

计算机视觉期中作业

王逸群 19307110397 程子琛 19307110417

2022.4-5

GitHub repo 链接: <https://github.com/quniLcs/cv-mid>

网盘链接: [百度网盘计算机视觉期中作业](#)

1 ResNet

1.1 数据集

本项目使用 CIFAR-100 数据集，其中包含 60000 张 32×32 的彩色图片，其中训练集 50000 张，测试集 10000 张，被平均分为 100 类。

1.2 网络结构

本项目使用 ResNet-18 网络结构，其中激活函数为 ReLU，最大的特征为残差连接。后者包括两种单元结构如图1 和图2所示。

对于输入的图像，先进行步长为 1 的 $3 \times 64 \times 3 \times 3$ 卷积操作，并进行批归一化和激活，维度变为 $64 \times 32 \times 32$ ；接着通过两次第一种单元结构，维度不变；再通过第二种单元结构，维度变为 $128 \times 16 \times 16$ ；再通过第一种单元结构，维度不变；再通过第二种单元结构，维度变为 $256 \times 8 \times 8$ ；再通过第一种单元结构，维度不变；再通过第二种单元结构，维度变为 $512 \times 4 \times 4$ ；再通过第一种单元结构，维度不变；最后通过全连接得到输出。

1.3 超参数设置

参数初始化： MSRA；

学习率：由 0.1 开始每 10 个回合阶梯下降一个数量级；

优化器：带有 0.9 动量的随机梯度下降算法；

正则化参数： 0.0005；

回合数： 40；

批量大小： 128；

每回合循环数： 391；

总循环数： $40 \times 391 = 15640$ ；

损失函数： 交叉熵损失函数；

评价指标： 精确度。

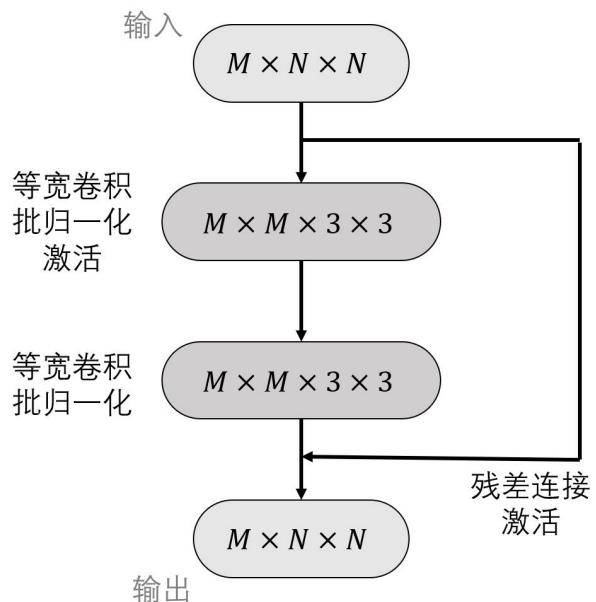


图 1: 残差连接第一种单元结构

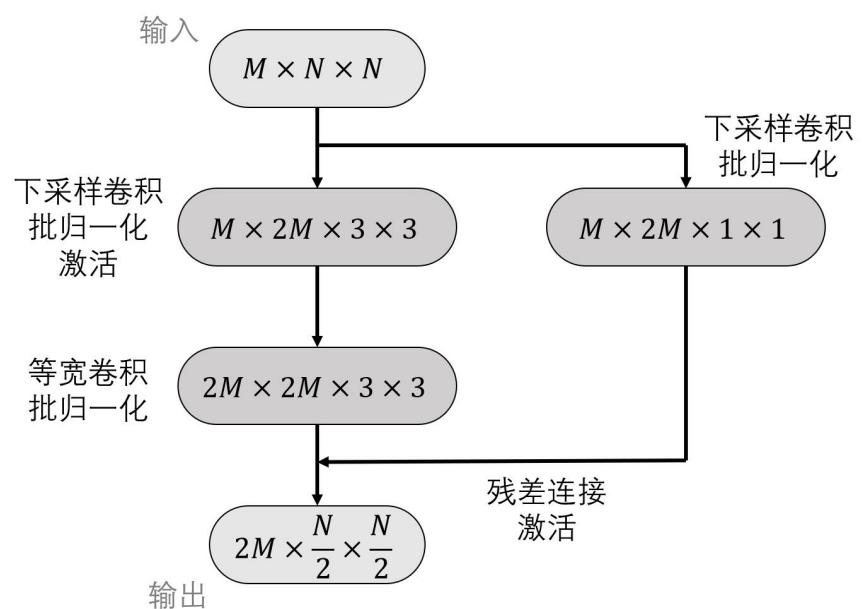


图 2: 残差连接第二种单元结构

1.4 数据增强

1.4.1 Cutout

该数据增强方法对于输入的图像，随机选取一点作为中心点，将其周围的正方形区域置为 0。具体效果如图3所示。

1.4.2 Mixup

该数据增强方法对一对输入的图像及标签进行凸组合，凸组合系数服从 Beta 分布。具体操作为，对样本 x_i 、 y_i 、 x_j 、 y_j ，凸组合系数 $\lambda \sim Beta(\alpha, \alpha)$ ，产生新的样本和标签：

$$x = \lambda x_i + (1 - \lambda)x_j$$
$$y = \lambda y_i + (1 - \lambda)y_j$$

具体效果如图3所示。

1.4.3 Cutmix

该数据增强方法结合了以上两种方法。具体操作为，对样本 x_i 、 y_i 、 x_j 、 y_j ，凸组合系数 $\lambda \sim Beta(\alpha, \alpha)$ ，先从 $H \times W$ 的样本 x_i 中随机选取一点作为中心点，将其周围 $H\sqrt{1-\lambda} \times W\sqrt{1-\lambda}$ 的正方形区域置为样本 x_j 的值，即正方形区域面积占比为 $1 - \lambda$ 。由于实际正方形区域可能超出样本区域，最后将 λ 修正为保留原样本值的区域面积占比，并产生新的标签：

