西安交通大学
XI'AN JIAOTONG UNIVERSITY

Week 1,2 Report

杨舜禹

Contents

一、神经网络的起源

二、训练方法

三、正则化方法

四、卷积神经网络

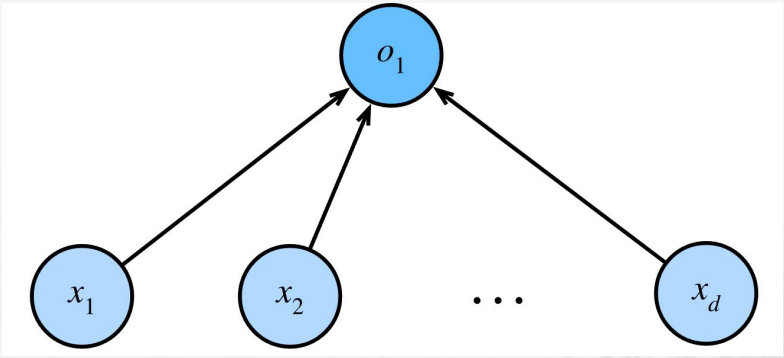
五、数据集介绍

六、模型训练结果

神经网络的起源

线性回归

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b$$



损失函数

$$l(\mathbf{w}, b) = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(\mathbf{w}^T \mathbf{x} + b - y)^2$$

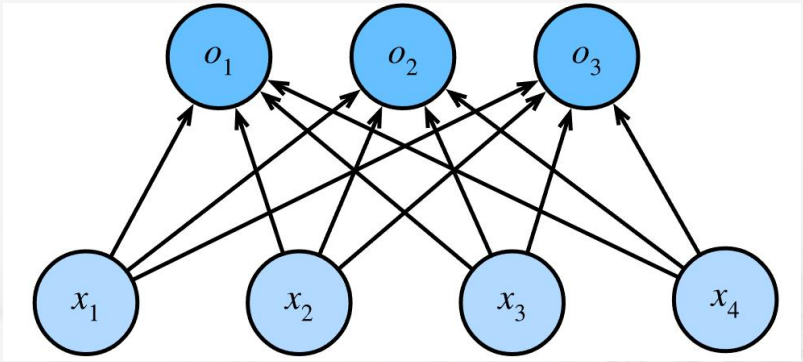
$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l(\mathbf{w}, b)$$

神经网络的起源

线性模型 + Softmax = 多分类器

$$\mathbf{o}_{q\times 1} = \mathbf{W}_{q\times d}\mathbf{x}_{d\times 1} + \mathbf{b}_{q\times 1}$$

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}), \hat{y}_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)}$$



损失函数

$$l(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{j=1}^{q} y_j \log \hat{y}_j$$

神经网络的起源

线性模型的局限性

默认输入与输出之间呈线性 (Linear) 关系，或者至少是单调 (Monotonic) 关系

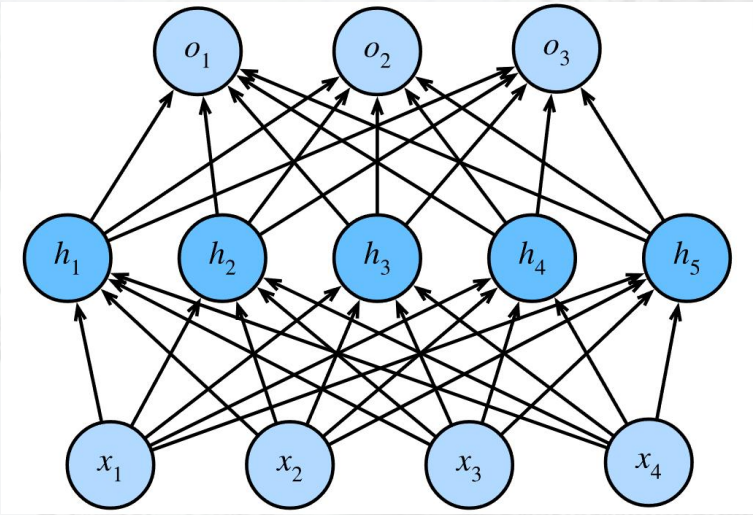
多个线性层 + 非线性函数 = 多层感知机

将多个线性层叠在一起，中间添加非线性函数

$$\mathbf{h}_{h\times 1} = \sigma(\mathbf{z}_{h\times 1}) = \sigma(\mathbf{W}^{(1)}_{h\times d}\mathbf{x}_{d\times 1} + \mathbf{b}^{(1)}_{h\times 1})$$

$$\mathbf{o}_{q\times 1} = \mathbf{W}^{(2)}_{q\times h}\mathbf{h}_{h\times 1} + \mathbf{b}^{(2)}_{q\times 1}$$

