计算机视觉

第5章 图像分类

福州大学 陈飞
chenfei314@fzu.edu.cn

本章内容

- ❑ 图像分类任务
- ❑ 评价指标
- ❑ ILSVRC竞赛
- ❑ 神经网络
- ❑ 卷积神经网络
- ❑ 样本增强
- ❑ AlexNet网络
- ❑ GoogLeNet
- ❑ ResNet残差网络
- ❑ 案例

5.1图像分类任务

- 图像分类任务是计算机视觉中的核心任务，其目标是根据图像信息中所反映的不同特征，把不同类别的图像区分开来。

- 图像分类：从已知的类别标签集合中为给定的输入图片选定一个类别标签。

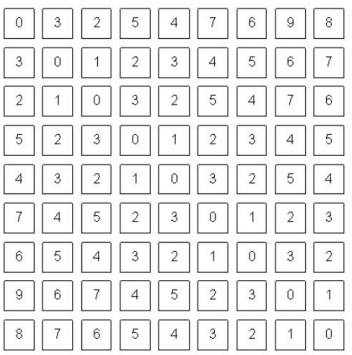
标签: {狗，猫，卡车，飞机，...}

狗

语义特征

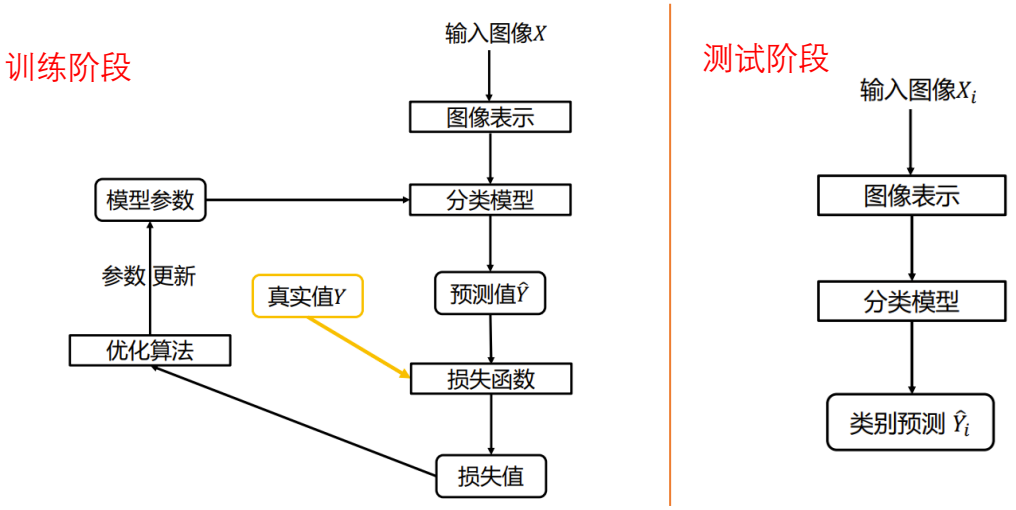
跨越“语义鸿沟”建立像素到语义的映射

我们看到的



机器看到的

分类器设计



5.2图像分类任务的评价指标

正确率（accuracy）= 分对的样本数/全部样本数
错误率（error rate）= 1 – 正确率
Top1指标与Top5指标

预测

TOP1 预测1: 猫 狗 车 树 椅子 √
预测2: 狗 猫 车 树 椅子 ×

TOP5 预测1: 猫 狗 车 树 椅子 √
预测2: 狗 猫 车 树 椅子 √

混淆矩阵

混淆矩阵		真实值	
		Positive	Negative
预测值	Positive	TP	FP
	Negative	FN	TN

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

准确率（Accuracy）

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

精确度（Precision）

$$\text{Precision} = \frac{TP}{TP + FP}$$

召回率（Recall）

$$\text{Recall} = \frac{TP}{TP + FN}$$

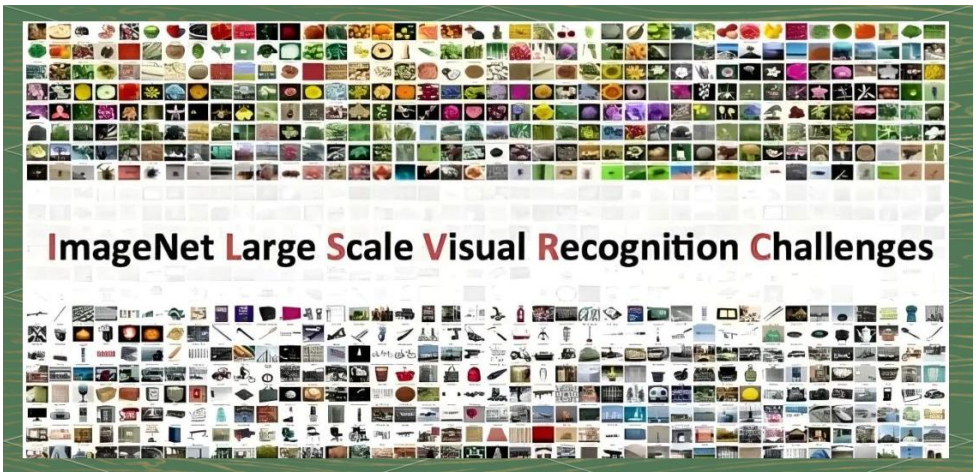
7

5.3 数据集

CIFAR-10 由多伦多大学计算机科学系的Alex Krizhevsky、Vinod Nair和Geoffrey Hinton于2009年创建。

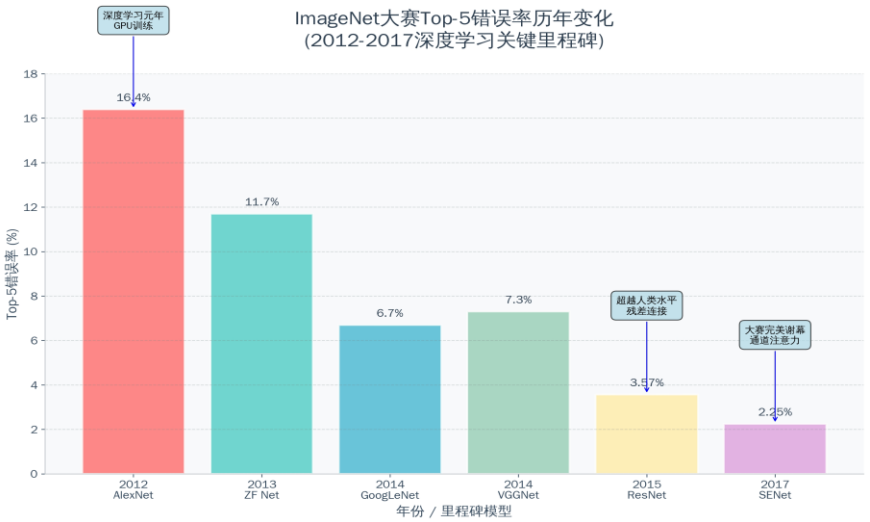
- **规模适中**：总共包含 **60,000** 张 **32×32** 像素的彩色RGB图像。
- **类别均衡**：共分为 **10** 个类别，包括：飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船、卡车。每个类别都有 **6,000** 张图像，数据分布非常均衡。
- **数据划分**：数据集被明确分为两个部分，**50,000** 张图像用于训练，**10,000** 张图像用于测试。