$$y = \lambda y_i + (1 - \lambda)y_j$$

具体效果如图3所示。



图 3: 数据增强效果

1.5 实验结果

实验结果如图4和表1所示。

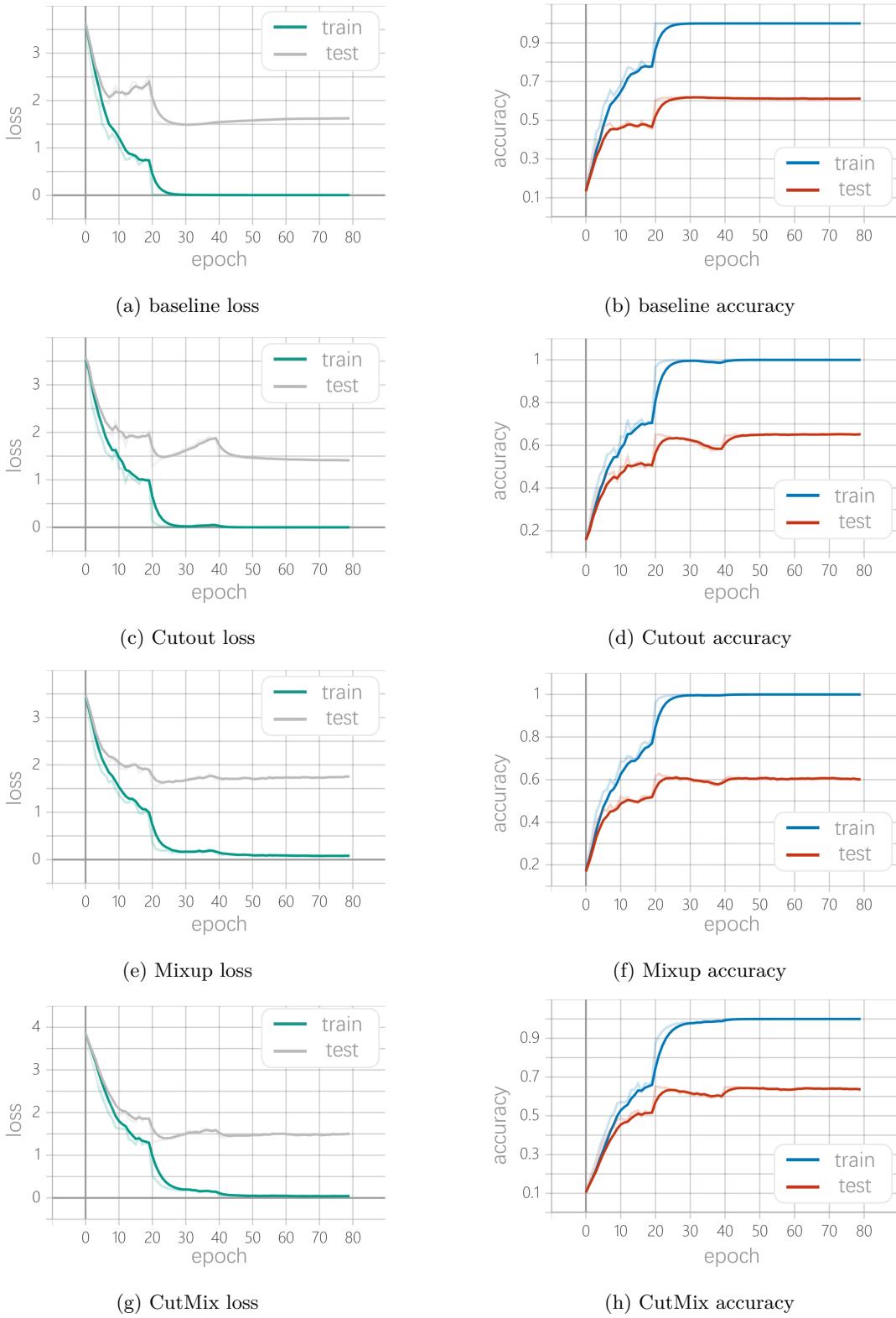


图 4: 实验结果

模型	训练集 top1 精确度	训练集 top5 精确度	测试集 top1 精确度	测试集 top5 精确度
baseline	0.99982	1.00000	0.61110	0.84930
Cutout	0.99982	1.00000	0.65360	0.87540
Mixup	0.99980	1.00000	0.60050	0.81830
CutMix	0.99980	1.00000	0.63310	0.85980

表 1: 实验结果

可以看到，模型训练的难点在于，训练集的精确度快速达到近乎 100%，测试集的精确度便不再有上升的空间。数据增强的意义在于，加大训练集收敛的难度，使得测试集有更大的提升机会。

例如，在第 15 个回合，baseline 模型的训练集精确率达到了 0.75324，但测试集精确率仅有 0.46340；相比之下，cutout 模型的训练集精确率仅有 0.69558，但测试集精确率有 0.51210。

最后，mixup 对模型没有明显提升，cutmix 和 cutout 都优化了模型结果。

2 目标检测

2.1 数据集

本项目基于 VOC 数据集进行训练和测试，其中，Faster R-CNN 网络使用 VOC2007 的训练（验证）和测试集，YOLOv3 网络使用 VOC2007、VOC2012 的训练（验证）集和 VOC2007 测试集（VOC2012 测试集未公开）。

数据集包括四大类，细分至 20 小类，VOC2007 数据集 Train/validation/test 共有 9963 张图片，包含 24640 个已标注的目标 objects；VOC2012 用于检测的数据集规模为：train/val : 11540 张图片，包含 27450 个已被标注的 ROI annotated objects。