理论上，可以拟合任意函数关系



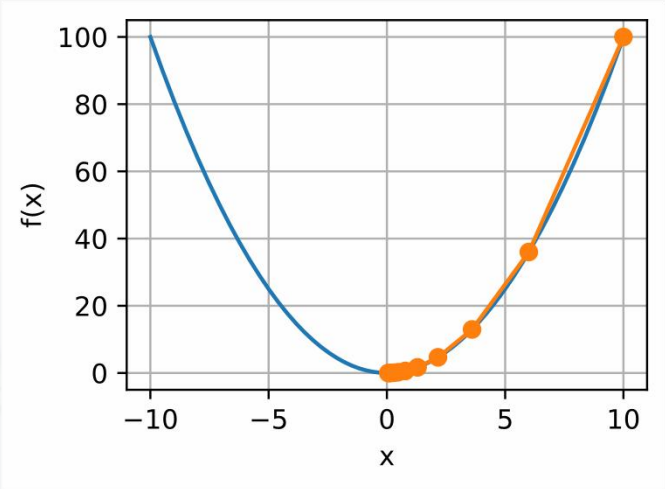
训练方法

梯度下降 GD

向损失函数的负梯度方向前进

$$\mathbf{W}^{(1)} = \mathbf{W}^{(1)} - \alpha \frac{\partial L}{\partial \mathbf{W}^{(1)}}$$

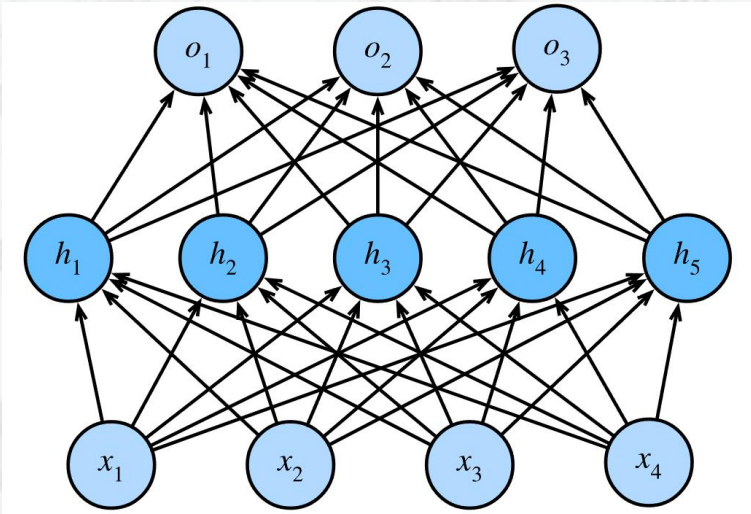
每次迭代，都根据全体样本计算 Loss



反向传播计算梯度

依据梯度的链式法则

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial L}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}}$$



训练方法

随机梯度下降 SGD

每次迭代，用一个样本计算梯度

$$\mathbf{W}^{(1)} = \mathbf{W}^{(1)} - \alpha \frac{\partial l^{(i)}}{\partial \mathbf{W}^{(1)}}$$

小批量随机梯度下降

GD 和 SGD 的折中方案：每次迭代，用一组样本计算梯度

动量 Momentum

每次迭代，除了本轮的梯度 g 外，还要参考上次的更新方向

$$v_t = \beta v_{t-1} + g$$

$$\mathbf{W}^{(1)} = \mathbf{W}^{(1)} - \alpha \mathbf{v}_t$$

正则化方法

Weight Decay - L2 正则化

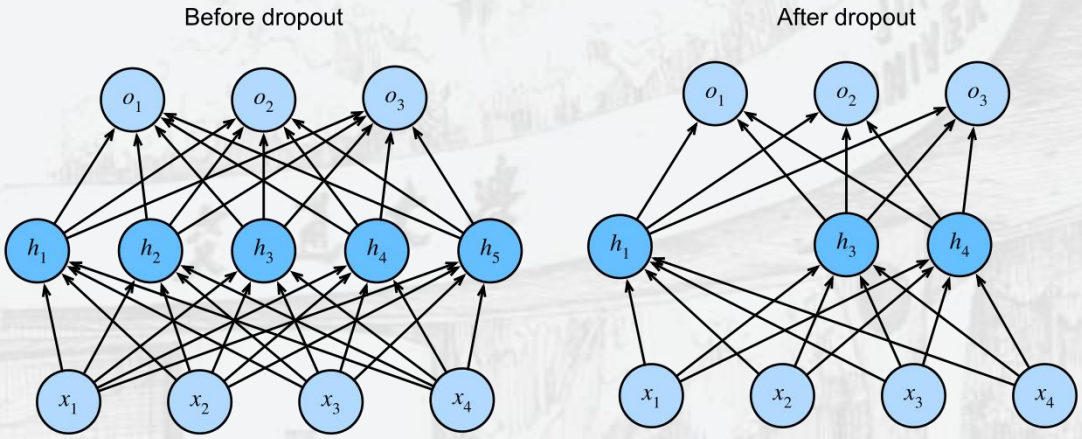
在损失函数中，惩罚过大的参数，以此限制参数的大小

相比直接限制参数量（比如限制多项式回归中的次数），更加温和

$$L'(\mathbf{w},b) = L(\mathbf{w},b) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Dropout