5.3 ILSVRC竞赛



ImageNet数据集——由斯坦福大学李飞飞教授主导制作，包含128万张训练图像、1000个物体类别的数据集

ILSVRC竞赛



从2010年到2017年，不仅是计算机视觉领域最重要的标杆，更直接推动了深度学习时代的到来。

WebVision

- **WebVision数据集是由瑞士苏黎世联邦理工学院的计算机视觉实验室创建的；**
- **WebVision 1.0**（2017年）包含1000个类别、240万张图像；
- **WebVision 2.0**（2018年），包含5000个类别，**超过1600万张；**
- **从互联网爬取，包含大量错误标注、“噪声”数据，且各类别图像数量极不均衡；**
- **“弱监督学习”和“噪声标签学习”领域的基准数据集。**

11

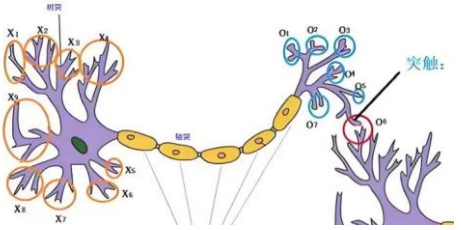
投票：AI 招聘系统中的“图像分类器”

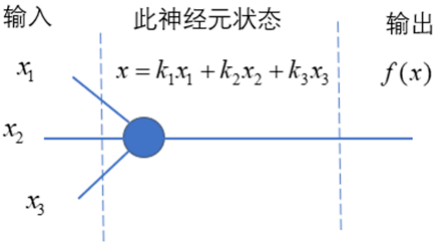
- 某科技公司开发了一款用于“初筛面试者”的图像分类系统。该系统仅通过分析面试者的**证件照**（面部特征、穿着、表情等），来预测该面试者是否“具备高职业素养”，并决定是否将其推送给HR进行下一轮面试。

12

5.4神经网络

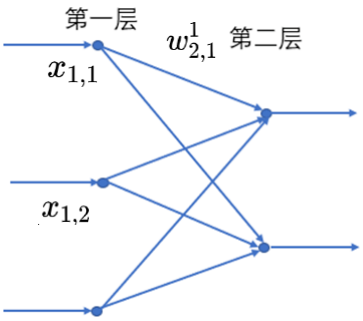
- **神经网络**是深度学习的核心算法架构，其设计灵感来源于人脑神经元的工作方式。它是一种通过多层非线性变换，从数据中自动学习层次化特征的数学模型。
- 神经元细胞——工作机理及其数学模型



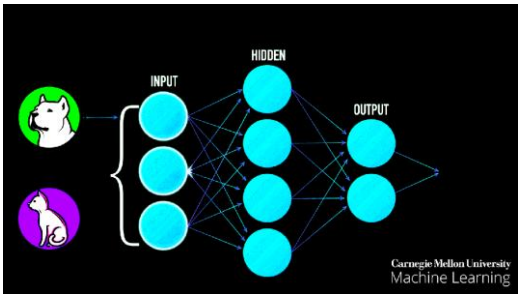


感受机制——神经网络

末端神经元受到的刺激要传到大脑里，不是一个神经元细胞就能完成的。通常需要很多个神经元首尾相接，层层传递。 由于末端需要很多神经元来收集信号，因此每一层需要若干个神经元细胞。



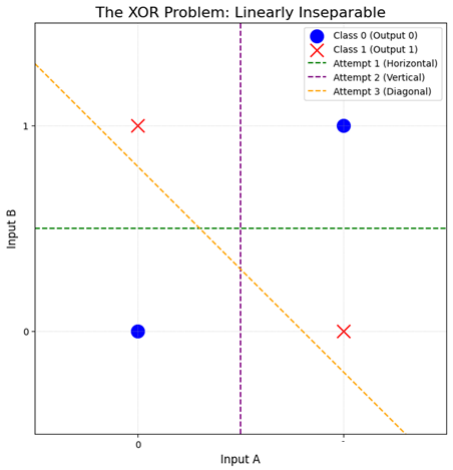
$$x_{2,1} = w^1_{2,1}f(x_{1,1}) + w^2_{2,1}f(x_{1,2}) + w^3_{2,1}f(x_{1,3})$$



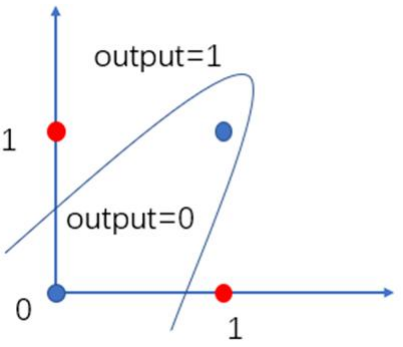
多层神经元细胞相互连接，构成一个网络，称为 **神经网络**。

线性不可分

- 异或（XOR）问题



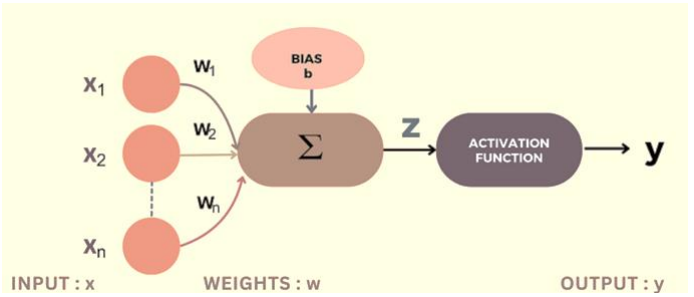
你无法画出任何一条直线，能将O和X完美地分在两边。



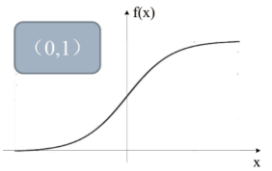
15

激活函数

激活函数通过在每层的输出上施加一个非线性变换，使得多层网络可以拟合复杂的、非线性的函数关系，从而具备更强大的学习和表达能力。

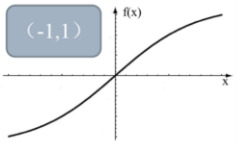


Sigmoid 函数



$$f(x)=\frac{1}{1+e^{-x}}$$

Tanh 函数



$$f(x)=\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

激活函数

引入非线性，使神经网络能够拟合任意复杂函数。

ReLU

$$f(z) = \max(0, z)$$

Leaky ReLU

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{if } z \leq 0 \end{cases}$$

α 是一个很小的正数，通常为 0.01。

ELU

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha(e^z - 1) & \text{if } z \leq 0 \end{cases}$$

