机器学习-第九章 深度卷积神经网络Normalization

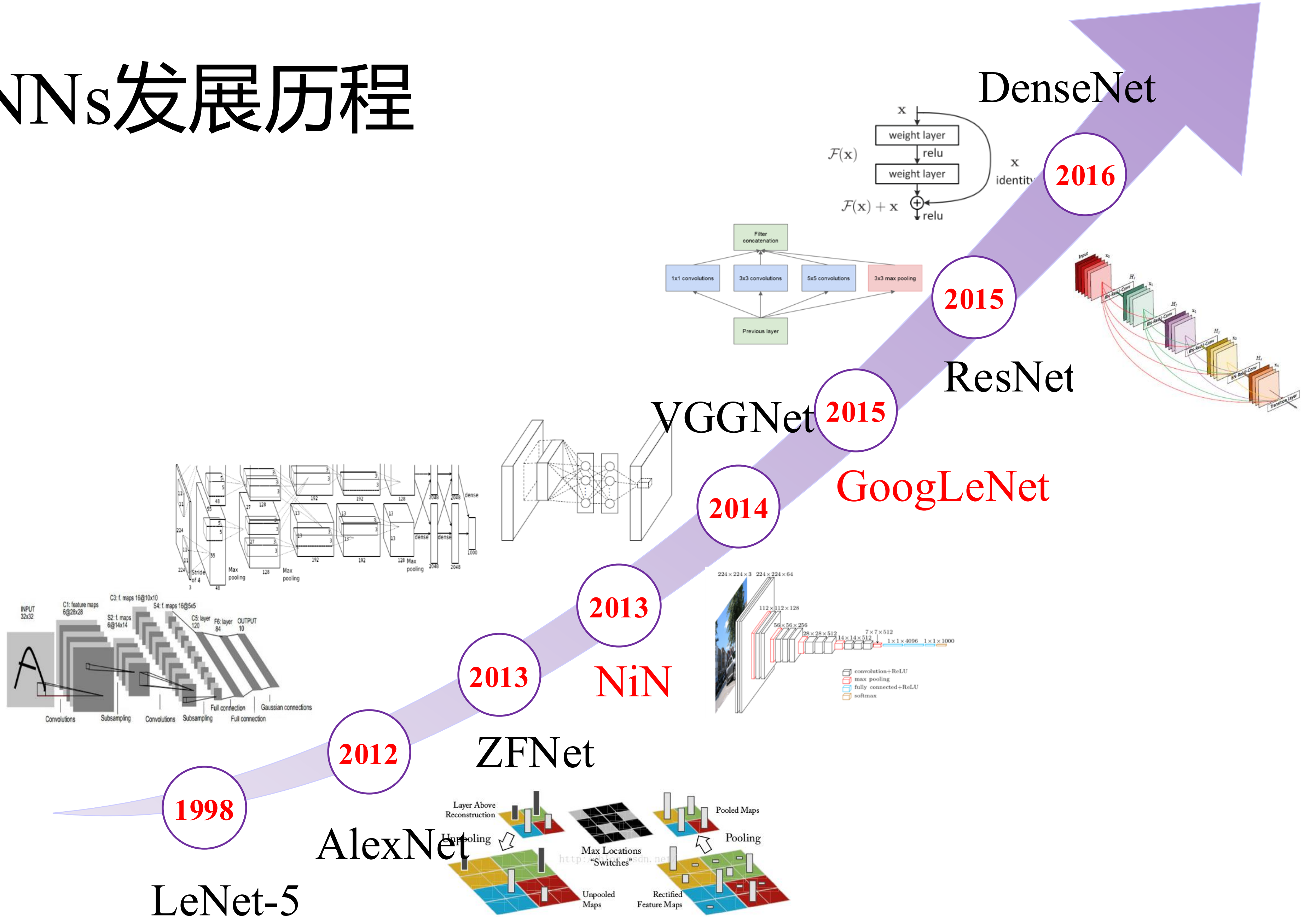
教师：胡俊杰 助理研究员

邮箱：hujunjie@scu.edu.cn

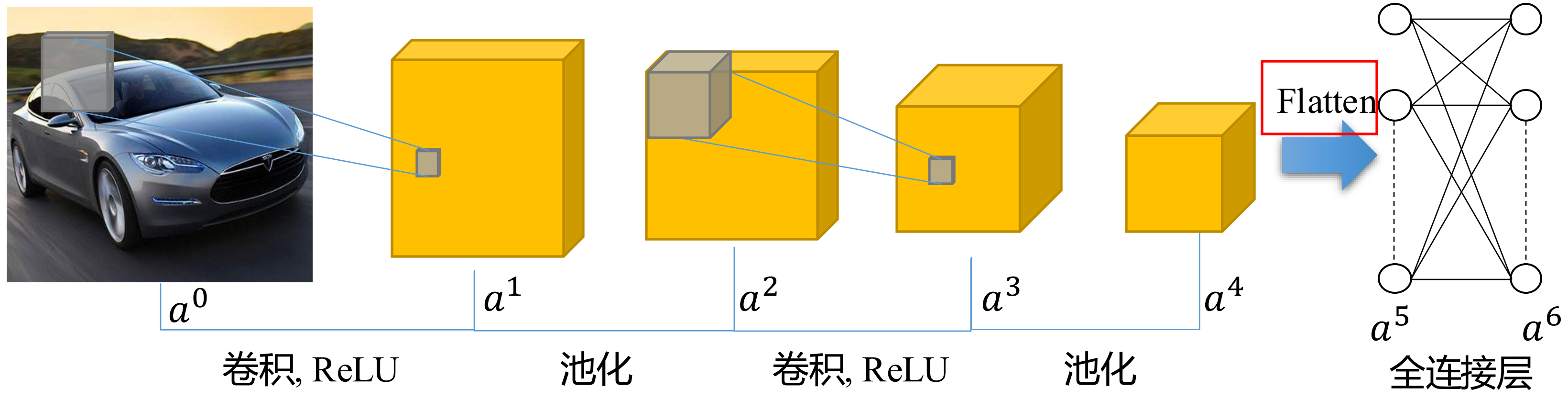
大纲

- Inception模块
- Batch Normalization
- Batch Normalization变体

CNNs发展历程


DenseNet
2016
ResNet
2015
VGGNet
2015
GoogLeNet
2014
NiN
2013
2013
ZFNet
2012
AlexNet
1998
LeNet-5
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
x
identity
Filter concatenation
1x1 convolutions
3x3 convolutions
5x5 convolutions
3x3 max pooling
Previous layer
INPUT 32x32
C1: feature maps 6@28x28
