深度学习

2022年4月23日 21:34

经典网络

2022年5月7日 9:17

1. VGG16采用3*3卷积核的优势？（深度学习-P35）
   - 2个3*3卷积核串联与5*5的卷积核的感受野相同， 3个3*3卷积核串联与7*7感受野相同
     - 计算利用3*3卷积后的输出特征图的大小与5*5（7*7）卷积后的输出特征图大小进行比较
   - 在达到相同效果的前提下，前者的参数量更少（（C_i * K * K + 1）* C_o））
   - 多层的卷积串联增加了更多的非线性映射，增强了模型的学习能力
2. 为什么ResNet和DenseNet网络可以构建的很深
   - ResNet
     - 添加了残差模块，构建了浅层到深层之间的直接映射，防止了梯度消失问题
   - DenseNet
     - 构建了浅层与之后所有层之间的密集链接，实现了特征重用，每一层都可以直达最后的误差信号，避免了梯度消失问题
3. Attention机制的作用?
   - 给予了模型区分辨别的能力，让模型更容易从输入信息中找到与当前数据相关的有用信息
   - CV中的Attention使得模型更加关注重点信息而忽略无关信息（一种权重分配机制）
4. GAN网络的思想
   - GAN包括生成模型和判别模型
   - 生成模型：用于生成一副与给定图像很相似的图像
   - 判别模型：用于判断给定的图片是否是真实图像
   - 生成模型和判别模型一起进行对抗训练，生成模型生成图片欺骗判别模型，判别模型判别真假，最终两个模型在训练过程中达到稳态
5. LeNet-5网络的结构（深度学习-P33）
   - 用于手写体识别的CNN网络
   - 结构：卷积+池化+卷积+池化+FC+FC+FC+MLP分类（conv:5*5）
6. AlexNet网络
   - 首个用于图像分类的CNN网络
   - 卷积（11*11）+池化+卷积（5*5）+池化+卷积（3*3）+卷积（3*3）+卷积（3*3）+FC+FC+FC+Softmax分类
7. DNN，CNN，RNN的区别
   - DNN：包含多个隐藏层的MLP，层与层之间的神经元都是全连接
   - CNN：通过卷积核将上下层进行连接，实现局部连接和权值共享
   - RNN：考虑了时间序列上的建模，DNN和CNN中每一层的神经元信号只能向下一层传播，RNN中的神经元不仅可以向下一层传播还可以在下一个时间段作用于自身
8. DNN在图像识别上的效果为什么没有CNN好?
   - DNN的输入是向量形式，无法考虑到平面图像的二维信息
   - CNN的输入是张量形式，可以是二维矩阵，通过卷积核可以获取局部特征，较好的保留了平面图像的结构信息
9. Anchor-Free和Anchor-Base的区别以及Anchor-Free的实现
   - 区别：
     - 是否利用Anchor提取候选目标框
     - Anchor-Base需要计算GT与Anchor之间的IOU，并设置阈值，判断正负样本
     - Anchor-Free（CenterNet）只取GT的中心一点作为Anchor，不需要计算IOU，也不需要NMS
   - Anchor-Free网络：CenterNet，ExtremeNet，CornerNet
   - Anchor-Free实现：
     - 基于多关键点预测：
       - CornerNet：左上角+右下角
       - CenterNet（Keypoint Triplets for Object Detection）：左上角点+右下角点+中心点
     - 基于单中心点预测
       - CenterNet（Object as Points）：中心点+高度+宽度
10. 为什么全连接网络不适合图像识别任务?
    - 参数太多
    - 输入是向量形式，无法利用图像像素之间的位置关系
    - 全连接参数固定，对输入图像尺寸有限制
11. ASPP模块
    - 空洞卷积池化金字塔
    - 作用：在不改变特征尺寸的前提下，增大网络的感受野，提高网络获得多尺度上下文信息的能力
    - 对于输入特征，利用不同空洞率的空洞卷积进行并行采样

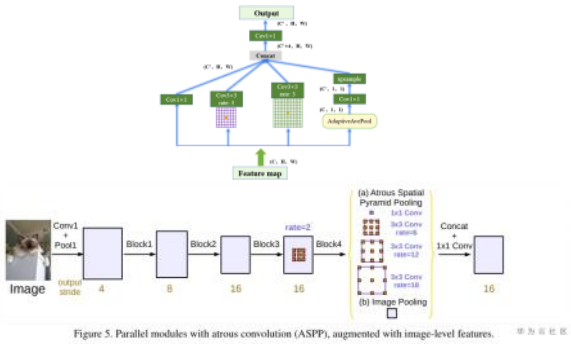
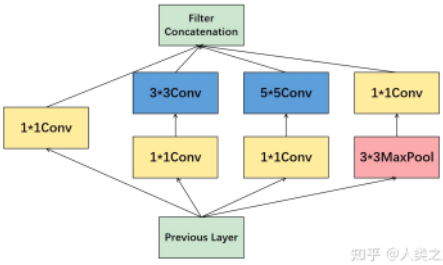
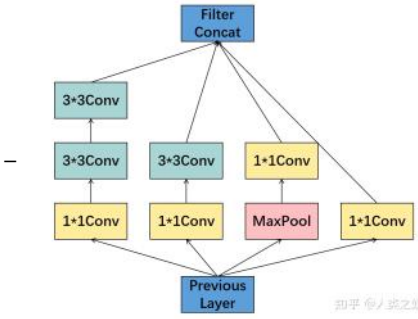


Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

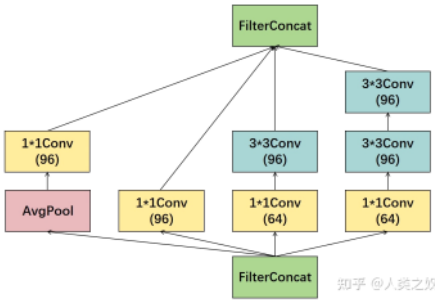
12. 如何实现模型轻量化
    - 网络剪枝
    - 网络量化
    - 设计轻量化模块（组卷积，深度可分离卷积）
    - 知识蒸馏
13. GoogleNet（串并联网络）
    - 动机
        - VGG，LeNet、AlexNet等经典网络都是串行结构
        - 感受野固定，导致提取的特征单一
        - 网络过深，易引起梯度消失，过拟合等问题
    - 基本组件：Inception
        - Inception-v1：将多尺度的卷积层、池化层提取的特征图拼接输入下一层，提升模型多尺度特征提取能力



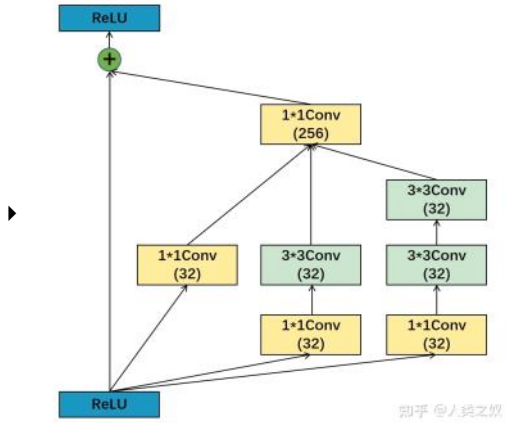
        - Inception-v2：将Inception-v1中的大尺寸卷积核替换为多个小尺寸卷积核的堆叠，减少模型参数量，增加网络深度，提升模型拟合复杂分布的能力



        - Inception-v3：使用RMSProp替代Inception-v2的SGD
        - Inception-v4：基本沿袭了Inception-v2/v3的设计，它的各个模块在结构上更加统一



        - Inception-ResNet：借鉴了ResNet的残差连接思想，达到特征复用的目的

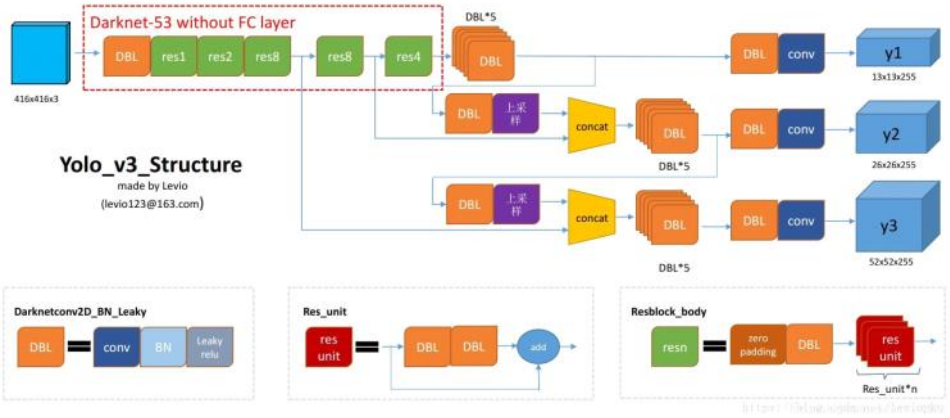


14. Yolov3
    - Anchor框设置（对预测的目标范围进行约束，并加入了尺寸先验知识）
        - Anchor框尺寸是从GT中聚类得到
        - 每个cell设置3个Anchor
    - 三个基本组件
        - CBL：Conv+BN+Leaky Relu
        - Res unit：CBL + CBL + add残差（残差结构为了构建更深的网路）
        - ResX：CBL+X个Res unit（最前面的CBL为了下采样）
    - 主干网络：Darknet53

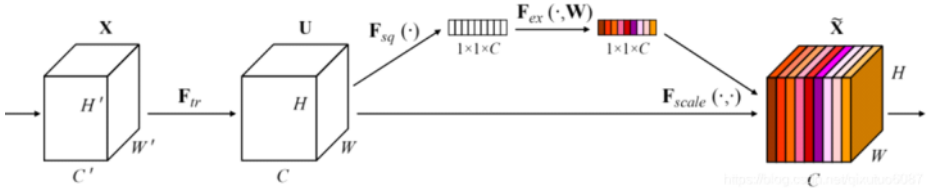
      Backbone中卷积层的数量：

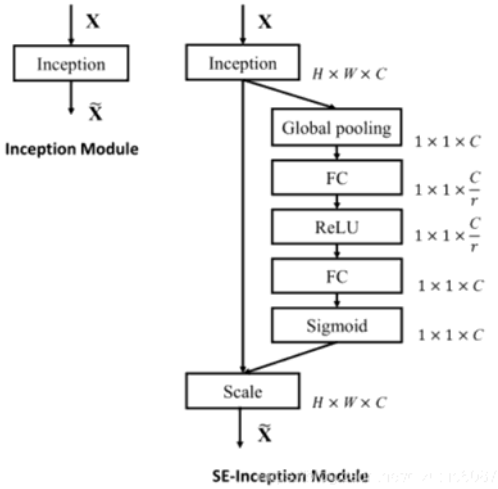
      每个ResX中包含1+2*X个卷积层，因此整个主干网络Backbone中一共包含
        - **1+（1+2*1）+（1+2*2）+（1+2*8）+（1+2*8）+（1+2*4）=52**，再加上一个FC全连接层，即可以组成一个**Darknet53分类网络。**不过在目标检测Yolov3中，去掉FC层，不过为了方便称呼，仍然把**Yolov3**的主干网络叫做**Darknet53结构。**

    - 激活函数采用Leaky Relu的原因：不会产生死亡神经元



12. SE-Net





13. SK-Net
    - 提出了一种非线性方法来聚合来自多个卷积核的信息，以实现神经元的自适应感受野大小

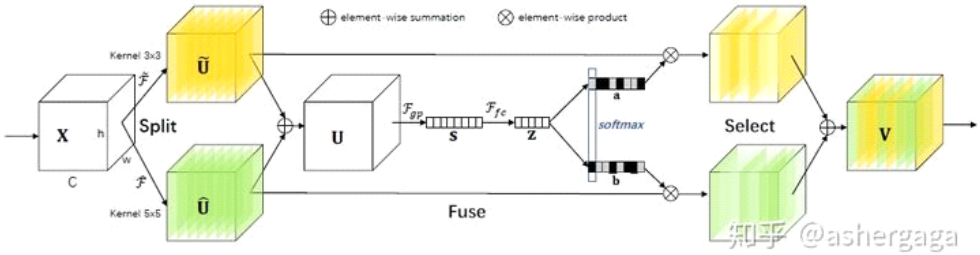
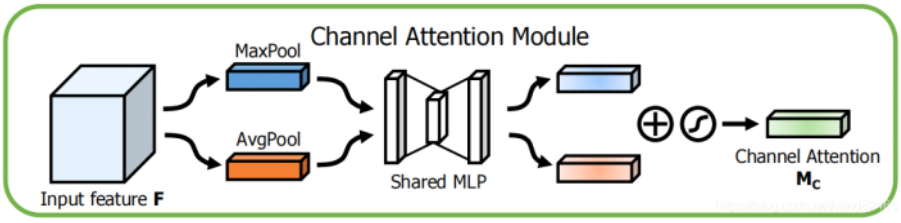


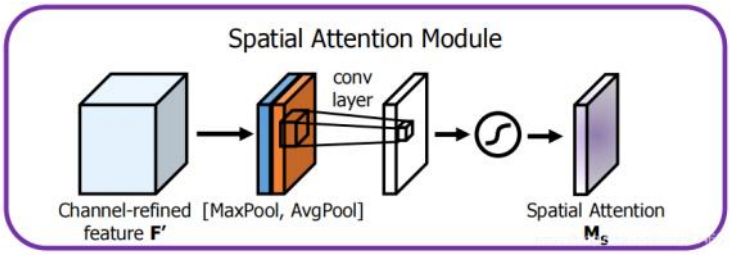
Figure 1. Selective Kernel Convolution.

14. CBAM
    - 混合域的注意力机制，CBAM将通道域和空间域同时融合到网络模型中
        - 通道维



$$\begin{aligned}\mathbf{M_c}(\mathbf{F}) &= \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F}))) \\ &= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),\end{aligned} \tag{2}$$

        - 空间维



$$\begin{aligned}\mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),\end{aligned} \tag{3}$$

    - 整体框架

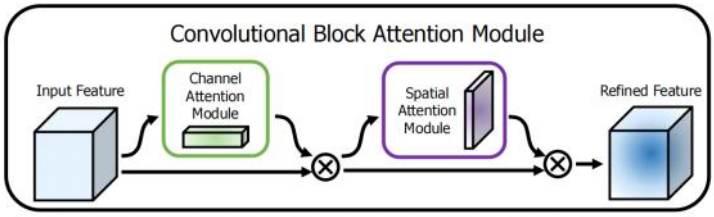


Fig. 1: **The overview of CBAM**. The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.

