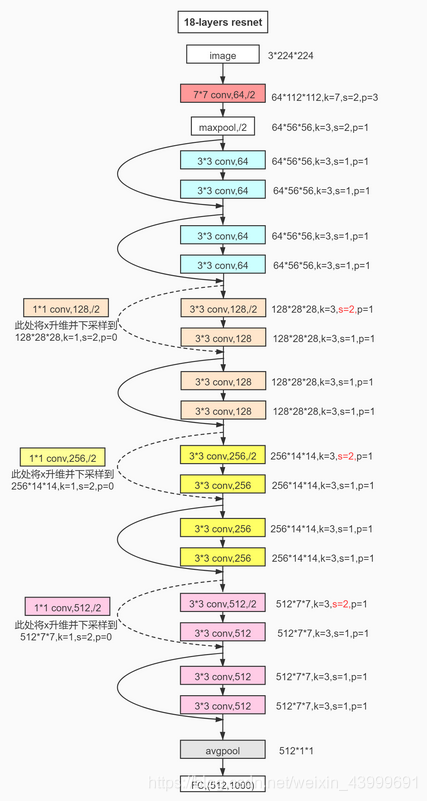
### 实验一：使用Resnet18完成CIFAR-100上图像分类任务

### 数据集介绍及划分

### 本次使用数据集为CIFAR-100数据集，由Alex Krizhevsky，Vinod Nair和Geoffrey Hinton收集。该数据集由100个类，每类600个图像组成。其中每类中有500个训练图像和100个测试图像。每个图像大小为32\*32像素，具有“精细”的类别标签以及“粗糙”的超类标签。

1. **网络结构**

本次实验使用CNN网络为Resnet-18，其网络结构图如下：



其由17个卷积层和1个FC层组成，通过一系列残差块搭载堆积而成，其中图片在layer2、layer3、layer4的第一个BasicBlock进行下采样，其余层保持原通道数。通过残差跳跃连接，缓解了网络深层有可能出现的梯度消失问题。

1. **实验参数设置**

（原作者地址：<https://github.com/weiaicunzai/pytorch-cifar100）>

**卷积网络：Resnet18**

**预训练：**无预训练模型

**训练方法：**使用Nesterov动量法，设定动量值为0.9

**epoch：**200

**batchsize：**128

**Iteration**：125

**learning\_rate**：Init\_lr=0.1（最大学习率）,Min\_lr=8e-4（最小学习率）

**学习率下降策略**：学习率在第60，120，160个epoch时除以5

**Loss function**：baseline、cutout使用CrossEntropyLoss计算，cutmix及mixup首先计算出图像关于两个混合类各自的隶属度，输出两个标签，然后对两个标签分别通过CrossEntropyLoss计算loss后，按照隶属度大小加权求和得到。

**cutout:** n\_holes=1, length=8

**mixup:** alpha=0.5

**cutmix:** alpha=1

**训练时长：约3.5h**

1. **评价指标**

baseline与通过cutout、mixup及cutmix图像增强后训练最终结果指标如下表：

|  |  |  |
| --- | --- | --- |
|  | top1 err | top5 err |
| Resnet18 | 23.36% | 7.21% |
| Resnet18+cutout | 24.06% | 7.04% |
| Resnet18+mixup | 22.32% | 6.83% |
| Resnet18+cutmix | 21.11% | 5.52% |

从上表中可以看出：cutout图像增强对Resnet18在CIFAR-100数据集上分类误差无明显下降，本次实验top1 err相较于baseline略有提升，top5 err略有下降，总体而言有一定的图像增强效果，但不明显。mixup方法top1 err比baseline降低了超过一个百分点，并且top5 err降低大概0.4%，增强效果相较于cutout更强。cutmix方法top1 err下降了2.25%，并且top5 err下降了1.69%，是这三种图像增强方法中最有效的，可以使模型泛化能力更强的方法。故本实验得到结论为（按照方法泛化性能好坏从小到大排序）：