α 是一个超参数，通常取 1。

SELU

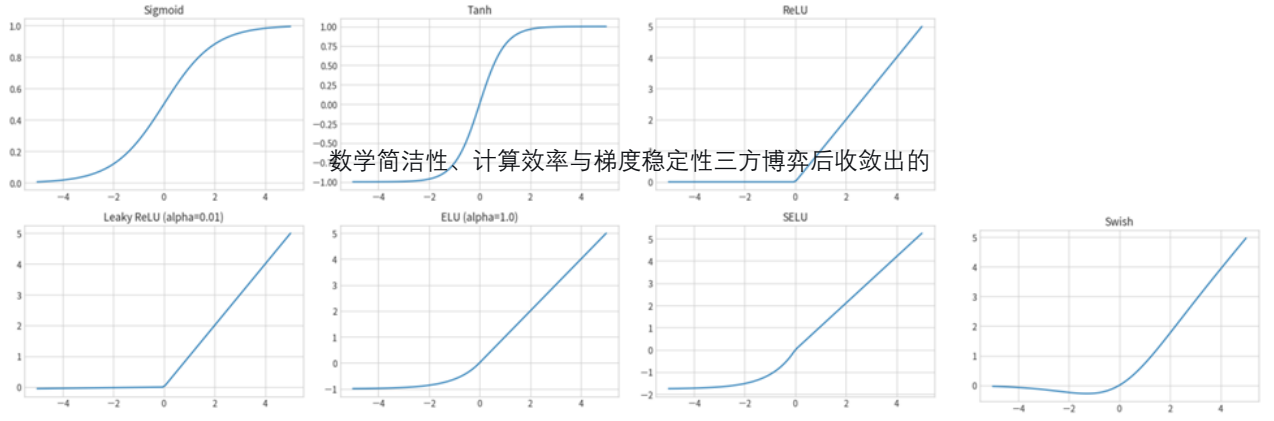
$$f(z) = \lambda \cdot \begin{cases} z & \text{if } z > 0 \\ \alpha(e^z - 1) & \text{if } z \leq 0 \end{cases}$$

Swish

$$f(z) = z \cdot \text{sigmoid}(z) = z \cdot \frac{1}{1 + e^{-z}}$$

激活函数

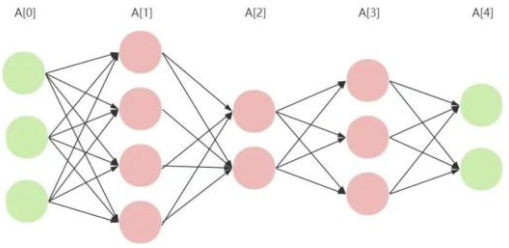
数学简洁性、计算效率与梯度稳定性三方博弈后收敛出的.



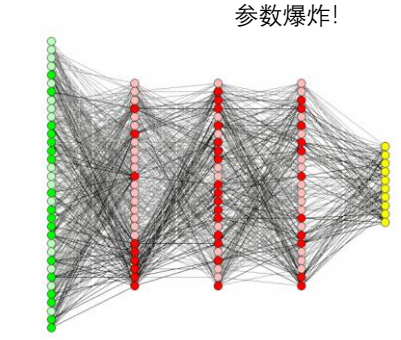
数学简洁性、计算效率与梯度稳定性三方博弈后收敛出的

全连接神经网络

- 神经网络是由大量神经元节点按一定体系架构连接成的网状结构，一般都有输入层，隐含层和输出层。
- 传统的浅层网络，一般有3~5层。



参数爆炸!



目标函数

目标函数（Objective Function），又称损失函数（Loss Function）或代价函数（Cost Function），是衡量模型预测值与真实值之间差距的函数。

1. 均方误差（MSE）

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2. 平均绝对误差（MAE）

$$\mathcal{L}_{\text{MAE}} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- y_i：真实值
- $\hat{y}_i$：预测值

3. Huber Loss

$$\mathcal{L}_{\text{Huber}} = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}\frac{1}{2}(y_i - \hat{y}_i)^2, & |y_i - \hat{y}_i| \le \delta \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2, & \text{otherwise}\end{cases}$$

- δ：阈值参数（通常取1）

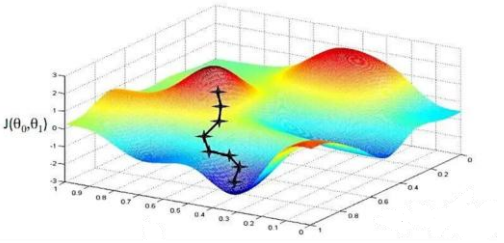
梯度下降

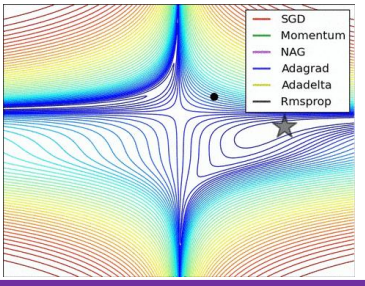
梯度下降法是神经网络训练的核心优化算法，通过迭代更新参数，使目标函数值不断减小。

参数更新公式

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

- θ：模型参数（权重和偏置）
- η：学习率（learning rate），控制步长大小
- $\nabla_\theta \mathcal{L}(\theta)$：目标函数关于参数$\theta$的梯度
- t：迭代次数





Softmax层

Softmax层的作用是突出“最大值”并转换成概率的形式。

$$a_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

C_1: w^1, b_1 $\quad z_1 = w^1 \cdot x + b_1$

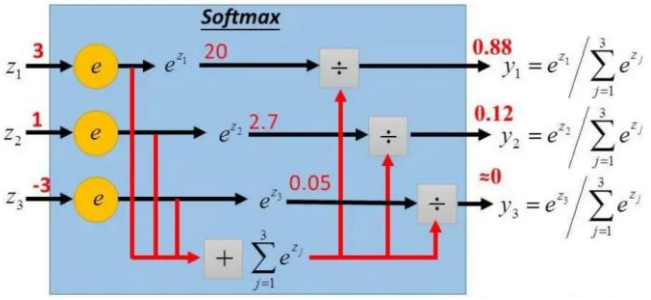
C_2: w^2, b_2 $\quad z_2 = w^2 \cdot x + b_2$

C_3: w^3, b_3 $\quad z_3 = w^3 \cdot x + b_3$

Probability:

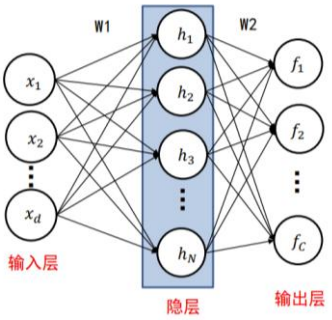
- $1 > y_i > 0$
- $\sum_i y_i = 1$

$$y_i = P(C_i \mid x)$$



全连接神经网络的瓶颈

两层全连接网络



全连接层的参数量为 $O(d_{\text{in}} \times d_{\text{out}})$，当处理高维输入时呈指数级增长。

具体示例：

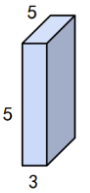
- 输入：ImageNet图像 $224 \times 224 \times 3 = 150,528$ 维
- 第一层隐藏层宽度：1024
- 仅第一层参数量：$150,528 \times 1024 \approx 1.54 \times 10^8$（1.54亿）
- 加上后续层，总参数量轻松突破10亿

