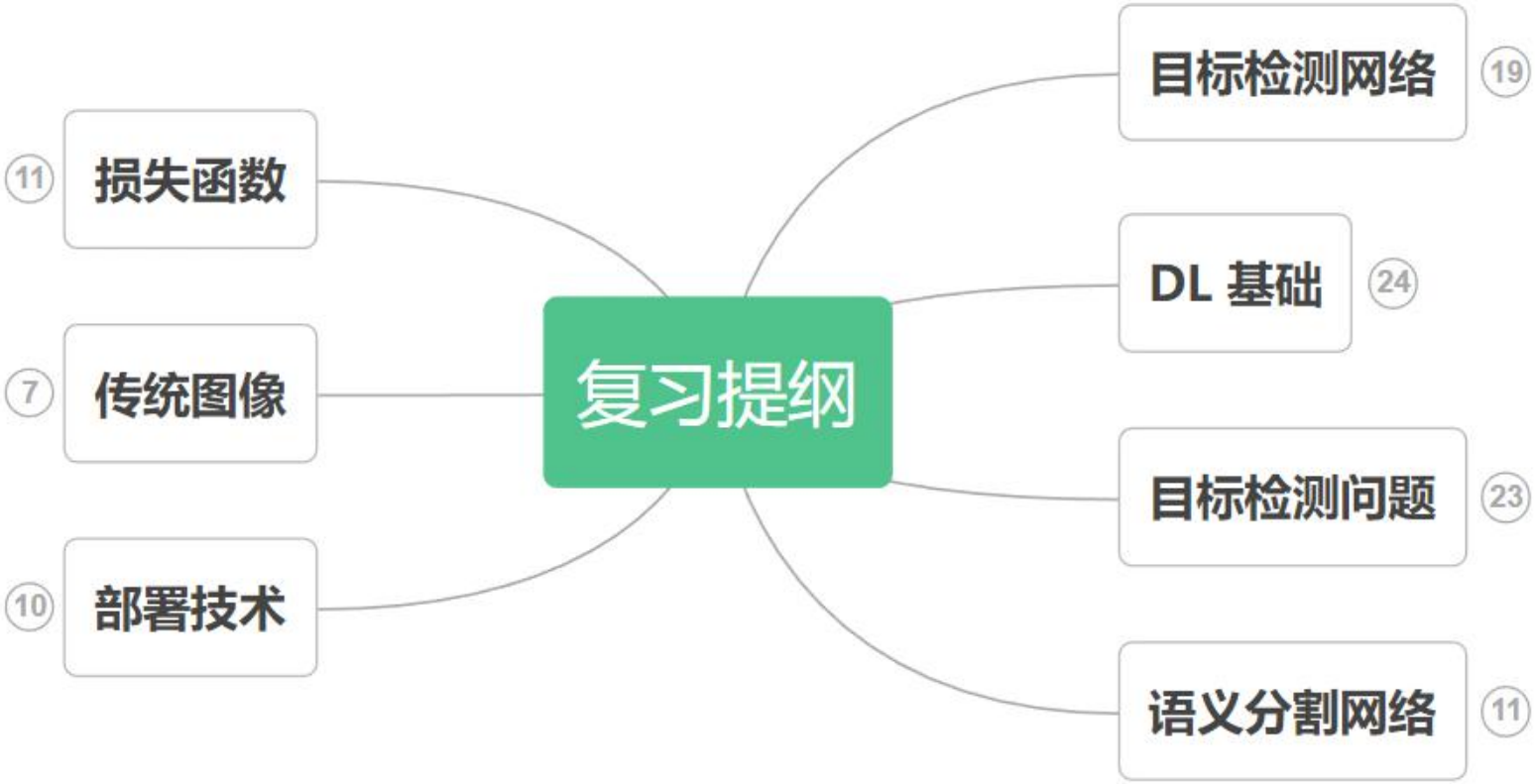
