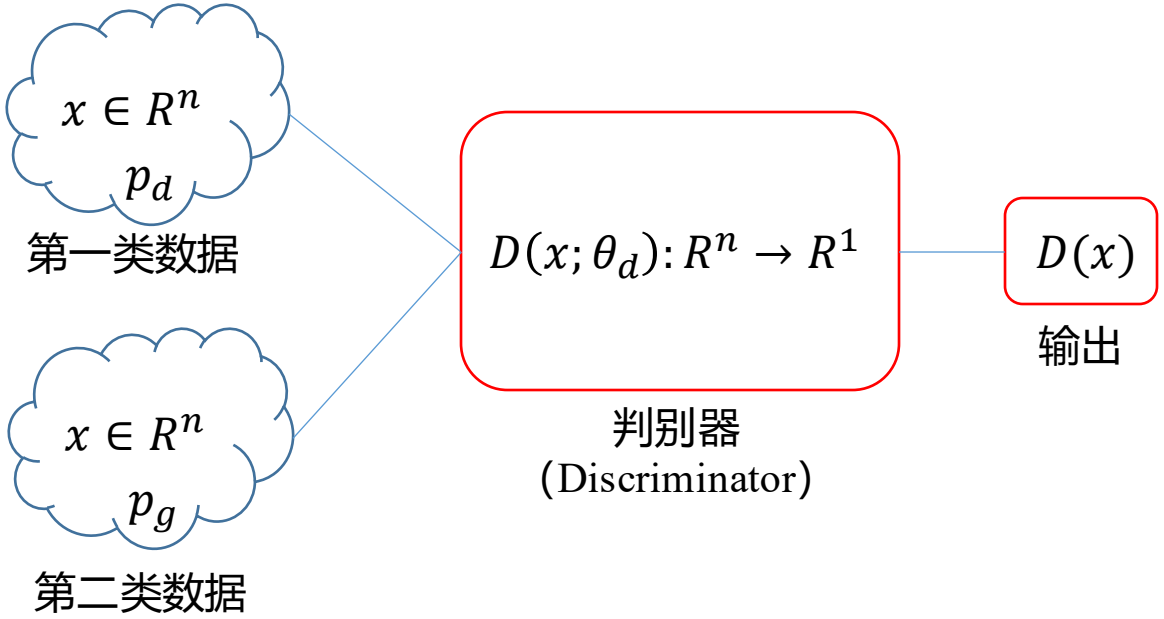
Discriminative Model



假设有两类数据$x \in R^n$，其中一类来自概率分布p_d，另一类来自于p_g

构建模型（Discriminative model, 判别器）对两类数据分类

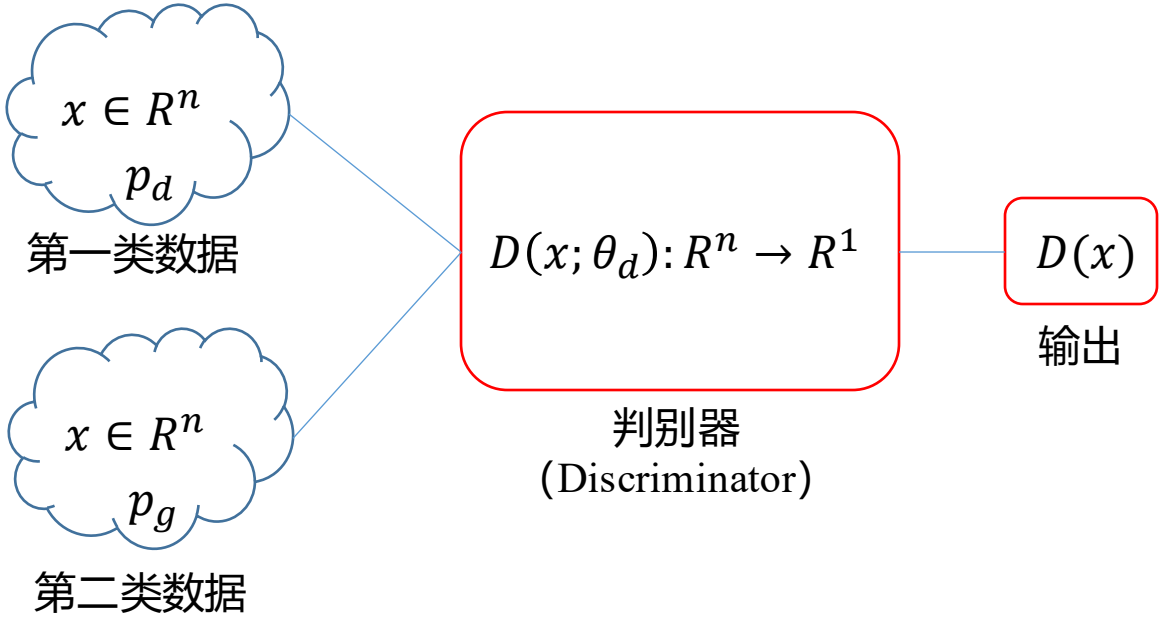
Discriminative Model



目标:学习θ_d

- 定义判别器D，其参数为θ_d，其输入为x（来自于p_d 或p_g分布），输出为$D(x) \in (0,1)$
- $x \sim p_d$对应标签为1， $x \sim p_g$对应标签为0
- 针对$x \sim p_d$，模型输出$D(x)$ 应接近1；针对$x \sim p_g$， $D(x)$ 应接近0

Cost Function of Discriminative Model



- 对于 $x \sim p_d$, 我们希望$D(x) \to 1$

$$D(x) \to 1 \Leftrightarrow \min_D(1 - D(x)) \Leftrightarrow \max_D D(x) \Leftrightarrow \max_D log D(x)$$

- 对于 $x \sim p_g$,我们希望$D(x) \to 0$

$$D(x) \to 0 \Leftrightarrow \min_D D(x) \Leftrightarrow \max_D log(1 - D(x))$$

Viewpoint of Probability Distribution

概率分布的角度

- $x \sim p_d$时，$\max_D logD(x) \Leftrightarrow \max_D E_{x\sim p_d}[logD(x)]$
- $x \sim p_g$时，$\max_D log(1-D(x)) \Leftrightarrow \max_D E_{x\sim p_g}[log(1-D(x))]$
- $V(G,D) = E_{x\sim p_d}[logD(x)] + E_{x\sim p_g}[log(1-D(x))]$, 目标是$\max V(G,D)$

Algorithm of Optimization of Discriminative Model

模型训练过程中, 随机采样 m 个样本来估计 $V(G,D)$

$$V(G,D) \approx \frac{1}{m}\sum_{i=1}^{m}[logD(x^i) + log\left(1 - D(x^i)\right)]$$