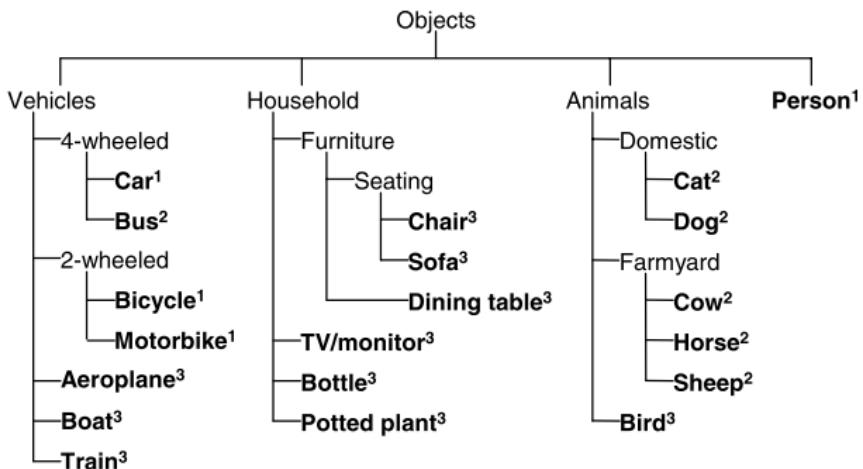


图 5: VOC 数据集分类

2.2 网络结构

2.2.1 Faster R-CNN

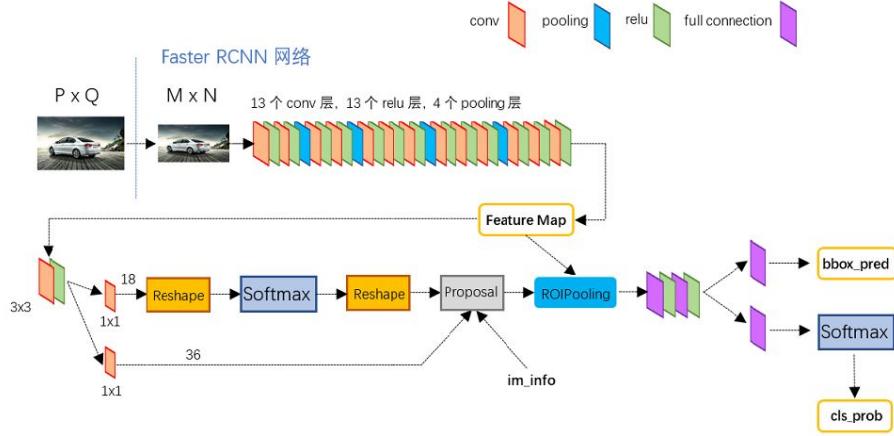


图 6: Faster R-CNN 网络架构

2.2.2 YOLOv3

layer	filters	size	input	output	FLOPs
0 conv	32	3 x 3 / 1	416 x 416 x 3	416 x 416 x 32	0.299 BFLOPs
1 conv	64	3 x 3 / 2	416 x 416 x 32	208 x 208 x 64	1.595 BFLOPs
2 conv	3	1 x 1 / 1	208 x 208 x 64	208 x 208 x 32	0.177 BFLOPs
3 conv	64	3 x 3 / 1	208 x 208 x 32	208 x 208 x 64	1.595 BFLOPs
4 res	1		208 x 208 x 64	208 x 208 x 64	
5 conv	128	3 x 3 / 2	208 x 208 x 64	104 x 104 x 128	1.595 BFLOPs
6 conv	64	1 x 1 / 1	104 x 104 x 128	104 x 104 x 64	0.177 BFLOPs
7 conv	128	3 x 3 / 1	104 x 104 x 64	104 x 104 x 128	1.595 BFLOPs
8 res	5		104 x 104 x 128	104 x 104 x 128	
9 conv	64	1 x 1 / 1	104 x 104 x 128	104 x 104 x 64	0.177 BFLOPs
10 conv	128	3 x 3 / 1	104 x 104 x 64	104 x 104 x 128	1.595 BFLOPs
11 res	8		104 x 104 x 128	104 x 104 x 128	
12 conv	256	3 x 3 / 2	104 x 104 x 128	52 x 52 x 256	1.595 BFLOPs
13 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
14 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
15 res	12		52 x 52 x 256	52 x 52 x 256	
16 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
17 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
18 res	15		52 x 52 x 256	52 x 52 x 256	
19 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
20 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
21 res	18		52 x 52 x 256	52 x 52 x 256	
22 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs

layer	filters	size	input	output	FLOPs
23 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
24 res	21		52 x 52 x 256	52 x 52 x 256	
25 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
26 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
27 res	24		52 x 52 x 256	52 x 52 x 256	
28 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
29 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
30 res	27		52 x 52 x 256	52 x 52 x 256	
31 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
32 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
33 res	30		52 x 52 x 256	52 x 52 x 256	
34 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
35 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
36 res	33		52 x 52 x 256	52 x 52 x 256	
37 conv	512	3 x 3 / 2	52 x 52 x 256	26 x 26 x 512	1.595 BFLOPs
38 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
39 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
40 res	37		26 x 26 x 512	26 x 26 x 512	
41 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
42 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
43 res	40		26 x 26 x 512	26 x 26 x 512	
44 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
45 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
46 res	43		26 x 26 x 512	26 x 26 x 512	
47 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
48 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
49 res	46		26 x 26 x 512	26 x 26 x 512	
50 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
51 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
52 res	49		26 x 26 x 512	26 x 26 x 512	
53 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
54 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
55 res	52		26 x 26 x 512	26 x 26 x 512	
56 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
57 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
58 res	55		26 x 26 x 512	26 x 26 x 512	
59 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
60 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
61 res	58		26 x 26 x 512	26 x 26 x 512	
62 conv	1024	3 x 3 / 2	26 x 26 x 512	13 x 13 x 1024	1.595 BFLOPs

layer	filters	size	input	output	FLOPs
63 conv	512	1 x 1 / 1	13 x 13 x1024	13 x 13 x 512	0.177 BFLOPs
64 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x1024	1.595 BFLOPs
65 res	62		13 x 13 x1024	13 x 13 x1024	
66 conv	512	1 x 1 / 1	13 x 13 x1024	13 x 13 x 512	0.177 BFLOPs
67 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x1024	1.595 BFLOPs
68 res	65		13 x 13 x1024	13 x 13 x1024	
69 conv	512	1 x 1 / 1	13 x 13 x1024	13 x 13 x 512	0.177 BFLOPs
70 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x1024	1.595 BFLOPs
71 res	68		13 x 13 x1024	13 x 13 x1024	
72 conv	512	1 x 1 / 1	13 x 13 x1024	13 x 13 x 512	0.177 BFLOPs
73 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x1024	1.595 BFLOPs
74 res	71		13 x 13 x1024	13 x 13 x1024	
75 conv	512	1 x 1 / 1	13 x 13 x1024	13 x 13 x 512	0.177 BFLOPs
76 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x1024	1.595 BFLOPs
77 conv	512	1 x 1 / 1	13 x 13 x1024	13 x 13 x 512	0.177 BFLOPs
78 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x1024	1.595 BFLOPs
79 conv	512	1 x 1 / 1	13 x 13 x1024	13 x 13 x 512	0.177 BFLOPs
80 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x1024	1.595 BFLOPs
81 conv	75	1 x 1 / 1	13 x 13 x1024	3 x 13 x 75	0.026 BFLOPs
82 yolo					
83 route	79				
84 conv	256	1 x 1 / 1	13 x 13 x 512	13 x 13 x 256	0.044 BFLOPs
87 conv	256	1 x 1 / 1	26 x 26 x 768	26 x 26 x 256	0.266 BFLOPs
88 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
89 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
90 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
91 conv	256	1 x 1 / 1	26 x 26 x 512	26 x 26 x 256	0.177 BFLOPs
92 conv	512	3 x 3 / 1	26 x 26 x 256	26 x 26 x 512	1.595 BFLOPs
93 conv	75	1 x 1 / 1	26 x 26 x 512	26 x 26 x 75	0.052 BFLOPs
94 yolo					
95 route	91				
96 conv	128	1 x 1 / 1	26 x 26 x 256	26 x 26 x 128	0.044 BFLOPs
97 upsample		2x	26 x 26 x 128	52 x 52 x 128	
98 route	97 36				
99 conv	128	1 x 1 / 1	52 x 52 x 384	52 x 52 x 128	0.266 BFLOPs
100 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
101 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
102 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs
103 conv	128	1 x 1 / 1	52 x 52 x 256	52 x 52 x 128	0.177 BFLOPs
104 conv	256	3 x 3 / 1	52 x 52 x 128	52 x 52 x 256	1.595 BFLOPs

layer	filters	size	input	output	FLOPs
105 conv	75	1 x 1 / 1	52 x 52 x 256	52 x 52 x 75	0.104 BFLOPs
106 yolo					