1. **图像增强方法可视化**

**选取样本图片为（1\_3432.jpg, 8\_4765.jpg, 65\_6731.jpg）**

**cutout(n=1,d=8):**

****

**mixup(alpha=0.5)：**

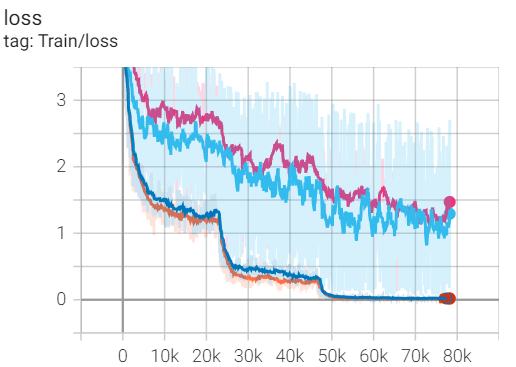
****

### cutmix**(alpha=1)**:

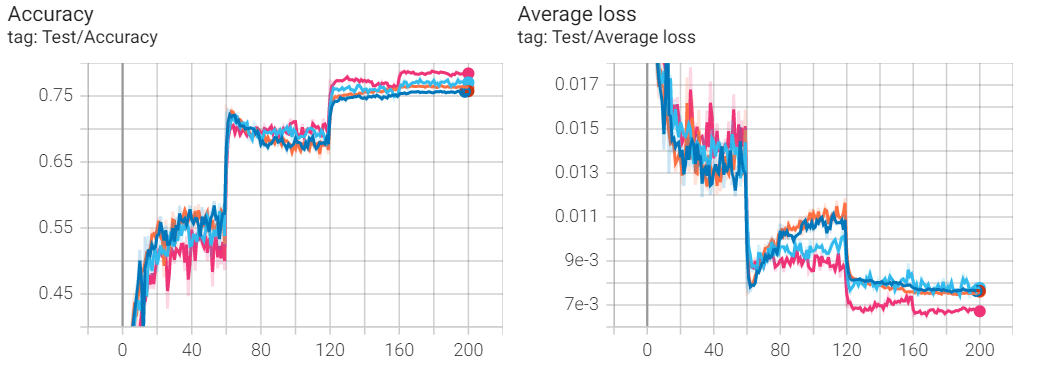


### 训练及测试loss曲线

训练loss（橙色baseline，深蓝cutout，浅蓝mixup，粉色cutmix）：



测试acc及loss（橙色baseline，深蓝cutout，浅蓝mixup，粉色cutmix）：



从上面曲线中可以看出，baseline与cutout的train\_loss十分接近，随着学习率下降而不断下降，最终接近于0，这说明其泛化性能并不好，cutout也产生这样的现象可能是由于depth设置太小，而mixup与cutmix的train\_loss下降比较缓慢，而且由于每个epoch都重新随机mix和cutmix区域和样本，故train\_loss波动较为剧烈，但这也带给它们更好的泛化性能，这一点可以从test\_acc上看到。每次学习率下降，导致test\_loss波动，可以看到受学习率下降影响更小的是cutmix，接下来是mixup，然后是cutout、baseline，即当模型泛化性能更好时，其受学习率影响也就变得更小。开始训练时学习率较大，故数据量更随机更大的mixup及cutmix曲线test\_acc比较低，但当后续不断训练，学习率不断下降后，其真正优势才体现出来，故当选择某些图像增强技术来提升泛化性能时，需要提供充分的时间及较小的学习率让其最终收敛，若最后结果不好，很可能是训练学习率较大，模型未收敛。