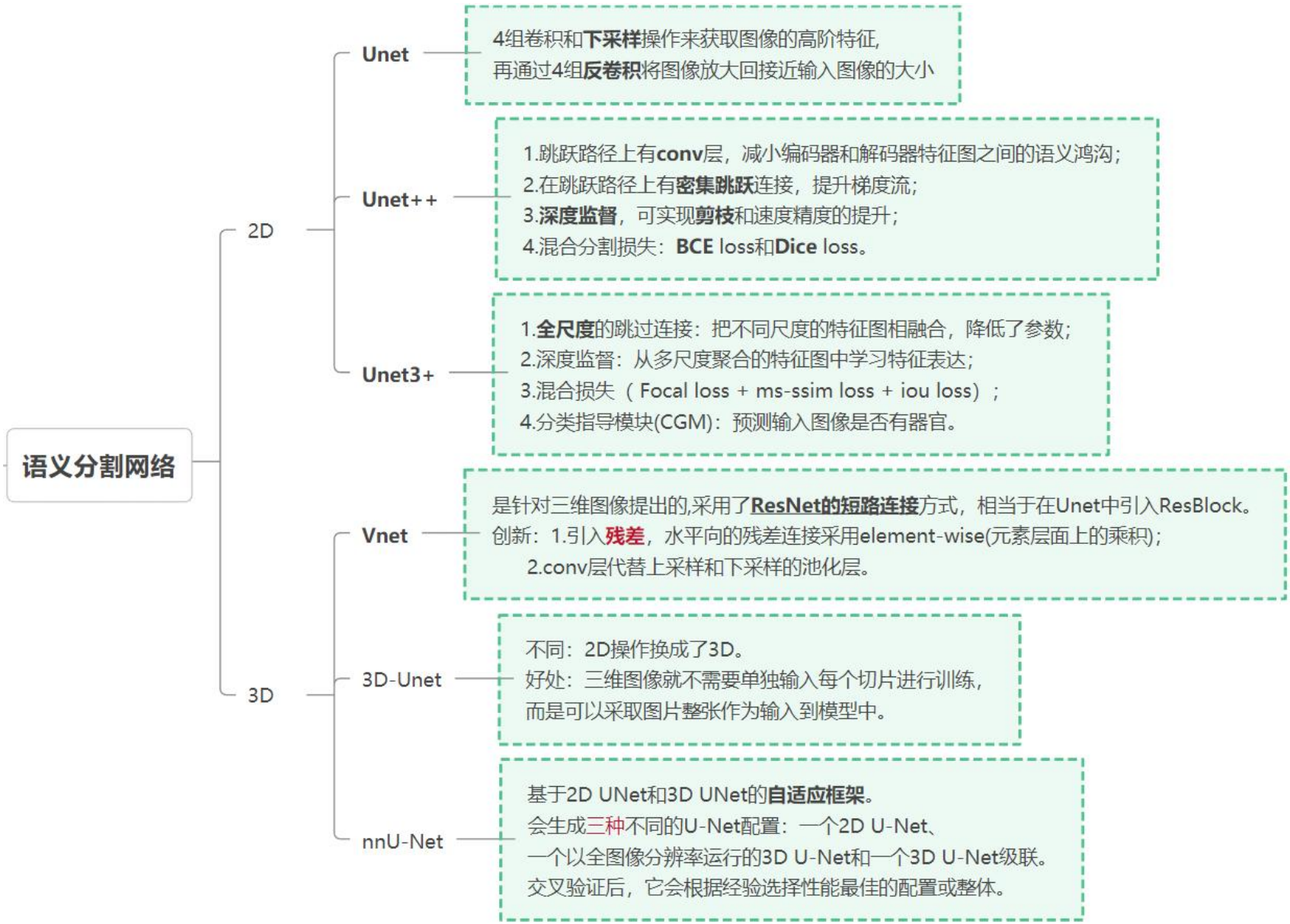

