Alexnet

多个CNN层堆叠而成的经典卷积神经网络，用于图像分类的任务

- 使用了ReLU激活函数，防止深层次的网络导致**梯度消失**
- 使用Dropout正则化，解决**过拟合问题**
- 使用池化层进行进行**降维**

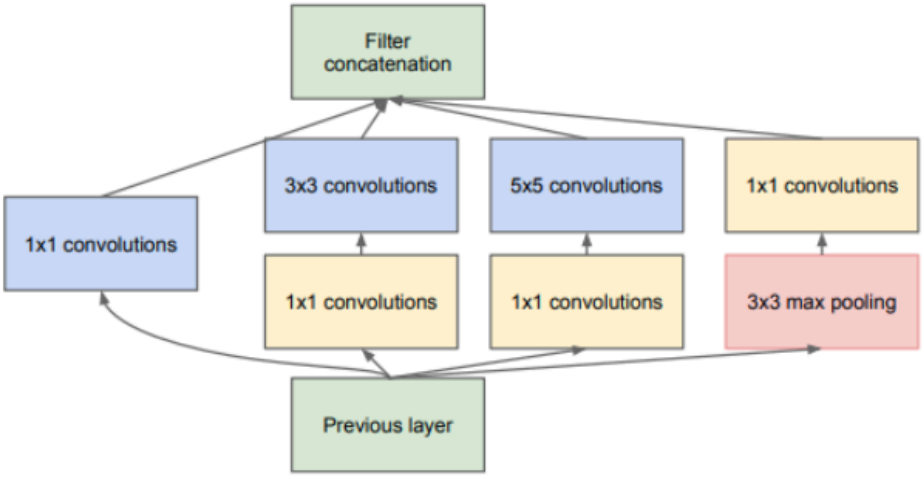
GoogleNet

传统CNN通多堆叠卷积层可以学习到图像更多特征，但会导致

- 计算量爆炸
- 梯度消失
- 过拟合

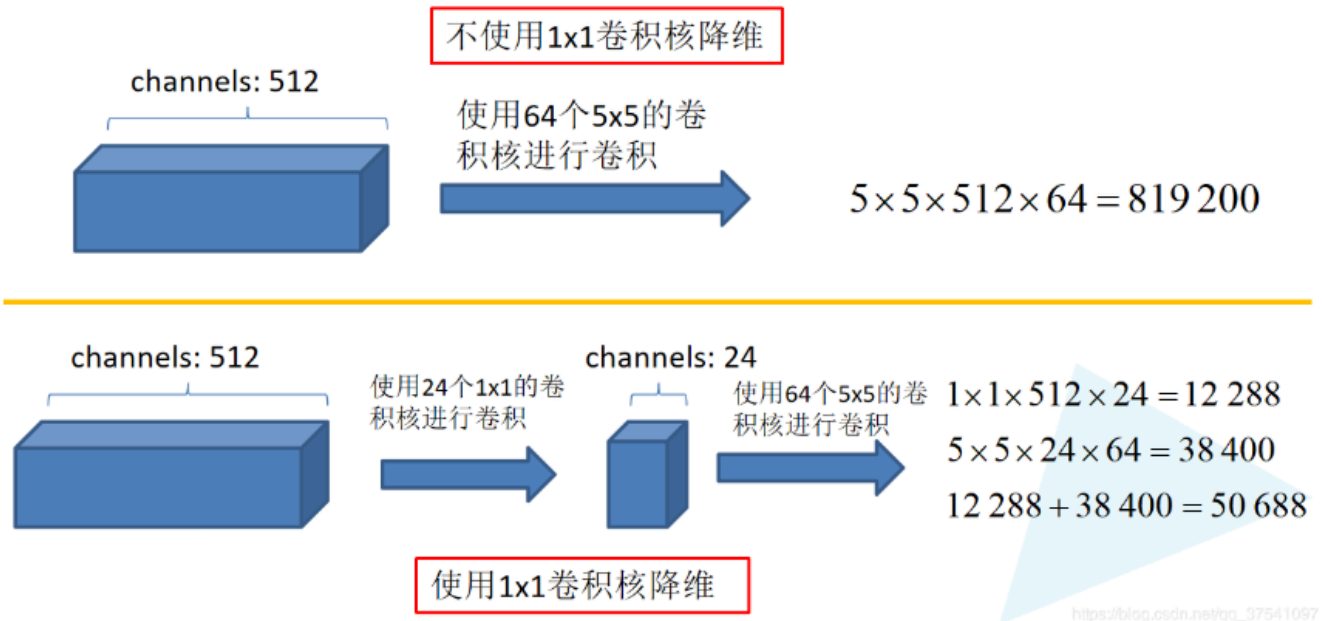
Googlenet设计了宽而深的结构，**创造性**引入了**Inception模块**和**辅助分类器模块**

Inception模块

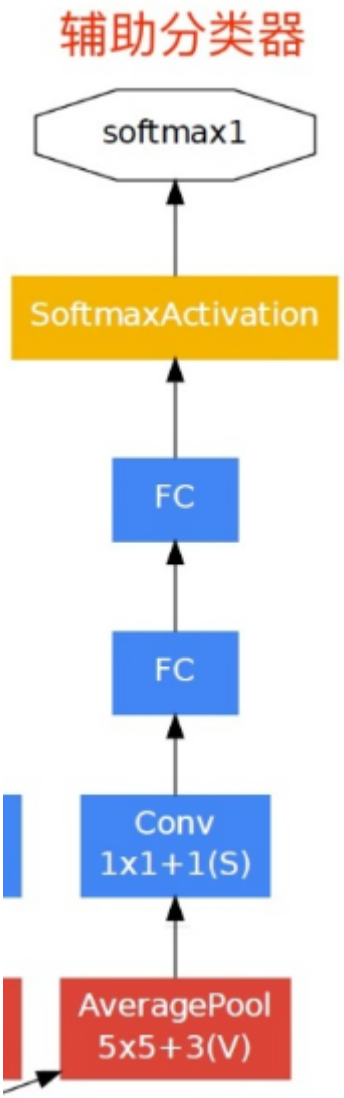


(b) Inception module with dimensionality reduction

- 并行多尺度卷积：在同一层并行使用1×1、3×3、5×5和3×3最大池化，融合不同感受野的特征
- 先使用1×1卷积进行降维，压缩通道数，同时加上ReLU，可以**显著降低参数量**