### 代码及模型地址

代码地址：https://github.com/theWaWang/Mid\_work/tree/main/CIFAR-100

模型地址：链接：https://pan.baidu.com/s/1vLqMysZgVP4qv2MfEz5\_TQ

提取码：2fw

### 实验2：在VOC数据集上训练并测试Faster R-CNN和YOLO V3

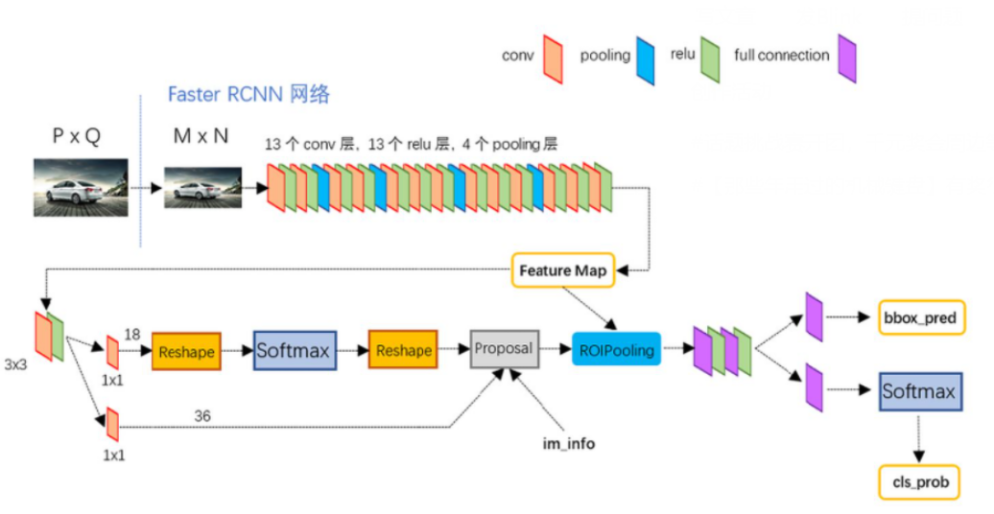
1. **数据集介绍及划分**

本次使用数据集为VOC2007及VOC2012数据集。VOC数据集始于2005年的Pascal VOC图像检测比赛，比赛给定训练数据集图片标注以及不带标注信息的测试数据集，通过比较测试数据集检测结果的准确率进行排名。该比赛于2012年停办，故数据集共有从2005-2012的8个版本。使用2007及2012版本数据集的原因是VOC2007相较于之前两年拓展为20个图类，而VOC2012是与VOC2007没有交集的从08-12年中最丰富完善的数据集。

VOC数据集文件结构相同，VOC+年份文件夹下设五个分立文件夹，Annotations文件夹下保存的为训练图片中的物体及bndbox标签，JPEGImages文件夹存放所有的图片信息（训练图片+测试图片），ImageSets文件夹下设Layout、Main以及Segmentation三个文件夹，其中Layout文件夹存放数据集划分txt文件（训练、验证、测试），Main文件夹存放按类统计数据，包括各类在图片中出现的数据，Segmentation是Main中数据的总和。SegmentationClass及SegmentationObject文件夹存放检测结果图。VOC2007共有20类9963张图片（2501+2510+4952），VOC2012共有20类17125张图片（5717+5823），VOC2012test共有20类16135张图片（10991）。