表 2: YOLOv3 网络架构

2.3 超参数设置

2.3.1 Faster R-CNN

参数初始化：预训练模型 vgg16；

学习率：初始 0.001，第 9 回合后变为 0.0001 并保持；

优化器：带有 0.9 动量的随机梯度下降算法；

正则化参数：0.0005；

回合数：14；

批量大小：1；

总循环数： $14 \times 5011 = 70154$ ；

损失函数：交叉熵损失函数；

评价指标：mAP。

2.3.2 YOLOv3

参数初始化：darknet53.conv.74；

学习率：采用线性 warmup 机制，由 0 开始，至 1000 批次上升为 0.001 并保持不变；

优化器：带有 0.9 动量的随机梯度下降算法；

正则化参数：0.0005；

批量大小：64；

总循环数：23620；

损失函数：二元交叉熵函数；

评价指标：mAP & mIoU。

2.4 训练

2.4.1 Faster R-CNN

基于[GitHub 上的 Simple Faster-RCNN 实现](#)。

使用 VOC2007 数据集进行训练和测试，将数据集解压至VOCdevkit文件夹。

修改代码文件：`utils/config.py`修改参数`voc_data_dir`为数据集对应路径；修改预训练模型路径以加载预训练参数。

源代码可视化部分基于visdom实现，对`trainer.py`文件`train_step`函数做调整，以输出训练过程 loss 曲线。

开始训练：运行`model_train.ipynb`中`train()`函数。

训练可视化：训练日志记录各批次的训练误差，通过`visualization.py`和Tensorboard可视化结果如下。

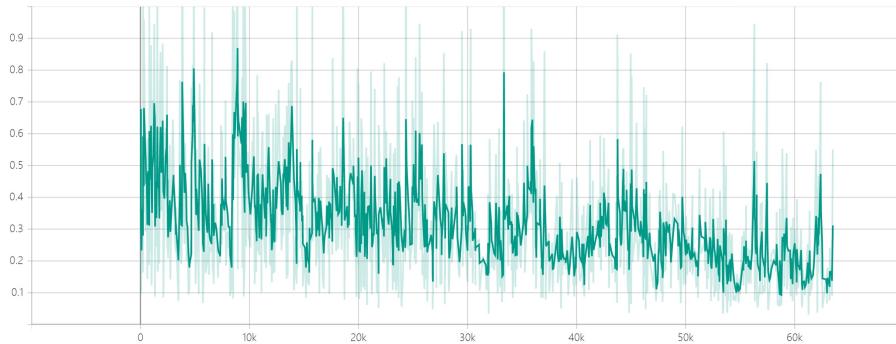


图 7: 训练误差

2.4.2 YOLOv3

模型训练基于[官网源码](#)实现。

首先进行数据预处理：解压 VOC2007、VOC2012 数据集，运行`python voc_label.py`，将 2007 年的训练和验证图像和 2012 年的图像放入`train.txt`用于集中训练。

修改配置文件：`cfg/voc.data`确定需要检测的目标类别数量和名称，修改训练集和验证集的图片路径。

`yolov3-voc.cfg`定义 YOLO 的卷积神经网络模型，和超参数配置等。考虑到内存大小，训练时设置 batch-size 为 64，subvisions 为 16，测试时将 batch-size 和 subvisions 同时设为 1。

代码修改：原版的`.weights`文件当训练次数小于 1000 时，每隔 100 次保存一次，大于 1000 每 10000 次保存一次，为更好记录训练过程，将`darknet/example/detector.c`修改为每 250 次迭代保存一次。

开始训练：采用`darknet53.conv.74`权重进行训练，运行以下代码训练并记录训练日志：

```
./darknet detector train cfg/voc.data cfg/yolov3-voc.cfg \
darknet53.conv.74 | -gpu 0 tee train_yolov3.log
```