辅助分类器模块



- GoogleNet有**3个输出层**，其中**2个是辅助分类层**
- 训练模型时，将2个辅助分类器的损失乘以权重加到总损失上

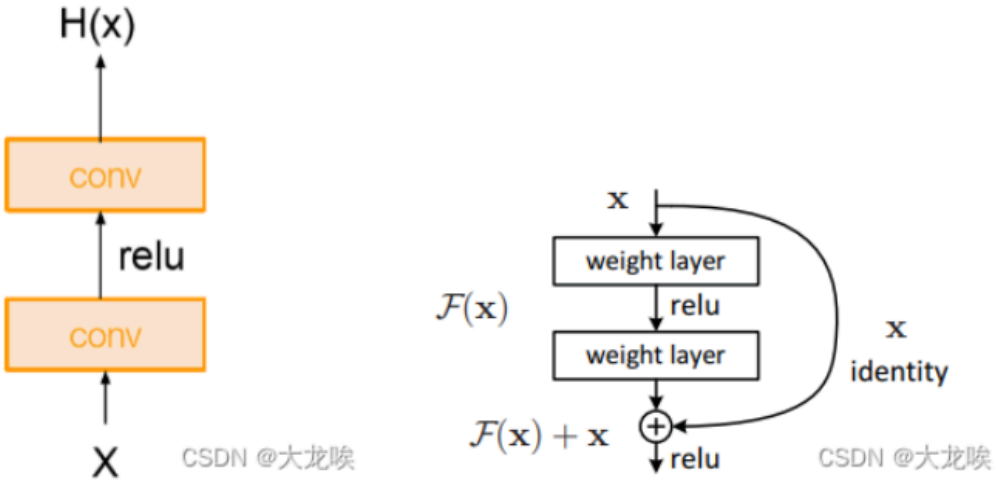
- 辅助分类器也能预测类别，在整个网络中起到一个调整的作用，可以防止网络发生**过拟合**
- 也可以加速梯度传递，防止梯度消失

Resnet

2015年由何凯明团队提出，利用**短路连接**解决深度卷积神经网络中**梯度消失**的问题

$$H(x) = x + F(x)$$

这样使得网络在**最差情况下也能获得和输入一样的输出**，不会出现**网络退化**的问题



ResNet Block

`BasicBlock`不会对每一个block的输出进行升维

`BottleNeck`会对每个layer的第一个block的输出进行升维，其输出通道数是输入中间通道数的4倍

二者结构如下所示

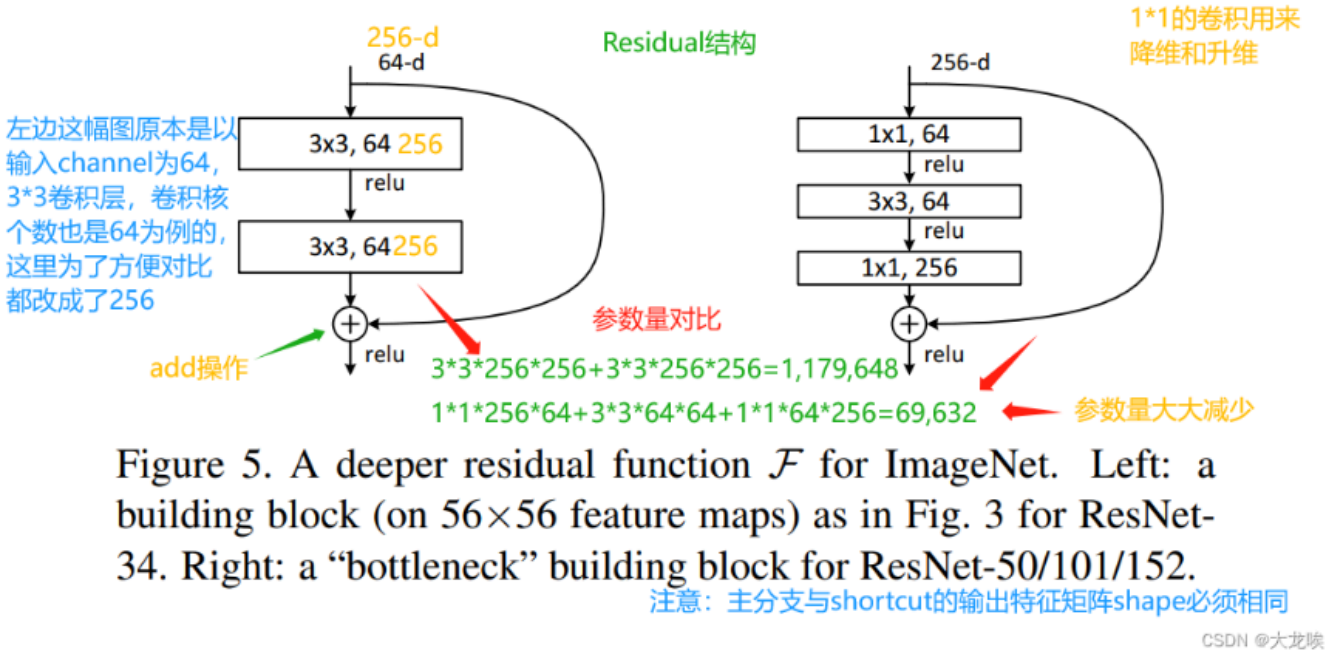


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

注意：主分支与shortcut的输出特征矩阵shape必须相同

- **BasicBlock**

- **从Layer2开始，每个 layer 的第一个 Block 都会升维**，而后续 Block 只是保持这个维度不变
- 两个3×3的卷积核，输出通道为64
- **block**的输入输出通道数相同
- 重点
  1. 在**第1个layer**中通道数**不增加**
  2. 进入到第2个layer的第1个block，**通道数增加**，但在后面的block（相同layer）中**通道数不变**