1. **网络结构**
2. **Faster R-CNN**

网络结构图（其中conv layers以vgg16为例）：



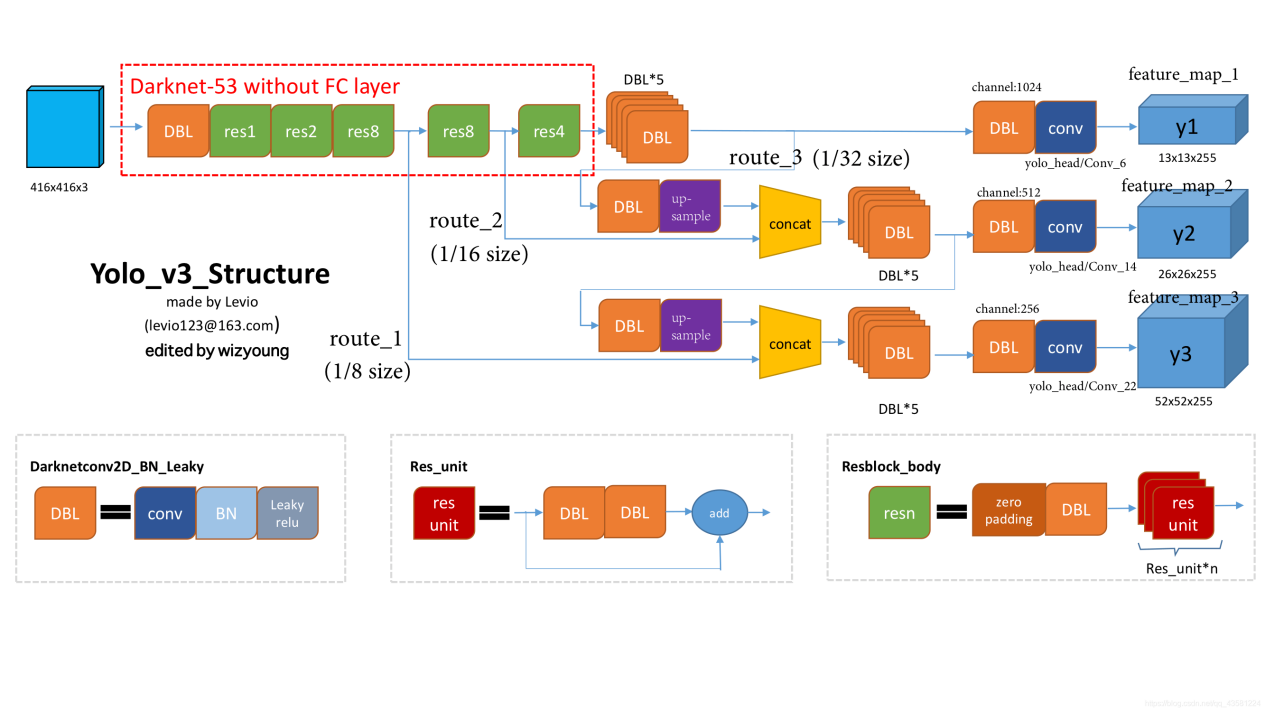
Faster R-CNN主要可分为四个模块：

1. conv layers，特征提取网络，由一组conv+relu+pooling层来提取图片的特征图，一般采用Resnet或VGG。
2. RPN net，区域候选网络，判断anchor内是否有目标，修正anchors以使检测框更加精确。
3. RoI Pooling，兴趣域池化，用于收集RPN生成的proposal框信息，并结合conv layers产生的特征图，生成具有特征的proposal框信息，送入后续全连接层进行分类及回归。
4. 全连接层，根据特征框分类计算框内物体类别及置信度，然后通过回归得到检测框精确位置。