15. K-means对检测框聚类
    - Yolov2聚类的度量单位不是距离，而是两个框的交并比
    - 最终的结果是K个聚类中心的宽和高

已知：$\mathbf{box_1} = (w_1, h_1)$，$\mathbf{box_2} = (w_2, h_2)$

$\mathbf{box_1}$ 和 $\mathbf{box_2}$ 的交并比：

- $$J(\mathbf{b_1}, \mathbf{b_2}) = \frac{\min(w_1, w_2) \cdot \min(h_1, h_2)}{w_1 h_1 + w_2 h_2 - \min(w_1, w_2) \cdot \min(h_1, h_2)}$$

YOLOv2中不使用欧式距离，使用如下公式来计算两个框的距离

$$d_J(\mathbf{b_1}, \mathbf{b_2}) = 1 - J(\mathbf{b_1}, \mathbf{b_2})$$

16. 大目标中包含小目标，NMS会不会将小目标抑制掉
    - 首先判断是否属于同一类别，NMS属于类内抑制
    - 如果是类内，可以采用 Soft-NMS 或者 增大IOU阈值设定
17. 3D视觉与2D视觉的区别
    - 最大区别：处理的数据类型不同
        - 3D视觉处理的是3维点云数据，2D视觉处理的是平面图像信息
        - 3D不仅可以感知场景中物体的有无，也能够准确感知物体离我们距离的远近，物体尺寸，物体位置朝向等
    - 目标识别
        - 3D视觉需要确定目标在场景中的位置和姿态
18. 深度学习的发展瓶颈
    - 数据匮乏
        - 深度学习主要由数据驱动，数据可获得性，数据质量，数据标注成本等都是制约深度学习发展的一大因素
    - 对标注数据的依赖性大
        - 难以得到较好的迁移学习效果
    - 不够透明
        - 内部机制难以解释，对于医学以及金融等行业，决策机制相当重要
    - 应用场景局限
        - 目前发展较好的只有分类，匹配，翻译等领域
        - 聊天机器人很容易遇到障碍
19. 目标检测的实际应用
    - 人脸识别
    - 智慧交通领域：自动驾驶（车辆，行人以及路况的检测）
    - 医疗领域：医学影像识别
    - 工业检测：产品缺陷检测
20. 语义分割的实际应用
    - 地理，农业，测绘，环境监测
21. 小目标问题
    - 小目标的定义
        - 绝对尺寸：在COCO数据集（400-600）中，尺寸小于32×32像素的目标被认为是小目标
        - 相对尺寸：目标的尺寸小于原图的0.12%，则认为是小目标
    - 小目标检测精度低的原因
        - 可利用特征少
            - 小目标可视化信息少，难以提取到具有判别力的特征，容易被环境因素干扰
        - 目标尺寸与深度网络提取特征之间的矛盾

        - 随着网络加深，目标信息量进一步减少，导致深层特征对小目标的表达能力较弱
    - 定位精度要求高
        - 小目标在图像中覆盖的面积小，在预测过程中，预测边界框偏移一个像素点都会造成很大的误差
    - 后处理操作NMS导致样本不均衡
        - 设定固定的阈值来判断Anchor是否属于正样本
        - 小目标的训练正样本远远小于大目标的正样本，导致模型忽略对小目标的检测
        - 小目标得不到充分的训练，训练的模型倾向于拟合大目标特征，在小目标上的泛化能力较弱
- 提升小目标检测精度的方法
    - 数据方面
        - 数据增强：增加小目标的数量，提高网络对小目标的拟合能力
            - 常见：旋转、剪裁、缩放、翻转等
            - 对包含小目标的样本进行过采样（易过拟合）
            - 将小目标拷贝并在一定范围内进行尺度和角度的随机变换，之后粘贴在图像的不同位置上
                - 增加小目标匹配到的锚点框数量，并提高小目标位置的多样性，促使模型更加关注小目标
    - 锚框设置
        - Anchor对多尺度目标的适应性较弱，特别是对小目标检测召回率过低
        - 铺设密集Anchor有助于小目标捕获，但会引入较多的计算量和参数量
        - 使用K-means算法对训练数据集目标的真实边界框进行聚类分析，获得合适的锚点框尺寸以降低目标定位的优化难度
    - 特征方面
        - 多尺度特征融合
            - 思想：多尺度特征融合通过自上而下的横向连接将低层特征与高层特征相互融合，构建具有细粒度特征和丰富语义信息的特征表示，有利于小目标的检测
            - 举例
                - FPN
                - DSSD：反卷积操作恢复特征图尺寸
                - RSSD：多尺度特征级联
                - YOLO V2：pass-through层
        - 采用多尺寸感受野
            - 思想：目标周围区域的上下文信息有利于增强小目标的细粒度特征
            - 举例
                - 空洞卷积，空洞卷积
        - 引入注意力
            - 空间注意力增强
    - 损失函数
        - Focal Loss：增加小目标对损失函数的贡献度

22. 大小目标共存问题
- 问题根源：CNN提取大小目标特征时特征表征不一致
    - 随着网络加深，目标信息量进一步减少，导致深层特征对小目标的表达能力较弱
        - 细节信息
            - 目标的细节信息都随着模型层数的加深而衰退
        - 语义信息
            - 小目标尺寸小，语义信息可能在较浅层提取完毕，之后随着层数增加，小目标的语义信息也会快速被环境信息稀释
            - 大目标尺度大，要在更深层才能提取到足够的语义信息，但此时小目标的语义信息已经丢失严重
- 解决方法
    - 多分支并行获取不同大小的感受野，以分别处理不同尺度的目标
        - 不同空洞率的卷积，形变卷积
            - 空洞卷积、可形变卷积扩张感受野使检测器对尺度跨度大的问题适应性更强
        - SNIP
            - 只对尺寸在指定范围内的目标，进行梯度回传
            - 每个目标在训练时都会有几个不同的尺寸，总有一个尺寸在指定的尺寸范围内
            - ROI在训练中是否回传梯度和预训练模型的数据尺寸相关
                - valid anchor的定义是和invalid ground truth的IOU小于0.3的anchor

23. 像素级分类与语义分割的区别
- 语义分割
    - 输入是一整张原始图片，通过编码-解码结构
        - 编码器使用卷积和池化将特征图尺寸缩小，使其成为更低维的表征

        - 解码器接收到这一表征，通过转置卷积执行上采样而「恢复」原始尺寸
    - 最终，解码器输出一个与原始输入图像大小相同的特征张量，且它的通道数就是类别数
    - 通过Softmax处理后确定每个位置的类别
- 像素级分类
    - 输入是以每个待预测像素点为中心的Patch
    - 通过神经网络提取特征
    - 通过全局平均池化压缩为一维向量
    - 利用全连接层对每个像素点进行类别预测

24. 密集目标检测常用方法
    - Anchor设置：小而密集
    - Loss：带权重Loss，关注难样本（Focal Loss）
    - Soft-NMS：避免重叠较多，直接过滤
    - 感兴趣区域：用ROI Align 代替 ROI Pooling
    - 边界框：旋转边界框 代替 水平边界框
25. MobileNet（轻量化网络）
    - 核心：深度可分离卷积（整个网络是深度可分离卷积的堆叠）
    - MobileNet V1
        - 结构

$$\textbf{MobileNet V1} \longrightarrow \underset{\textbf{DW}}{3\times 3} \xrightarrow{ReLU6} \underset{\textbf{PW}}{1\times 1} \xrightarrow{ReLU6}$$

        - $\textbf{MobileNet V2} \longrightarrow \underset{\textbf{PW}}{1\times 1} \xrightarrow{ReLU6} \underset{\textbf{DW}}{3\times 3} \xrightarrow{ReLU6} \underset{\textbf{PW}}{1\times 1} \xrightarrow{Linear}$ 知乎 @cicada
        - 缺点
            - 当输入特征维度较低时，DW只能在低维计算，ReLU也在低维计算，信息丢失严重
            - 网络结构是简单的堆叠，缺乏特征融合和复用，特征提取不充分
    - MobileNet V2
        - 结构

$$\textbf{ResNet} \xrightarrow{C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{0.25\times C_{in}} \underset{\textbf{Standard CONV}}{3\times 3} \xrightarrow{0.25\times C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{C_{out}(\approx C_{in})}$$

        - $\textbf{MobileNet V2} \xrightarrow{C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{6\times C_{in}} \underset{\textbf{DW}}{3\times 3} \xrightarrow{6\times C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{C_{out}(\approx C_{in})}$ 知乎 @cicada
        - 改进
            - V2 去掉了第二个 PW 的激活函数改为线性激活
                - 激活函数在高维空间能够有效的增加非线性，而在低维空间时则会破坏特征，不如线性的效果好
            - V2 在 DW 卷积之前新加了一个 PW 卷积
                - 专门用来升维
        - 与ResNet的区别

$$\textbf{ResNet} \xrightarrow{C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{0.25\times C_{in}} \underset{\textbf{Standard CONV}}{3\times 3} \xrightarrow{0.25\times C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{C_{out}(\approx C_{in})}$$

        - $\textbf{MobileNet V2} \xrightarrow{C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{6\times C_{in}} \underset{\textbf{DW}}{3\times 3} \xrightarrow{6\times C_{in}} \underset{\textbf{PW}}{1\times 1} \xrightarrow{C_{out}(\approx C_{in})}$ 知乎 @cicada
            - ResNet：先降维+卷积+再升维（沙漏形）
            - MobileNet V2：先升维+卷积+再降维（纺锤形）
                - 为了使用DW卷积而作的适配，希望特征提取能够在高维进行
    - MobileNet V3
        - 思想
            - 使用AutoML 技术为给定的问题找到最佳的神经网络架构
                - MnasNet（一种自动移动神经体系结构搜索（MNAS）方法）

- 结构
  - 继承V1的深度可分离卷积
  - 继承V2的纺锤形残差结构
  - 残差结构的DW之后添加SE-Net注意力机制
  - 使用新的激活函数h-swish(x)代替ReLu6，速度优化
    - $\text{h-swish}[x] = x\frac{\text{ReLU6}(x+3)}{6}$

26. CNN、RNN、LSTM和Transformer之间的优缺点
    - CNN
      - 优点
        - 提取特征时具有平移不变性，更好的表征目标特征
        - 局部连接、权值共享使得网络可以较好的处理高维数据
      - 缺点
        - 网络过深时，反向传播过程中易出现梯度消失和梯度爆炸
        - 采用梯度下降法迭代时，训练结果易收敛于局部最优解
        - 池化层会丢失一些有价值信息
        - 解释性不足
    - RNN
      - 优点
        - 时间维度的建模
        - 每一时刻隐藏层的信息不仅由该时刻的输入层决定，还可以由上一时刻的隐藏层决定
        - 更好的挖掘数据的时序信息
      - 缺点
        - 反向传播局限性，长序列场景处理时，易出现梯度消失或者梯度爆炸
        - 具有短期记忆性，缺乏长程依赖性
    - LSTM
      - 优点
        - LSTM引入遗忘门、输入门、输出门结构，有效的缓解了梯度消失问题，建立了较好的长程依赖性
      - 缺点
        - 当时序跨度较大，网络较深时，会出现计算量较大和耗时偏多的问题
        - 细胞状态，上一状态的所有信息，并且会一直向后传递，门控机制是对有效信息的筛选
    - Transformer
      - 优点
        - 相比于RNN，可以并行计算序列数据
        - 注意力机制更具有解释性，各个注意力头可以学习执行不同任务
      - 缺点
        - 局部信息的获取能力弱于CNN
        - 位置信息编码依赖于人为的设计

27. 目标检测标签分配方法
    - 为了训练目标检测器，为每个anchor 分配 cls 和 reg 目标，这个过程称为标签分配或者正采样
    - 方法
      - 使用 Anchors 与 GT 的 IoU 阈值来区分正负样本
        - Faster R-CNN、SSD
      - 集合预测方式，一一对应标签分配
        - DETR、YOLOS
      - 中心点落在哪个网格中，哪个网格负责预测
        - YOLO v1
      - 纵横比匹配
        - YOLO v5
          - 将GT与同一特征图上的所有Anchor进行纵横比匹配
          - 如果纵横比大于设定阈值，则说明GT和对应的Anchor不匹配
          - 如果某个GT有匹配到Anchor，那么就算GT的中心点落在哪个cell中
          - 寻找与该cell最近邻的两个cell，这三个cell中的Anchor都负责预测该GT的目标

28. 多模态遥感数据的配准
    - 不同传感器通过轨道参数以及严格的几何定位已经进行了粗配准，消除了地物之间的旋转和尺度等几何形变问题
    - 但不同模态数据中，同一地物呈现出完全不同的灰度信息，导致同名点的匹配依旧十分困难
    - 基于结构相似性的进行同名点匹配，利用方向梯度直方图进行相位一致性匹配

29. YOLO V7
    - 在COCO数据集上，取得了当前已知的目标检测器中的最高精度
    - 从 模型重参数化 和 动态标签分配 两个维度进行改进
    - 动态标签分配
        - 根据预测输出结果选出最优的 Anchor 对 GT 进行匹配，确定正样本
        - 而不是使用先验的固定规则：IOU最大、最接近Anchor的中点、根据尺寸比例等 进行匹配
        - OTA
            - 思想
                - 两个供应商
                    - GT供应商：提供一定数量Labels
                    - Background供应商：为Anchor提供 Negative 标签
                - 需求方：Anchor
                    - 需要唯一Label的需求方
                    - 如果 Anchor 从 某个GT 那儿获得了 足够的Label，那么这个 Anchor 就是此 GT 的一个正样本
            - 目标问题：供应商 如何分配 Labels 给 需求方，可以让 Cost 最低
                - 对于GT-Anchor对：Cost是 类别损失（交叉熵损失）和 位置损失(IOU损失）的加权和
                - 对于BG-Anchor对：Cost是 类别损失
    - 模型重参数化
        - 一种模型压缩技术
        - 思想：
            - 训练时使用复杂的结构，通常使用多分支结构，使得模型具备某种好的性质
                - 多分支架构：训练更加稳定且容易，但是推理速度慢，占用的内存大
            - 测试推理时，将复杂模型转换为推理模型，通常使用单分支结构，使得推理模型结构较小，且保留这种好的性质
                - 单分支结构：推理速度快，节省内存，但是训练比较困难，训练的性能较低。
            - 训练时的结构对应一组参数，推理时的结构对应另一组参数；只要能把前者的参数等价转换为后者，就可以将前者的结构等价转换为后者
        - 举例
            - 训练时采用多分支的网络使模型获取更好的特征表达，测试时将并行分支融合成串行，降低计算量和参数量，提升速度
            - RepVGG
                - 训练阶段
                    - 三分支结构：3*3卷积，1*1卷积，恒等映射
                - 推理阶段
                    - 单分支：3*3卷积
                - 转换过程
                    - 卷积核权重（卷积核的可加性质）
                        - 1*1卷积：看做有很多0填充的3*3卷积
                        - 恒等映射：全是1填充的3*3卷积
                        - $conv(x, W1) + conv(x, W2) + conv(x, W3) = conv(x, W1 + W2 + W3))$
                    - BN层参数（“吸BN”方法)
                        - 将BN层参数合并到卷积中