5.5卷积神经网络

- 卷积核：

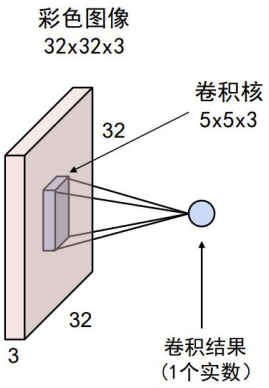
➢ 不仅具有宽和高，还具有深度，常写成：宽度 x 高度 x 深度

➢ 卷积核参数不仅包括核中存储的权值，还包括一个偏置值



5×5×3的卷积核

卷积网络中的卷积操作



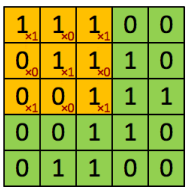
计算过程：

- 将卷积核展成一个5x5x3的向量，同时将其覆盖的图像区域按相同的展开方式展成5x5x3的向量
- 计算两者的点乘。
- 在点乘的结果上加上偏移量

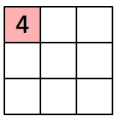
数学公式：

$$w^T x + b$$

w 为卷积核的权值， b 为卷积核的偏置

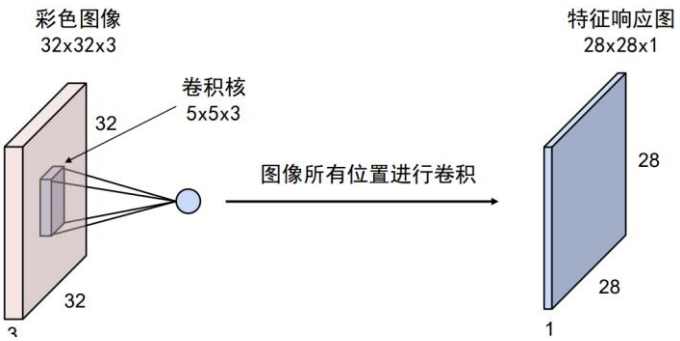


Image



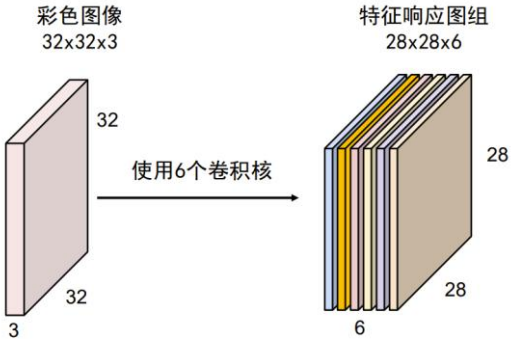
Convolved Feature

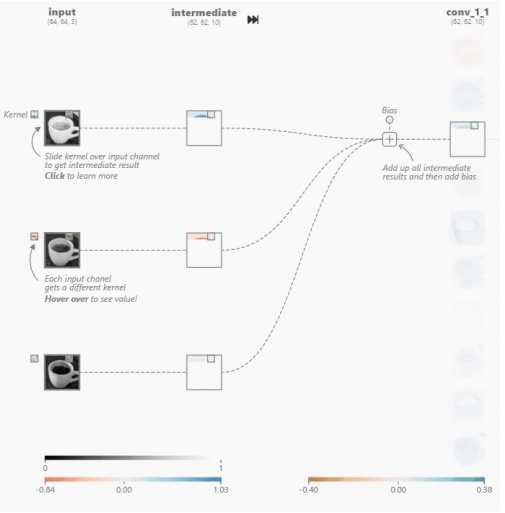
卷积网络中的卷积操作



特征响应图中每个位置上的值反 映了图像上对应位置是否存在卷 积核所记录的基元结构信息。

卷积层

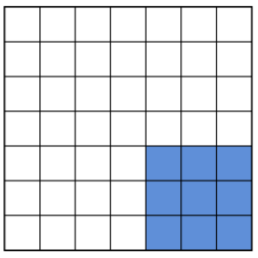




不同的特征响应图反映了输入图像对不同卷积核的响应结果

卷积步长（stride）

- 卷积神经网络中，卷积核可以按照指定的间隔进行卷积操作，这个间隔就是卷积步长。



卷积结果：3x3

图像尺寸： 7x7

卷积核尺寸：3x3

卷积步长： 2

输入数据矩阵尺寸： W1×H1

输出特征图组尺寸： W2×H2

W2与W1关系如下：

W2=(W1-F)/S+1

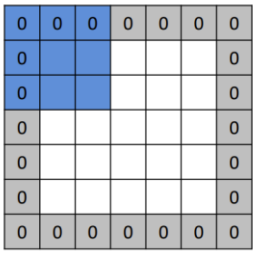
H2=(H2-F)/S+1

F——卷积核尺寸

S——卷积步长

边界填充

- 卷积神经网络中最常用的填充方式是零值填充。



图像尺寸： 5x5
卷积核尺寸：3x3
卷积步长： 1
零值填充： 1
卷积结果： 5x5

F——卷积核尺寸
S——卷积步长
P——零填充数量

输入数据矩阵尺寸： W1×H1
输出特征图组尺寸： W2×H2
W2与W1关系如下：

$W2=(W1-F+2P)/S+1$

$H2=(H2-F+2P)/S+1$

作用：保持输入、输出尺寸的一致！

池化操作

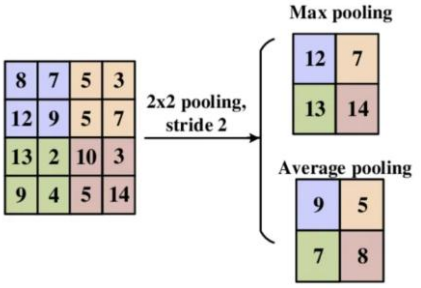
- 池化的作用：对每一个特征响应图独立进行，降低特征响应图组中每个特征响应图的宽度和高度，减少后续卷积层的参数的数量，降低计算资源耗费，进而控制过拟合。

➢ 池化操作:对特征响应图某个区域进行池化就是在该区域上指定一个值来代表整个区域。

➢ 常见的池化操作：

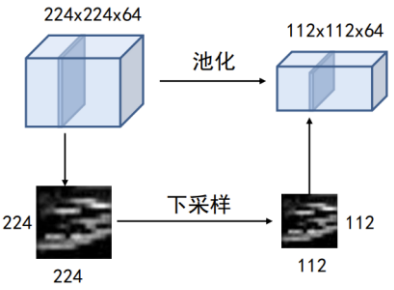
- 最大池化:使用区域内的最大值来代表这个区域；
- 平均池化:采用区域内所有值的均值作为代表。

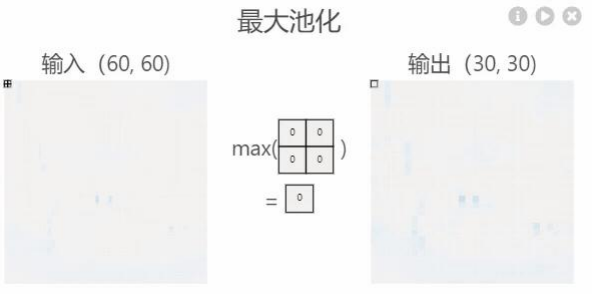
➢ 池化层的超参数：池化窗口和池化步长



池化操作示例

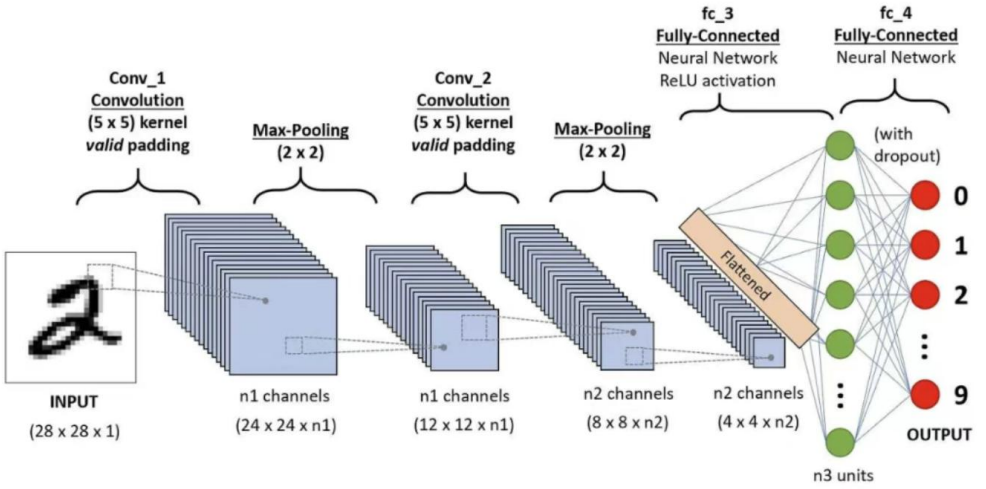
- 池化操作对每一个特征响应图独立进行;
- 对特征响应图某个区域进行池化就是在该区域上指定一个值来代表整个区域。