S2: f. maps 6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer 120
F6: layer 84
OUTPUT 10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections
Layer Above Reconstruction
Unpooling
Max Locations "Switches"
Pooled Maps
Pooling
Unpooled Maps
Rectified Feature Maps
convolution+ReLU
max pooling
fully connected+ReLU
softmax

Global Average Pooling (GAP)

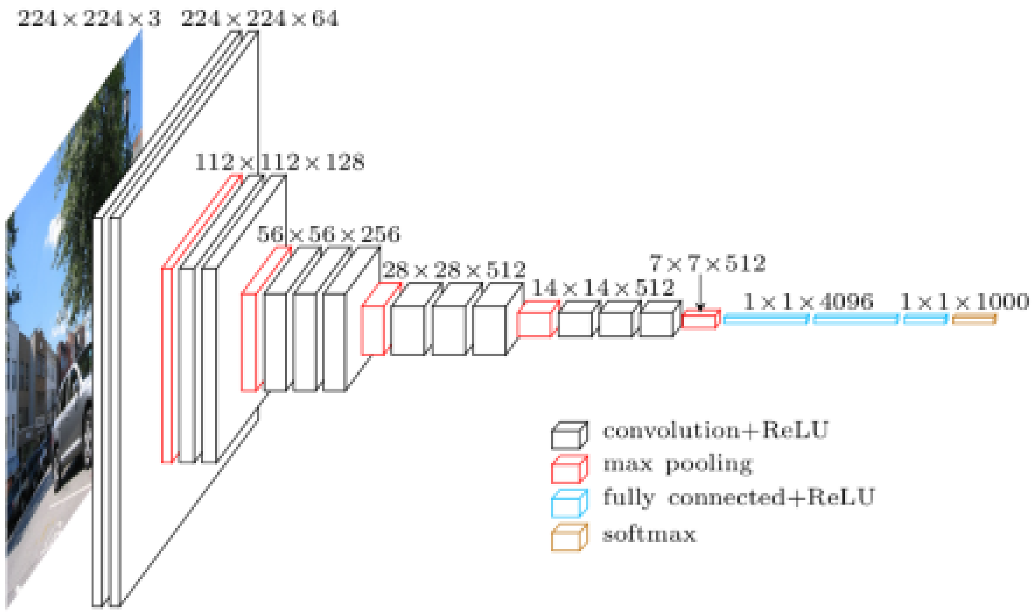


$$a_{GAP} = \frac{1}{WH}\sum_{w=1}^{W}\sum_{h=1}^{H} a$$

- GAP可显著降低最后全连接层的参数量

[*] M. Lin, Q. Chen, and S. Yan. Network in Network[C]. ICLR, 2014.