一个批次的日志输出解读如下：每组包含三条信息，分别是：Region 82, Region 94, Region 106。

每行参数中：

- **Avg IOU:** 当前迭代中，预测的 box 与标注的 box 的平均交并比，越大越好，期望数值为 1；
- **Class:** 标注物体的分类准确率，越大越好，期望数值为 1；
- **.5R:** 以 $\text{IOU}=0.5$ 为阈值时候的召回率，即检出的正样本与实际的正样本的比值；
- **0.75R:** 以 $\text{IOU}=0.75$ 为阈值时候的召回率；
- **count:** 正样本数目。

每个批次最后输出行代表批次数，总损失，平均损失，当前学习率，当前批次训练时间，目前为止参与训练的图片总数。

训练可视化：训练日志记录各批次的训练误差，通过`visualization.py`和Tensorboard可视化结果如下。

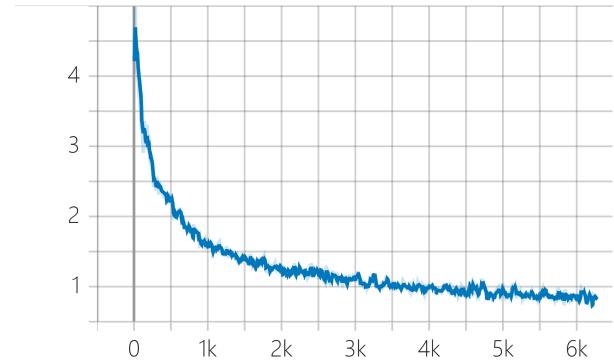


图 8: 训练误差

2.5 测试

以下为不在 VOC 数据集内，但包含有 VOC 中类别物体的图像检测结果。

分别运行model_test.ipynb和执行命令对图片进行测试。

```
./darknet detector test cfg/voc.data cfg/yolov3-voc.cfg \
backup/yolov3-voc_23500.weights <path/to/image>
```