网络结构图（其中conv layers以vgg16为例）：

1. **YOLO V3**

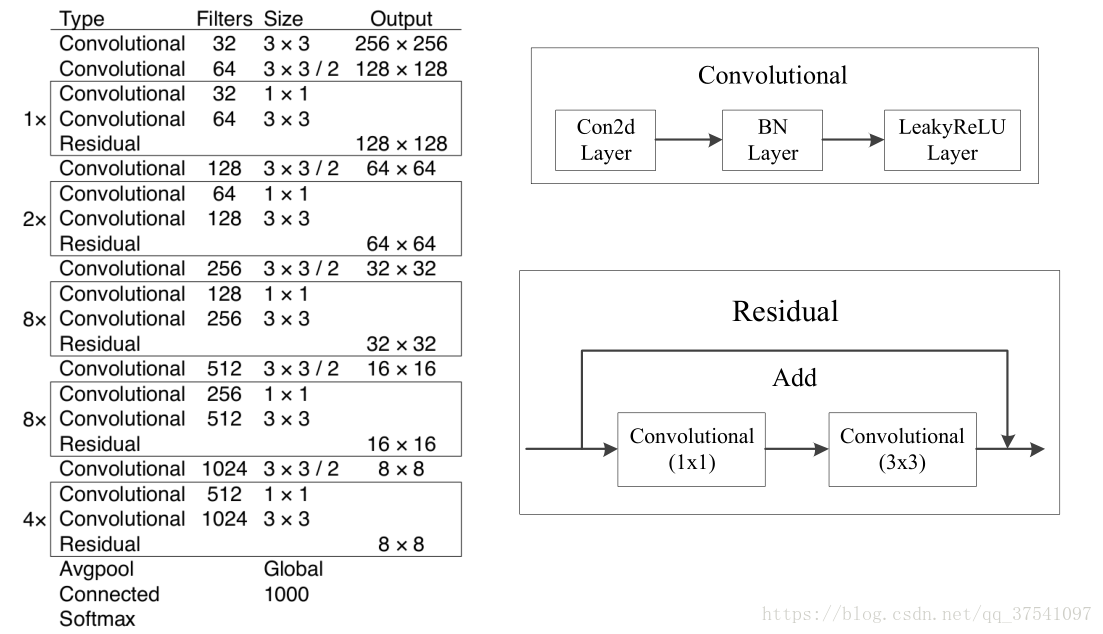
网络结构图：



Yolov3主要可分为3个部分：

1. Darknet-53结构：与Faster R-CNN网络中的特征提取网络功能相同，用于提取图像特征，从结构图中可以看到darknet53提取三个不同尺度的特征信息，这也为后续检测提供更多信息。

Darknet-53网络结构：



1. 特征层融合结构：通过1\*1、3\*3conv+BN+LeakyRelu层将Darknet输出的三个尺度特征信息进行总和，为后续分类回归任务提供信息。
2. 全连接层：与Faster R-CNN中类似。
3. **实验参数设置**

**Faster R-CNN实验**（使用bubbliiiing作者复现代码）：

（原作者代码地址：<https://github.com/bubbliiiing/faster-rcnn-pytorch）>

**卷积网络：Resnet50**

**预训练模型：**使用作者给定的voc\_weights\_resnet.pth训练模型

**训练方法：**将训练过程划分为冻结阶段和解冻阶段，冻结阶段特征提取网络不发生改变，网络主干冻结，使网络训练更加稳定，不至崩溃，后续解冻阶段，在冻结阶段基础上继续训练，提高精确度。

**优化器：**Adam（根据历史梯度震荡情况及过滤震荡后真实历史梯度进行变量更新，既能适应稀疏梯度，又能缓解梯度震荡）

**epoch：**Freeze\_Epoch=50, UnFreeze\_Epoch=100（这里UnFreeze\_Epoch代表总epoch数，即冻结阶段epoch=50，解冻阶段epoch=50）

**batchsize：**Freeze\_batch\_size=4, UnFreeze\_batch\_size=2（这里由于冻结阶段使用显存较小，故可设为4，事实上当解冻阶段batchsize设为4及其之上时，电脑显存就已经不够）

**Iteration**：625

**Num\_workers：**0（电脑显存限制）

**learning\_rate**：Init\_lr=1e-4（最大学习率）,Min\_lr=1e-6（最小学习率）

**学习率下降策略**：余弦下降策略

**Loss function**：RPN\_loss+Classification\_loss，其中RPNloss由rpn\_cls\_loss及rpn\_reg\_loss两部分组成，rpn\_cls\_loss使用CrossEntropy计算，rpn\_reg\_loss使用L1 Loss计算；Classification\_loss也由标签分类loss及boxloss组成，其中各类loss同RPNLoss中计算方式。

**训练时长：约15h**

**YOLOV3实验**（使用bubbliiiing作者复现代码）：

（原作者代码地址：https://github.com/bubbliiiing/yolo3-pytorch）

**预训练模型：**使用作者给定的yolo\_weights.pth及darknet53\_backbone\_weights.pth训练模型

**训练方法：**将训练过程划分为冻结阶段和解冻阶段，冻结阶段特征提取网络不发生改变，网络主干冻结，使网络训练更加稳定，不至崩溃，后续解冻阶段，在冻结阶段基础上继续训练，提高精确度。

**优化器：**SGD（随机梯度下降）

**epoch：**Freeze\_Epoch=50, UnFreeze\_Epoch=300（这里UnFreeze\_Epoch代表总epoch数，即冻结阶段epoch=50，解冻阶段epoch=250）

**batchsize：**Freeze\_batch\_size=16, UnFreeze\_batch\_size=8（这里由于冻结阶段使用显存较小，故可设为16，事实上当解冻阶段batchsize设为8及其之上时，电脑显存就已经不够）

**Iteration**：156

**Num\_workers：**0（电脑显存限制）

**learning\_rate**：Init\_lr=1e-2（最大学习率）,Min\_lr=1e-4（最小学习率）

**学习率下降策略**：余弦下降策略

**Loss function**：loc\_loss+conf\_loss+cla\_loss，（目标定义偏移量损失+目标置信度损失+目标分类损失，正样本计算）。其中loc\_loss使用Sum of Squared Error Loss，conf\_loss及cla\_loss均使用二值交叉熵损失。