VGG

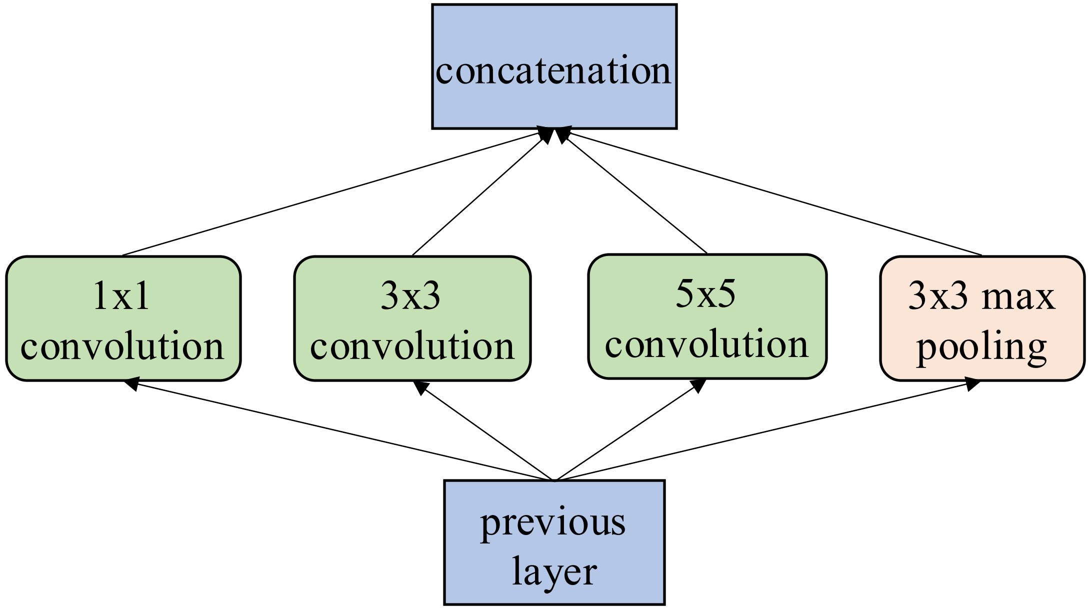


Very deep convolutional networks for large-scale image recognition
K Simonyan, A Zisserman - arXiv preprint arXiv:1409.1556, 2014 - arxiv.org
In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3x3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16-19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the …
☆ 保存 99 引用 被引用次数：78397 相关文章 所有 39 个版本

- VGG交替使用卷积和池化
- 小卷积核（3x3）可高效地实现图像中的抽象特征提取

[*] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[C]. arXiv, 2014.

Inception-V1 普通版本

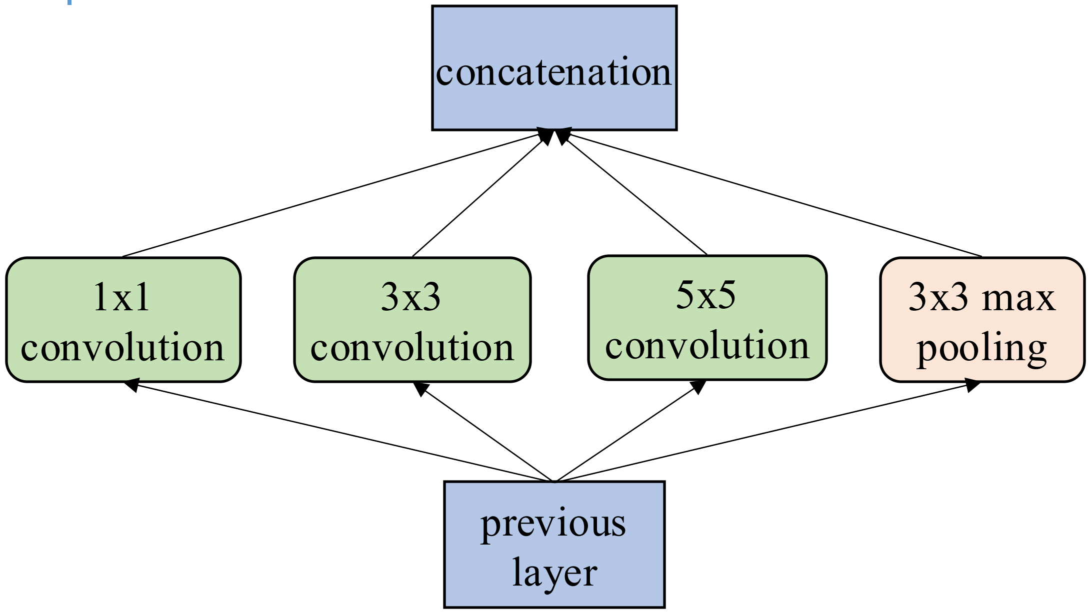


Inception-V1普通版本

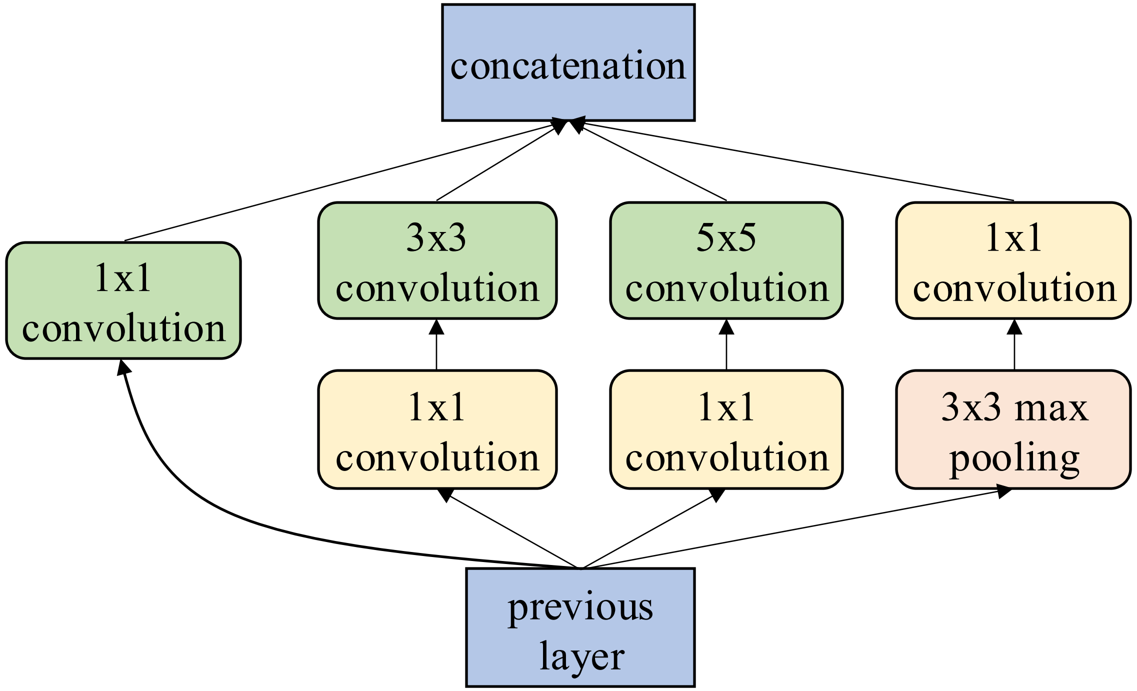
- 使用1x1, 3x3, 5x5大小的卷积核，步长均为1
- 使用3x3 大小的最大值池化