                          而BN操作可以等价为：

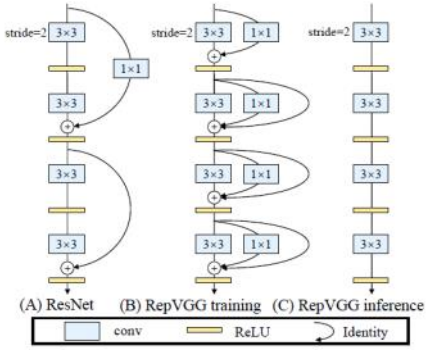
                          $$\mathrm{bn}(\mathrm{M}, \mu, \sigma, \gamma, \beta)_{:,i,:,:} = (\mathrm{M}_{:,i,:,:} - \mu_i)\frac{\gamma_i}{\sigma_i} + \beta_i \,. \qquad (1.2)$$

                          把BN的参数合并到卷积中的过程为：

                        - $$\mathrm{W}'_{i,:,:,:} = \frac{\gamma_i}{\sigma_i}\mathrm{W}_{i,:,:,:}\,, \quad \mathbf{b}'_i = -\frac{\mu_i\gamma_i}{\sigma_i} + \beta_i \,. \qquad (1.3)$$

                          合并之后卷积就成为了：

                          $$\mathrm{bn}(\mathrm{M} * \mathrm{W}, \mu, \sigma, \gamma, \beta)_{:,i,:,:} = (\mathrm{M} * \mathrm{W}')_{:,i,:,:} + \mathbf{b}'_i \,. \qquad (1.4)$$
                - 结构图

- ◇ 

30. Center-Net
    - □ 使用关键点预测来确定中心点，然后回归其他的属性（宽、高）
    - □ 思想
        - 首先将图像输入一个全卷积网络产生热点图
        - 图上的峰值点对应目标的中心
        - 根据每个峰值点附近的图像特征来预测目标的宽、高
    - □ Center-Net与Anchor-base的区别
        - CenterNet确定的 关键点 只与位置相关，与框的重叠度无关
        - 每个目标只对应一个关键点，因此不需要NMS
        - CenterNet输出的分辨率较大OS=4
    - □ Center-Net训练过程
        - 令 $I \in R^{W \times H \times 3}$ 表示宽高分别为 W, H 的输入图像，目标是预测一个关键点热力图 $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$，其中 R 表征输出的特征图尺寸，C 表示关键点类型的数量（或者目标检测中的目标类别数），论文中 $R=4$。在热力图中 $\hat{Y}_{x,y,c}=1$ 对应一个被检测到的关键点，$\hat{Y}_{x,y,c}=0$ 表示背景。
        - 训练
            - □ 关键点损失

训练过程，对于每个类别为 c 的GT关键点 $p \in R^2$，模型都会计算一个低分辨率的替换值 $\tilde{p}=\lfloor \frac{p}{R} \rfloor$，然后使用一个高斯核函数 $Y_{xyc}=exp\left(-\frac{(x-\hat{p}_x)^2+(y-\hat{p}_y)^2}{2\sigma_{\hat{p}}^2}\right)$ 将所有的GT关键点赋予到一幅热力图 $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$，如果同一类别的两个高斯生成点重叠，则以较大值为准，训练的目标损失函数如下：

$$L_k=\frac{-1}{N}\Sigma_{xyc}\begin{cases}(1-\hat{Y}_{xyc})^{\alpha}log(\hat{Y}_{xyc}) & if\ Y_{xyc=1}\\ (1-Y_{xyc})^{\beta}(\hat{Y}_{xyc})^{\alpha}log(1-\hat{Y}_{xyc}) & otherwise\end{cases}$$

其中 α, β 是focal loss的超参数，论文中 $\alpha=2, \beta=4$，N 表示图像 I 中关键点的数量，除以 N 是为了对所有的正损失归一化。

            - □ 偏移损失

为了消除由于OS带来的离散误差，论文为每个中心点额外预测了一个偏移量 $\hat{O} \in R^{\frac{W}{R} \times \frac{H}{R} \times 2}$，所有的类别 c 共享同一个偏移与测量，这里使用如下所示L1损失函数：

$$L_{off}=\frac{1}{N}\Sigma_p\left|\hat{O}_{\hat{p}}-\left(\frac{p}{R}-\hat{p}\right)\right|$$

此监督旨在关键点位置 $\hat{p}$ 起作用。

            - □ 尺寸损失

令 $(x_1^{(k)}, y_1^{(k)}, x_2^{(k)}, y_2^{(k)})$ 表示目标类别为 c_k 的目标 k 的边界框，其中心点 $p_k = (\frac{x_1^{(k)} + x_2^{(k)}}{2}, \frac{y_1^{(k)} + y_2^{(k)}}{2})$，使用关键点估计 $\hat{Y}$ 来预测所有的中心点，另外，为每个目标 k 进行尺寸 $s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - x_1^{(k)})$ 的回归，使用L1损失来指导训练：

- $$L_{size} = \frac{1}{N}\Sigma_{k=1}^{N}\left|\hat{S}_{p_k} - s_k\right|$$

最终检测损失如下，其中 $\lambda_{size} = 0.1, \lambda_{off} = 1$：

$$L_{det} = L_k + \lambda_{size}L_{size} + \lambda_{off}L_{off}$$

- 最终实现了一个单一网络来预测关键点、偏移量和尺寸，每个位置对应 （C+4） 个模型输出

- Center-Ne推理过程

在推理过程中，首先提为每个类别单独的从热力图中提取峰值点，方法是检测3邻域中的最大值，令 $\hat{P}_c$ 表示 n 个被检测到的类别为 c 的中心点 $\hat{P} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{n}$ 的集合，每个关键点的位置坐标由 (x_i, y_i) 给出，使用关键点的值 $\hat{Y}_{x_i y_i c}$ 作为检测的置信度然后在该位置产生一个边界框：

- $$(\hat{x}_i + \delta\hat{x}_i - \hat{\omega}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,$$
  $$\hat{x}_i + \delta\hat{x}_i + \hat{\omega}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2)$$

其中 $(\delta\hat{x}_i, \delta\hat{y}_i) = \hat{O}_{\hat{x}_i, \hat{y}_i}$ 是偏移预测量，$(\hat{\omega}_i, \hat{h}_i) = \hat{S}_{\hat{x}_i, \hat{y}_i}$ 是尺寸与测量，其中峰值点提取可以使用一个 3×3 最大池化，不再需要NMS后处理。

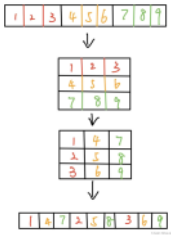
31. CornerNet角点匹配
    - 输出：Heatmap、Offset、Embedding Vector
    - 关键点预测损失
      - 预测关键点热力图 与 经过高斯映射的GT热力图匹配
    - 偏移预测损失
      - 预测关键点偏移值 与 真实偏移值
    - 关联损失
      - 属于同一目标的两个关键点的Embedding（嵌入向量）距离最小
      - 基于不同角点的Embedding vector距离找到每个目标的一对角点，如果一个左上角角点和一个右下角角点属于同一个目标，那么二者的Embedding vector之间的距离应该很小
32. 各种分割算法
    - 语义分割
      - 对图像中的每个像素点进行分类
      - 同一类别的不同实例不需要区分检测
        - 
    - 实例分割
      - 常用分割算法：Mask R-CNN
      - 目标检测与语义分割的结合
      - 相比于目标检测的边界框标定目标，实例分割可以具体到目标的边缘
      - 相比于语义分割，实例分割需要对同一类别目标的不同实例进行区分标记
        - 
    - 全景分割
      - 实例分割与语义分割的结合
      - 实例分割只需要对图像中的目标进行检测，并对检测到的目标进行分割
      - 全景分割需要对图像中的目标以及背景进行检测和分割

- ◇ 

33. 维度灾难
    - 随着数据维度的增加，数据量倍增
    - 当维度增加时，使得可用的数据变得稀疏
    - 解决方法：特征分解，特征降维
34. 语义分割损失函数
    - 逐像素交叉熵损失
        - 平等对待每个像素
        - 当前景像素的数量远远小于背景像素的数量时，即出现样本不平衡的的时候，损失函数中背景的成分占据主导，使模型偏向背景
    - 加权交叉熵
        - 更关注少样本类别
        - 解决样本不平衡问题
    - Foacl Loss
        - 解决难易样本不平衡问题，更关注难分样本的损失
        - 将高置信度样本的损失降低，低置信度样本的损失提高
    - Dice loss
        - 评估集合相似度的度量函数
        - 比较适用于样本极度不均的情况
        - $$s = \frac{2|X \bigcap Y|}{|X|+|Y|}$$
        - $$loss = 1 - \frac{2|X \bigcap Y|}{|X|+|Y|}$$
35. 多模态数据融合
    - 基于GAN
        - 定义生成器：由LRMS生成HRMS
        - 两个判别器：PAN判别器+MS判别器
        - 分别使得生成的HRMS包含更多的光谱信息和空间信息
36. 语义分割边界模糊
    - 语义分割经典网络：FCN，U-Net，Deeplab，PspNet
    - 边界模糊原因
        - 边界像素点占比较少，但损失函数同等看待所有像素点，导致优化偏向于目标内部区域，边界效果差
        - 边界像素点类别不同，但提取的邻域信息相近
    - 方法
        - 基于先验知识的边界细化
            - 从基线得到一张粗糙mask
            - 根据预测边界以及原始图像生成一系列Image Patch与相应mask Patch
            - 将concat后的bounding mask送入二分类网络进行前景、背景分类
            - 将得到的结果重新组装，求和求平均，根据阈值（0.5）处理
        - 语义分割与边缘检测结合进行联合优化
            - 预测粗粒度语义分割掩膜
            - 根据语义分割掩膜求解梯度图
            - 根据梯度图进行边缘检测，GT监督，反向优化分割掩膜边界
        - 主体和边界解耦，分别进行处理，再将结果整合
            - 动机：利用边界信息进行粗分割结果优化的方式中，边界信息一般从中间层特征获得，容易导致错误的边缘估计向后传播
            - 根据高低频将图像特征解耦为主体和边缘特征
            - 生成的主体分割结果和边缘分割结果分别由对应的损失函数监督
            - 合并主体特征和边缘特征，重建原始特征
37. 点云配准及其应用
    - 步骤
        - 输入：点云坐标（x，y，z）
        - 特征提取：ICP、PCRNet
        - 对应关系：求解源点云和模板点云之间的对应关系

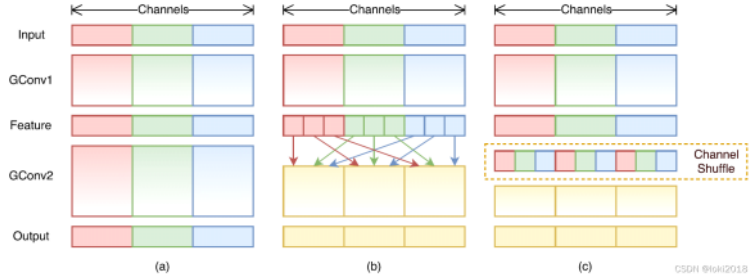
        - 预测：根据对应关系预测源点云输出
        - 输出：旋转和平移矩阵
        - 误差：根据刚性变换矩阵求解与另一个点云的误差
        - 根据旋转和平移矩阵调整源点云位姿，实现与模板点云配准
    - 应用
        - 三维重建，自动驾驶的高精地图构建

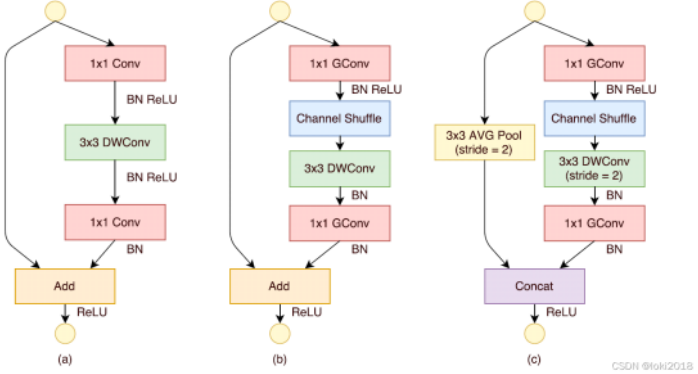
38. ShuffleNet
    - 组卷积弊端（通道稀疏连接方式）
        - 不同组之间的通道难以进行信息交流，降低网络的特征提取能力
    - 动机
        - 利用Channel Shuffle来缓解组卷积的弊端
    - 核心思想
        - 将每次分组卷积后的结果进行组内分组，再互相交换各自组内的子组（均匀的Channel Shuffle)
        - ShuffleNet组件：1*1组卷积+Channel Shuffle+3*3深度可分离卷积+1*1组卷积
            - Channel Shuffle：解决了多个组卷积结果叠加出现的边界效应（通道间信息难交互）
            - 组卷积+深度可分离卷积：减少参数量和计算量
    - 操作：对ResNet组件BottleNeck中的层做了改进
        - Channel Shuffle的实现
            - （B，C，H，W）-> （B，G，N，H，W）-> （B，N，G，H，W）-> （B，C，H，W）



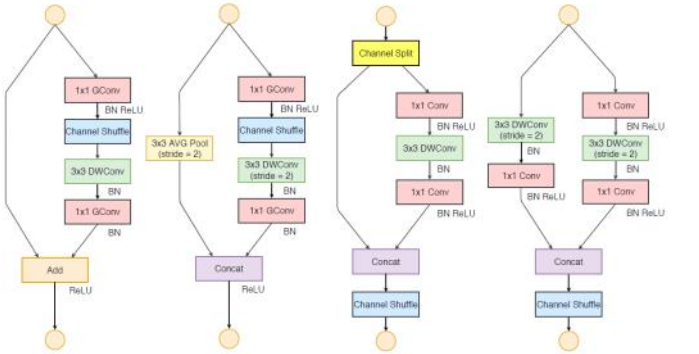
```
def channel_shuffle(x, groups):
    # x[batch_size, channels, H, W]
    batch, channels, height, width = x.size()
    channels_per_group = channels // groups  # 每组通道数
    x = x.view(batch, groups, channels_per_group, height, width)
    x = torch.transpose(x, 1, 2).contiguous()
    x = x.view(batch, channels, height, width)
    return x
```