31

卷积神经网络



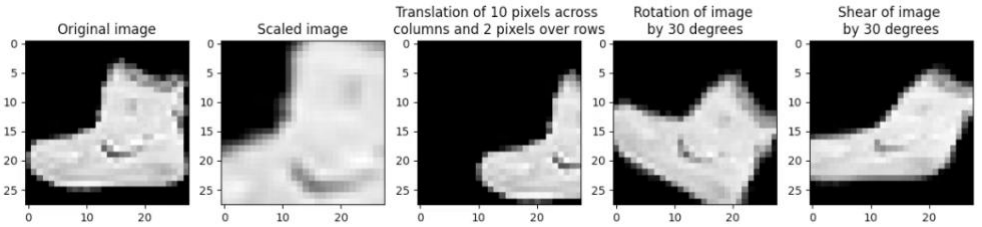
32

5.6样本增强

➢ 存在的问题：过拟合的原因是学习样本太少，导致无法训练出能够泛化到新数 据的模型。

➢ 数据增强：是从现有的训练样本中生成更多的训练数据，其方法是利用多种能 够生成可信图像的随机变换来增加样本。

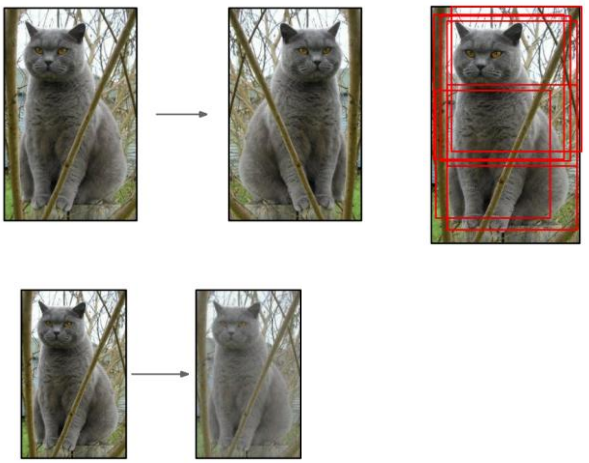
➢ 数据增强的目标：模型在训练时不会两次查看完全相同的图像。这让模型能够 观察到数据的更多内容，从而具有更好的泛化能力



33

样本增强

- 翻转
- 随机缩放&抠图
- 色彩抖动
- 平移
- 旋转
- 拉伸
- 径向畸变
- 裁剪



34

卷积神经网络进化

网络进化

- 网络：AlexNet → VGG → GoogLeNet → ResNet
- 深度：8 → 19 → 22 → 152
- VGG结构简洁有效
  - 容易修改，迁移到其他任务中去
  - 高层任务的基础网络
- 性能竞争网络
  - GoogLeNet：Inception v1 → v4
    - Split-transform-merge
  - ResNet：ResNet1024 → ResNeXt
    - 深度、宽度、基数(cardinality)

5.7 AlexNet网络

AlexNet网络

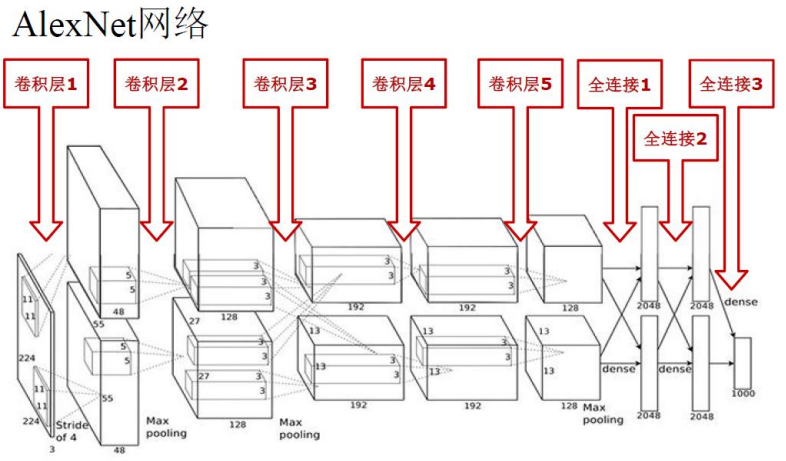
- ImageNet-2012竞赛第一
- 标志着DNN深度学习革命的开始
  - 5个卷积层 + 3个全连接层
  - 60M个参数 + 650K个神经元
  - 2个分组 → 2个GPU（3GB）
    - 使用两块GTX 580 GPU训练了5~6天
  - 新技术
    - ReLU非线性激活
    - Max pooling池化
    - Dropout regularization

精度提升超过10个百分点!

AlexNet网络

结构:

- CONV1
- MAX POOL1
- NORM1
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8



AlexNet网络

结构:

- CONV1
- MAX POOL1
- NORM1
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8

- 第一层 (CONV1): 96 个11x11 卷积核，步长为4，没有零填充

问题：输入:227x227x3 大小的图像，输出特征图个数及尺寸为多少?

尺寸：(227-11)/4+1 = 55

个数：96

问题：这层有多少个参数?

参数: (11*11*3+1)*96 = 35K

AlexNet网络

结构:
- CONV1
- MAX POOL1
- NORM1
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8

• Max POOL1: 窗口大小3x3，步长为 2（重叠有助于 对抗过拟合）

作用：降低特征图尺寸，对抗轻微的目标偏移带来的影响

输出尺寸： (55-3)/2+1 = 27

特征图个数：96

参数个数：0

39

AlexNet网络

结构:
- CONV1
- MAX POOL1
- NORM1
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8

• 局部相应归一化层（NORM1）作用:

➢ 对局部神经元的活动创建竞争机制；

➢ 响应比较大的值变得相对更大；

➢ 抑制其他反馈较小的神经元；

➢ 增强模型的泛化能力

后来的研究表明： 更深的网络中该层 对分类性能的提升 效果并不明显，且 会增加计算量与存储空间。

40

AlexNet网络

结构:

- CONV1
- MAX POOL1
- NORM1
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8

- 第二层 (CONV2): 256 个5x5 卷积核，步长为 1，使用零填充p=2

问题：输入: 27x27x256 大小的特征图组，输出特征图个数及尺寸为多少?

尺寸：(27 - 5 + 2*2)/1+1 = 27

个数：256

41

AlexNet网络

结构:

- CONV1
- MAX POOL1
- NORM1
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8

- 第三、四层 (CONV3、CONV4): 384 个3x3 卷积核，步长为 1，使用零填充 p=1

问题：CONV3输入: 13x13x256 大小的特征图组，输出特征图个数及尺寸为 多少?