在训练时，每轮计算都随机删除一些节点，减少模型对特定 activation pattern 的依赖，从而减少受噪声的影响



正则化方法

Early Stopping

原理：在含噪声的数据中，模型倾向于先拟合干净的数据，然后再拟合噪声

操作：训练时监测 validation error，若减小幅度不超过 ε，则终止训练

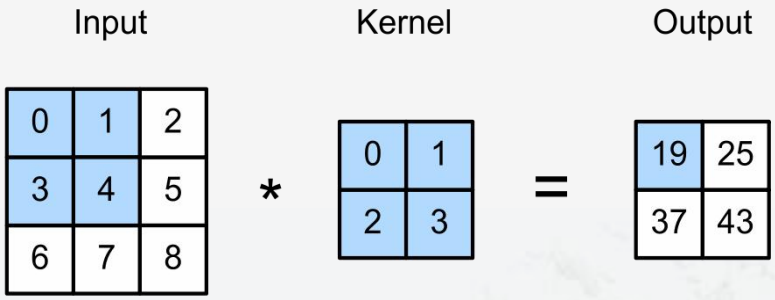
Batch Normalization

- 让模型数值稳定，避免不同层数值差异引起的学习率不同
- 正则化（限制模型数值过大），限制过拟合

$$\mathbf{h} = \sigma(BN(\mathbf{W}\mathbf{x} + \mathbf{b}))$$

卷积神经网络

卷积操作



卷积的意义——局部性

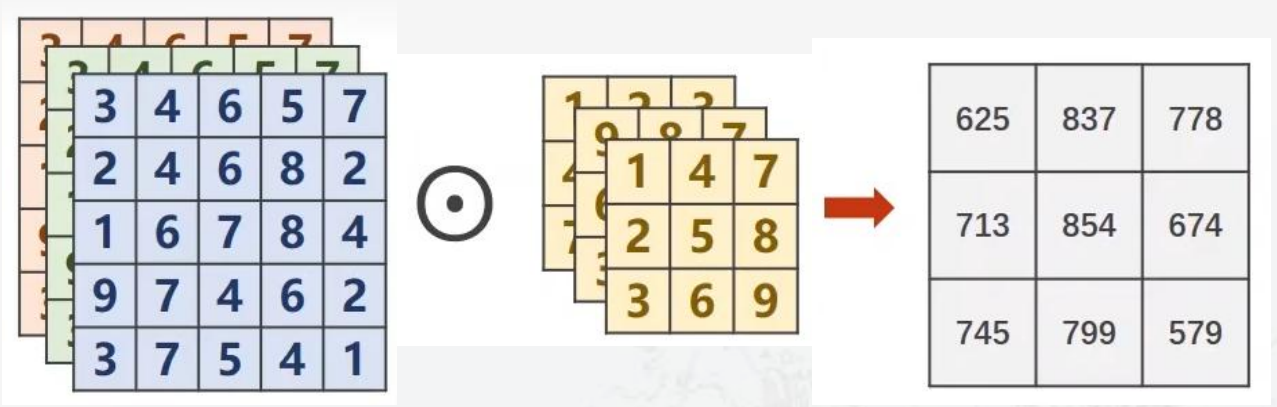
对不同区域的相同特征

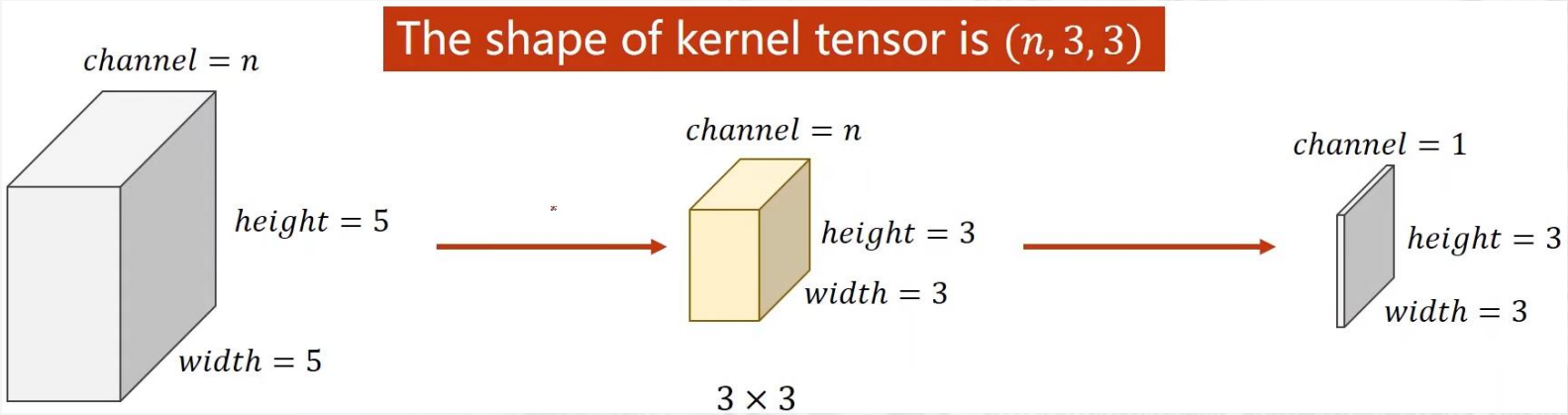
应当有相同的反应

*当然，卷积的参数量也远少于全连接

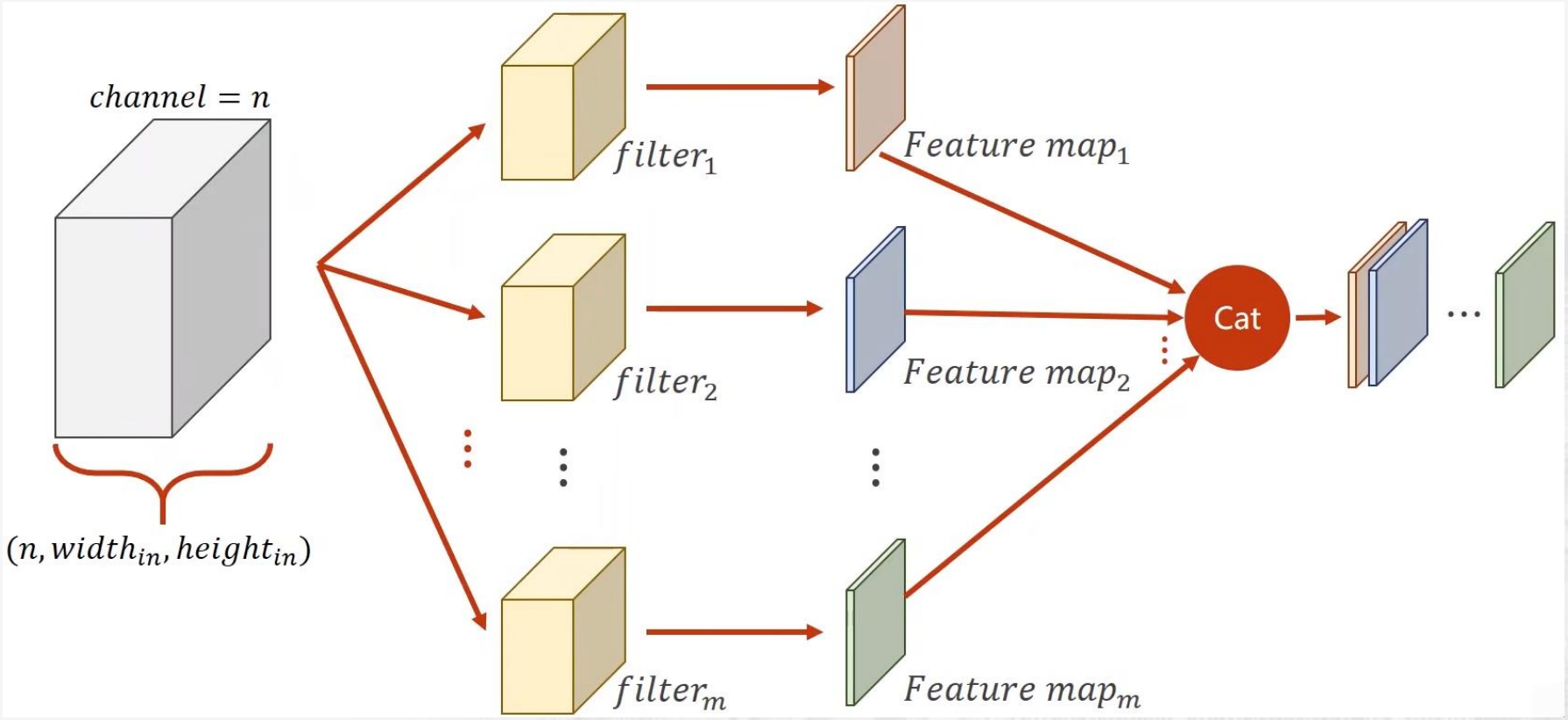
“What Waldo looks like does not depend upon where Waldo is located”