    - 结构图





    - ShuffleNet V2（替换了1*1分组卷积，并且尽量避免了add操作）

- 

39. Unet

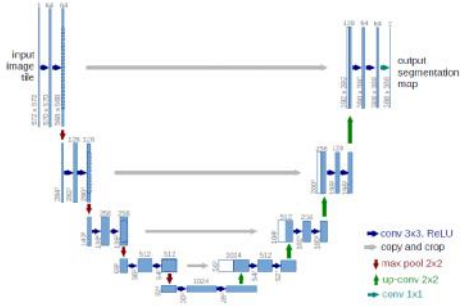
- 

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

40. 深度学习和强化学习的区别
    - 深度学习
        - 一般基于神经网络进行
        - 一般采用带标签的数据集进行监督学习
            - 人类为深度神经网络准备数据集（题目和答案），然后训练神经网络做题，希望它在刷题后碰到相似的题目也能答对
        - 应用：图像识别，自然语言处理
    - 强化学习
        - 强化学习本质上是决策问题
        - 通过环境的反馈（奖励或惩罚）找到某种行为使得利益最大化
            - 要与环境交互，再通过环境的反馈来学习
        - 最终从固定的数据集里推测出最优的分布是哪一个
        - 应用：机器人控制
    - 深度学习可以用于实现强化学习

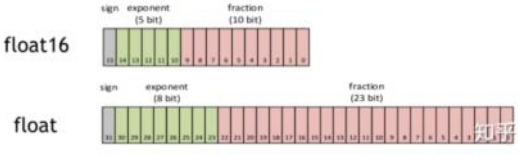
数据预处理，模型评估

2022年5月6日 22:14

1. 项目中遇到的困难
    - 不收敛
        - 数据预处理不到位，特征分布不一致（灰度图像，RGB图像都有）（RGB转灰度之后，再与灰度按比例混淆，减少差异）
        - Batch-size较小：单卡训练，后来换成多卡
        - 学习率：学习率太大，出现NAN
    - 过拟合
        - 刚开始接触阶段，魔改，将各种好的思想或者组件都揉进网络，模型太复杂
        - 数据集比较小：进行数据增强
    - 粗心
        - 自己写的一个模块，在添加到主体网络框架中时，没有用ModelList()实例化，导致模型不收敛
2. 项目的落地
    - 目标检测：军方项目，构建一个全方位的定位系统，对飞机分布情况，以及机场，舰船分布情况进行监测
    - 融合分类：主要是环境监测的，指定区域内，近几年的植被覆盖率，建筑覆盖率这些进行评估分析
    - 特征分解：老师的国家自然科学基金项目
3. 导致模型不收敛的原因有哪些?
    - 数据问题：数据标注不准确，数据没有进行归一化
    - Batch-size：太小，模型难以学习到正确的数据分布特性
    - 学习率问题：太大易震荡，太小难收敛
    - 模型问题：问题的复杂程度与模型的复杂程度不匹配（模型太简单，数据太复杂）
4. Batch-size为什么一般设置为2的整数次幂
    - 为了符合 CPU、GPU 的内存要求，利于并行化处理
    - 不使用偶数时，损失函数是不稳定的
5. 图像处理中的锐化操作和平滑操作?
    - 锐化：增强图像中的高频分量减少模糊（在增强图像边缘的同时增加了图像的噪声）
    - 平滑：过滤掉图像的高频分量，减少图像噪声，使图像变得模糊
6. 为什么要进行模型微调？（深度学习-P15)
    - 神经网络通常需要大量的样本进行训练，但我们自己的数据集质量参差不齐，模型得不到较好的训练
    - 因此我们通过对在大型数据集上训练好的模型进行微调，使得它适用于我们自己的数据集。
7. Dropout（深度学习-P20)
    - 在每一次训练过程中，网络会以一定的概率P丢弃一部分节点，每次迭代过程中丢弃的节点是不同的，因此每一次迭代都相当于在训练一个独一无二的模型，最终集成在同一个模型上，集成时不是直接平均的方式，而是将权值乘以概率P
    - 减少多层神经元之间的联合共适应关系（链式求导）
    - 模型预测时不需要Dropout,需要关掉
8. Dropout与Dropconnect的区别
    - Dropout：对神经元的输出值，以一定的概率清零（a）
    - Dropconnect：对与神经元连接的权重，以一定的概率清零（w）
9. 怎么提升模型的泛化能力?
    - 数据层面：收集更多的数据，数据增强，数据缩放，数据变换，特征选择等
    - 算法层面：Dropout，正则化，较大的batch-size， 调试学习率，损失函数以及优化方法等
    - 集成学习：随机森林，Bagging
10. 传统图像处理算法
    - 形态学方法
        - 腐蚀：让黑色更多，白色更少，去除目标周围的小颗粒噪声，有助于消除孤立点和噪声
        - 膨胀：让白色更多，黑色更少，合并目标周围的背景点
        - 开运算：先腐蚀再膨胀，消除物体边界粘连的现象，将两个密集目标分开
        - 闭运算：先膨胀再腐蚀，将两个密集排列的目标合并
        - 顶帽：将原始图像与开运算图像作差
        - 黑帽：将原图像与闭运算后的图像作差
        - RGB转灰度图：相当于三维空间的像素点像R、G、B三个坐标轴作映射，映射为一维，常用方法是RGB三个通道颜色的加权和

- 边缘检测算子
  - Sobel算子：【1、0、-1】
  - 拉普拉斯算子：对孤立像素点的响应强一些，适用于无噪声图像
  - Canny算子：高斯滤波器去噪+一阶偏导计算梯度幅值和方向+对梯度幅值进行非极大值抑制
- 图像去噪（模糊）
  - 均值滤波
    - 当前像素点的值是周围像素点值的加权平均
  - 高斯模糊
    - 中间像素(x,y)的权值是最大的，周边像素的加权系数随着它们与中间像素的距离增大而减小
    - $G(x,y) = Ae^{\frac{-(x-u_x)^2}{2\sigma_x^2}+\frac{-(y-u_y)^2}{2\sigma_y^2}}$
  - 中值滤波
    - 将图像中的每个像素用领域像素的中值来代替
  - 导向滤波
    - 导向滤波器是导向图和滤波输出图之间的局部线性模型
    - 一个复杂函数可以由很多局部的线性函数来表示
  - 双边滤波
    - 双边滤波的核函数由定义域核和值域核的综合结果得到
    - 双边滤波的输出像素值同样是依赖于领域像素的加权组合
      - $g(i,j) = \frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$
    - 权重系数w(i,j,k,l)的选取：定义域核和值域核的乘积
      - $w(i,j,k,l) = exp(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{||f(i,j)-f(k,l)||^2}{2\sigma_r^2})$
- 图像对比度增强（去模糊）
  - 直方图均衡化：对图像的灰度值做非线性映射，使图像的灰度值大致符合均匀分布，从而增强图像对比度
  - 拉普拉斯算子：锐化增强
  - 伽马变换：主要用于图像校正，将灰度过高或灰度过低的图像进行校正，提高对比度

11. 图像的高频分量，低频分量
    - 图像频率：用于衡量图像灰度值变化的剧烈程度，是灰度在平面空间上的梯度
      - 低频：灰度缓慢变化的部分，表示连续渐变的一块区域，目标内部，大致概貌，纹理信息
      - 高频：灰度变化较快的部分，表示目标的边缘，细节部位，噪声
12. 模型压缩方法
    - 网络剪枝
    - 模型量化：利用更低的精度来代替原本的浮点精度，神经网络模型的参数通常是用32bit长度的浮点数表示，可以通过量化，比如用0~255表示原来32个bit所表示的精度，通过牺牲精度来降低每一个权值所需要占用的空间
    - 低秩分解：将原始矩阵映射到更低的线性子空间
    - 知识蒸馏：将已经训练好的教师模型（复杂）包含的知识，蒸馏到学生模型（简单）中
13. FP16和FP32
    - FP16
      - 采用2字节(16位)进行编码存储的一种数据类型
    - FP32
      - 采用4字节(32位)进行编码存储的一种数据类型



- 如上图，fp16第一位表示+-符号，接着5位表示指数，最后10位表示分数；
- 公式：

$$(-1)^{signbit} \times 2^{(exponent-15)} \times (1+\frac{fraction}{1024})$$

- 其中，sign位表示正负，exponent位表示指数（ ），fraction位表示的是分数（ ）。其中当指数为零的时候，下图加号左边为0，其他情况为1。

- FP16和FP32相比对训练的优化：
  - 内存占用减少：FP16内存占用比原来更小,可以设置更大的Batch_Size
  - 加速计算：加速计算只在最近的一些新GPU中，FP16训练速度可以是FP32的2-8倍
- 问题
  - FP16的值区间比FP32的值区间小很多
  - 在计算过程中很容易出现上溢出（Overflow，>65504）和下溢出（Underflow，<6x10^-8）的错误
  - 溢出之后就会出现“Nan”的问题

14. 什么样的数据集不适合深度学习
    - 数据集太小：深度学习是数据驱动的学习方式
    - 数据集没有局部相关性
      - 目前深度学习表现较好的领域主要是图像、语音、自然语言处理等，这些领域的共性就是局部相关性
      - 局部相关性：图像的像素组成物体，文本数据中单词组成句子，当元素被打乱时，表示的含义也会发生改变
      - 对于没有局部相关性的数据集，深度学习不太适应
        - 预测一个人的健康状况：相关参数包括年龄，职业，收入，家庭状况等，将这些元素打乱，并不会影响相关结果
15. 对模型微调的理解
    - 使用预训练网络的好处：使用训练好的SOTA模型权重去做特征提取，可以节省我们训练模型和调参的时间
    - 需要微调最后几层权重的原因：
      - 模型的浅层卷积层提取的是通用特征，深层是抽象的更加专业化的语义特征，深层特征更能反应具体任务中的数据特性
      - 训练的参数越多，越容易过拟合，预训练的SOTA模型通常拥有千万级的参数，在小的数据集上训练有过拟合的风险
16. HOG算法
    - HOG：方向梯度直方图（Histogram of Oriented Gradient）
      - 在一副图像中，梯度信息能够较好地描述局部目标区域的特征，HOG正是利用这种思想，对梯度信息做出统计，并生成最后的特征描述
      - 在深度学习之前，HOG特征与SVM结合，广泛应用于图像识别中
17. 常见的数据增强方法
    - 翻转：Fliplr，水平镜面翻转，上下翻转
    - 旋转：rotate，顺时针，逆时针旋转
    - 缩放：zoom，图像被放大或者缩小
    - 裁剪：crop，随机裁剪
    - 平移：translation，沿x轴或者y轴移动
    - 仿射变换：Affine，包括平移、旋转、缩放
    - 添加噪音：随机添加噪音
    - 亮度，对比度增强：属于图像色彩增强操作
    - 锐化：Sharpen
18. 离线数据增强和在线数据增强的区别
    - 离线数据增强
      - 直接对硬盘上的数据进行处理，并保存增强后的数据
    - 在线数据增强
      - 获得一个Batch-size的数据后，对这个Batch-size中的数据进行增强，包括平移、旋转、翻转等操作
19. 移动端深度学习框架
    - TensorFlow、小米MACE、腾讯的NCNN
20. 验证集和测试集的作用
    - 验证集：训练过程中检测模型的训练情况，从而确定合适的超参
    - 测试集：模型训练结束后，检测模型的泛化能力
21. NMS和IOU原理
    - NMS
      - 作用：在目标检测中，在解析模型的输出预测框时，存在多个重复预测框定位到同一个目标的现象，NMS可以用于过滤掉重复预测框，保留真正的目标框。NMS在计算过程中使用了IOU。
      - 过程：
        - 先对预测框按照置信度进行排序
        - 选择置信度最大的预测框与后面的预测框计算交并比IOU
        - 当IOU大于某个阈值时，则认为两个预测框定位到了同一个目标，删除置信度较低的预测框
        - 重复上述步骤，直到得到所有预测框
    - IOU

- 用于衡量两个边界框之间的相关程度，相关程度越高，IOU越大
- 计算：交集/并集

22. 模型评估
    - 分割数据集
        - 训练集：用于模型训练
        - 验证集：用于模型选择，选择合适的超参
        - 测试集：用于模型评价
    - 经验误差和泛化误差
        - 经验误差：模型在训练数据集上的误差
        - 泛化误差：模型在测试集和验证集上的误差
    - 回归问题评估指标
        - 均方误差MSE：预测结果与样本标签之间的差值的平方的均值
    - 分类问题评估指标
        - 正确率：正确分类样本数/总样本数
        - 错误率：错误分类样本数/总样本数
        - 精度：TP/TP+FP
        - 召回率：TP/TP+FN
        - P-R曲线：Recall为横坐标，Precision为纵坐标，曲线下的面积为目标检测中的AP
        - ROC曲线：假正率为横坐标，真正率为纵坐标，曲线下的面积为AUC
        - F1值：2*Precision*Recall/Precision+Recall
23. 排序算法时间复杂度
    - 排序算法稳定性：多个具有相同关键词的记录在重新排序后，相对位置保持不变

各种常用排序算法							
类别	排序方法	时间复杂度			空间复杂度	稳定性	复杂性
		平均情况	最好情况	最坏情况	辅助存储		
插入排序	直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定	简单
	希尔排序	$O(n^{1.3})$	$O(n)$	$O(n^2)$	$O(1)$	不稳定	复杂
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定	简单
	堆排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(1)$	不稳定	复杂
交换排序	冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定	简单
	快速排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(n^2)$	$O(log_2n)$~$O(n)$	不稳定	复杂
归并排序		$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(n)$	稳定	复杂
基数排序		$O(d(r+n))$	$O(d(r+n))$	$O(d(r+n))$	$O(rd+n)$	稳定	复杂
注：基数排序的复杂度中，r代表关键字的基数，d代表长度，n代表关键字的个数							

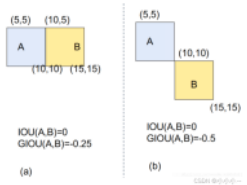
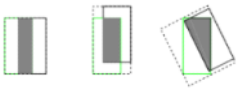
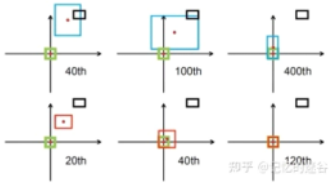
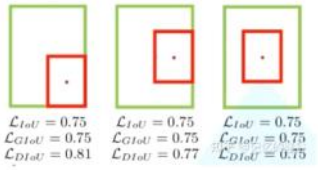
24. 查找算法时间复杂度