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

Inception-V1



Inception-V1普通版本

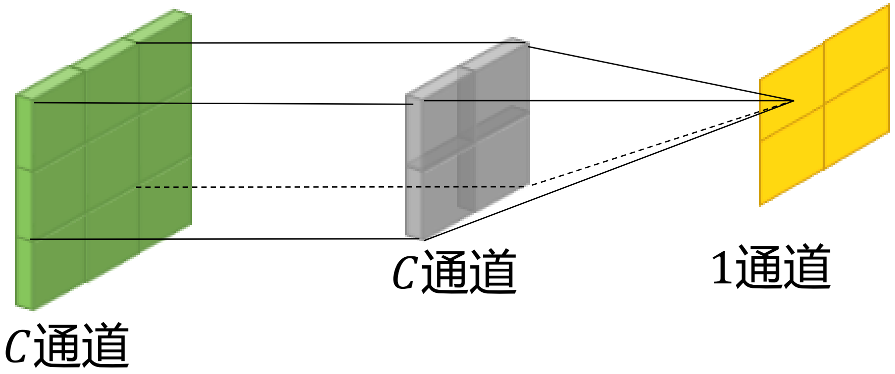


Inception-V1

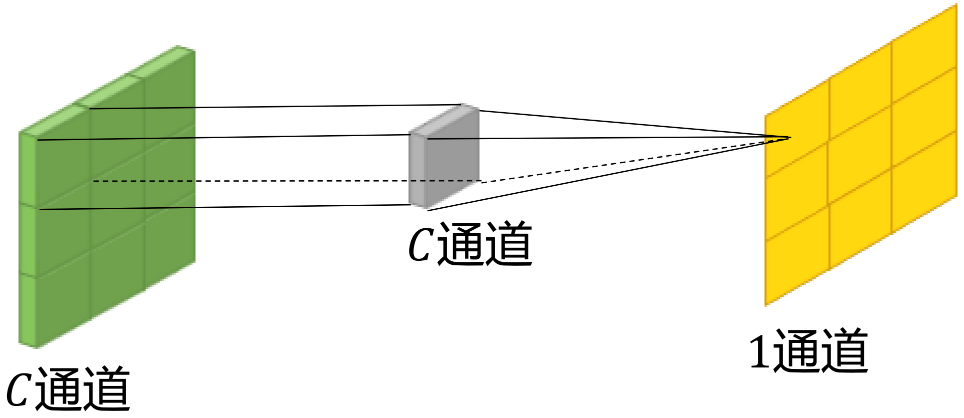
- 对3x3 、5x5和最大值池化使用1x1卷积减少通道数目

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

Inception-V1



3x3卷积

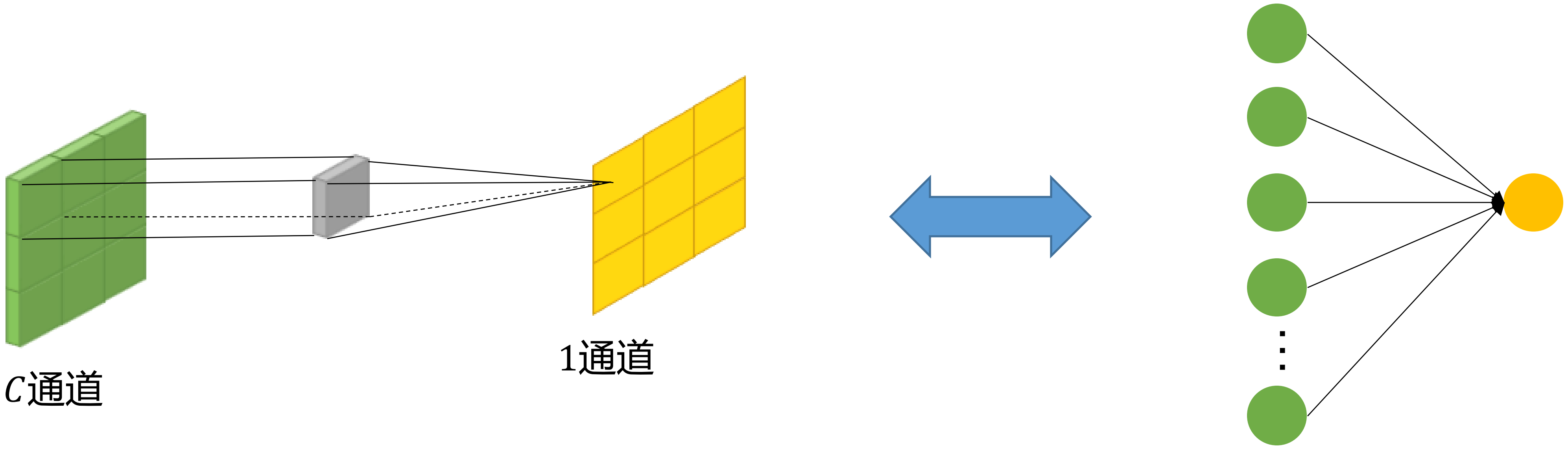


1x1卷积

- C代表通道数

- 单次1x1卷积运算等价于全连接网络
- 1x1卷积可高效降低特征的通道数目

Inception-V1



1x1卷积

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

GoogLeNet

Auxiliary classifier

Auxiliary classifier

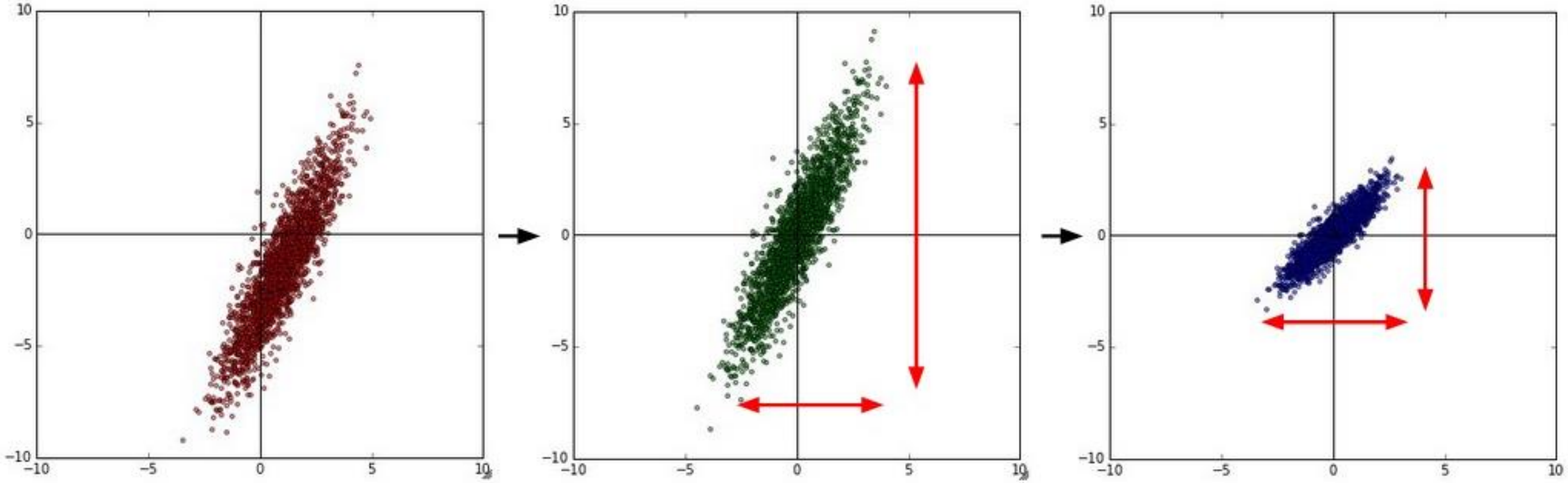
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