- **BottleNeck**
  - **每个Layer的第一个 Block 都会升维**，而后续 Block 只是保持这个维度不变
  - 将3×3的卷积层替换为$1 \times 1 + 3 \times 3 + 1 \times 1$
  - 先通过1×1的卷积核进行通道降维，巧妙**扩张或缩减特征图的维度**，一般降到输入维度的四分之一
  - 再用3×3进行主卷积
  - 最后用1×1进行通道升维，一般升到**输入中间通道数**的4倍
  - 重点
    1. 经过每个layer的第1个block之后通道数都会上升，但在后面的block（相同layer）中通道数不变
    2. 通道数增加至**输入中间通道数**，也就是经过1×1的卷积的输出通道数的**4倍**
    3. 输入中间通道数会比输入通道数小**二分之一**

可以减小计算量，同时保证**输入输出维度一致，可进行残差连接**

各类别ResNet一览

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				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	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

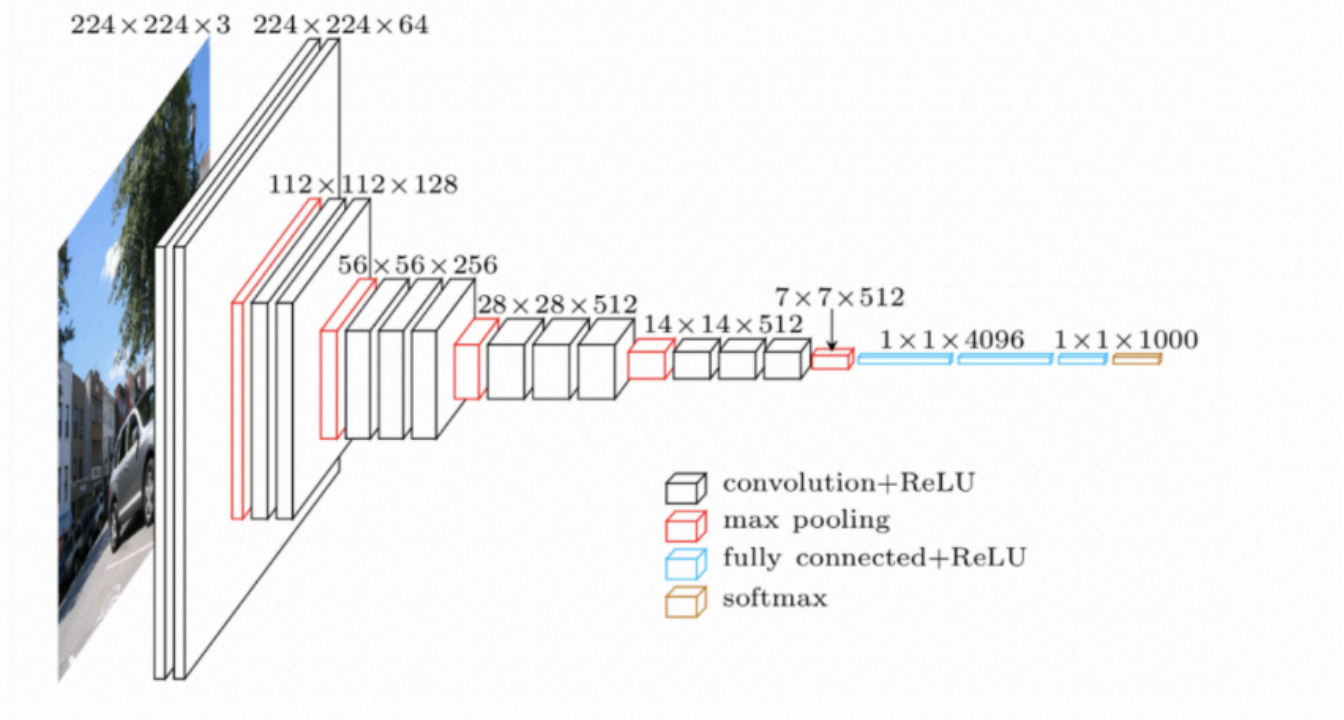
VGG

深度卷积神经网络，固定卷积核大小为3×3，一个非常经典的架构

使用多个小卷积核（3×3）代替大卷积核（如 $5 \times 5, 7 \times 7$）

- 每层使用固定大小的 3×3 **卷积核**

- 步幅为 1（stride = 1），**padding=1**，保证特征图大小不变
- 通过**堆叠多个** 3×3 **卷积**，获得更大的感受野



MobileNet

轻量级神经网络，可部署在边缘设备上

其**计算时间95%**都花费在1×1卷积上

深度可分离卷积

由**深度卷积和逐点卷积**组成（显著减少**计算量和参数量**），最后进行**归一化和非线性激活**

- **深度卷积**用于**滤波**
  - 对每个**输入通道单独进行空间滤波**，提取局部特征
  - 与普通卷积不一样，每个输入通道**独立使用一个卷积核**，不与其他通道**交互**
- **逐点卷积**用于**合并**
  - 混合通道信息，生成新的特征图

宽度乘子

通过系数α等**比例减少所有层的通道数**，进一步压缩模型

分辨率乘子

通过系数ρ**降低输入图像分辨率**，减少计算量

激活函数

使用了`ReLU6`，防止**激活值过大**导致量化时**精度损失**

$$ReLU6(x) = min(ReLU(x), 6)$$

倒残差结构（Inverted Residuals）

借鉴了ResNet里面的**残差结构**

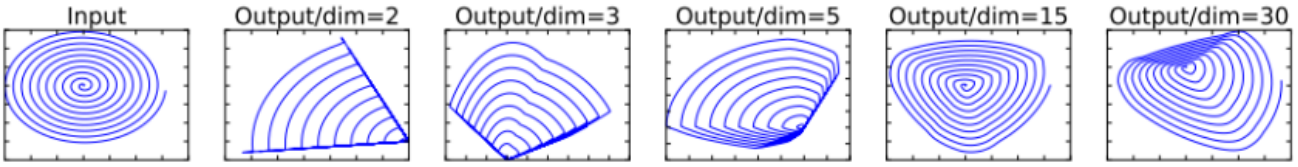
- ResNet里面的BottleNeck是先使用1×1的卷积进行**降维**，再使用3×3的卷积进行**特征提取**，最后使用1×1的卷积进行**降维**
- **MobileNetv2**里面先使用1×1的卷积进行**升维**，再使用3×3的**逐通道卷积（深度可分离卷积）**进行**特征提取**，最后使用1×1的卷积进行**降维**