判别器训练算法

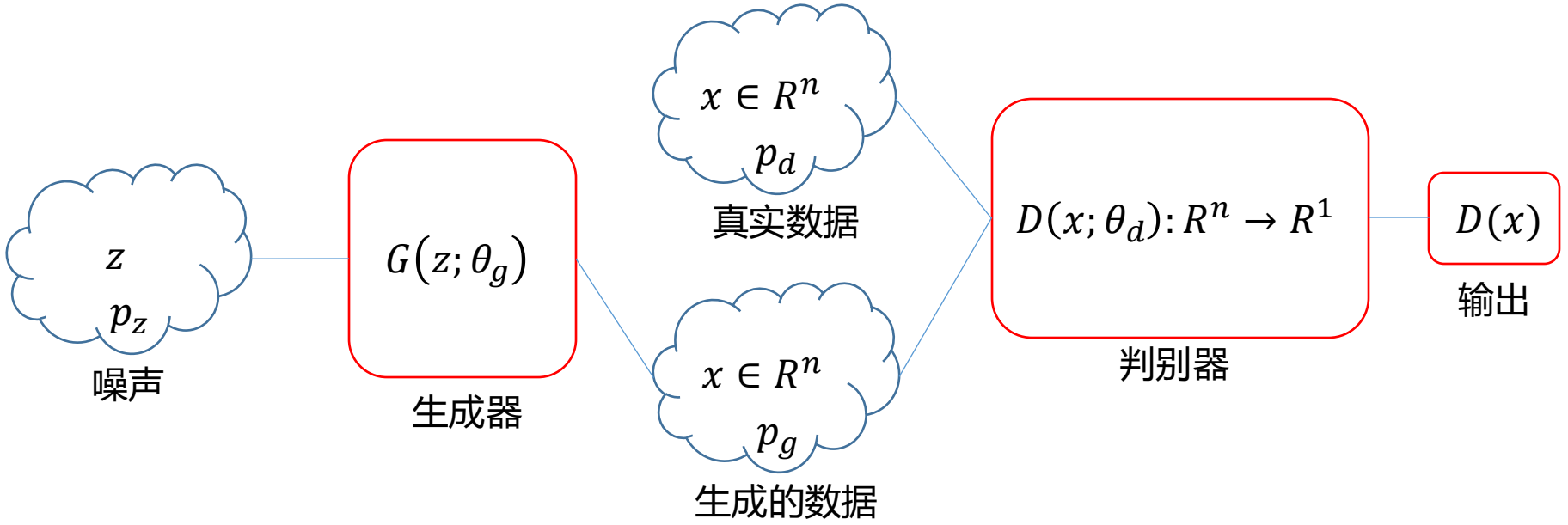
for k steps do

- 从 p_d分布中采样 m 个样本 $\{x_d^1, \dots, x_d^m\}$
- 从p_g 分布中采样m 个样本 $\{x_g^1, \dots, x_g^m\}$
- 沿着梯度上升的方向更新判别器:

$$\nabla_{\theta_d}\frac{1}{2m}\sum_{i=1}^{2m}[logD(x_d^i) + log\left(1 - D(x_g^i)\right)]$$

endfor

Generative Model



- 生成器和判别器“左右互搏”，生成器生成假钞，判别器辨别钞票