复习提纲
目标检测网络
19
DL 基础
24
目标检测问题
23
语义分割网络
11
损失函数
11
传统图像
7
部署技术
10


One Stage
SSD
conv提取特征-> 下采样-> featuremap划分为nxn格 ->
中心点出发，生成Anchor -> 对anchor进行reg和cls ->检测框+类别+置信度
YoloV1
没有设计anchor,最后采用full层进行类别和目标位置的回归预测。
YoloV2
1.大尺度预训练分类；2.Darknet-19; 3.加入Anchor
YoloV3
1.多尺度预测(FPN)； 2.Darknet-53(类似ResNet引入残差结构);
3.分类loss: BCE loss，不适应softmax
YoloV4
1.输入端：Mosaic数据增强、cmBN、SAT自对抗训练
2.BackBone：CSPDarknet53、Mish、DropBlock
3.Neck:SPP(任意尺寸的输入产生固定大小的输出)、FPN+PAN结构
4.预测：CIOU，预测框筛选：nms变为DIOU_nms
YoloV5
1.输入端改进：Anchor的计算加入了train，自适应计算不同训练集的best anchor-v
2.BackBone：CSPDarknet53，图片输入前加入Focus切片(邻近下采样,3->12)；
四种不同大小的网络结构：s,m,l,x,depth+width控制
3.Neck: SPP+PAN多尺度特征融合
4.预测：GIOU，输出：置信度、边框信息、分类信息


Two Stage
R-CNN
SS提取RP -> CNN提取特征 -> SVM分类-> BBOX回归
缺点：训练繁琐(微调网络+训练SVM+训练Bbox)，慢，占空间。
SPP-Net
Fast R-CNN
SS提取RP -> CNN提取特征 -> softmax分类-> 多任务Loss Func边框回归
缺点：特征提取0.3秒;未实现端到端；RP在CPU实现的。
Faster R-CNN
简述： RPN提取RP -> CNN提取特征 -> softmax分类-> 多任务Loss Func边框回归
优点：真正端到端； 缺点：无法实施检测；获取RP计算量还是大。
1.Conv层提取特征图
2.RPN：生成RP ->生成Anchor ->softmax判断anchors属于前(后)景,二分类;
bbox修正 anchor box
3.Roi Pooling: RPN生成的proposals和VGG16最后一层得到的F-map，得到固定大小的F-map
4.Classifier：将Roi Pooling-> 固定大小的F-map->全连接->Softmax分类;
利用L1 Loss完成bbox回归 ->物体的精确位置。
R-CNN与 Faster R-CNN区别?
Faster R-CNN设计了提取候选区域的网络RPN，代替了费时的SS，使得检测速度大幅提升。


语义分割网络
2D
Unet
4组卷积和下采样操作来获取图像的高阶特征,
再通过4组反卷积将图像放大回接近输入图像的大小
Unet++
1.跳跃路径上有conv层，减小编码器和解码器特征图之间的语义鸿沟;
2.在跳跃路径上有密集跳跃连接，提升梯度流;
3.深度监督，可实现剪枝和速度精度的提升;
4.混合分割损失：BCE loss和Dice loss。
Unet3+
1.全尺度的跳过连接：把不同尺度的特征图相融合，降低了参数;
2.深度监督：从多尺度聚合的特征图中学习特征表达;
3.混合损失（Focal loss + ms-ssim loss + iou loss）;
4.分类指导模块(CGM)：预测输入图像是否有器官。
3D
Vnet
是针对三维图像提出的,采用了ResNet的短路连接方式，相当于在Unet中引入ResBlock。
创新：1.引入残差，水平向的残差连接采用element-wise(元素层面上的乘积);
2.conv层代替上采样和下采样的池化层。
3D-Unet
不同：2D操作换成了3D。
好处：三维图像就不需要单独输入每个切片进行训练，
而是可以采取图片整张作为输入到模型中。
nnU-Net
基于2D UNet和3D UNet的自适应框架。
会生成三种不同的U-Net配置：一个2D U-Net、
一个以全图像分辨率运行的3D U-Net和一个3D U-Net级联。
交叉验证后，它会根据经验选择性能最佳的配置或整体。

1.**Anchor**的理解？怎么设置？

概念: 图像上预设好的不同大小，不同长宽比的参照框。
使用**K-means**获得anchor box：
1.随机选取K个box作为初始anchor；
2.使用IOU度量，将每个box分配给与其距离最近的anchor；
3.计算每个簇中所有box宽和高的均值，更新anchor；
4.重复2和3，直到anchor不再变化，或者达到了最大迭代次数。