线性瓶颈

低维空间使用ReLU会**丢失部分特征信息**（如负值被置零），因此在最后一层移除ReLU，改用**线性激活**

原因：

- 在变换过程中，需要将低维空间的信息转换到高维空间，再从高维空间转换为低维空间。**如果输出维度高，信息损失小，输出维度低，信息损失大**，如下图所示：



- 在MobileNetV1中，深度可分离卷积的卷积核中的数值**大部分都为0**，作者猜想是ReLU会把**小于0的部分截断取0**，容易造成信息的丢失，因此把ReLU更换为线性激活函数

Vit

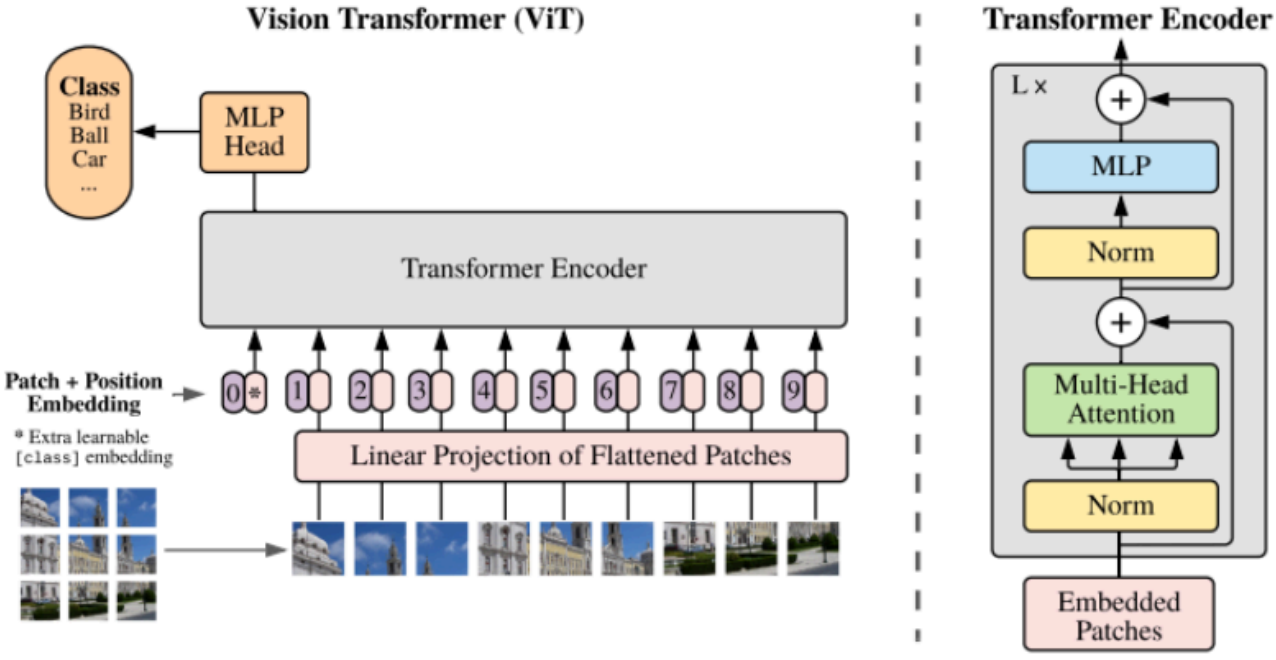
贡献：**首次将Transformer应用于图像分类**，证明自注意力机制可**完全替代传统卷积操作**

工作流程：

1. 将图像进行分块，**分成多个patches**，将图像展平为序列
2. 送入**线性投影层**，将patches投影成D（通常是**768**）维向量，并且加**入位置编码（绝对位置编码）**，再加上一个`[CLS]` Token（类似Bert）
3. 送入transformer的encoder层，**[CLS] Token会与所有patch交互**，但其他patch之间**也互相计算注意力**
4. 从encoder层出来后，将`[CLS]` Token**送入**全连接层进行分类

局限性

1. 依赖**大规模数据集**，在小规模数据集上泛化能力不佳
2. **不像传统CNN具有针对图像的归纳偏置（Instructive Bias）**，也就是模型没有潜意识要去怎么处理图像，在小的数据集上鲁棒性差
3. **对算力要求高，难以处理大分辨率图像**， 因为要把图像分割成很多个patch，还要进行全局注意力计算，对算力要求高



Deit

旨在解决**Vit在小数据集上泛化能力不佳的**问题，引入了**知识蒸馏**并且**改进了训练策略**

知识蒸馏（distillation）

- 使用CNN或者Vit为教师模型进行知识蒸馏，**双教师协同蒸馏**
- Deit相对于Vit在encoder的输出层加入了一个`[distill]`Token，用于接受教师模型的知识
  - `[distill]`专门从教师模型中提取知识，计算教师模型输出的KL散度损失
  - `[CLS]`计算真实标签的交叉熵损失
- 硬标签（hard distillation）：限制两种模型输出的**类别标签**尽可能接近
- 软标签（soft distillation）：限制两种模型输出的**类别分布**尽可能接近，使用**KL散度**进行分布距离的衡量