**训练时长：约18h**

1. **评价指标**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **training data** | **test data** | **mAP** | **mIoU** |
| **Faster R-CNN**  **+Resnet50** | **VOC07+**  **VOC12** | **VOC12test** | **76.04%** | **0.4945** |
| **YOlO V3** | **VOC07+VOC12** | **VOC12test** | **86.64%** | **0.7253** |

从上述表格中可以看出，就本次实验训练得到的模型而言，yolov3训练得到模型的评价指标高于Faster R-CNN得到模型，其检测效果应较好。但考虑到训练时长及模型复杂度等条件，YOLOV3模型大小约240MB，Faster R-CNN模型大小约110MB，故可知Faster R-CNN更加轻便且迅捷，而YOLOV3则是准确率较高，检测效果较好。（同时Faster R-CNN训练时出现问题可能造成的影响，具体问题请见七）

1. **检测结果可视化**

检测图片均为网络下载图片（不在VOC数据集内），包括两张街道图以及一张猫狗图。

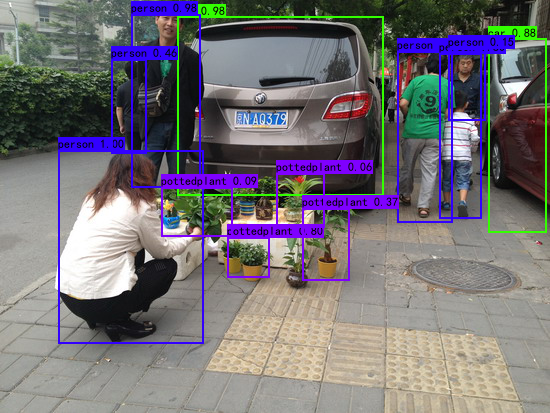
街道图1（从上到下分别为原图、FasterRCNN检测图、YOLOV3检测图）：