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

大纲

- Inception模块
- Batch Normalization
- Batch Normalization变体

Normalization



x（二维）的分布

$\hat{x} := x - \mu$

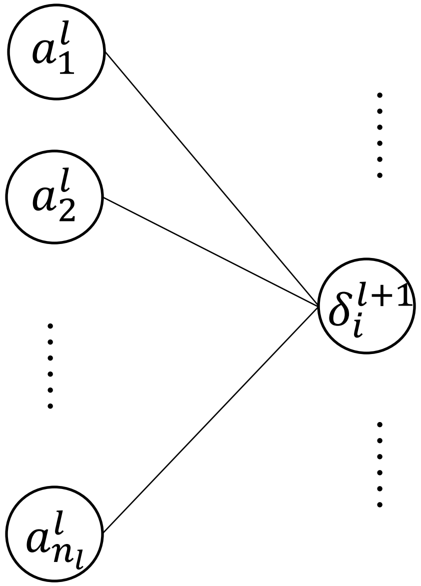
Zero-centered:
以原点为中心

$\tilde{x} := \dfrac{\hat{x}}{\sigma}$

Scaled: 方差为1

[*] CS231n Convolutional Neural Networks for Visual Recognition

Normalization

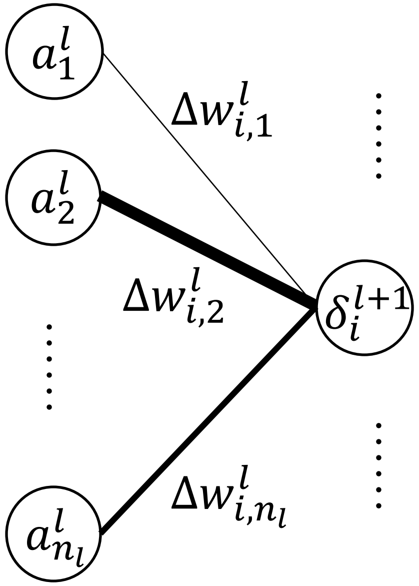


$$\frac{\partial J}{\partial w_{i,*}^l} = \delta_i^{l+1} a_*^l$$

- **Zero-centered**