2.**NMS**,**Soft Nms** 原理及区别？

NMS

目的：消除多余（交叉重复）的窗口，找到最佳物体检测位置。
原理：将所有的预测概率**从高到低**排列，每个bbox只保留概率**最高**的那个，移除其他的。
最后列表就只保留了每个 bbox 预测概率最高的那个值。

Soft Nms

Nms略显粗暴（hard），因为NMS直接将和得分最大的box的IOU大于某个阈值的box的得分置零。
softNms 用稍低一点的分数来代替原有的分数，而不是直接置零。
Soft-NMS: 提升多个重叠物体检测的平均准确率(mAP)。

3.**漏检**问题该怎么解决？

原因：anchor的大小和长宽比与待检测的物体尺度不一致，anchor与物体的IoU小于阈值。

1.类别样本数量少：**数据增强**；2.**Focal Loss**； 3.用**Kmeans**来设置anchor；4.网络结构： **FPN** 5.**OHEM**策略

4.小目标物体检测如何处理？

1.数据增强 2.**MTCNN** 3.特征融合：**FPN** 4.降低下采样率与空洞卷积

5.正负样本不均衡如何解决?

1.**样本数据**方面： 各种的上采样、下采样方法。
2.**损失函数**角度： 比如增加样本少的类别的损失权重，减少样本多的类别的损失权重。

6.误检的问题如何解决?

训练负样本可 降低误检测率；收集负样本后，将正负样本集及其标签作为训练集送入模型训练；
1.对误检的图像生成一个空的xml（文件中没有任何对象）
2.误检图像中含有正样本，但不是误检的类别。训练结果：对网络进行训练时，会产生loss，并且训练后的模型，不会再把误检图像中的背景误检为人物。

7.评价指标有哪些?

- mAP，AP
  - AP是在**单个类别**下的，mAP是AP值在**所有类别**下的均值；
  - 计算mAP的**置信度阈值**: 选取TOP N（置信度分数从高到低排名的前N个检测）的样本来统计TP,FP,FN，计算AP。
- P-R曲线
  - 精准率(P) = TP/(TP+FP) = 所有被预测为正的样本中实际为正的样本的概率
  - 召回率(R) = TP/(TP+FN) = 在实际为正的样本中被预测为正样本的概率
  - 横轴为**R**(查全率)，纵轴为**P**(查准率);
  - 平衡点(BEP)：P=R时的取值；值越大表明分类器性能越好。
- F1-Score
  - **调和平均**： 2/F1 = 1/P + 1/R
  - P和R的权衡: 只有在P和R都高时，F1 score才会很高，比BEP更为常用。
- ROC和AUC
  - ROC曲线：横轴为FPR，纵轴为TPR(真正率)
  - AUC: **ROC曲线下的面积**，AUC的取值：[0.5,1] AUC值越高, 预测准确率越高。

1.BN原理？作用？

原理：计算mini-batch均值,方差 ->归一化 ->尺度**缩放**(γ)和**偏移**(β)操作
-> 变换回原始的分布，恒等变换
目的：补偿网络的非线性表达能力，因为经过标准化之后，偏移量丢失。
BN -> 激活值落入非线性函数的线性区内，导数**远离导数饱和区**，加速训练收敛；
增加缩放和偏移两个参数，训练学习得到，对变换后的激活反变换，网络表达能力增强。