训练策略优化

- 优化器的改进
- 知识增强
- 正则化等

图像分割

语义分割（Semantic Segmentation）

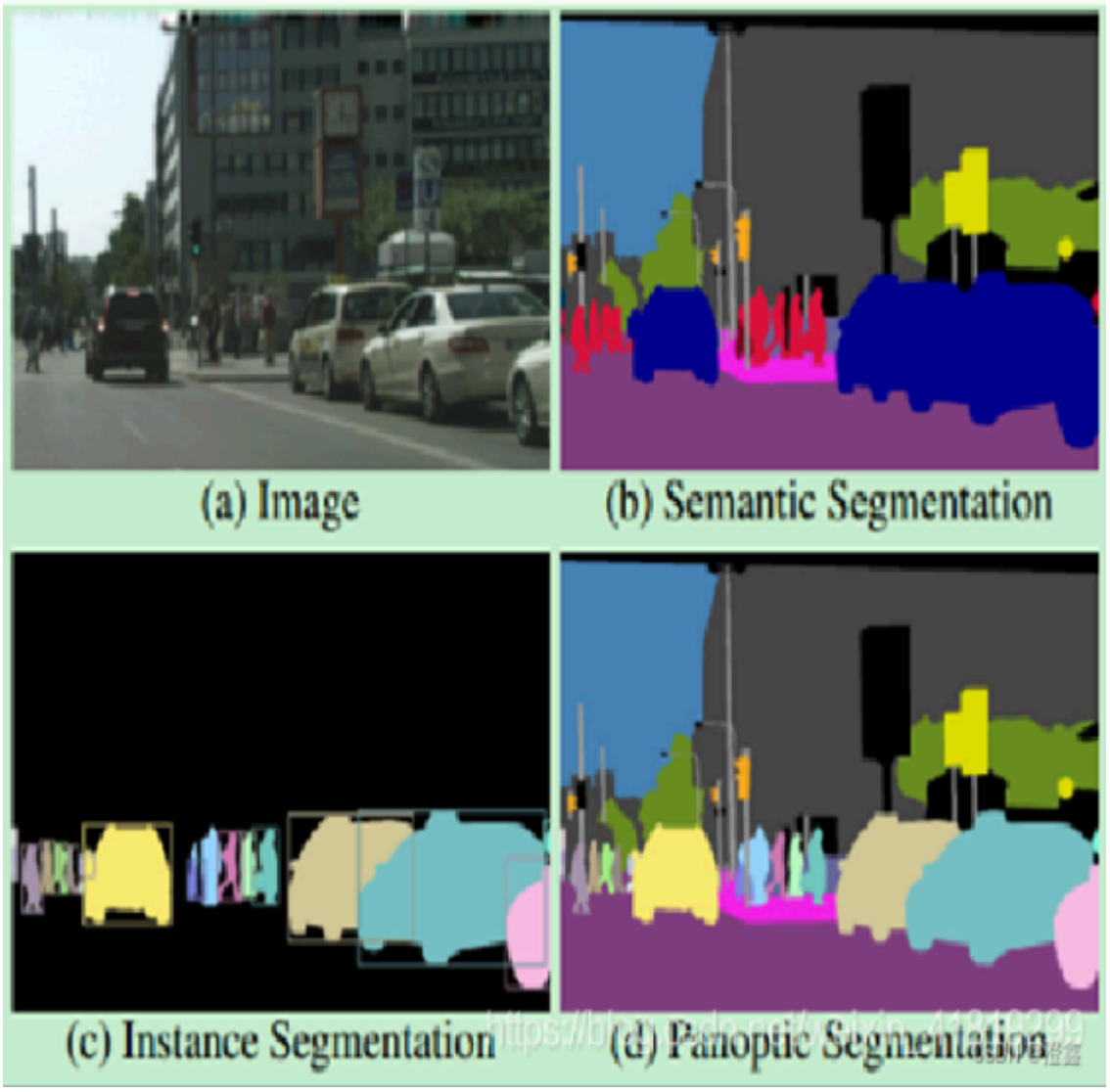
- 为图像中的**每个像素**分配一个类别标签，**不区分同类对象的不同实例**（会将图中所有的人归为**同一类**）
- **只关注像素类别**