尺寸：(13 - 3 + 2*1)/1+1 = 13

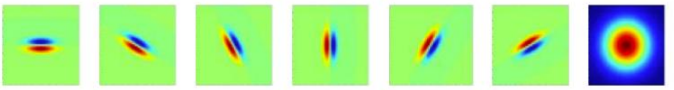
个数：384

42

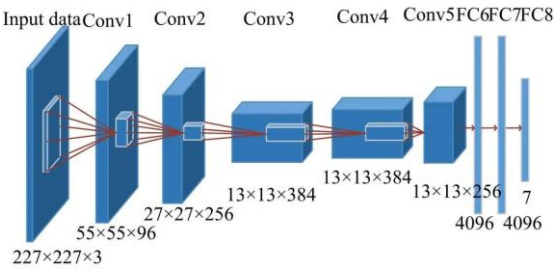
AlexNet网络

结构:

- CONV1
- MAX POOL1
- NORM1
- CONV2
- MAX POOL2
- NORM2
- CONV3
- CONV4
- CONV5
- Max POOL3
- FC6
- FC7
- FC8



AlexNet网络



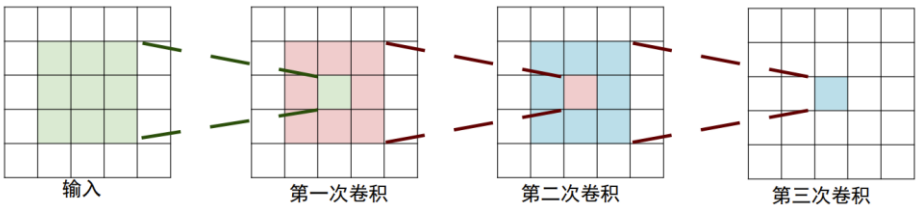
超级卷积核组

特征响应图

```
       Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 96, 54, 54]          34,944
              ReLU-2           [-1, 96, 54, 54]               0
         MaxPool2d-3           [-1, 96, 26, 26]               0
            Conv2d-4          [-1, 256, 26, 26]         614,656
              ReLU-5          [-1, 256, 26, 26]               0
         MaxPool2d-6          [-1, 256, 12, 12]               0
            Conv2d-7          [-1, 384, 12, 12]         885,120
              ReLU-8          [-1, 384, 12, 12]               0
            Conv2d-9          [-1, 384, 12, 12]       1,327,488
             ReLU-10          [-1, 384, 12, 12]               0
           Conv2d-11          [-1, 256, 12, 12]         884,992
             ReLU-12          [-1, 256, 12, 12]               0
        MaxPool2d-13            [-1, 256, 5, 5]               0
          Flatten-14                 [-1, 6400]               0
           Linear-15                 [-1, 4096]      26,218,496
             ReLU-16                 [-1, 4096]               0
          Dropout-17                 [-1, 4096]               0
           Linear-18                 [-1, 4096]      16,781,312
             ReLU-19                 [-1, 4096]               0
          Dropout-20                 [-1, 4096]               0
           Linear-21                   [-1, 10]          40,970
================================================================
Total params: 46,787,978
Trainable params: 46,787,978
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 10.22
Params size (MB): 178.48
Estimated Total Size (MB): 189.28
----------------------------------------------------------------
```

思考

- 问题1：小卷积核有哪些优势？
- 回答：多个小尺寸卷积核串联可以得到与大尺寸卷积核相同的感受野； 使用小卷积核串联构建的网络深度更深、非线性更强、参数也更少。



3个3x3卷积核串联，感受野为7x7

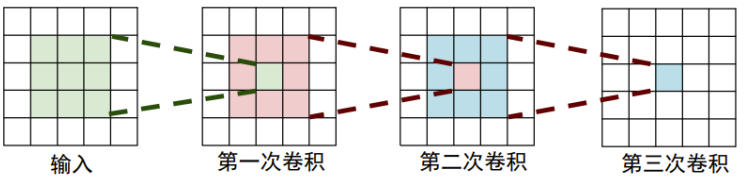
小卷积核优势

- 假设卷积层输入和输出的特征图个数均为C：
- 三个3 × 3的卷积串联参数个数

$(3 \times 3 \times C) \times C \times 3 = 27C^2$

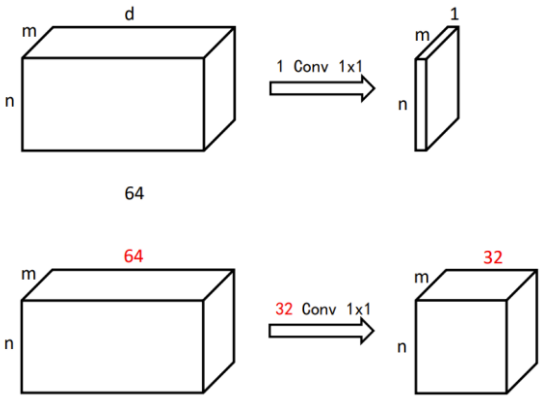
一个7 x 7的卷积层卷积参数个数

$(7 \times 7 \times C) \times C = 49C^2$

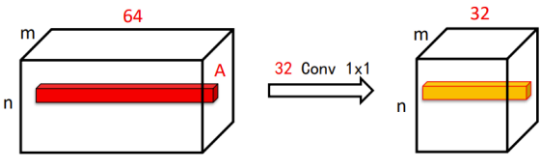


1x1卷积

- 利用1x1卷积进行压缩会损失信息吗？



1x1卷积



这种压缩是否会损失信息呢？
位置A的这个64维向量是一个非常稀疏向量
利用1x1卷积进行非线性压缩通常不会损失信息。

- 1×1的卷积有两个方面的作用
  - 实现跨通道的交互和信息整合
  - 进行卷积核通道数的降维和升维

AlexNet的贡献

- AlexNet——验证了深度卷积神经网络的高效性
- 主体贡献

1. 提出了一种卷积层加全连接层的卷积神经网络结构
2. 首次使用ReLU函数做为神经网络的激活函数
3. 首次提出Dropout正则化来控制过拟合
4. 使用加入动量的小批量梯度下降算法加速了训练过程的收敛；
5. 使用数据增强策略极大地抑制了训练过程的过拟合；
6. 利用了GPU的并行计算能力，加速了网络的训练与推断。

5.8 GoogLeNet

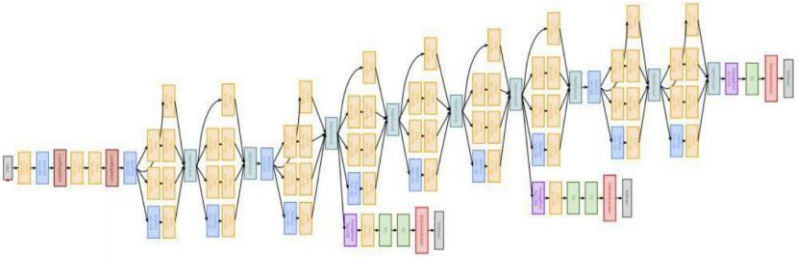
- GoogleNet的创新点：

➢ 提出了一种Inception结构，它能保留输入信号中的更多特征信息；

➢ 去掉了AlexNet的前两个全连接层，并采用了平均池化，这一设计使得GoogLeNet只有500万参数，比AlexNet少了12倍；

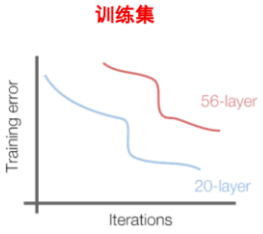
➢ 在网络的中部引入了辅助分类器，克服了训练过程中的梯度消失问题。

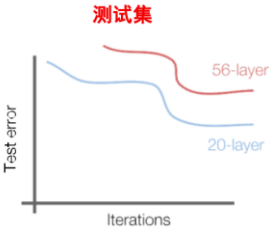
GoogLeNet是2014年ImageNet的冠军,亚军是著名的VGG。两类模型共同的特点是更深了。VGG继承了AlexNet的很多思想,而GoogLeNet在结构上则有了大胆的尝试。