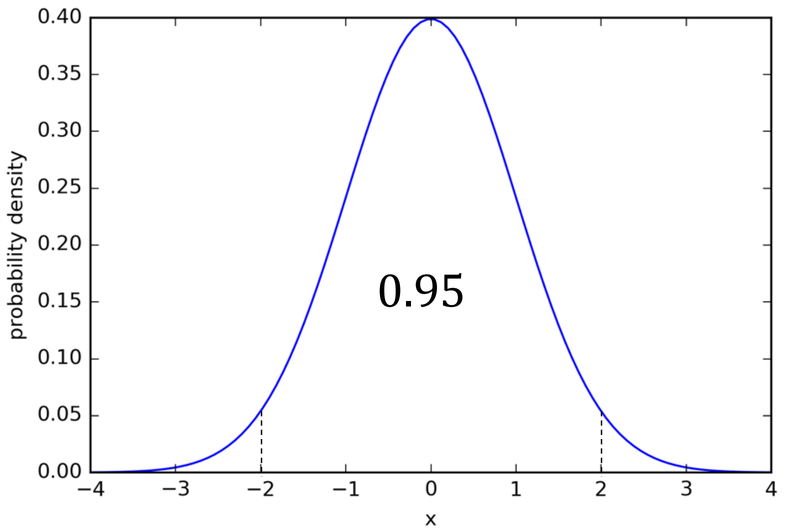
  当a_*^l均为正或均为负时，sign($\frac{\partial J}{\partial w_{i,*}^l}$)将相同，且只取决于 sign($\delta_i^{l+1}$)

- **Scaled**

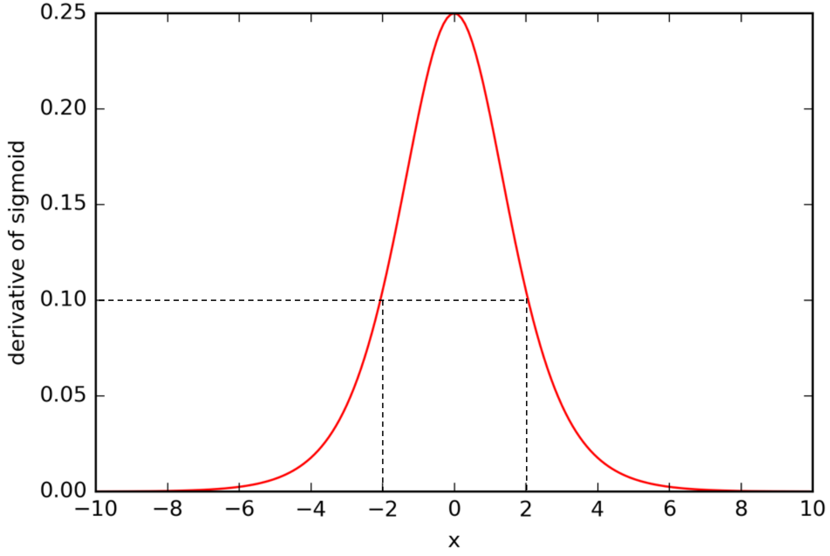


[*] LeCun Y A, Bottou L, Orr G B, et al. Efficient backprop[M]. Neural networks: Tricks of the trade, 1998.

Normalization



标准正态分布的概率密度函数

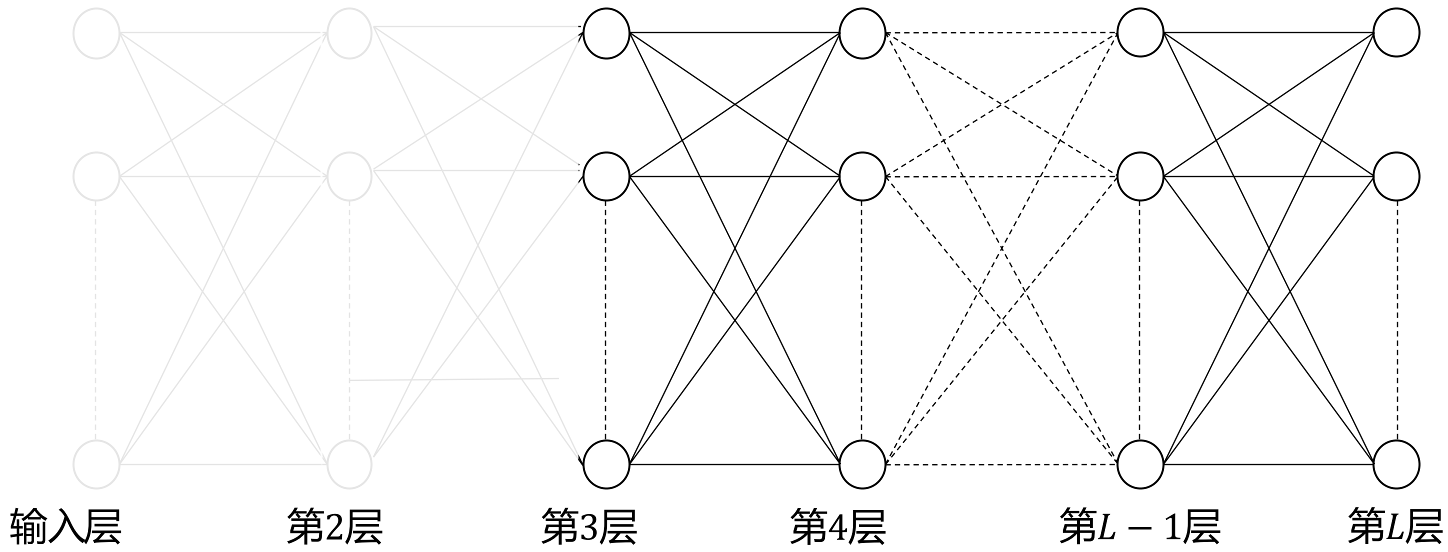


Sigmoid的导数

$$\delta_i^l = \boxed{\dot{f}\left(z_i^l\right)} \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^l$$