2.Dropout原理？作用？

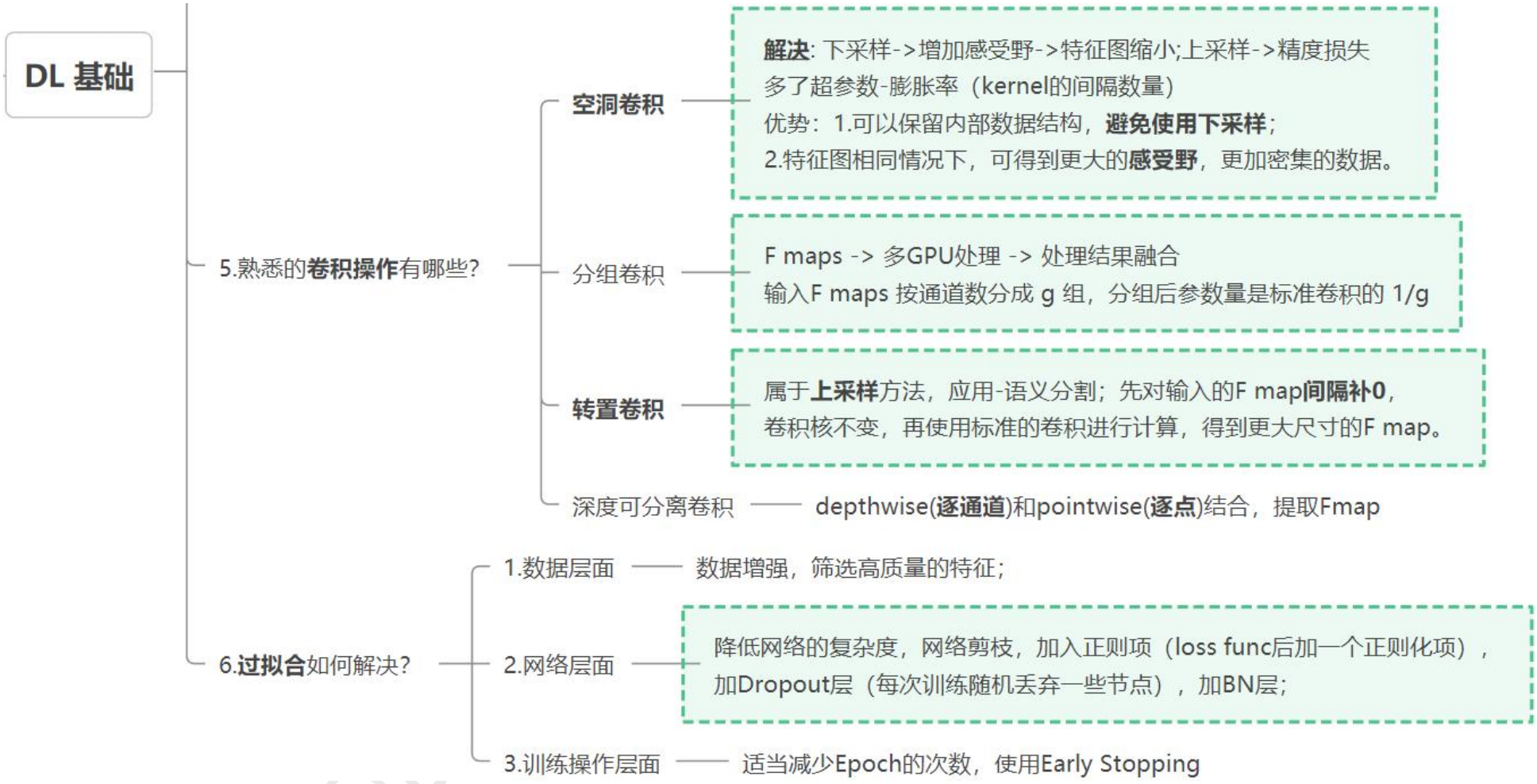
概念：Forward时，某个神经元的激活值以一定的**概率P**停止，增强模型泛化。
缺点：每次迭代都会随机消除一些神经元结点的影响 -> 无法确保成本函数单调递减。
使用dropout时，先设置 **keep_prob==1.0** -> J(w,b)函数单调递减，再打开 dropout。