5.9 ResNet残差网络

- 实验：持续向一个“基础”的卷积神经网络上面叠加更深的层数会发生什么？





梯度消失

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_L} \cdot \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_{L-1}} \cdots \frac{\partial \mathbf{x}_{l+2}}{\partial \mathbf{x}_{l+1}} \cdot \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{w}_l}$$

退化问题

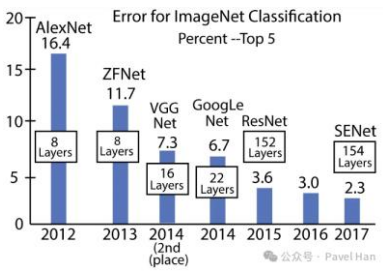
原因：训练过程中网络的正、反向信息流动不顺畅， 网络没有被充分训练。

猜测：加深网络层数引起过拟合，导致错误率上升

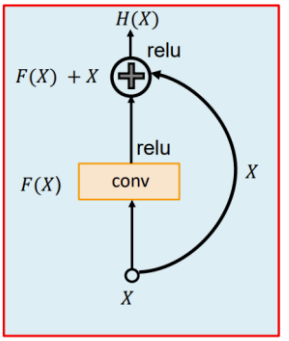
ResNet残差网络

- **ResNet**（Residual Neural Network，残差网络）是由何恺明等人在2015年提出的深度卷积神经网络架构。它通过引入**残差连接**（跳跃连接），解决了极深网络中的**退化问题**（梯度消失/爆炸、网络过深反而性能下降），使得网络深度可以突破数百甚至上千层。

2023年引用量超过18万次。



ImageNet-2015竞赛第一



残差模块

典型CNN的参数与效果对比

模型名	AlexNet	VGG	GoogLeNet v1	ResNet
时间	2012	2014	2014	2015
层数	8	19	22	152
Top-5错误	16.4%	7.3%	6.7%	3.57%
Data Augmentation	+	+	+	+
Inception(NIN)	–	–	+	–
卷积层数	5	16	21	151
卷积核大小	11,5,3	3	7,1,3,5	7,1,3,5
全连接层数	3	3	1	1
全连接层大小	4096,4096,1000	4096,4096,1000	1000	1000
Dropout	+	+	+	+
Local Response Normalization	+	–	+	–
Batch Normalization	–	–	–	+

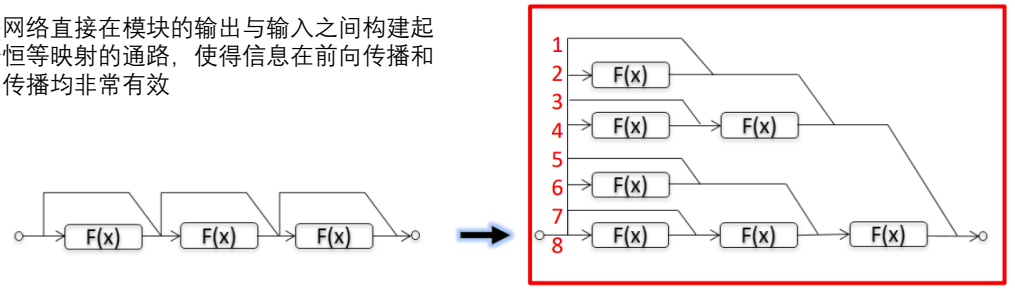
为什么残差网络性能这么好?

- 一种典型的解释：残差网络可以看作是一种集成模型！

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)$$

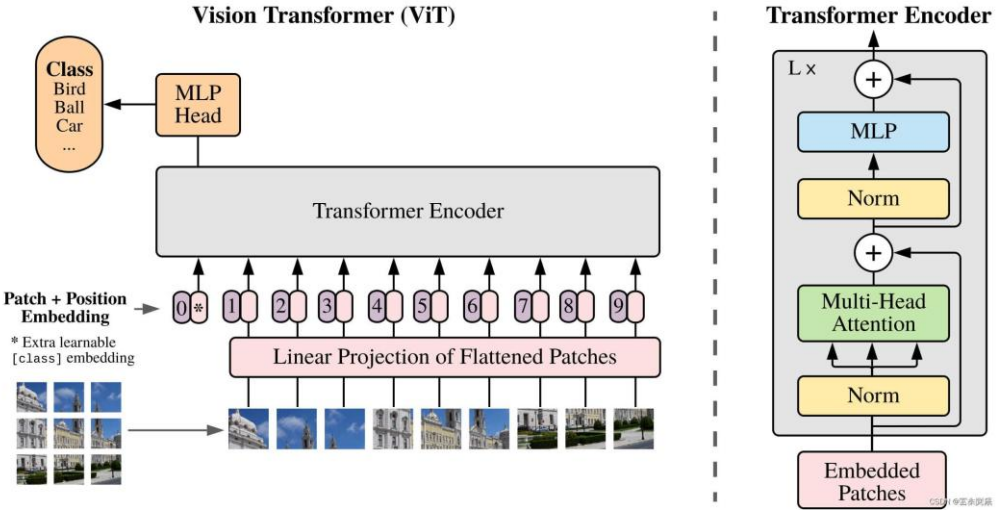
$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_L} \cdot \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_L} \cdot \left(1 + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)\right)$$

残差网络直接在模块的输出与输入之间构建起一个恒等映射的通路，使得信息在前向传播和后向传播均非常有效

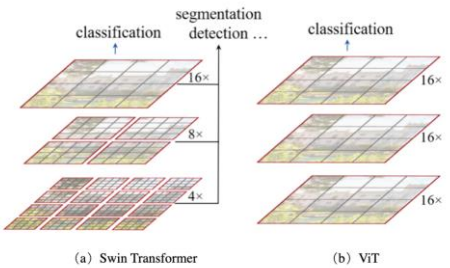


残差结构　　展开后的残差结构

5.10 Vision Transformer （ViT）

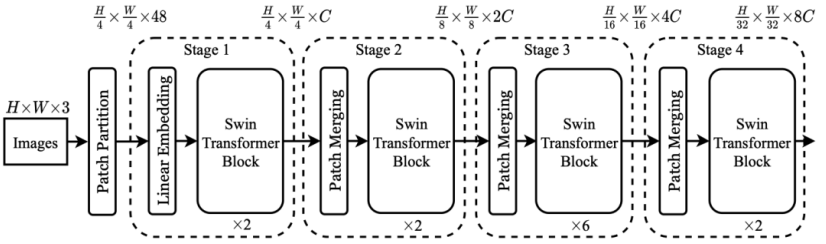


Swin Transformer



步骤	ViT	Swin Transformer
分块方式	使用较大 Patch (16x16)，数量少	使用较小 Patch (4x4)，数量多
位置编码	需要添加可学习/正弦位置编码	**不需要**位置编码（通过层级结构隐含位置信息）
特征聚合	使用额外的"类别令牌"	使用**全局平均池化**
注意力范围	全局自注意力（计算量大）	窗口 + 滑动窗口（计算量小）
输出特征	单一尺度	多尺度（4 种分辨率）