实例分割（Instance Segmentation）

- 在语义分割的基础上，**区分同一类别的不同实例**（如区分图中不同的个体）

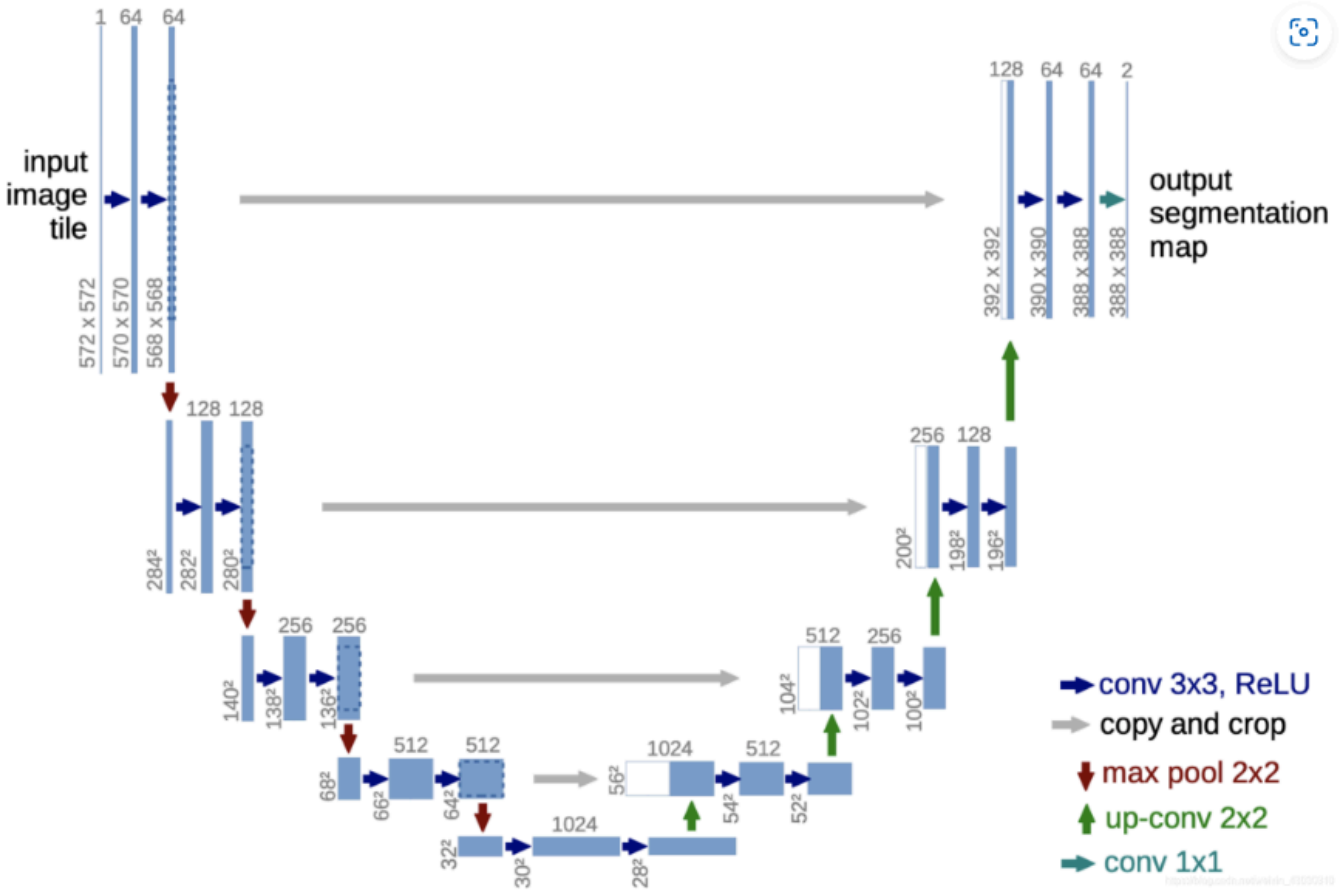
全景分割（ Panoptic Segmentation）

- 统一**语义分割和实例分割**，要求对图像中**所有像素**进行分类，并区分**可数对象（如车辆、人）和不可数区域（如天空、道路）**



U-Net

- Unet采用了**编码器-解码器架构**，并加入了**跳跃连接**，将**浅层的高分辨率特征**直接传递到解码器中，有助于**弥补上采样过程中的细节丢失**
- 适合应用于小型数据集，常用于医学影像分割
- 常用于**语义分割**



工作流程

- 输入图像为$1 \times 572 \times 572$大小
- 蓝色箭头卷积参数为`in_channels=1, out_channels=64, kernel_size=3, stride=1, padding=0`，每进行一次卷积和ReLU，**图像尺寸**`-2`，通道数**逐层增加至1024**
- 最大池化的卷积核和步长都设置为2，每进行一次池化，**输出尺寸减半，通道数不变**
- 每进行一次上采样（转置卷积），**通道数减半，图像尺寸加倍**，同时在解码器模块中也会有卷积参与
- 在进行跳跃连接时，需要对下采样的图像进行**裁剪**，之后**与上采样图像进行拼接**

深监督

原始Unet并未使用这个方法，但这个方法在后来对**Unet的改进中被使用**

我个人认为其作用和**GoogleNet的辅助分类器**大差不差，都是加入了**辅助损失函数防止梯度消失**

深监督指的是在网络的**中间层**添加**辅助损失函数**，简单来说就是加网络的中间层加一个输出层，有利于

- 缓解梯度消失：通过多路径反向传播加速训练
- 改善特征学习
- 早期收敛：浅层网络也能输出初步结果
- 防止过拟合

IoU（交互比）

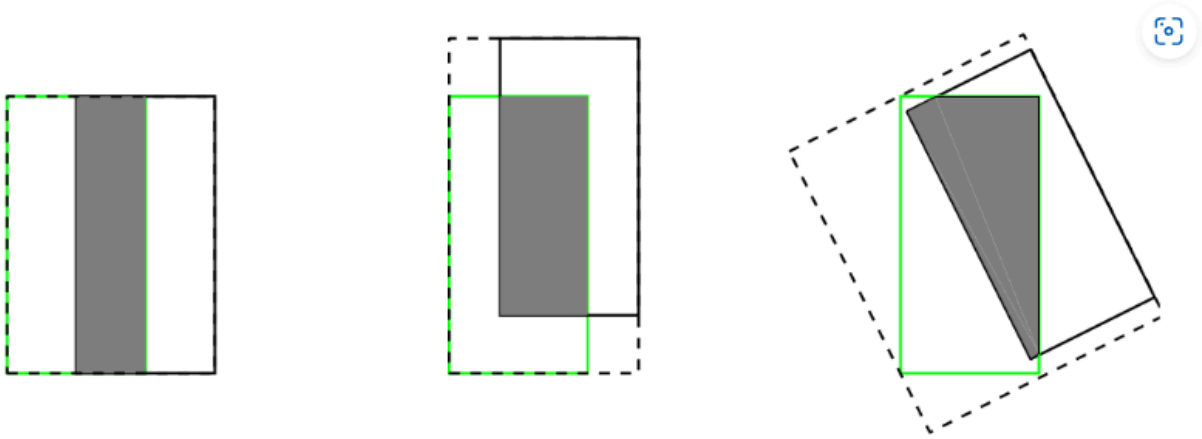
用于衡量**预测区域与真实区域**重叠程度的指标，核心思想是计算**两者的交集与并集的比值**，如下，用预测框（A）和真实框（B）的交集除二者并集

$$IoU = \frac{A \cap B}{A \cup B}$$

常用于**目标检测，图像分割**等场景

但IoU也有如下缺点：

- 当预测框和真实框完全不重合时，梯度为0，无法优化
- 无法精确反映两者重合度大小，相同的IoU可能**有不同的重合度情况**。如下所示，三种情况**IoU都相等**，但他们的重合度情况不一样，左边最好，右边最差。



Dice Loss

基于**Dice系数**，用于衡量预测分割掩码和真实掩码的重叠程度，主要用于二分类任务

$$DiceLoss = 1 - \frac{2 \cdot |A \cap B|}{|A| + |B|}$$

其中：

- A：预测的分割掩码（通常为概率图，值在0~1之间）。
- B：真实的分割掩码（二值图，0或1）。
- $|A \cap B|$：预测与真实掩码的交集（逐像素相乘后求和）。
- $|A| + |B|$：预测和真实掩码的像素值之和。

不依赖**绝对像素数量**，只关注**重叠比例**，对像素少的**目标进行分割仍可以有效优化**

BCE Loss

二分类交叉熵损失函数，衡量预测概率分布与真实分布的差异，在图像分割领域常与**DiceLoss**结合，

$$BCE = -\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i) log(1 - p_i)]$$