- 归一化可缓解sigmoid函数的导数落入饱和区

Normalization

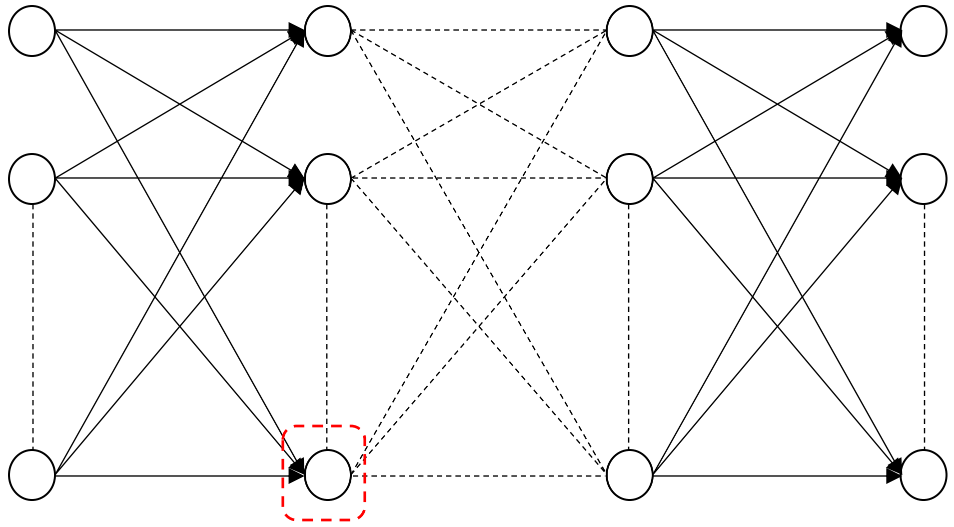


- 任意中间隐藏层及之后的部分均可视为子网络（sub-network）
- 除开对网络输入进行归一化，是否可对隐藏层特征也归一化?

Batch Normalization

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

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

- 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

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

- 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?

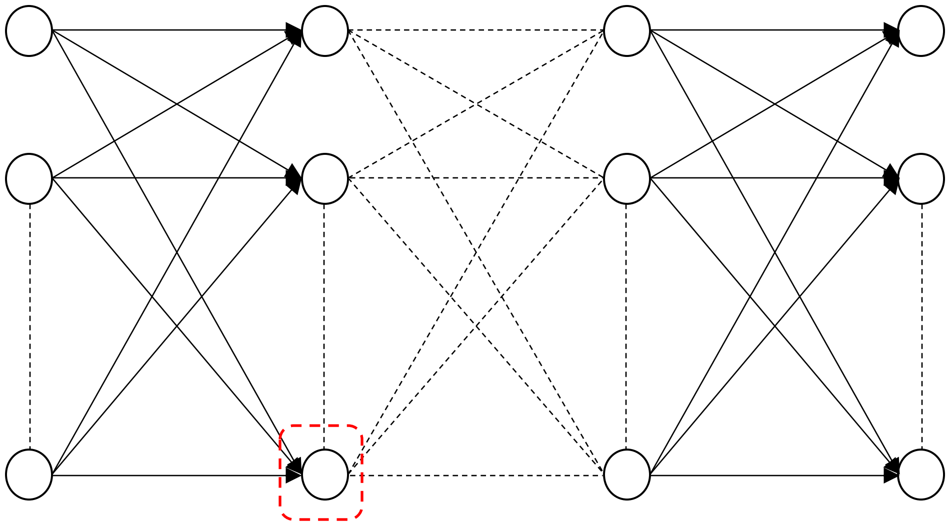
- 在训练时，BN使用指数滑动平均（Exponential moving average）方法估计全体训练集的μ和σ

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

- 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



$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$\hat{z}_i^{l+1} = \frac{z_i^{l+1} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