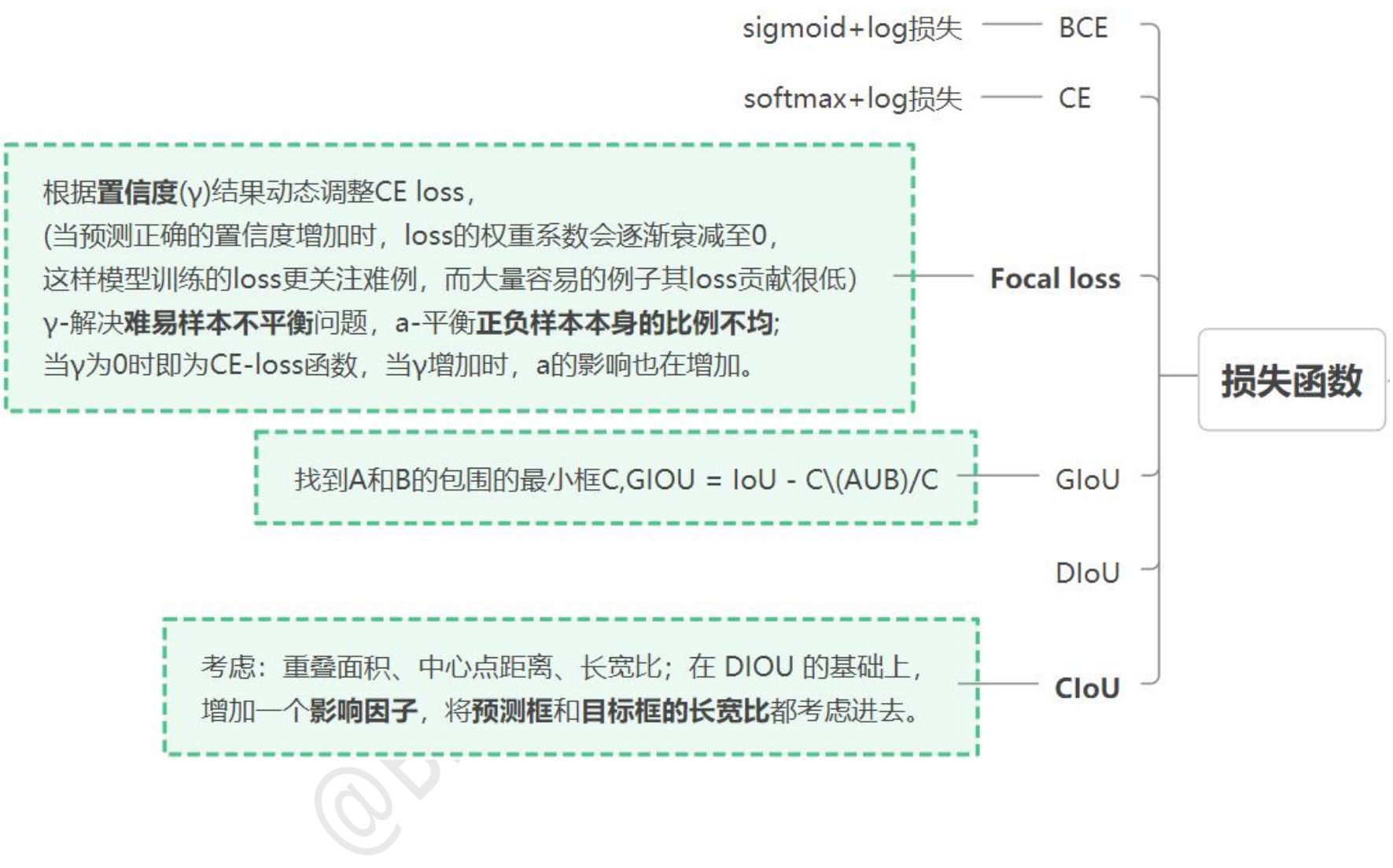
3.余弦退火原理？

cos函数中随着x的增加余弦值->缓慢下降->加速下降->缓慢下降。
执行完Ti个epoch后开始热重启,通过增加 lr 来模拟，
并且重启之后使用权重作为初始解，重启就是为了通过增大学习率来跳过局部最优。

4.权重衰减 (L2正则化)

权重衰减系数**λ**越小，对过拟合的抑制更好


DL 基础
5.熟悉的卷积操作有哪些?
空洞卷积
解决: 下采样->增加感受野->特征图缩小;上采样->精度损失
多了超参数-膨胀率（kernel的间隔数量）
优势：1.可以保留内部数据结构，避免使用下采样；
2.特征图相同情况下，可得到更大的感受野，更加密集的数据。
分组卷积
F maps -> 多GPU处理 -> 处理结果融合
输入F maps 按通道数分成 g 组，分组后参数量是标准卷积的 1/g
转置卷积
属于上采样方法，应用-语义分割；先对输入的F map间隔补0，
卷积核不变，再使用标准的卷积进行计算，得到更大尺寸的F map。
深度可分离卷积
depthwise(逐通道)和pointwise(逐点)结合，提取Fmap
6.过拟合如何解决?
1.数据层面
数据增强，筛选高质量的特征；
2.网络层面
降低网络的复杂度，网络剪枝，加入正则项（loss func后加一个正则化项），
加Dropout层（每次训练随机丢弃一些节点），加BN层；
3.训练操作层面
适当减少Epoch的次数，使用Early Stopping