  - 生成器：制作出最逼真的假钞
  - 判别器：辨别出所有真钞与假钞

Cost Function of Generative Model

- 生成器生成的数据为$G(z)$，其中z代表随机噪声
- 训练生成器时，我们希望生成的数据骗过判别器，因此$G(z)$对应标签为1，即$\max_G \log D(G(z))$
- $\max_G \log D(G(z)) \Leftrightarrow \min_G \log(1 - D(G(z)))$

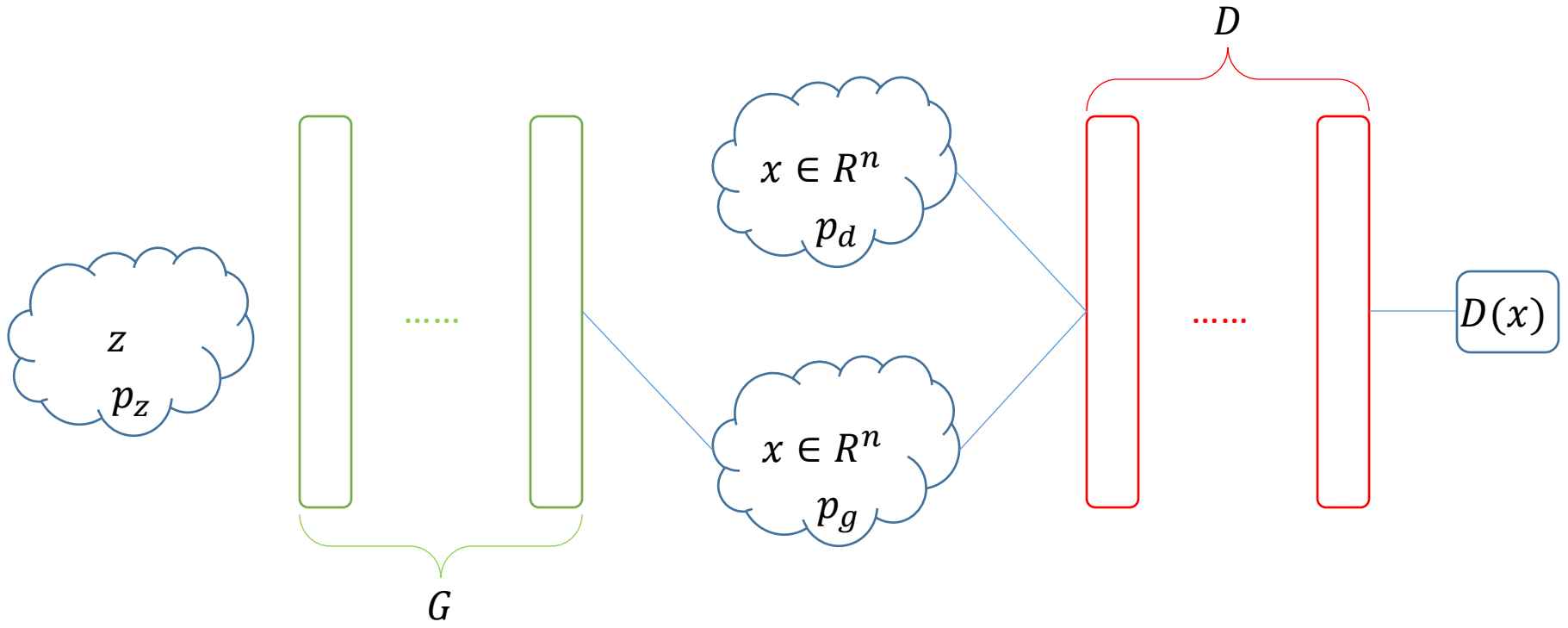
从概率分布p_z中采样m个样本$\{z^1, \dots, z^m\}$，训练生成器的代价函数如下：