$$\tilde{z}_i^{l+1} = \gamma_i \hat{z}_i^{l+1} + \beta_i$$

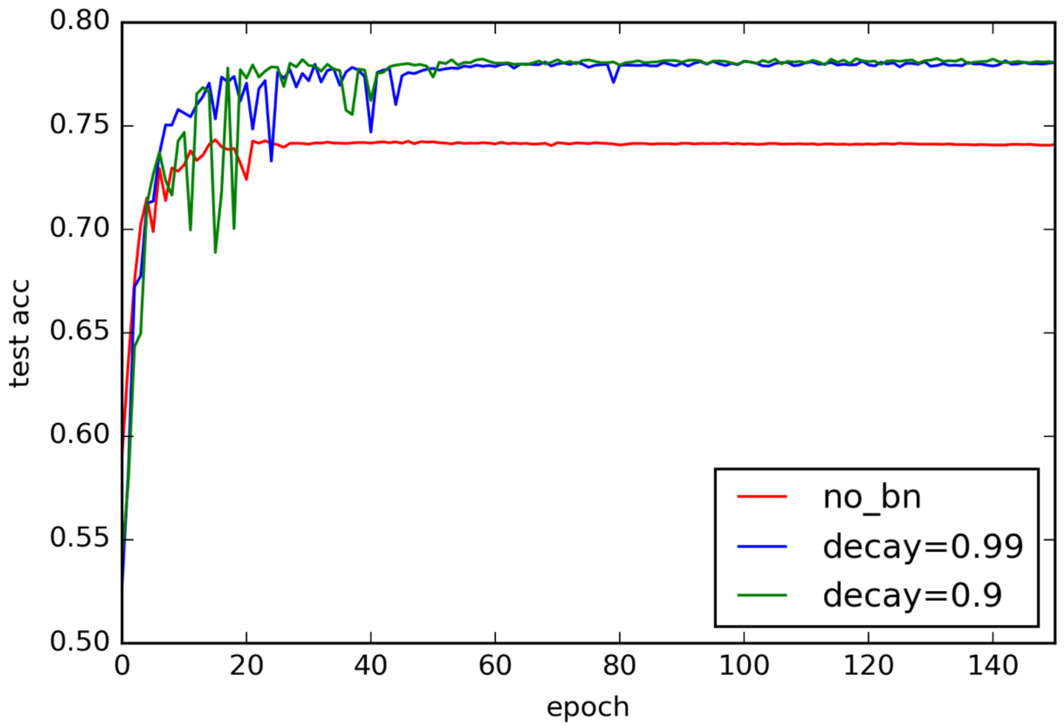
卷积神经网络

- 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对比实验

- 数据集: Cifar10
- 网络: LeNet-5
- 模型
  - LeNet-5 无 batch normalization
  - LeNet-5 有 batch normalization，且$decay = 0.99$
  - LeNet-5 有 batch normalization，且$decay = 0.9$

Batch Normalization对比实验



	无BN	有BN，Decay=0.99	有BN，Decay=0.9
测试集准确率	74.33%	78.16%	78.26%

大纲

- Inception模块
- Batch Normalization
- Batch Normalization变体

Batch Normalization (BN)

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

- 训练阶段，BN中μ和σ依赖于batch大小
- batch过小时，难以准确估计全局μ和σ，这将影响网络在测试阶段性能

若μ和σ的计算不依赖batch，则无上述问题，即非batch的Normalization

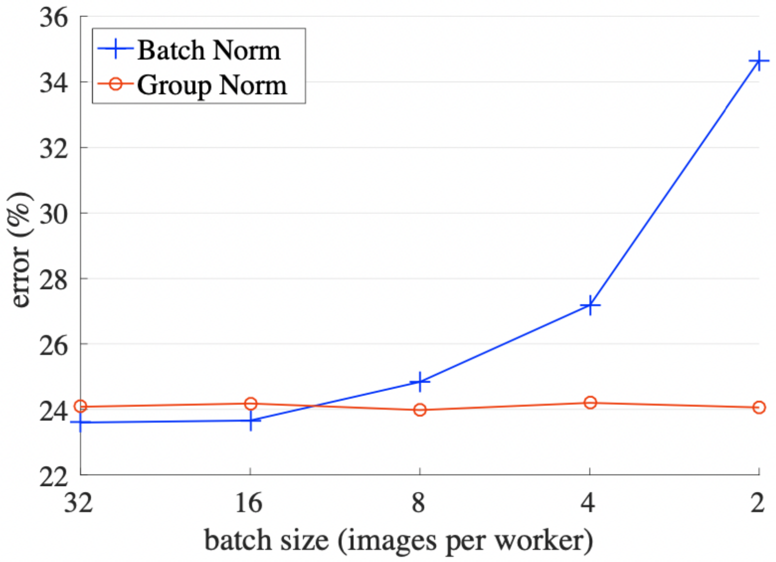


Figure 1. **ImageNet classification error *vs*. batch sizes.** This is a ResNet-50 model trained in the ImageNet training set using 8 workers (GPUs), evaluated in the validation set.

Batch Normalization (BN)

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

- 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

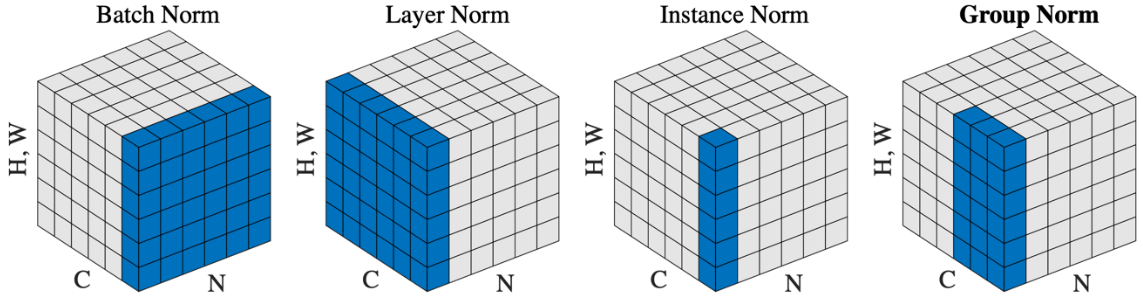


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

课后作业

- 使用pytorch实现Batch Normalization模块，针对CIFAR-10数据集
  - 对比有/无Batch Normalization的训练集和验证集loss曲线
  - 对比有/无Batch Normalization的测试集准确率

```
class BatchNorm2d(nn.Module):
    def __init__(self, num_features, num_dims, momentum=0.9, eps=1e-5):
        super().__init__()
        if num_dims == 2:
            shape = (1, num_features)
        else:
            shape = (1, num_features, 1, 1)
        self.gamma = nn.Parameter(torch.ones(shape))
        self.beta = nn.Parameter(torch.zeros(shape))
        self.moving_mean = torch.zeros(shape)
        self.moving_var = torch.ones(shape)
        self.momentum = momentum
        self.eps = eps

    def forward(self, X):
        # TODO 实现BN
        return Y
```

谢 谢！