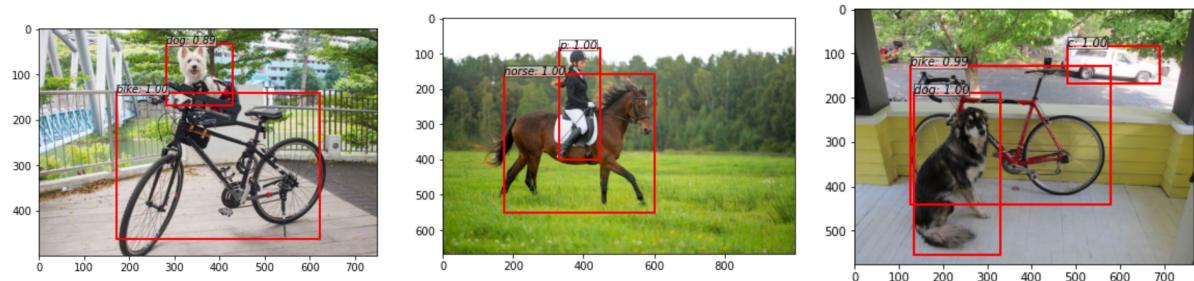


图 9: Faster R-CNN 测试结果

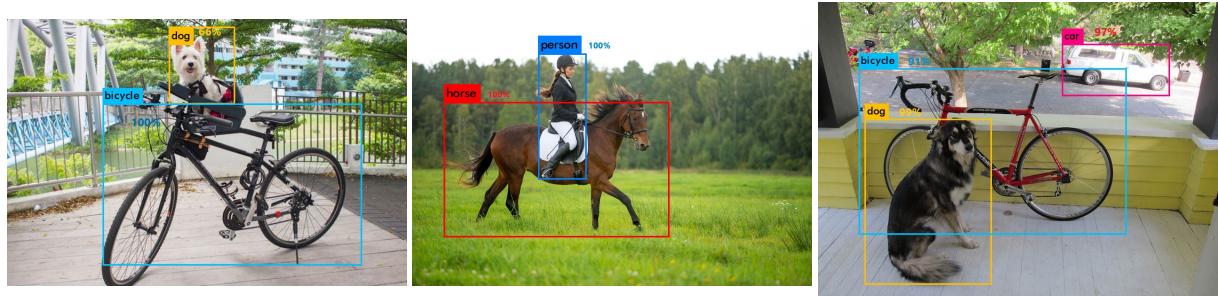


图 10: YOLOv3 测试结果

2.5.1 Faster R-CNN

在 4 张测试图片上可视化 Faster R-CNN 第一阶段的 proposal box 结果如下。