多个输入通道——卷积核也要多通道

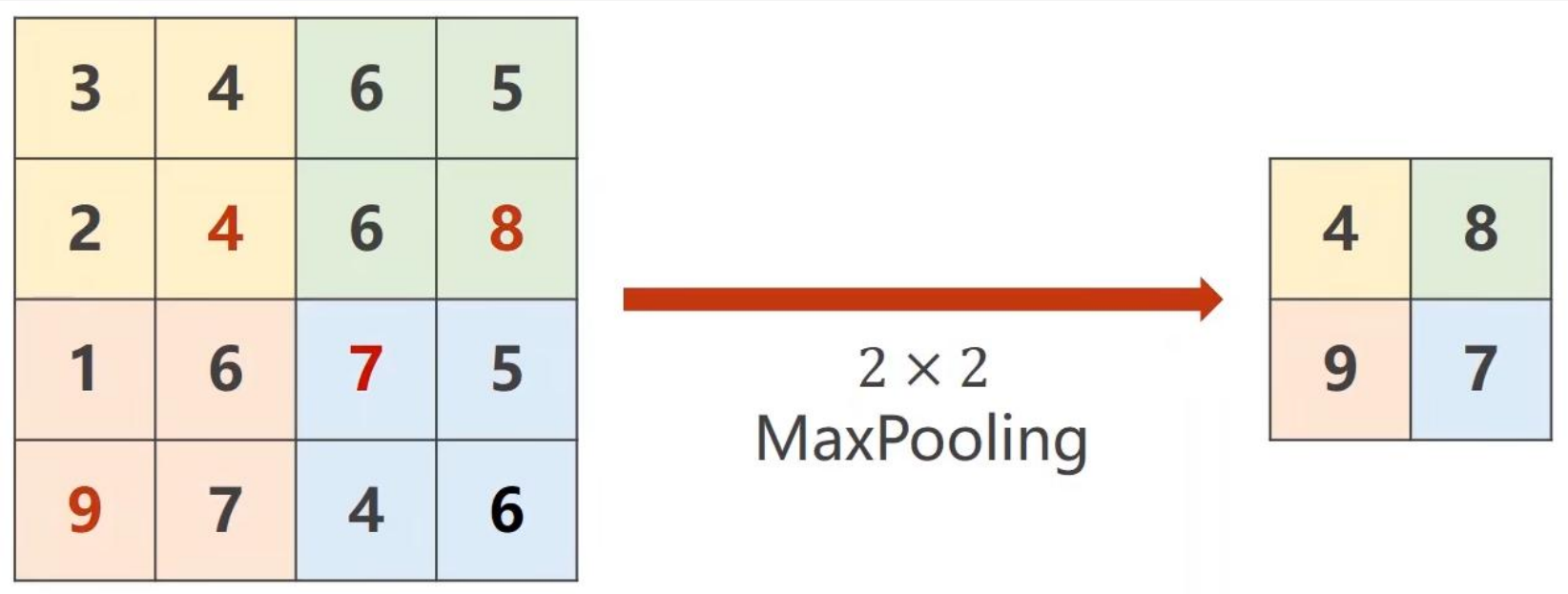




多个输出通道——使用多个卷积核

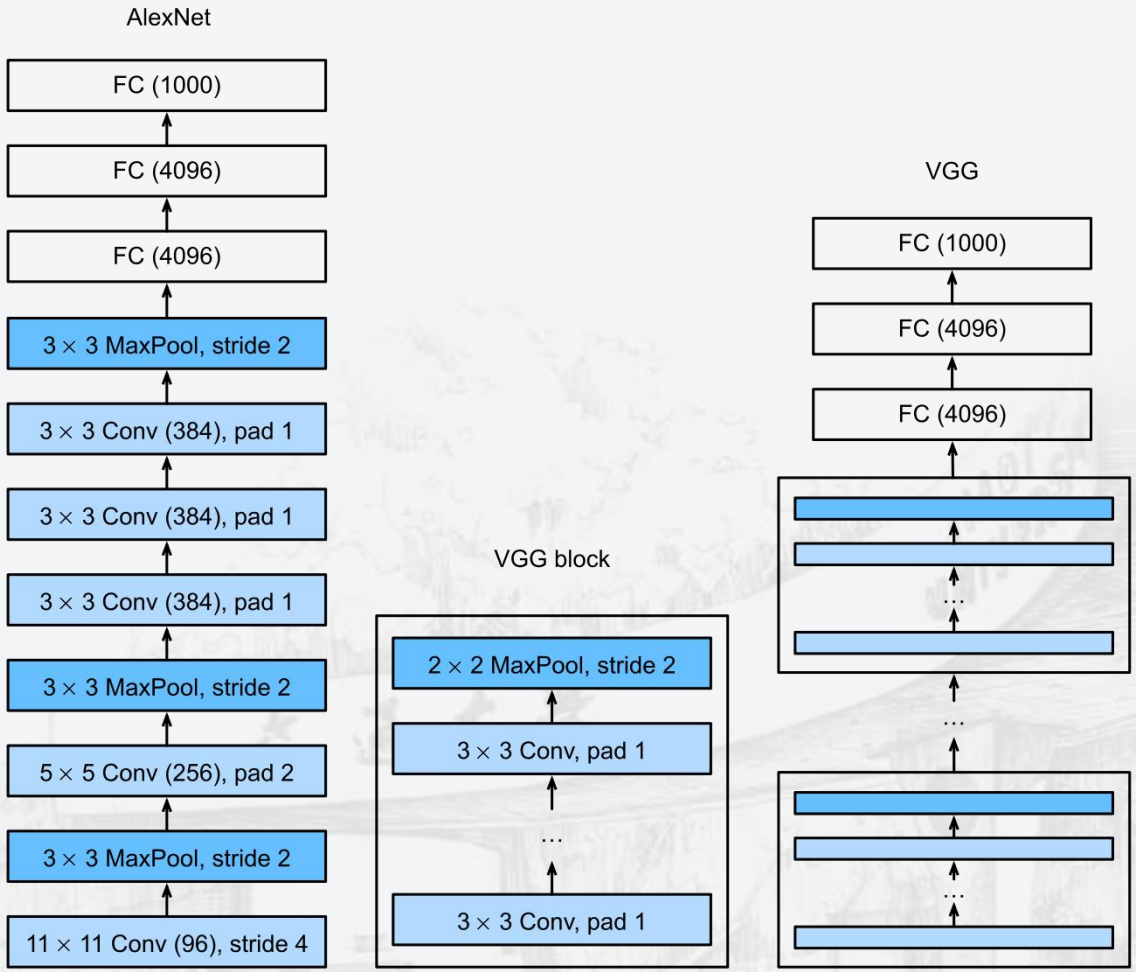


池化操作



VGG 设计思想

将卷积、激活、池化
“老三样”打包成块
最后附上全连接层



VGG 结构

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