街道图2（从上到下分别为原图、FasterRCNN检测图、YOLOV3检测图）：

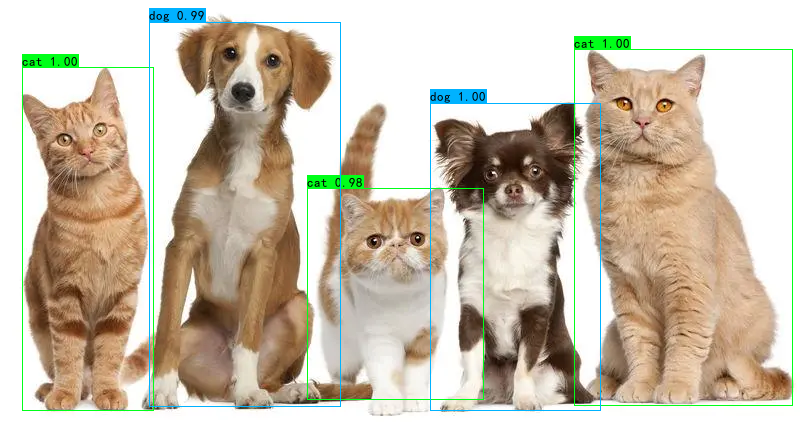


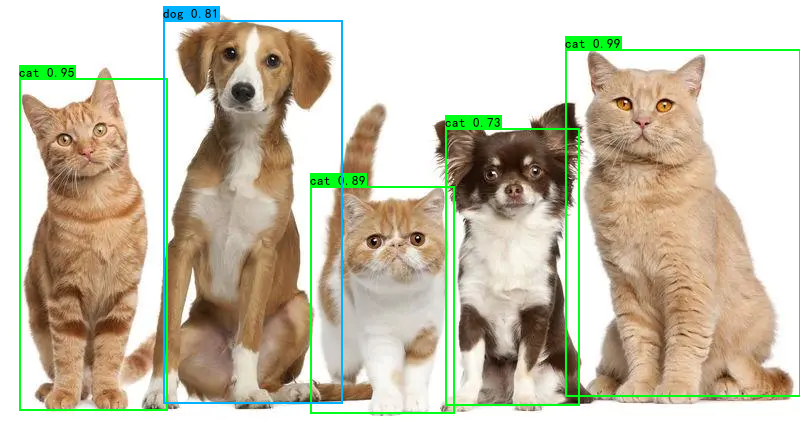




猫狗图3（从上到下分别为原图、FasterRCNN检测图、YOLOV3检测图）：



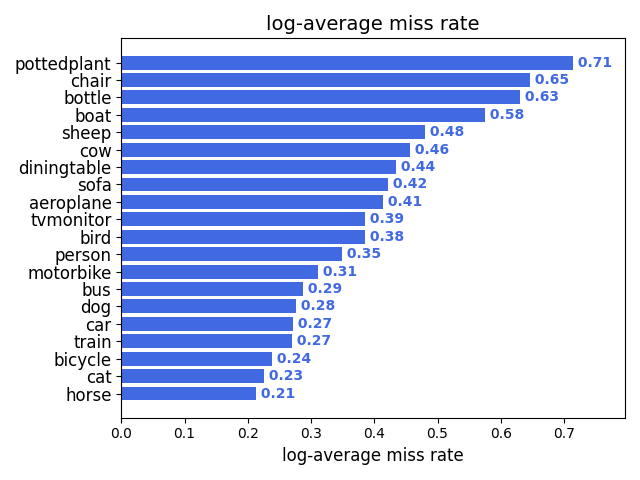
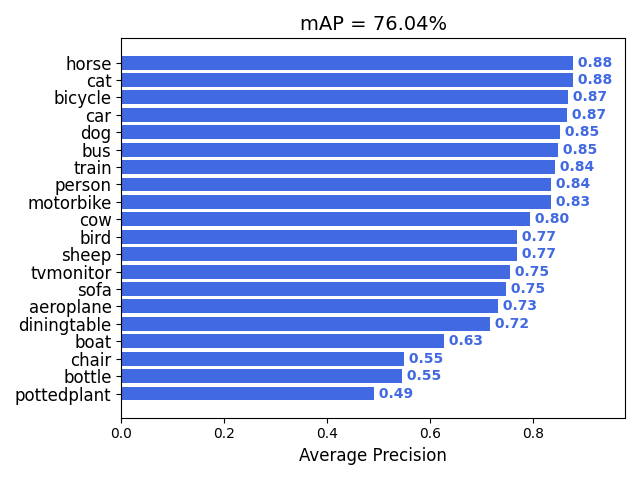




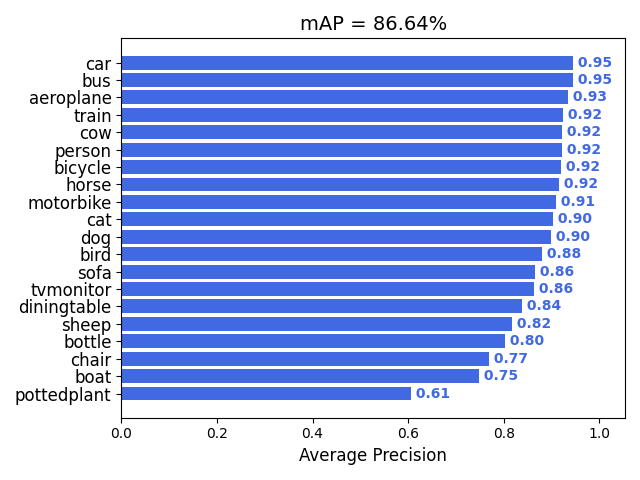
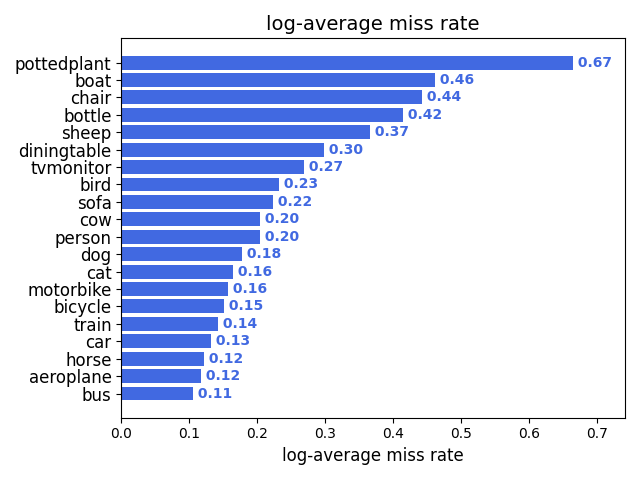
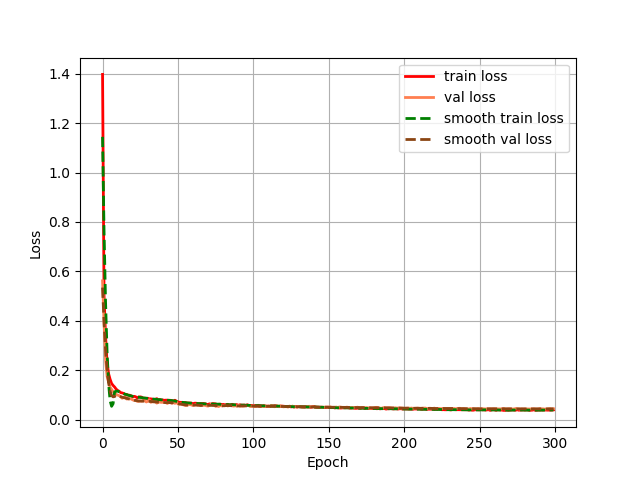
从上述可视化结果中可以看出，FasterRCNN检测结果proposal box数目更多，即检测目标更全，并且置信度普遍较高，即物体检测精度较高，而YOLO虽没有那么高的目标检测率，但其准确性更高，并不会出现误识别的现象，如街道一图中，FR把地上的盘子误检测分类为car，以及右侧房屋误识别为bus，街道二图中，FA将右侧不完整的红色汽车头部误检测分类为sofa，但YOLO并不会存在这样的问题，即检测准确性更高，分类准确率更高。而在猫狗图三中可以发现，二者检测框范围结果均较好，但YOLO存在将狗识别为猫的未知错误，其原因可能由于猫狗类图像较为相像，故虽然YOLO泛化性更好，但有时会导致严重的单体误分类以及置信度不高的问题。