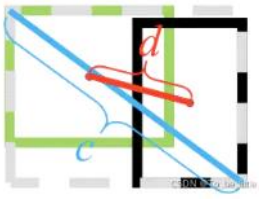
查找	平均时间复杂度	查找条件	算法描述
顺序查找	O(n)	无序或有序队列	按顺序比较每个元素，直到找到关键字为止
二分查找（折半查找）	O(logn)	有序数组	查找过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。 如果在某一步骤数组为空，则代表找不到。
二叉排序树查找	O(logn)	二叉排序树	在二叉查找树b中查找x的过程为： 1. 若b是空树，则搜索失败 2. 若x等于b的根节点的数据域之值，则查找成功； 3. 若x小于b的根节点的数据域之值，则搜索左子树 4. 查找右子树。
哈希表法（散列表）	O(1)	先创建哈希表（散列表）	根据键值方式(Key value)进行查找，通过散列函数，定位数据元素。
分块查找	O(logn)	无序或有序队列	将n个数据元素"按块有序"划分为m块（m ≤ n）。 每一块中的结点不必有序，但块与块之间必须"按块有序"；即第1块中任一元素的关键字都必须小于第2块中任一元素的关键字；而第2块中任一元素又都必须小于第3块中的任一元素，……。然后使用二分查找及顺序查找。

https://blog.csdn.net/zs742946530

25. 如何提高模型训练速度
    - 合理的超参选择
        - Batchsize, Epoch, 学习率

- 减少模型参数量
- 多卡训练，数据并行

26. IOU变体
    - 利用Smooth L1 Loss作为边界框回归损失：不同情况计算出来的损失值相同，不能准确反应GT与BBox的重叠程度，IOU Loss应运而生
    - IOU缺陷
        - 当GT与BBox不相交时，IOU为0，无法准确判断GT与BBox的距离，IOU loss（1-IOU）的梯度为0，无法更新网络性能
            - 
        - IOU无法准确反应GT与BBox重合度的大小
            - 
    - GIOU
        - $$GIoU = IoU - \frac{A^c - u}{A^c}$$
          $$-1 \le GIoU \le 1$$
        - 
        - Ac表示GT与BBox的最小外接矩形的面积，u表示GT与BBox的并集，蓝色部分就是（Ac-u）
            - 当IOU为0时，意味着GT与BBox非常远，u=0，GIOU趋近于 -1
            - 当IOU为1时，两框重合，（Ac-u）=0，GIOU趋近于 1
            - GIOU的取值为【-1, 1】
        - GIOU Loss = 1 - GIOU（值域【0,2】）
        - 优点
            - 有效缓解了IOU loss的梯度消失问题
        - 缺点
            - 收敛速度慢，且不易收敛（预测框需要先变大与GT接近，再去契合GT的形状）
            - 当GT与BBox水平对齐或者垂直对齐时，退化为IOU
            - 
            - 
    - DIOU
        - $$DIoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2} = IoU - \frac{d^2}{c^2}$$
          $$-1 \le DIoU \le 1$$
          DIOU计算公式

- 

  - d表示GT中心点与BBox中心点之间的欧式距离，c表示GT与BBox的最小外接矩阵的对角线距离
    - 当GT与BBox不相交时：IOU=0，d与c无限趋近，DIOU趋近于 -1
    - 当GT与BBox重叠时：IOU=1，d=0，DIOU趋近于1
  - $$L_{DIoU} = 1 - DIoU$$
  - $$0 \le L_{DIoU} \le 2$$
  - 优点
    - 最小化GT与BBox中心点之间的距离，收敛速度更快（避免了GIOU中，当GT与BBox距离较远时，产生较大的外接矩形，损失较大，难收敛）
    - 当GT与BBox水平对齐或者垂直对齐时，c不变，但d依旧可以度量GT与BBox的距离
- CIOU
  - 优秀的边界框回归损失应该满足三个要素：重叠面积，中心点距离，长宽比
    - GIOU：初步缓解了IOU为0的情况
    - DIOU：同时考虑了重叠面积和中心坐标
    - CIOU：在DIOU中当GT与BBox的中心点重合时，d和c值都不在变化，因此CIOU还考虑了长宽比，使得形状收敛速度更快
  - $$CIoU = IoU - (\frac{\rho^2(b, b^{gt})}{c^2} + \alpha\upsilon)$$
  - $$\upsilon = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2$$

    $$\alpha = \frac{\upsilon}{(1 - IoU) + \upsilon}$$
  - alpha是权重函数，v用来度量长宽比的一致性
  - CIOU loss = 1 - CIoU

27. 样本不均衡问题
    - 三部分：正负样本不均衡、难易样本不均衡、类间样本不均衡
    - 正负样本不均衡
      - 问题：大量负样本参与训练，导致损失优化方向发生倾斜
        - 例如：SSD中总计8732个Anchor，十个左右目标，正Anchor数为60个左右，其他全为负样本
    - 难易样本不均衡
      - 难分样本：置信度较低或者易错分的样本，一般处于边界区域（前景与背景的过渡区域）
      - 易分样本：置信度较高的样本，通常处于目标内部，平坦区域，背景区域
      - 单个难样本损失较大，但样本数量较少
    - 类别间样本不均衡
      - A类别样本数量较多，B类别样本数量较少，网络优化可能由A类损失主导，B类精度大大降低
    - 解决方法
      - 数据层面：数据增强，数据过采样，数据欠采样
        - 过采样：
          - 随机过采样：随机选择少数类别的样本进行复制，增加少数类样本的数量（易过拟合）
          - Smote方法：根据少数类样本之间的相似性构建新的人工数据
            - 计算每个少数类样本与其他少数类样本的欧氏距离，获得K近邻
            - 从K近邻中选择若干个样本，与当前的少数类样本结合，构造新的数据
        - 欠采样：
          - 随机欠采样：随机剔除掉一部分多数类样本
          - Tomek Links方法：剔除掉与少数类样本距离最近的多数类样本集中的样本
      - 算法层面：
        - Faster RCNN 或者 SSD根据IOU阈值使正负样本比例处于合理范围内

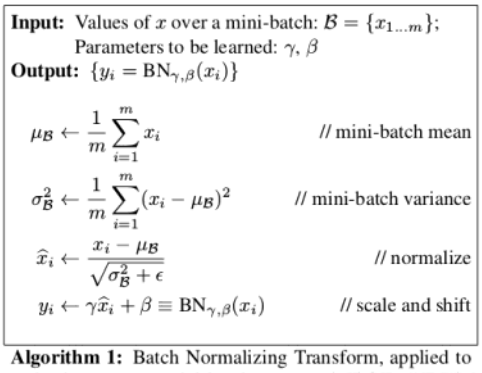
- Faster RCNN在RPN阶段通过置信度得分，保留2000个ROI，将大量负样本和简单样本过滤掉
- OHEM：在梯度反向传播过程中，将简单样本的梯度设置为0，只回传难样本的梯度
- S-OHEM：采用分层抽样的方法选择难样本（e.g., 高分类误差，低回归误差），进行梯度回传

- 损失函数层面：
  - 平衡交叉熵损失：给损失函数添加权重因子，提高样本少的类别在损失函数中的权重（权重因子是超参）（更加注重正负样本不均衡问题）
  - Focal Loss：降低简单负样本在损失函数中的权重，增加难样本的贡献程度，权重动态调整
- 评价指标：选用ROC，F值等

卷积，归一化，激活函数

2022年5月6日 22:03

1. Sigmoid函数特性？(深度学习-P6)
    - 定义域为(-inf, inf) 值域为(0，1)
    - 函数在定义域内为连续且光滑函数
    - 处处可导， 但饱和区域梯度较小
    - 收敛速度较慢，易梯度消失
2. BN层的作用（深度学习-P13)
    - ICS现象
        - ICS：训练集的数据分布和预测集的数据分布不一致，导致在训练集上训练的分类器，在预测集上不会取得比较好的效果
        - 随着网络的不断加深，每一层输入的分布都会随着参数的更新发生变化，高层输入的分布变化更大，使得高层需要不断的适应底层参数的更新。
    - BN层实现的效果：
        - 使得每一层的输入尽量属于同一分布，减少变化带来的不确定性（保持独立同分布，如果每层的数据分布一直改变的话，不容易训练）
        - 每一层都自适应的去拟合自己的输入分布，相对独立
    - 方法
        - 数据标准化为均值为0方差为1，定义两个可学习参数用于控制数据的归一化程度



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

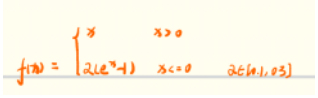
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

知乎 @王二的石锅拌饭

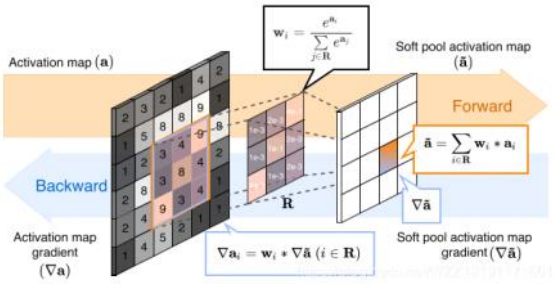
    - BN作用：
        - 加速网络收敛
        - 防止梯度消失和梯度爆炸
        - 防止过拟合，可以少用或者不用Dropout或者L2正则化
        - 降低对学习率的敏感程度
    - BN层中的学习参数贝塔和伽马的作用?
        - BN层对输入数据的分布进行了约束，提高模型的泛化能力
        - 但BN层会将原始数据的分布强制转换为均值为0方差为1的分布，这也降低了模型对复杂数据分布的拟合能力
        - 参数贝塔和伽马实现缩放和偏移操作，使数据尽可能恢复为原始最优的数据分布，提高模型的非线性表达能力
    - BN层如何防止过拟合的?
        - BN层的使用是建立在一个Batch-size上的，一个Batch-size内的样本被关联在一起，同一个样本的输出不再仅仅取决于样本自身
        - 而且网络的每一次迭代都是随机选取Batch-size，这就使得网络不会只朝着一个方向学习
        - 因此在一定程度上避免了过拟合
    - BN放在激活函数之前还是之后?
        - 通常情况下，卷积+BN+激活
            - BN将卷积操作的输出进行标准化，使得激活值跳出激活函数的饱和区，梯度更大一下，有利于模型更新
        - 部分实验表明，卷积+激活+BN效果更好
            - 对于Sigmoid来说：如果在激活之前，使用BN标准化，输入数据的分布大部分会落在非饱和区，Sigmoid的非饱和区近似于线性区域，降低模型对非线性的拟合能力
            - 对于ReLu来说：其值域没有上限，会把数据过于放大化，因此需要BN层来限制其输出范围，否则深层的网络，会使得数值越来越大

- BN层在训练和测试中的区别
  - 在训练阶段，每次的输入是一个Batch-size，BN的均值和方差计算都是基于Batch-size的
  - 在测试阶段，每次只输入一个样本，此时没有Batch-size的概念，因此均值和方差的计算是基于全量训练数据的
    - 训练时每个Batch-size都会得到一组（均值、方差），训练时对这些数据求数学期望
- 训练过程中，BN为什么不用整个训练数据集的均值和方差
  - 整个训练数据集的均值和方差是固定值，会引起模型的过拟合
  - 而Batch-size中的数据在每次迭代过程中是打乱的，每一次的均值和方差都有一定的区别，可以提高模型鲁棒性，避免过拟合

3. BatchNorm和LayerNorm的区别
   - 概念：
     - BN：针对一个Batch-size，对一个Batch-size所包含的样本内的每个特征分别做归一化（输出：1*C*1*1）
     - LN：针对每个样本，对每个样本的所有特征做归一化（输出：B*1*1*1）
     - 举例：对于一个二维矩阵，行表示样本，列表示样本的特征，BN相当于对每一列做归一化，LN相当于每一行做归一化
   - 相同点：都会为了让当前层的参数更稳定，避免梯度消失或者梯度爆炸
   - 不同点：
     - BN层考虑了不同样本中同一特征的大小关系，不考虑不同特征之间的关系
     - LN不考虑不同样本之间的关系，保留的是同一样本中不同特征之间的关系
     - NLP领域常用LN，CV领域常用BN
4. BN，GN，LN，IN的区别
   - BN是对B、H、W维度进行标准化，适合CV任务
   - LN是对C、H、W维度进行标准化，适合NLP任务
   - IN是对H、W维度进行标注化，适合CV中生成式任务
   - GN是对H、W、(C/group)进行标准化，适合CV任务
5. BatchNorm和GN，WN， LRN的区别
   - LRN
     - LRN模仿了生物神经元中活跃神经元对周围神经元的抑制现象
     - LRN使值比较大的神经元值更大，并且抑制值比较小的神经元
   - GN
     - GN仅针对与同一个特征，与Batch-size无关
     - 在特征图的通道维进行分组，每组内计算均值和方差
   - WN
     - WN是对网络权重进行归一化，BN是对输入数据进行归一化
     - WN是对网络权重进行归一化，不依赖于Batch-size
6. 梯度消失
   - 神经网络的激活函数一般采用Sigmoid函数，Sigmoid函数的饱和区梯度趋近于0
   - 反向传播过程中采用链式求导，当网络很深的时候，浅层权值的梯度相当于乘了多个小于1的数，因此会逐渐趋向于0，产生梯度消失的现象
7. 组卷积
   - 方法：对输入特征图的通道进行分组，在每个组内分别进行卷积，将每个组的输出通道叠加在一起得到最终的输出通道数
   - 作用：常用于轻量级网络，与标准的卷积相比，参数量减少为原来的1/M（M为分组个数）
8. Relu与Sigmoid相比优势是什么？（深度学习-P7）
   - 计算速度更快，Sigmoid计算梯度时需要进行指数运算，相对来说更加耗时
   - 解决了梯度消失问题，Sigmoid函数在饱和区的梯度接近于0，易导致梯度消失，Relu的正半区的梯度为常数1，不会导致梯度衰减
   - Relu在负半区的值为0，梯度为0，此部分神经元不参与训练，提供神经网络的稀疏表征能力
9. 神经网络的权值共享
   - 卷积神经网络通过卷积核实现了空间位置上实现权值共享
   - 循换神经网络可以看做是在时间位置上实现权值共享
10. 激活函数的作用
    - 引入非线性因素：增加网络的非线性表达能力
    - 提高模型对复杂关系，非线性关系的学习和理解能力
11. 激活函数的比较（深度学习-P6）
    - Sigmoid函数
      - 数学公式
      - 函数图像及梯度图像(0,0.25]