其中：

- y_i：真实标签
- p_i：预测概率
- N：像素总数

其梯度稳定，且对**小目标友好**（因为**其对像素级分类更敏感**）

但是**在类别不平衡时，会被多数类（背景）主导，导致模型忽视少数类（肿瘤）**

Deeplab

用于语义分割

v2、v3的backbone（骨干网络）为ResNet

问题：

1. 图像多次下采样分辨率降低
2. 要对目标进行多尺度特征提取

LargeFov

空洞卷积

Deeplab中引入空洞卷积扩大了卷积核的感受野而**不增加参数量或者计算量**！！！，有效捕捉**多尺度上下文信息**

- 无需下采样即可获得大感受野
- 高膨胀率下，卷积核采样点过于稀疏，可能丢失**局部连续性信息**

相较于VGG的改进

1. 将VGG的全连接层（`FC6`、`FC7`）转换为 7×7卷积以保留空间信息。
2. 进一步用 3×3 空洞卷积（`r=12`）替换 7×7 卷积，**以极低计算量实现全局感受野**
3. 最后VGG的**全连接分类层**用1×1卷积核进行分类**（卷积核也可以进行分类）**

ASPP model（V3）

为了水论文而水。。。

核心：多尺度空洞卷积并行处理

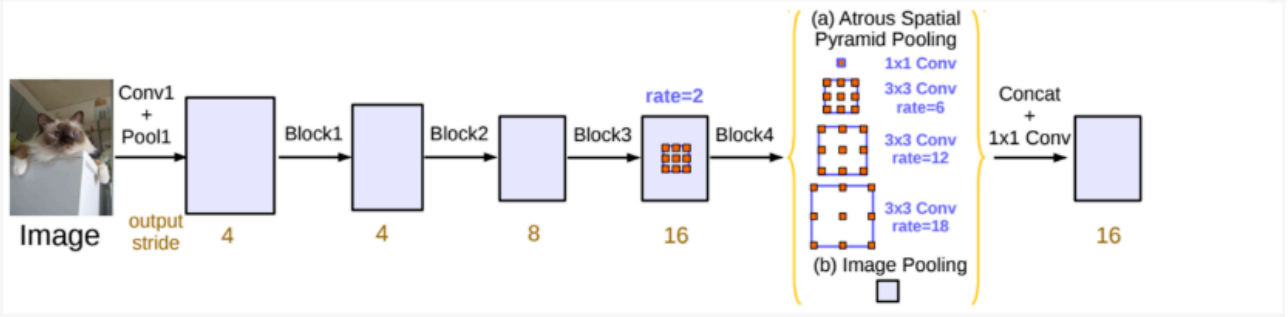


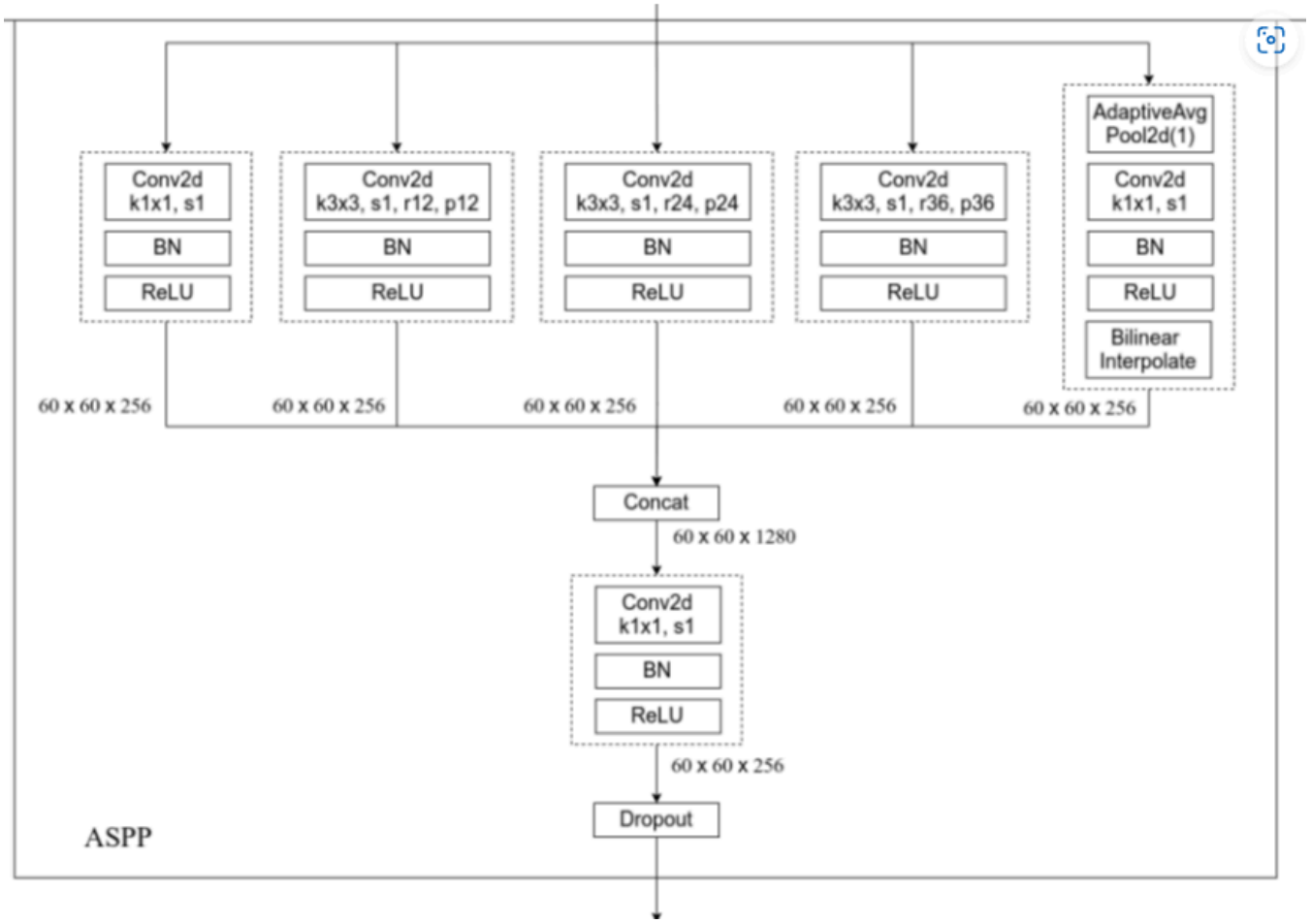
Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