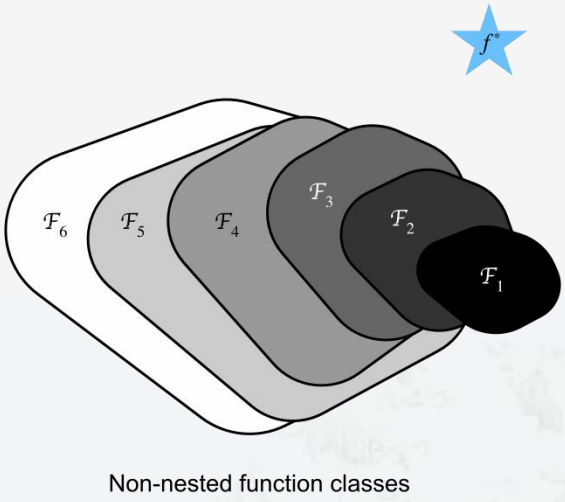
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

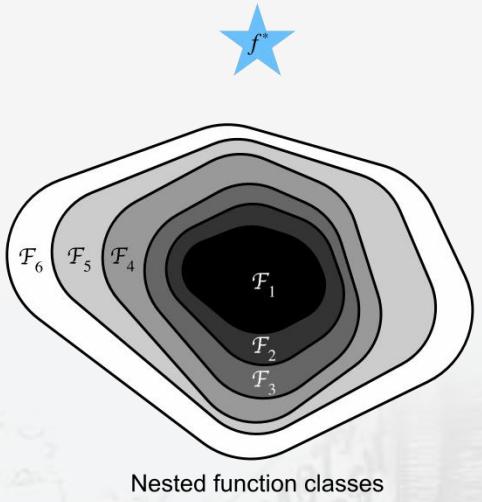
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

表达性 v.s. 复杂度?