        - 优点：
            - 便于求导的平滑函数
        - 缺点
            - 饱和区梯度接近于0，易梯度消失
            - 求梯度时有指数运算，计算量大
            - Sigmoid函数输出不是Zero-centered（对于某一层的一个神经元来说，与它连接的所有权重在更新时的变化方向是一样的，同时增大或者同时减小，要想实现w1增，w2减，只能走Z字形实现，导致网络收敛缓慢）
    - Tanh函数
        - 数学公式
        - 函数图像及梯度图像
        - 优点：
            - 函数输出是Zero-centered
            - 收敛速度更快
        - 缺点：
            - 饱和区梯度接近于0，依旧容易导致梯度消失
            - 梯度计算存在指数运算，计算量大
    - Relu函数
        - 数学公式
        - 函数图像以及梯度图像
        - 优点：
            - 解决了梯度消失问题（当激活值处于Relu的正半区时，导数为1，在反向传播过程中，梯度不会衰减）
            - 计算速度快，不存在幂指运算
            - 收敛速度快
            - 提供神经网络的稀疏表达能力（当激活值处于负半区，梯度为0，那么这些神经元不再参与训练）
        - 缺点：
            - 死亡神经元：随着网络的训练，部分激活值落入负半区之后，梯度为0，对应的神经元不再参与训练，这个过程是不可逆的，导致对应的参数永远不会被更新，使得数据多样化丢失
    - Elu函数
        - 软饱和：负半区趋向于0，但不等于0（非线性）
        - 存在指数运算
        - 
    - Leaky Relu函数
        - 软饱和：负半区趋向于0，但不等于0（线性）
        - 没有指数运算，计算更快
12. 梯度消失和梯度爆炸
    - 梯度消失：
        - 神经网络中的激活函数通常采用Sigmoid时，Sigmoid函数在饱和区的梯度几乎为0，并且Sigmoid函数的梯度值域是(0,0.25).
        - 当神经网络足够深时，在反向传播过程中，浅层的权重的梯度相当于乘了很多<1的数得到，因此也接近于0，产生梯度消失
        - 导数小于1
    - 梯度爆炸：
        - 将权重初始化为很大的数，那么在反向传播过程中，0.25*大数>1，每一层的权重梯度相当于乘了很多>1的数，因此得到一个非常大的权重更新，产生梯度爆炸
        - 导数大于1 或者 权重非常大
    - 梯度消失和梯度爆炸产生原因
        - 根本原因：反向传播训练法则（前一层的梯度来自后面层梯度的乘积），属于先天不足
        - 参数初始化不当
        - 激活函数选取不当
    - 解决方法：
        - 采用非饱和的激活函数（Relu）
        - 合适的参数初始化方法
        - BN层

        - 梯度截断
        - 合适的优化器
        - CNN的残差结构
13. 1*1卷积核的作用
    - 实现信息的跨通道整合与共享
    - 实现卷积核通道数的降维与升维
    - 实现多个特征图的线性组合，相当于全连接
14. 卷积层和池化层的区别（深度学习-P27）
    - 相同点：卷积层和池化层的结构相似，都是对感受野内的特征进行提取，然后根据步长获得不同维度的输出
    - 不同点：
        - 结构：
            - 卷积层：输出特征的空间维度和通道维度都会发生变化
            - 池化层：输出特征的空间维度发生变化，通道维通常不会发生变化
        - 作用：
            - 卷积层：
                - 提取感受野内的局部关联特征
                - 局部感知，参数共享的特点大大降低了网络参数量，保证了网络的稀疏性
                - 模型的浅层提取局部的，高度通用的图像特征
                - 模型深层提取更加抽象的语义特征
            - 池化层：
                - 提取感受野内的泛化特征
                - 特征不变性：池化使模型更加关注于是否存在某特征，而不是特征的具体位置，能容忍一些特征微小的位移
                - 特征降维：在空间范围内做了维度约简，从而使模型可以抽取更广范围内的特征
                - 减小下一层输入特征图的大小，从而减少计算量和参数个数
                - 在一定程度上防止过拟合，更方便优化
        - 稳定性：
            - 卷积层：输入特征发生变化时，输出特征也会发生变化
            - 池化层：输入特征发生轻微变化时，输出特征未必会发生变化
        - 参数：
            - 卷积层：参数可学习，参数量根据卷积核尺寸以及通道数决定
            - 池化层：不会引入额外参数
15. 卷积操作的平移不变性
    - 不变性：意味着无论目标的外观发生什么变化，依然能够识别出来
    - 平移不变性：无论输入发生怎样的平移，系统都可以产生完全相同的输出
    - 卷积的平移不变性：无论目标处于图像的那个位置，得到的预测结果是一样的
        - 卷积：卷积操作在不同的位置产生的作用是一样的（目标在左上角，特征图中目标特征也在左上角）
            - 目标平移，特征图中目标特征也会平移
    - 无论目标发生怎样的平移，卷积始终能检测出它的特征
16. 池化操作SoftPool
    - 目的：减少池化操作过程中的信息损失
    - 形式：softmax+avg
    - 操作：
        - 计算感受野内每个元素的权重：利用softmax
        - 对感受野内所有值进行加权求和



17. 卷积核和滤波器
    - 在只有一个通道的情况下，卷积核和滤波器的概念是等价的
    - 当有多个通道时，一个滤波器相当于是多个卷积核的堆叠
18. 卷积，反卷积，空洞卷积、池化输出特征图大小
    - 卷积
        - $W = \frac{W - F + 2P}{S} + 1 \qquad H = \frac{H - F + 2P}{S} + 1$
    - 池化
        - $W = \frac{W - F}{S} + 1$
        - $H = \frac{H - F}{S} + 1$
    - 空洞卷积
        - $W = \frac{W - d(k-1) - 1 + 2p}{s} + 1$
        - d为空洞率，k为卷积核大小，p为padding
    - 反卷积
        - $o = s(i-1) + 2p - k + 2$
        - i：输入尺寸
19. 卷积核的填充padding
    - valid：代表只进行有效卷积，对边界数据不处理
    - same：对原图像进行边界补零，使卷积核可以到达图像边缘
20. openCV
    - openCV读取图像返回的矩阵格式shape是(H、W、C)
    - openCV读取彩色图像时，返回的通道顺序是BGR
21. softmax作用
    - 作用：将输出值转换为值为正且和为1的概率分布
    - 直接使用输出层的输出存在的问题
        - 神经网络输出层的输出范围不确定，难以直观上判断这些值的意义
        - 真实标签是离散值，离散值与不确定的输出之间的误差难以衡量
22. 感受野理解
    - 感受野是指后一层神经元在前一层神经元上的感受空间
    - 感受野也可以认为是每层特征图上的一个像素点映射到原始输入图像中时，所表示的区域大小
    - 感受野的计算
        - 第一层卷积层：输出特征图中每个像素点的感受野大小等于滤波器的大小
        - 深层卷积层的感受野的大小和它之前所有卷积层的滤波器的大小和步长有关
        - 计算公式
            - $$l_k = l_{k-1} + \left[(f_k - 1) * \prod_{i=1}^{k-1} s_i \right]$$
            - 其中 l_{k-1} 为第 $k-1$ 层对应的感受野大小，f_k 为第 k 层的卷积核大小，或者是池化层的池化尺寸大小。
23. CNN的结构特点
    - CNN具有局部连接，权值共享，下采样，多层次结构的特点
        - 局部连接：使网络提取数据的局部特征
        - 权值共享：大大降低了网络的训练难度
        - 池化操作和多层次结构一起实现了特征降维，将浅层局部特征组合成高层次特征，实现对整个图像的表示
24. 卷积神经网络特征的层次性
    - 卷积操作可以获得不同类型的特征，而池化操作可以对特征进行融合和抽象
    - 随着卷积和池化的多层堆叠，各层得到的特征逐渐从泛化特征（边缘、纹理信息等）过度到高层语义表示（躯干，头部等）
25. 模型过拟合的原因

- 数据角度
  - 数据分布不均，模型只关注到部分特征
  - 用重采样，交叉验证，规则化
- 模型角度
  - 模型的复杂度与数据的复杂度不匹配
  - 用dropout，剪枝，正则化

26. 过拟合和欠拟合问题
    - 过拟合
      - 概念：模型过于复杂时，他很好的记住了训练数据中随机噪声的部分，而忽略了数据中的通用趋势
      - 表现：训练数据集上损失函数较小，精度较高；测试数据集上损失函数较大，精度降低
      - 解决方法：
        - 数据增强：增加数据多样性
        - 正则化：L1，L2正则化
        - DropOut
        - 模型融合：Bagging
        - BN层
        - 提前终止训练
    - 欠拟合
      - 表现：模型不收敛
      - 如何判断欠拟合：
        - 模型训练足够长时间后，loss依旧很大，且精度很低
      - 解决方法：
        - 寻找最优的权重初始化方式
        - 选用合适的激活函数
        - 选择合适的优化器和学习率
27. 导致模型不收敛的原因
    - 数据预处理不到位，没有对数据做归一化
    - Batch size太小
    - 学习率设置不当
    - 参数初始化方式不恰当
    - 网络中存在死亡神经元
    - 模型太简单，数据太复杂
28. 卷积核权值共享的理解
    - 用一个卷积核在不改变其内部权重系数的情况下处理整张图片
29. Faster R-CNN、Mask R-CNN为什么要使用ROI Pooling和ROI Align
    - 因为这两个网络的最后包含全连接层，全连接层的参数是固定的，需要将不同大小的感兴趣区域固定成同一尺寸，才能共享两个全连接层的参数
30. 上采样方法总结
    - 基于线性插值的上采样：最近邻算法、双线性插值、双三次插值
    - 基于深度学习的上采样：反卷积
    - Unpooling方法：简单的补零或者扩充操作
31. BN层和Dropout一起用会导致什么结果
    - BN和Dropout分别单独使用时都可以避免过拟合，但一起使用可能效果更差
    - Dropout在每一次迭代训练过程中都会以一定概率屏蔽掉一部分神经元，也就是使一部分神经元的输出值置0，当Dropout放在BN层之前使用时，BN层的计算是基于一部分神经元失活之后的数据计算的。但在预测阶段，不使用Dropout，不存在神经元失活，此时，与训练阶段相比，数据分布也会发生一定的变化，但BN层的参数是基于之前的结果计算的，从而导致一些负面影响
32. 形变卷积
    - 标准卷积的感受野是一个矩形区域
    - 形变卷积的感受野是不规则区域
    - 过程
      - 首先使用标准卷积核计算偏移量（B，W，H，2N）（delta x，delta y）
      - 将偏移量作用于输入特征图，给每个像素点加上 X 和 Y 方向的偏移量
      - 再利用标准卷积核与加上偏移量之后的像素点进行加权求和

损失函数、优化器

2022年4月23日 22:05

1. BGD, MBGD, SGD, SGD-Momentum, SGD-NAG, AdaGrad, AdaDelta, Adam的原理？(深度学习-P75)
    - 传统梯度下降方法：BGD, MBGD, SGD (采用的训练样本数不同，参数优化方式相同)
    - SGD-M: 引入一阶动量优化梯度方向。当前点的实际优化方向由当前点的梯度方向以及前期累积的梯度方向(动量)共同决定
    - SGD-NAG: 当前点的实际优化方向由超前梯度方向(沿累积梯度方向向前先走一步)决定
    - AdaGrad:引入二阶动量自适应的调节学习率大小。参数的学习率由其对应的累计梯度平方和的大小决定(反比)
    - AdaDelta:利用一段时间的累计梯度平方和来调节学习率，而不是所有历史进程中的梯度平方和
    - Adam:以上方法的集大成者。同时使用一阶以及二阶动量来优化梯度方向和学习率。
2. L1 Loss不可导怎么办？(融合分类-P5)
    - 当损失函数不可导时，可以采用坐标轴下降法，沿着坐标轴的方向进行参数更新。
    - 坐标轴下降法
        - 优化目标
            - 在权值系数W的n个坐标轴上对损失函数进行下降迭代，当所有坐标轴上的Wi都达到收敛时，损失函数的最小值
        - 方法
            - 假设有m个特征，坐标轴下降时，先固定m-1个特征值，求解另一个特征的局部最优解，从而避免损失函数不可导的问题
    - 坐标轴下降法与梯度下降法的区别
        - 坐标轴下降法
            - 沿一个坐标方向进行搜索，固定其他的坐标方向，不需要求目标函数的导数，只按照某一坐标方向搜索损失函数最小值
        - 梯度下降法
            - 总是沿着负梯度方向求函数的局部最小值，该梯度方向可能不与任何坐标轴平行
3. 为什么交叉熵可以用做代价函数
    - 为了让模型学习到的分布更加接近数据的真实分布，我们需要最小化模型数据分布与训练数据分布之间的KL散度
    - 因为训练数据的分布是固定的，因此最小化KL散度相当于最小化交叉熵，交叉熵计算更简单一些
    - 因此，交叉熵常作为分类问题的损失函数
4. 最优化方法
    - 一阶最优化方法（梯度下降）
        - BGD
            - 优点
                - 每一次迭代中采用所有样本进行计算，全数据集确定的方向能更好的代表样本整体
                - 模型参数可以更准确的向极值所在方向更新
                - 目标函数为凸函数时，BGD一定能够得到全局最优解
            - 缺点
                - 当数据集很大时，每一次迭代需要对所有样本进行计算，导致训练速度缓慢
        - SGD
            - 优点
                - 每次迭代只使用一个样本对参数进行更新，使得训练速度加快
            - 缺点
                - 可能会收敛到局部最优，因为单个样本并不能代表全体样本的趋势
        - MBGD
            - 优点
                - 每次迭代在一个batchsize上优化参数，是对SGD和BGD的折中