损失函数
BCE
sigmoid+log损失
CE
softmax+log损失
Focal loss
根据置信度(γ)结果动态调整CE loss，
(当预测正确的置信度增加时，loss的权重系数会逐渐衰减至0，
这样模型训练的loss更关注难例，而大量容易的例子其loss贡献很低)
γ-解决难易样本不平衡问题，a-平衡正负样本本身的比例不均;
当γ为0时即为CE-loss函数，当γ增加时，a的影响也在增加。
GIoU
找到A和B的包围的最小框C,GIOU = IoU - C\(AUB)/C
DIoU
CIoU
考虑：重叠面积、中心点距离、长宽比；在 DIOU 的基础上，
增加一个影响因子，将预测框和目标框的长宽比都考虑进去。


传统图像
高斯滤波原理？计算步骤？
作用：平衡对图像的噪声的抑制和对图像的模糊；
σ越大，高斯滤波器的频带就较宽，对图像的平滑程度就越好。
1.计算邻域的 dis(其他像素点 - 中心)，带入二维高斯函数，计算出高斯模板；
2.模板若为小数，归一化；
3.将高斯模板的中心对准待处理的图像矩阵 -> 对应元素相乘后相加，没有元素的地方补零;
4.每个元素分别进行上述计算 -> 输出矩阵: 高斯滤波的结果。
SIFT、HOG
图像的特征提取方法？(熟悉)
边缘检测算子知道哪些？(熟悉)
Canny算子
1.高斯滤波器平滑图像；
2.一阶偏导的有限差分来计算梯度的幅值和方向；
3.对梯度幅值进行NMS;
4.用双阈值算法检测和连接边缘。


部署技术
Flask + Docker部署
基于Ubuntu18.04系统部署， 显卡为Tesla的P40，进行Flask + Docker 部署:
总结：确定好输入和输出节点 -> 把模型导出为SavedModel格式 ->
用TF-Serving启动服务 -> 客户端发http（grpc）请求获取预测结果。
1.创建Dockerfile，添加部署到服务器上的路径以及启动的python文件名和变量名；
2.使用build命令构建Docker镜像，登陆远程服务器，pull 镜像(docker pull tf/serving:2.1.0);
3.部署到服务器上：apt-get install docker.io；
4.运行docker镜像;
5.更新项目,维护好配置文件,build ->push,在服务器pull下来,重新运行即可。
TensorRt原理
支持FP16和INT8的计算，训练使用32(16)位数据，推理时降低模型参数的位宽来进行低精度推理;
do_Inference原理:
host device的方式转化将数据传输到GPU -> 异步执行做推断 -> 推断的结果从GPU拿到 Host
(即设备到主机) -> 将流进行同步。多个input,output结点用do_inference_2方法。
使用python接口导入TF模型:
导入TensorRT -> 创建冻结的TF模型 -> 使用UFF转换器(convert-to-uff)将pb模型转为UFF文件
->更改路径以反映Samples中包含的模型的位置 ->创建builder,network和parser
基于tflite的Android部署
1.build.gradle: 添加tf-lite-gpu;
2.model放在asset/，tf相关的文件放于tflite/;
3.设置tfliteOption(cpu、gpu)，构造函数(loadModelFile)
掉用tfliteModel,初始化图像数据，分配内存;
4.做预测的构造函数recognizeImage,先把bitmap -> byteBuffer;
5.runInference操作(tflite.run()) -> 所有的概率值。
NCNN部署
pt模型转onnx-> 再转为ncnn-> 修改ncnn模型导出-> 部署在Android