1. **训练结果图**

**Faster R-CNN**(1各类检测图数，2平均丢失率，3mAP)

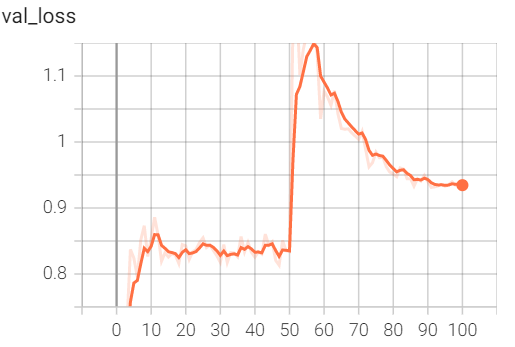
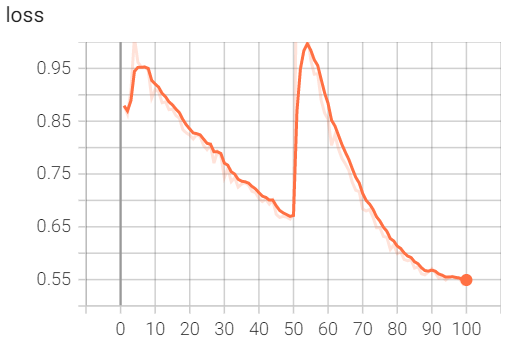
****

**YOLO V3**(1各类检测图数，2平均丢失率，3mAP，4loss)：

****

1. **训练中遇到的问题**

在使用原作者给定的预训练模型训练Faster RCNN时，train\_loss及val\_loss曲线图如下，可以看到解冻使得loss突升，并且后续训练时，虽然train\_loss和val\_loss持续下降，并且train\_loss比之前下降了超过三十个百分点，但是val\_loss自解冻提升后，永远下降不到原先的低loss值，故训练效果不如预训练模型好，该现象可能是由于过拟合导致。即原作者提供的预训练模型效果已经很好，再次训练导致其破坏了原先收敛到的极小值点，故应选择其他数据集预训练得到的模型或更加原始的模型。

****

1. **代码及模型地址**

代码地址：https://github.com/theWaWang/Mid\_work

模型地址：链接：https://pan.baidu.com/s/1vLqMysZgVP4qv2MfEz5\_TQ

提取码：2fwn