网络层数加深时，其能表示的函数并不一定包含旧的；也就是说，复杂度增加并不一定会让我们离目标函数越来越近



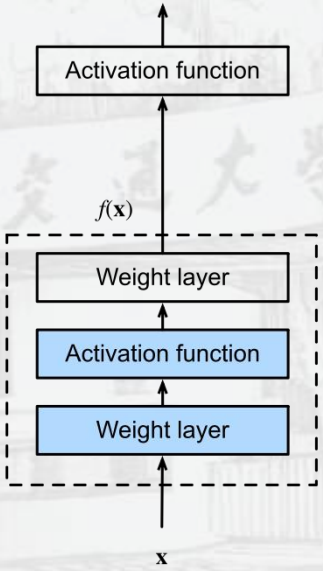
Non-nested function classes

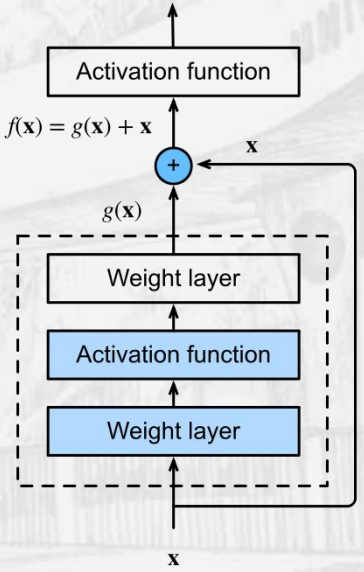


Nested function classes

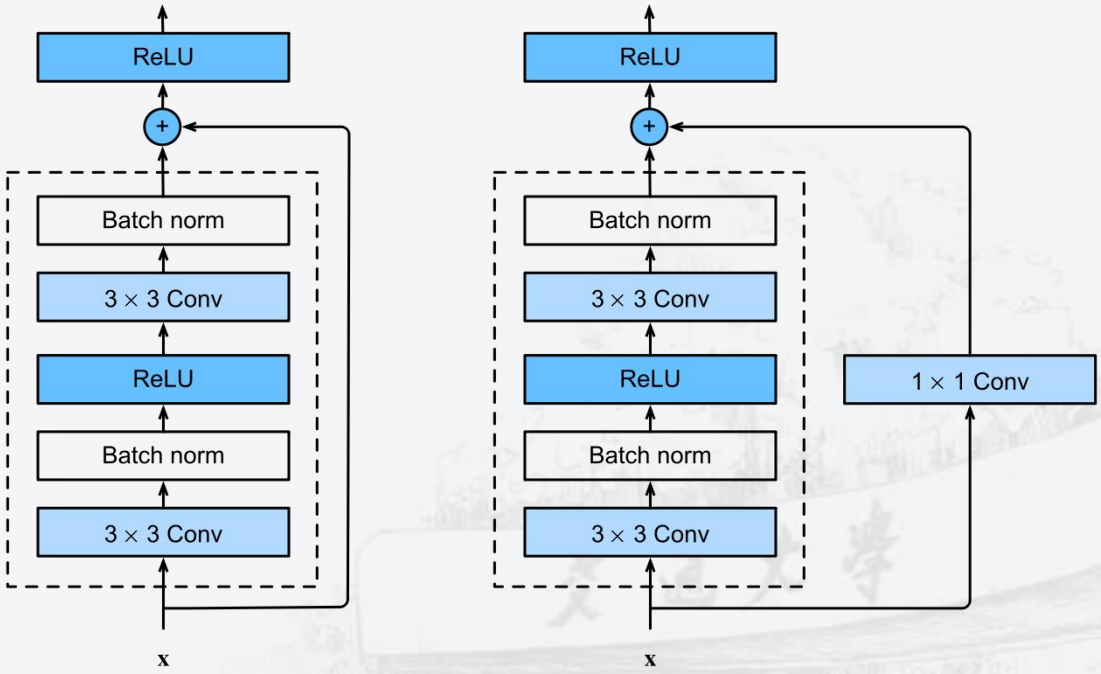
Residual Block

引用来自旧层的输出，让新层至少能产生旧层的函数，也就包含了之前的函数集合





ResBlock Design



如果第一个 3x3 conv 涉及 shape change (通常是：通道x2， 宽高/2)

那么对 residual 用 1x1 conv 做相同的 shape change

ResNet

开头:

7x7, stride=2 conv (c = 64, size halved)

3x3, stride=2 pool (size halved)

中间:

ResModule 2x64

ResBlock 64 (no shape change)

ResBlock 64

ResModule 2x128

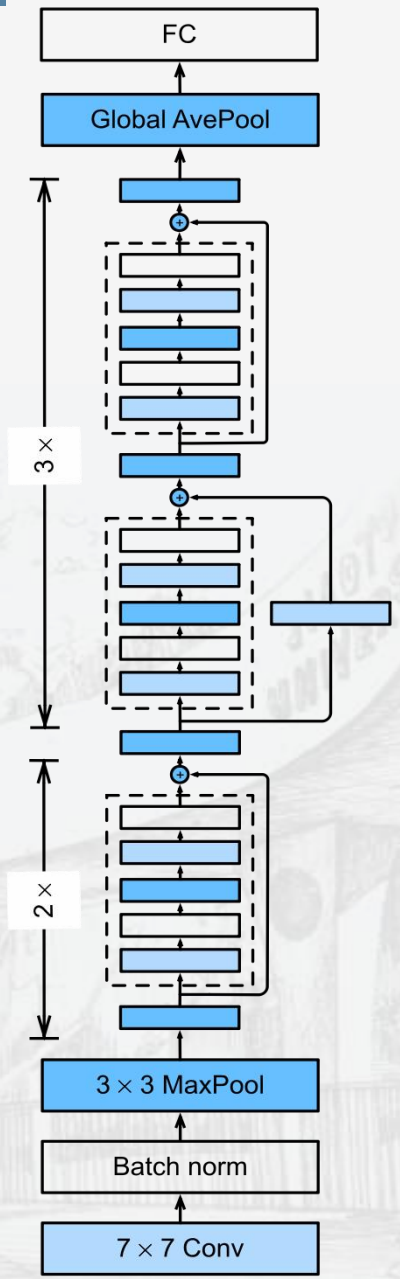
ResBlock 128 (shape changed!)

ResBlock 128

...

结尾:

AveragePool (size becomes 1x1)

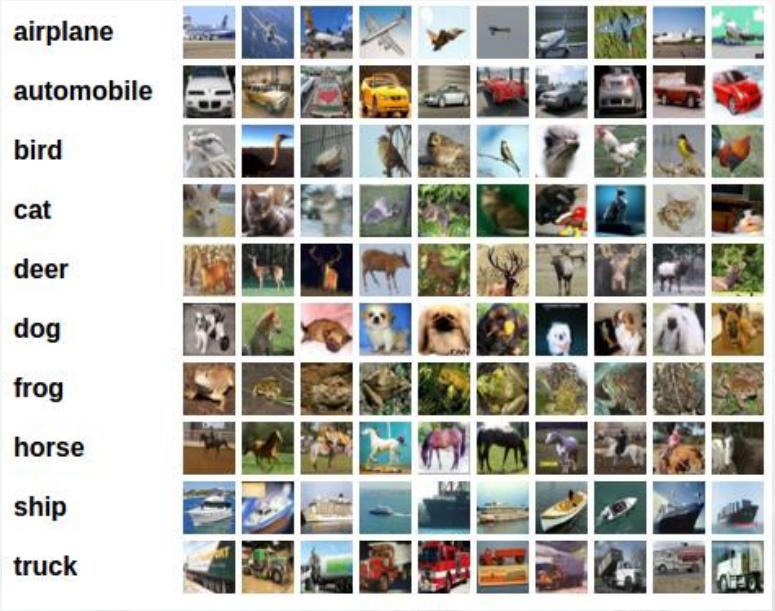


数据集介绍

CIFAR-10

60000 (50000+10000) 张 32x32 图片

分 10 类，每类 6000 (5000+1000) 张



CIFAR-100

60000 (50000+10000) 张 32x32 图片

分 100 类，每类 600 (500+100) 张；

这 100 类 (fine) 又被归做 20 个大类 (coarse)