输入为**backbone的输出**，也就是特征图

五个尺度：

- 1×1卷积
- 膨胀系数为12的3×3卷积
- 膨胀系数为24的3×3卷积
- 膨胀系数为36的3×3卷积
- 全局池化

最后进行**channel上的拼接**，再通过1×1的卷积层进行**进一步融合**



需要注意的是：

- 当下采样率设置为8的时候，上图的3×3膨胀卷积的膨胀系数分别为**6、12、18**
- 当下采样率设置为16的时候，上图的3×3膨胀卷积的膨胀系数分别为**12、24、36**（作者说要翻倍，我也不知道为什么）

poly

V2采用的学习率策略（炼丹炼上瘾了）

$$poly = lr \times (1 - \frac{iter}{max\_iter})^{power}$$

其中：

- $power$：衰减强度系数，通常取0.9-2.0
- $iter$：当前步数
- max_iter：总步数

RCNN

用于目标检测呀！

Backbone使用的是VGG16

总体来说分为4步（**非常慢，很耗时，因为不是端对端网络，要分别训练多个网络**）：

1. 一张图像**生成1k~2k个候选区域**
2. 对每个候选区域，使用深度网络**提取特征**（前面说的分类网络本质上都可以）
3. 深度网络输出的特征向量送入**SVM进行分类**
4. 使用回归器精细**修正侯选位置**

非极大值抑制剔除重叠建议框

1. 寻找**得分最高**的目标，也就是经过SVM分类器之后概率最高的
2. 计算**其他目标与得分最高的目标**的IoU值
3. 删除所有IoU值**大于给定阈值**的目标

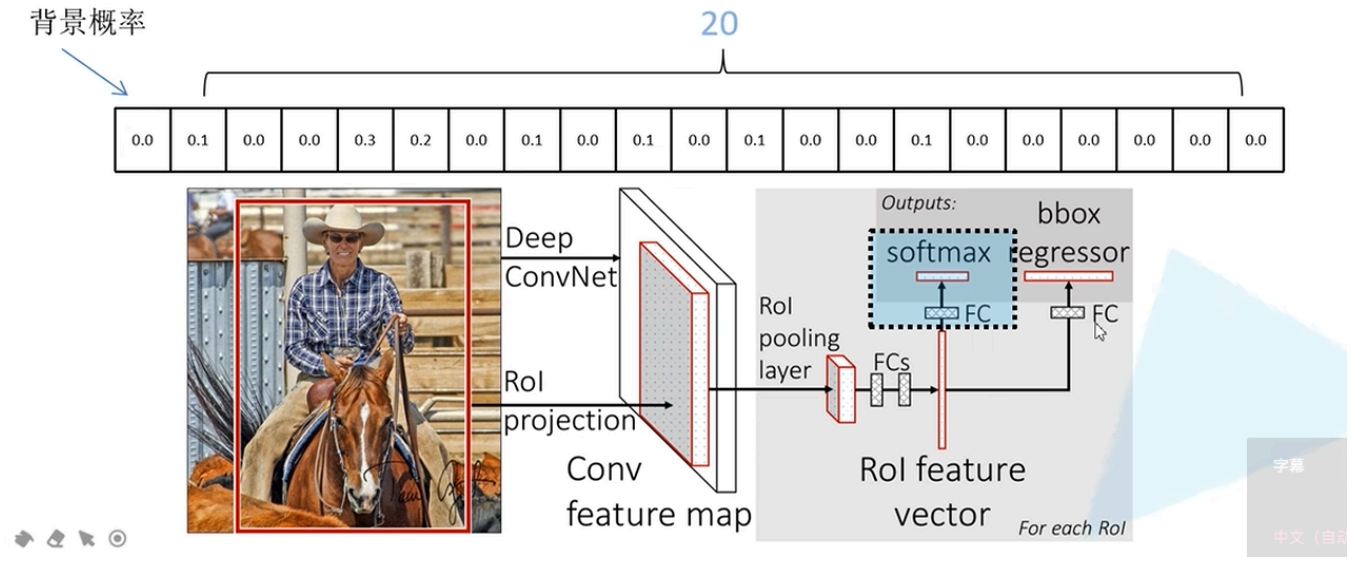
Fast R-CNN

Backbone使用的是VGG16

工作步骤

1. 一张图像**生成1k~2k个候选区域**
2. 将**图像**输入网络得到特征图，通过**SS算法**将生成的**候选框直接投影到特征图**上获得相应的**特征矩阵**
3. 将每个**特征矩阵**通过**ROI pooling**层统一缩放到7×7大小的特征图，将特征图**展平通过一系列全连接层得到预测结果**
    - 有两个输出通道，并联两个全连接层
    - 其中一个全连接层用于**边界框回归参数的预测**，另一个用于**目标概率的预测**（会输出**N+1个类别的概率，N为检测目标的种类，1为背景**）

从SS算法提供的区域进行**正负样本采样**解决数据不平衡的问题，原论文中提出**只要候选框和真实的目标框的IoU大于0.5就是正样本**



边界框回归参数的预测

ROI pooling

1. 输入图像后经过特征提取，得到特征图
2. RoI区域映射到特征图上（映射：与ROI在原图上的位置相对应）
3. 将映射后的区域划分成多个**部分**，**部分**的数量与目的输出的维度有关
4. 对每个部分进行**max pooling**操作

Faster R-CNN

核心：RFN + Fast R-CNN

Clip

Blip

- **上下采样指的是图像分辨率的变化，而不是通道数的变化**
- **消融实验是逐步移除或修改模型的某个组件**（如模块、层、技术等），观察性能变化，从而量化该组件对模型整体的贡献