                - 在一个batchsize上训练可以大大减小收敛所需的迭代次数，收敛结果更加接近梯度下降的效果
            - 缺点
                - 依赖于batchsize的大小
            - batchsize的影响
                - 合理范围内增大batchsize
                    - 内存利用率提高
                    - 跑完一次epoch，所需的迭代次数少了，处理速度更快
                    - 确定的梯度下降方向更准确，引起震荡的可能性更小
                - 盲目增大batchsize
                    - 导致内存溢出
                    - 跑完一个epoch的迭代次数减少了，但想达到相同的精度，其花费的时间会增加
                    - 当batchsize大到一定程度，其所确定的梯度下降方向，基本不再发生变化
    - 二阶最优化算法
        - 牛顿法
            - 思想：牛顿法是求方程根的，但求方程根和求极值本质上是一个问题，求极值也就是求导数为0的点，即求导数方程的根
            - 牛顿法推理：利用泰勒展开式
            - 牛顿法存在的问题
                - 需要求解Hessian矩阵的逆矩阵，计算量大，且可能不可逆
        - 拟牛顿法
            - 构造一个正定矩阵近似牛顿法中的Hessian矩阵的逆
    - 牛顿法和梯度下降法的区别
        - 牛顿法需要二阶导数，梯度下降法只利用了一阶导数，牛顿法的收敛速度更快，梯度下降法只考虑了方向，牛顿法还兼顾了学习的步长
5. 神经网络的损失函数为什么一般是非凸的?
    - 凸函数的要求非常严格（Hessian矩阵正定、二阶导大于等于0）
    - 在神经网络中，每层都加了非线性，最终的输入输出之间的关系会非常复杂，一般难以满足凸函数条件
6. Smooth L1 loss优点
    - 早期使用L1，梯度稳定，快速收敛，后期使用L2，逐渐收敛到最优解
    - L1的导数为正负1，L2的导数是2x，L1相比于L2来说对异常值没那么敏感，但L1在最低点不可导，导致在最优解附近震荡，而L2可以缓慢收敛到最优解
    - Smooth L1的优点
        - 当误差过大时，梯度不至于太大
        - 当误差很小时，梯度足够小
        - 收敛速度更快，对异常值不敏感
7. Triplet Loss
    - 一种三元组损失函数，主要用于训练差异性小的样本（人脸）
    - 步骤
        - 在训练数据集中随机选择一个样本Anchor
        - 再随机选择一个与Anchor属于同一类的样本P
        - 再随机选择一个与Anchor不属于同一类别的样本N
        - 针对三元组中的每个元素，训练一个网络，得到三个元素的特征表达F(a)，F(p)和F(n)
        - 使得F(a)和F(p)之间的距离尽可能小，F(a)和F(n)之间的距离尽可能大
    - 目标函数
        - 距离用欧式距离度量
        - F(a)和F(p)之间的距离与F(a)和F(n)之间的距离相差一个阈值Alpha
        - 设置Alpha的原因：避免模型走捷径，将P和N的Embedding训练成很相近
        - +表示[]内的值大于零的时候，取该值为损失，小于零的时候，损失为零

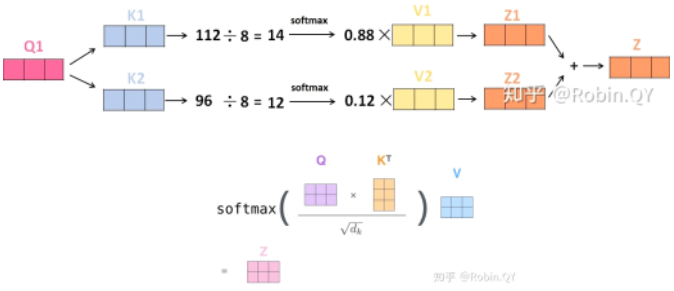
        - $\sum_{i}^{N}\left[\|f(x_i^a)-f(x_i^p)\|_2^2-\|f(x_i^a)-f(x_i^n)\|_2^2+\alpha\right]_+$

8. 训练过程中Loss突然增大的原因
    - 原因
        - 数据标注存在错误（label标准错误）
        - 数据预处理不到位，没有标准化
        - 学习率太大，导致震荡
    - 解决方法
        - 梯度截断（当梯度大于某个值时， 进行截断）
        - 学习率在训练过程中进行衰减

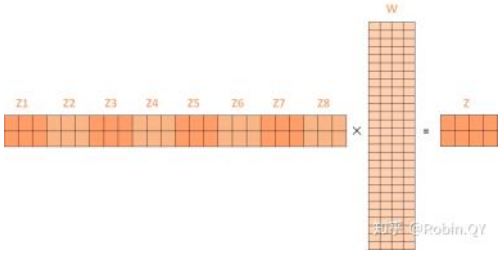
Transformer

2022年5月8日 10:07

1. Transformer
   - 完全基于注意力的模型
   - 组成：
     - 编码器（6组）
       - 组成：多头注意力+前馈神经网络
     - 解码器（6组）
       - 组成：掩蔽多头注意力+多头注意力+前馈神经网络（MLP）
   - 自注意力
     - 计算每个单词与其他单词的相关性
     - 除以根号dk的原因
       - 默认Q、K、V的维度是64
       - 原论文中作者发现当QK的乘积值太大时，经过Softmax映射后的值非0即1，产生的梯度太小，不利于网络训练
       - 除以根号dk，保证QK值在一个较小的范围内



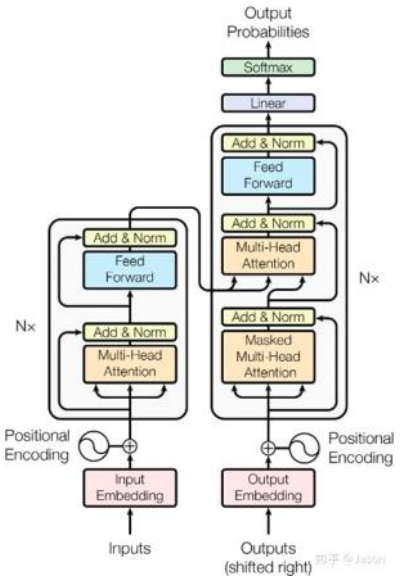
   - 多头注意力
     - 作用：
       - 扩展了模型专注于不同位置的能力
       - 为注意力层提供了多个“表示子空间”（8组）

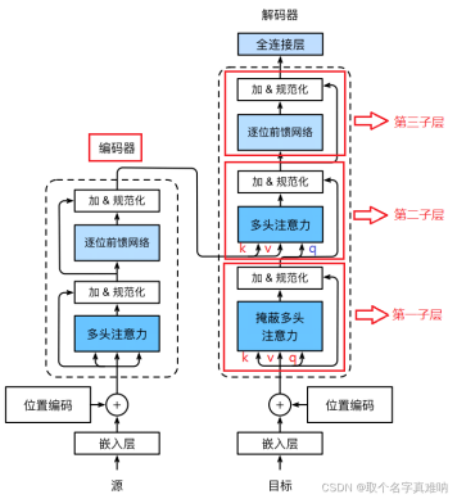


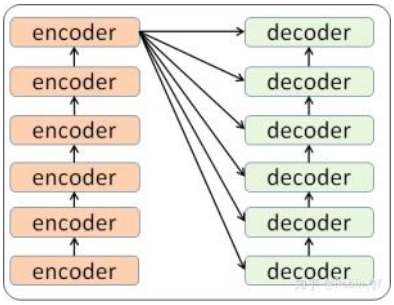
   - 掩蔽多头注意力
     - 解码器中，自注意力层只能作用于输出序列的较早位置，也就是需要在进行自注意力的softmax操作之前将未来位置的值屏蔽掉（设置为-inf）
   - 残差结构：缓解梯度消失
   - 位置编码
     - 作用：表示输入序列中单词的顺序
     - 函数：Sin+Cos
   - 最后的Linear+Softmax
     - Linear：全连接层，得到一个与单词数相同的向量（有100个词就得到100维的向量）
     - Softmax：将向量值转换为概率，概率最大的单词即为最终输出

	I	am	machine	he	is	learning
position1	0.01	0.02	0.93	0.01	0.01	0.01
position2	0.01	0.01	0.05	0.02	0.01	知乎 @Robin.QY

   - 结构图







2. ViT
   - 基于Transformer的图像分类模型（无解码器）
   - 与CNN相比，Transformer缺少归纳偏置，即先验知识
     - 目标周围的邻域信息对目标分类的影响
     - 卷积操作的平移不变性
   - 算法流程：

     (1) patch embedding：例如输入图片大小为224x224，将图片分为固定大小的patch，patch大小为16x16，则每张图像会生成224x224/16x16=196个patch，即输入序列长度为**196**，每个patch维度16x16x3=**768**，线性投射层的维度为768xN (N=768)，因此输入通过线性投射层之后的维度依然为196x768，即一共有196个token，每个token的维度是768。这里还需要加上一个特殊字符cls，因此最终的维度是**197x768**。到目前为止，已经通过patch embedding将一个视觉问题转化为了一个seq2seq问题

     (2) positional encoding（standard learnable 1D position embeddings）：ViT同样需要加入位置编码，位置编码可以理解为一张表，表一共有N行，N的大小和输入序列长度相同，每一行代表一个向量，向量的维度和输入序列embedding的维度相同（768）。注意位置编码的操作是sum，而不是concat。加入位置编码信息之后，维度依然是**197x768**

     (3) LN/multi-head attention/LN：LN输出维度依然是197x768。多头自注意力时，先将输入映射到q，k，v，如果只有一个头，qkv的维度都是197x768，如果有12个头（768/12=64），则qkv的维度是197x64，一共有12组qkv，最后再将12组qkv的输出拼接起来，输出维度是197x768，然后在过一层LN，维度依然是**197x768**

     (4) MLP：将维度放大再缩小回去，197x768放大为197x3072，再缩小变为**197x768**

     一个block之后维度依然和输入相同，都是197x768，因此可以堆叠多个block。最后会将特殊字符cls对应的输出 $\mathbf{z}_L^0$ 作为encoder的最终输出，代表最终的image presentation（另一种做法是不加cls字符，对所有的tokens的输出做一个平均），如下图公式(4)，后面接一个MLP进行图片分类

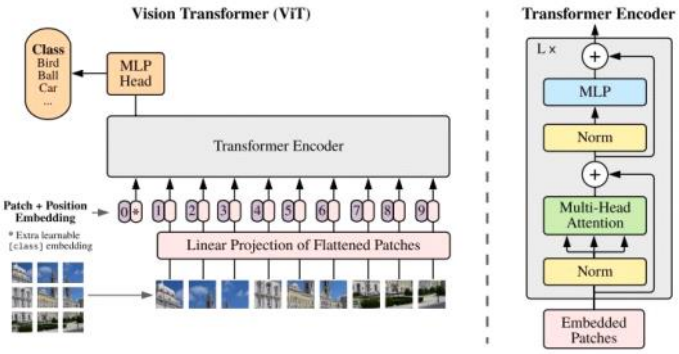
$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \quad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

   - 位置编码
     - 1-D编码：如果有9个patch，则位置编码为1-9
     - 2-D编码：编码为11,12,13,21,22,23,31,32,33. 同时考虑X轴和Y轴信息
   - 结构图



3. DETR
   - DETR是一种基于“集合预测”的目标检测框架（目标检测的新范式）
   - DETR结构
     - CNN网络：提取图像特征，输出为（C*H*W）
       - reshape为（W*H)*C　（行：共有W*H个像素，列：每个像素的特征维度为C）
     - Tansformer的编码器-解码器结构
       - 编码器输入输出维度一致
       - 解码器
         - 输入的object queries的个数是固定的，就是要预测的目标的个数N
           - object queries：可学习的位置编码，可以聚焦于图像的不同区域
           - Object Queries的含义
             - Object queries是一种可学习的位置编码
             - 用于计算Q矩阵：相当于提出问题（这个位置有什么目标）（Encoder提供了K,V）
             - Object queries是需要训练的，训练的目的相当于是学习如何提出更具体，更准确的问题
         - 并行解码N个object queries, 输出与输入是一一对应的，也是输出N个经过注意力映射后的token
     - 用于检测的前馈神经网络
       - 将N个token输入前馈神经网络进行预测，得到N个框的位置和类别分数
       - FFN负责预测边界框的位置参数，以及类别分数（1*1卷积）
   - 位置编码
     - 分别计算X和Y维度的编码，concat后与CNN的输出相加
   - 损失函数
     - 边界框匹配损失
       - DETR是集合预测，预测N个框的位置和类别，比原本的目标要多
       - 将GT扩展成N个框，此时预测集合和GT集合大小相同
       - 采用匈牙利算法进行二分图匹配，实现预测集合和GT集合的一一对应（原则：匹配损失最小）
     - 边界框损失
       - IOU损失和L1 损失的加权和
       - $$\mathcal{L}_{\text{Hungarian}}(y,\hat{y}) = \sum_{i=1}^{N}\left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + 1_{\{c_i \neq \varnothing\}}\mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\hat{\sigma}}(i)\right)\right]$$
         $$\mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\hat{\sigma}}(i)\right) = \lambda_{\text{iou}}\mathcal{L}_{\text{iou}}\left(b_i, \hat{b}_{\sigma(i)}\right) + \lambda_{\text{L1}}\left\|b_i - \hat{b}_{\sigma(i)}\right\|_1$$
   - 结构图

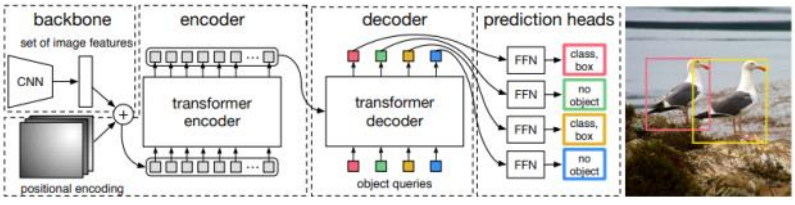
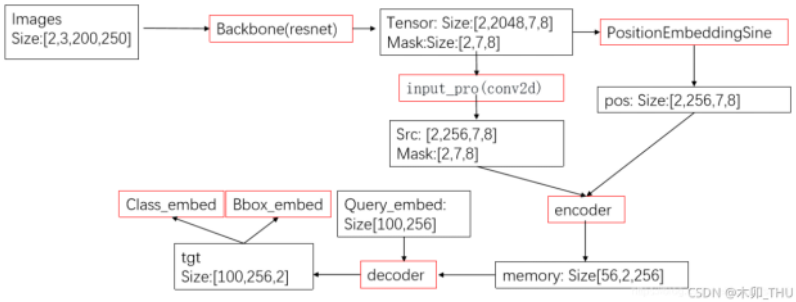
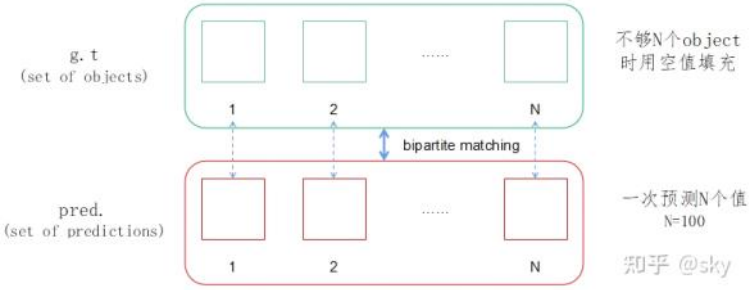


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a “`no object`” class.