$$\frac{1}{m}\sum_{i=1}^{m} \log(1 - D(G(z^i)))$$

沿着梯度下降的方向更新生成器：

$\nabla_{\theta_g} \frac{1}{m}\sum_{i=1}^{m} \log(1 - D(G(z^i)))$

Implementation of Neural Networks



两阶段优化

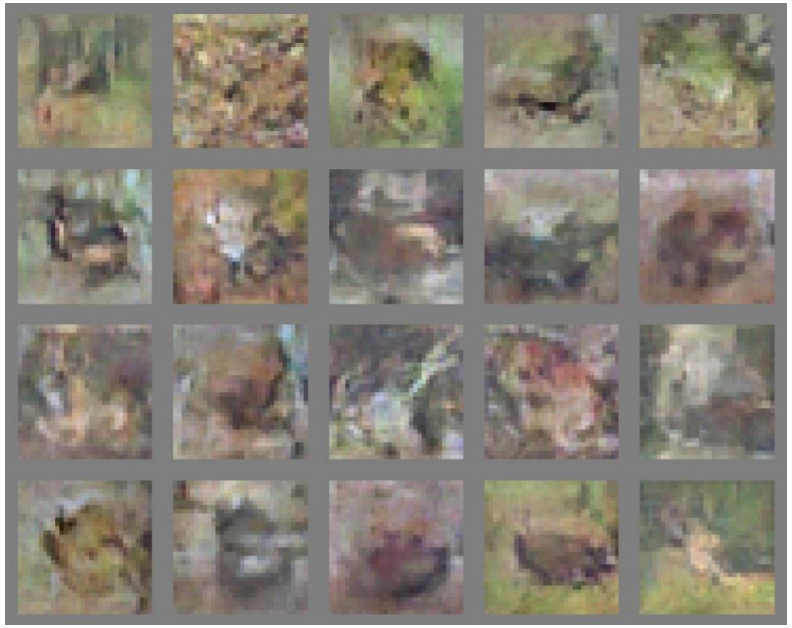
for number of training iterations do

第一阶段: 固定G，优化 D

第二阶段: 固定 D，优化 G

end for

生成图像的效果



初代GAN

如今的GAN

GAN的各类变体

GAN

ACGAN

BGAN

CGAN

DCGAN

EBGAN

fGAN

GoGAN

⋮ ⋮

- 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
- 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
- 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
- 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
- ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
- ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- ACGAN - Coverless Information Hiding Based on Generative adversarial networks
- acGAN - On-line Adaptative Curriculum Learning for GANs
- ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
- AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- AE-OT - Latent Space Optimal Transport for Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference (github)

https://github.com/hindupuravinash/the-gan-zoo

课后作业

- 使用ResNet分类CIFAR-10数据集
  - 对比普通CNN与ResNet在训练集和验证集上的loss曲线
  - 对比普通CNN与ResNet的测试集准确率

https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py

- 训练一个针对MNIST的GAN，并画出GAN生成的图像

https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/gan/gan.py

谢 谢!