数据集介绍

Imagenet

1400万+图片，20000+类别

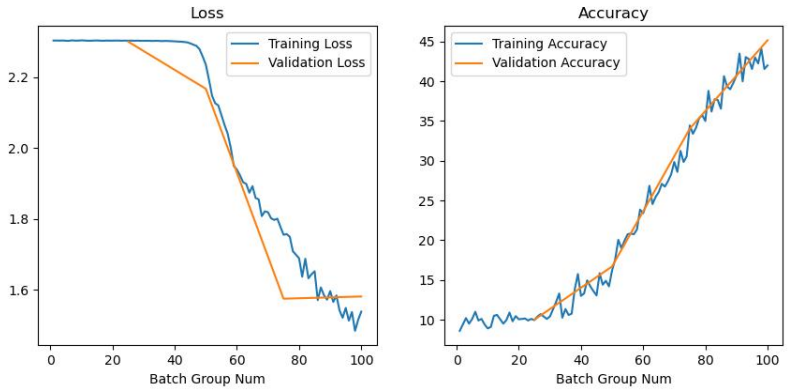
TinyImagenet

120000 (100000+10000+10000) 张 64x64 图片

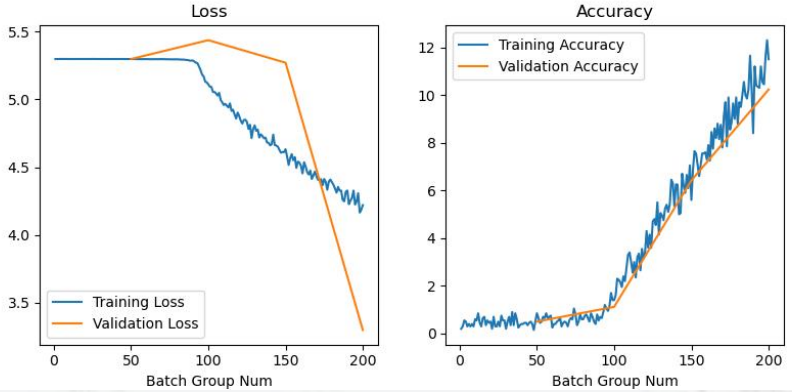
分 200 类，每类 600 (500+50+50) 张

模型训练结果

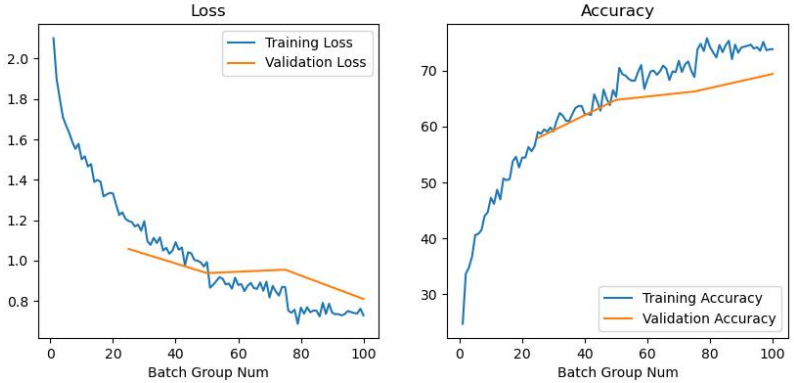
因为算力限制，VGG用了11层的，且都只训练了4 epochs



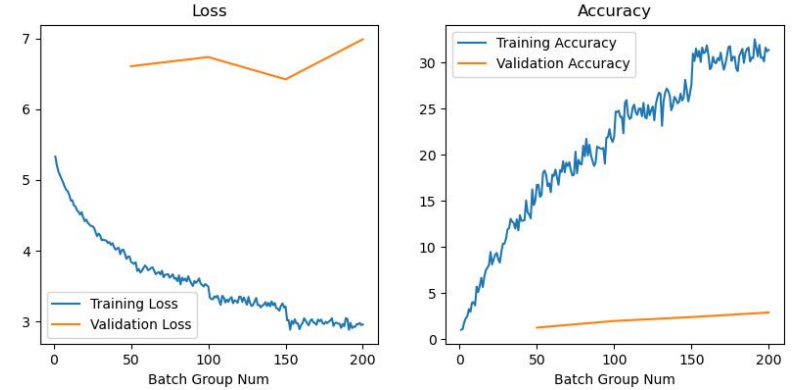
VGG-11 on CIFAR-10



VGG-11 on tiny Imagenet



Resnet-18 on CIFAR-10



Resnet-18 on tiny Imagenet

参考资料

《Dive into Deep Learning》- Zhang, Lipton, Li, Smola

《Pytorch 深度学习实践》- bilibili @刘二大人

Very deep convolutional networks for large-scale image recognition

Deep Residual Learning for Image Recognition

Accelerating Deep Network Training by Reducing Internal Covariate Shift