CSDN @木卯_THU





4. Swin Transformer
   - 架构

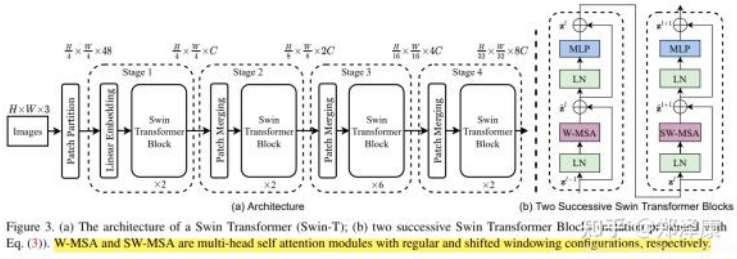
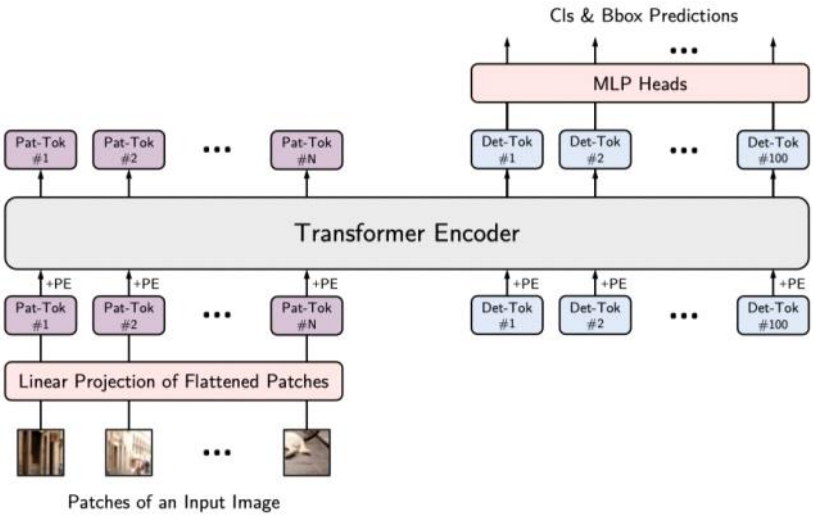


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

   - 结构解析
     - 总共分为4个阶段，每个阶段都会缩小特征图的分辨率，扩大感受野
     - Patch Embedding：将图片切成小patch，再展平为向量
     - Patch Merging：在每个stage中做下采样，缩小分辨率（在行和列方向上间隔2选取元素）
     - Swin Transformer Block
       - Window Attention：注意力计算限制在每个窗口内
       - Shift Window Attention：实现多个windows之间的交互
5. YOLOS
   - 将ViT结构迁移到目标检测任务
   - ViT是对全局上下文关系和长程依赖关系的建模，而不是局部区域关系的建模
   - YOLOS
     - YOLOS丢弃了ViT的【CLS】Token，而是添加了100个可学习的【DET】Token用于目标检测
     - YOLOS将ViT的图像分类损失替换为了匹配损失，按照DETR的预测方式执行目标检测
   - YOLOS相当于ViT和DETR的结合
   - 【DET】Token
     - 随机初始化的目标代理
     - 作用：
       - 标签分配：【DET】生成的预测框与GT集合中的目标进行一一匹配
       - 不需要将ViT的输出重新解释为2D结构再进行标签分配
   - 结构图



6. ViDT
   - 对ViT和DETR的高效集成模型
   - 动机：
     - 传统的目标检测方法的检测性能依赖于精心设计的组件：Anchor和NMS
     - DETR
       - 消除了这些精心设计的组件，基于Transformer的集合预测模型
         - ResNet提取图像特征
         - Transformer的Encoder-Decoder进行图像编码解码，Object Queries对应的Tokens通过FFN进行一一对应的最终的检测
       - 缺陷：
         - 收敛慢：传统37个epoch可以达到的效果，DETR需要500个epoch
         - 小目标检测效果差：用的是最后一层特征图生成的嵌入向量，小目标信息较少
       - 改进的Deformable DETR
         - 可形变注意力+多尺度特征图
           - 可行变注意力
             - 每一个Q不需要与所有的K进行计算
             - 以当前Q为参考点，通过线性映射计算偏移量，选择当前Q附近的M个点为采样点，计算注意力
             - $$\mathrm{DeformAttn}(z_q, p_q, x) = \sum_{m=1}^{M} W_m [\sum_{k=1}^{K} A_{mqk} \cdot W'_m x(p_q + \Delta p_{mqk})]$$
           - 多尺度特征图
             - 选取ResNet多个Stage的输出特征图生成嵌入向量，每一层的嵌入向量都添加了对应层的【层】标记
             - 训练过程中随机初始化【层】标记，Encoder-Decoder协同训练
     - ViT
       - 完全基于Transformer的图像分类模型
         - 额外添加了一个【CLS】Token负责类别预测
         - ViT也证明不需要额外的归纳偏置（先验知识：图像结构信息，卷积的平移不变性），模型依旧可以学习到有效的特征信息
       - 缺陷：复杂度较高，与图像大小呈二次增长
     - DETR（ViT）
       - 利用ViT代替ResNet提取图像特征，利用Encoder-Decoder作为Neck组件进行编码解码，Object Queries对应的Tokens通过FFN进行一一对应的最终的检测
       - 缺陷：
         - ViT的缺陷
         - Neck组件中Encoder-Decoder的注意力计算增加检测器的计算开销
         - 难扩展
     - YOLOS
       - 基于ViT的目标检测器
         - 在Encoder的输入中添加了【DET】Tokens，实现了无Neck组件的目标检测
           - 因为【DET】组件在ViT的Encoder中已经进行了全局注意力的交互，可直接用于检测，因此不再需要Decoder，从而实现无Neck组件的检测
         - YOLOS实现了高效的计算，同时也证明2D目标检测可以通过序列-to-序列的方式实现
         - 缺陷：

        - ◇ 继承了ViT的缺陷：高复杂度是由于全局注意力
        - ◇ 无Neck组件：无法使用多尺度特征提升检测性能
- ○ ViDT结构
    - ▪ RAM（重新配置的注意力模块）
        - □ 结构图

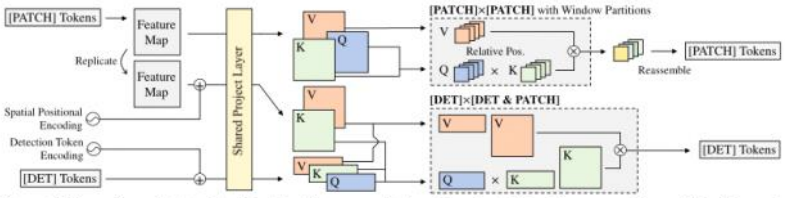


Figure 3. Reconfigured Attention Module (Q: query, K: key, V: value). The skip connection and feedforward networks following the attention operation is omitted just for ease of exposition.

        - □ RAM将【Patch】和【DET】的全局注意力分解为三部分：
            - ◆ 【Patch】*【Patch】
                - ◇ 作用：聚合全局关键内容
                - ◇ 操作：
                    - ▸ 窗口注意力：局部自注意力计算
                    - ▸ 移动窗口注意力：提供窗口之间的交互，捕获全局信息
            - ◆ 【DET】*【DET】
                - ◇ 作用：每个【DET】通过计算全局注意力来捕获他们之间的关系，有助于定位不同的目标
                - ◇ 操作：自注意力（self-attention）
            - ◆ 【Patch】*【DET]
                - ◇ 作用：每个【DET】都表示不同的对象，因此需要为每一个【DET】生成不同的embedding，【Patch】中的关键内容被聚合到每个【DET】上
                - ◇ 操作：交叉注意力
            - ◆ 利用RAM替换Swin Transformer中的所有注意力模块
            - ◆ 位置编码：
                - ◇ 【Patch】：Swin Transformer的相对位置编码
                - ◇ 【DET】：可学习的位置编码（【DET】没有特定的顺序）
            - ◆ 【Patch】*【DET】注意力只在Swin Transformer的最后一个Stage使用，避免增加额外的计算开销
    - ▪ Encoder-Free Neck组件
        - □ 无Encoder的原因：Swin Transformer已经提取了适用于目标检测的细粒度特征
        - □ Decoder：采用Deformable DETR的Decoder可以充分利用多尺度特征
            - ◆ 两个输入：每个Stage生成的【Patch】+最后一个Stage生成的【DET】
            - ◆ 使用多尺度形变注意力：用于生成新的【DET】，聚合从多尺度特征图中采样的一小部分关键内容
                - ◇ 【DET】*【DET】的自注意力
                - ◇ 【Patch】*【DET】的交叉注意力

$$\mathrm{MSDeformAttn}([\mathtt{DET}], \{\boldsymbol{x}^l\}_{l=1}^{L}) = \sum_{m=1}^{M} \boldsymbol{W}_m \Big[\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlk} \cdot \boldsymbol{W}'_m \boldsymbol{x}^l \big(\phi_l(\boldsymbol{p}) + \Delta \boldsymbol{p}_{mlk}\big) \Big], \qquad (1)$$

                - ◇ where m indices the attention head and K is the total number of sampled keys for content aggregation. In addition, $\phi_l(\boldsymbol{p})$ is the reference point of the $[\mathtt{DET}]$ token re-scaled for the l-th level feature map, while $\Delta \boldsymbol{p}_{mlk}$ is the sampling offset for deformable attention; and A_{mlk} is the attention weights of the K sampled contents. $\boldsymbol{W}_m$ and $\boldsymbol{W}'_m$ are the projection matrices for multi-head attention.
        - □ 辅助技术
            - ◆ 辅助解码损失
            - ◆ 迭代的边界细化
    - ▪ 基于标签匹配的知识蒸馏
        - □ 目的：降低计算成本
        - □ 操作：
            - ◆ 教师模型：预训练的大型ViDT
            - ◆ 学生模型：需要学习的小型ViDT
            - ◆ 将学生模型的【Patch】和【DET】与教师模型的【Patch】和【DET】匹配，将教师模型的知识转移到学生模型

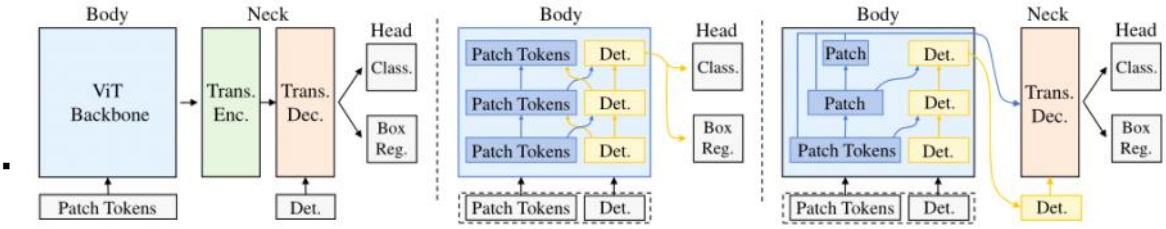
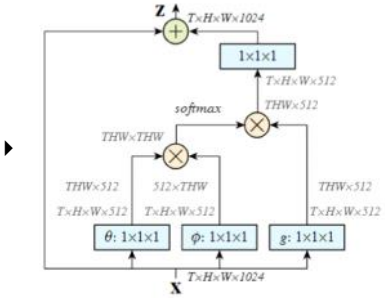
- $$\ell_{dis}(\mathcal{P}_s, \mathcal{D}_s, \mathcal{P}_t, \mathcal{D}_t) = \lambda_{dis}\Big(\frac{1}{|\mathcal{P}_s|}\sum_{i=1}^{|\mathcal{P}_s|}\Big\|\mathcal{P}_s[i] - \mathcal{P}_t[i]\Big\|_2 + \frac{1}{|\mathcal{D}_s|}\sum_{i=1}^{|\mathcal{D}_s|}\Big\|\mathcal{D}_s[i] - \mathcal{D}_t[i]\Big\|_2\Big), \qquad (2)$$
  - 结构图
  - 

Figure 2. Pipelines of fully transformer-based object detectors. DETR (ViT) means Detection Transformer that uses ViT as its body. The proposed ViDT synergizes DETR (ViT) and YOLOS and achieves best AP and latency trade-off among fully transformer-based object detectors.

- Transformer和CNN的区别
  - Transeformer仍属于机器学习，没有卷积，池化等操作
  - 输入形式
    - Transformer：一维序列
    - CNN：二维图像
  - 建模方式
    - Transformer：利用数据之间的全局相似性，更好的建立长程依赖
    - CNN：关注数据的局部关联性，更多的关注数据的空间结构信息
  - 先验知识
    - Transformer：位置编码信息
    - CNN：空间结构信息，平移不变性
- Transformer中的Q、K、V的含义
  - Q：Question查询向量
  - K：表示被查询的关键信息
  - V：表示被查询向量
  - 通俗解释：
  - 类似于搜索过程
  - 在网上搜索问题Q，每篇文章V都有对应的标题K，搜索引擎将问题Q与每篇文章V的标题K进行匹配，查看相关度QK，再对这些不同相关度的文章V进行加权求和得到一个新的Q'，而Q'融合了相关性强的文章V的更多信息，而融合了相关性弱的文章V的较少信息
- Transformer和Non-Local的区别
  - 相同点
    - 两者都基于注意力机制，有利于建立长距离依赖关系
  - 不同点
    - Transformer给出了一种建模方案，而Non-local只是自注意力机制的一种组件化模块（算子），即插即用
    - Non-local依旧作用于二维图像上，Transformer作用于一维序列上，两者作用在不同的输入模态上
  - Non-Local结构图
  - 

- Transformer中为什么用LN不用BN
  - Transformer用于处理文本数据，一个Batch_size对应着句子长度，每个样本相当于一个单词，通常来说，单词长度是不固定的
  - BN
    - BN是对多个样本的同一属性进行标准化处理
    - 如果使用BN层可以保证靠前的单词字符可以做相同的均值方差操作，而靠后的多余数据不能进行此操作，作使得句子与句子之间的关

联性消失

- LN
  - LN是对同一样本的所有属性进行标准化处理
  - LN操作依旧保留了句子的原始长度，使得句子中前后词之间的关联性更强

- 匈牙利算法
  - 作用：一种求解任务分配问题（指派问题）的组合优化算法
  - 指派问题模型
    -