1.图像分块
2. **线性嵌入**
3. **Swin Transformer 阶段（核心）**
4. **层归一化**
5. **全局池化**
6. **分类输出**



讨论：模型究竟在看什么？

- 在训练一个区分“鸡”和“公鸡”的二分类图像分类器时，我们发现了一个奇怪的现象：训练出来的模型在测试集上准确率很高，但在实际应用（如农场监控）中，它把很多背景里有“太阳”的“鸡”都错误地识别成了“公鸡”。
- **讨论任务：**
- **诊断问题：** 你认为导致这个模型“失效”的根本原因是什么？
- **解决方案：** 如果你是这个项目的工程师，你会如何修正这个模型？

案例1：鲜花分类

- 使用数据增强提升鲜花分类模型性能
- 在鲜花分类模型中加入数据增强机制，提高模型泛化能力。
- 有17种鲜花数据，每种鲜花有数十张图像。

案例：鲜花分类

案例步骤概述

不使用数据增强

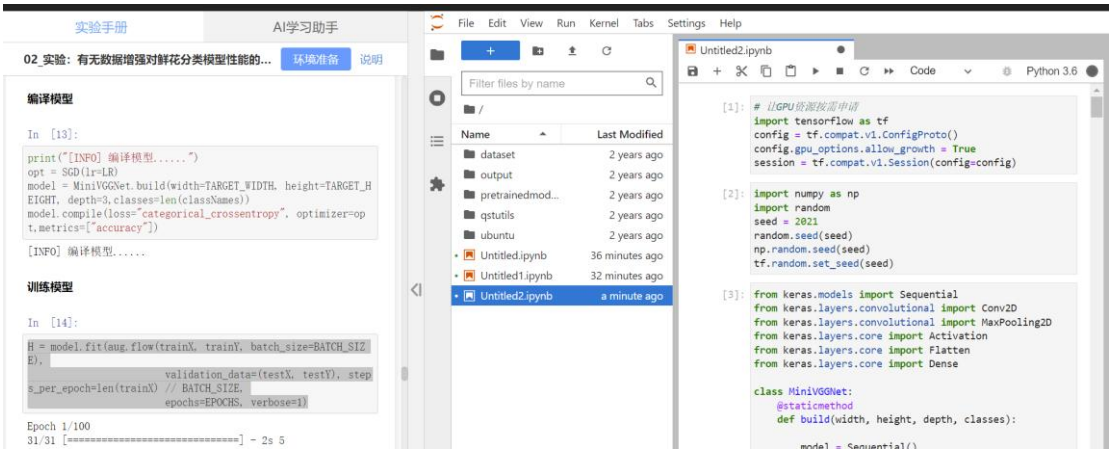
① 导入库；
② 设置模型超参数；
③ 设置全局变量；
④ 处理数据；
⑤ 构建并编译模型；
⑥ 训练模型；
⑦ 评估模型分类性能；
⑧ 画出loss曲线；
⑨ 画出accuracy曲线。

案例步骤概述

使用数据增强

① 导入库；
② 设置模型超参数；
③ 设置全局变量；
④ 处理数据；
⑤ 数据增强；
⑥ 构建并编译模型；
⑦ 训练模型；
⑧ 评估模型分类性能；
⑨ 画出loss曲线；
⑩ 画出accuracy曲线。

案例：鲜花分类



案例2：毒蘑菇识别

实验背景

全世界已知有2000多种野生食用菌的种类，云南占全国的80%、全世界的40%以上，全省境内有126个县城出产野生菌，每年吃菌的时间长达半年。 据说，每个云南人都有一个因吃菌中过毒的朋友。了解蘑菇种类，能帮助食客们分别毒蘑菇与可食用蘑菇。 本实验将使用蘑菇数据集，训练一个能分辨蘑菇种类的模型。

实验内容

本实验将使用kaggle的蘑菇数据集，采用MindSpore框架训练一个蘑菇分类器。

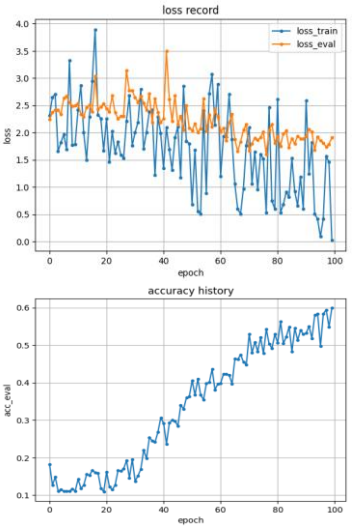
案例2：毒蘑菇识别

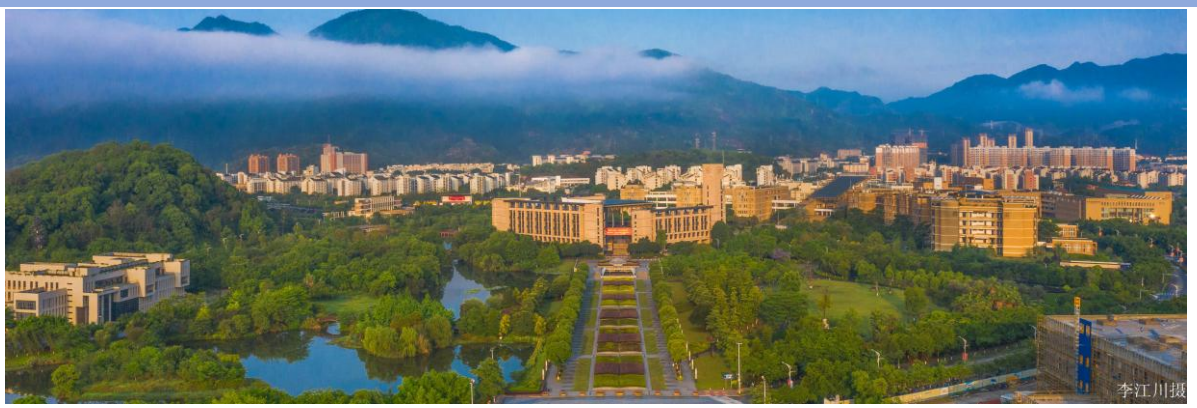
- 深度学习框架：MindSpore
- 训练硬件：Ascend 910
- 使用MindSpore构建ResNet-50网络

```
import mindspore.dataset as ds # 数据集载入
import mindspore.nn as nn # 各类网络层都在nn里面
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor, TimeMonitor # 回调函数
from mindspore.train import Model # 承载网络结构
from mindspore import load_checkpoint # 读取最佳参数
from mindspore import context # 设置mindspore运行的环境

from easydict import EasyDict as ed # 超参数保存
import numpy as np # numpy
import matplotlib.pyplot as plt # 可视化

# 文件处理相关
import os
```





Thank you!