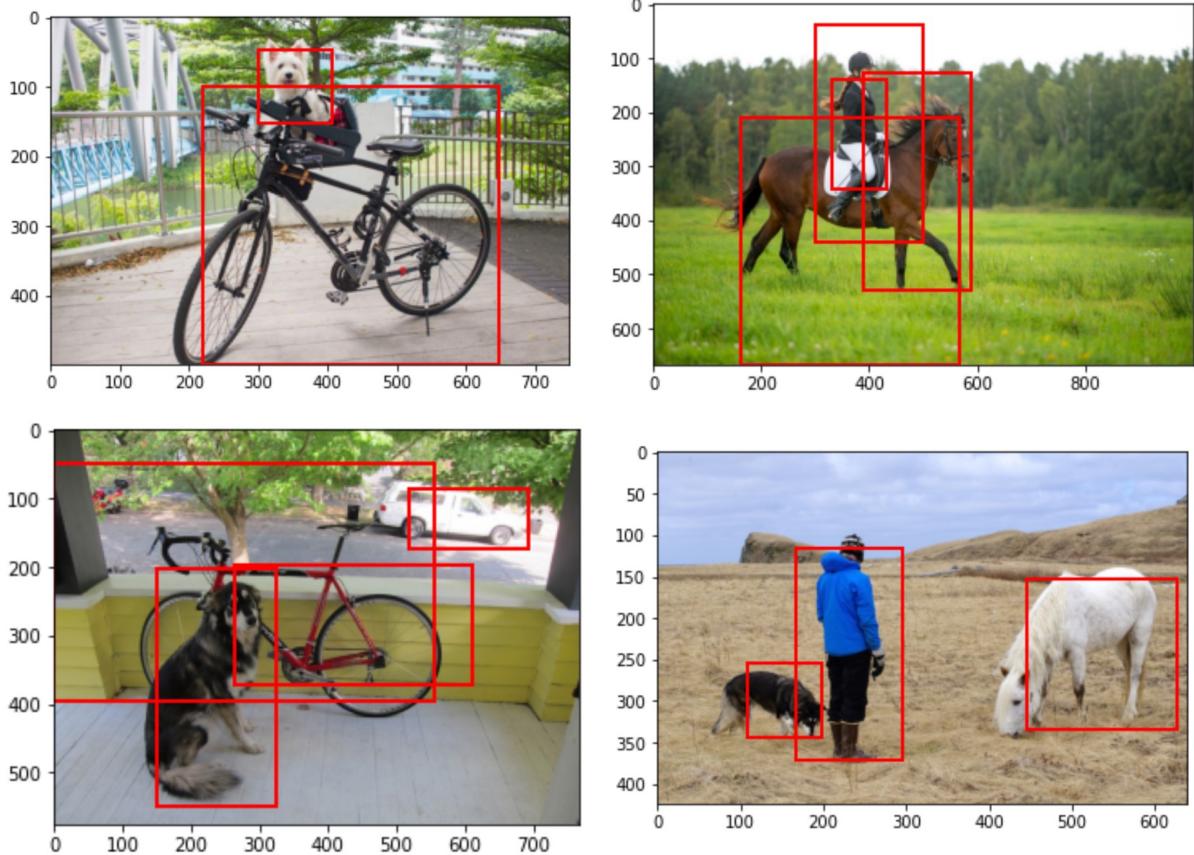


图 11: stage-1 proposal box

`model/util/bbox_tools.py` 文件中 `loc2bbox(src_bbox, loc)` 函数在已知源 `bbox` 和位置偏差 `dx`, `dy`, `dh`, `dw` 的情况下返回调整后的目标框 `G`。

第一阶段得到的 proposal box 和最终返回的目标框 `G` 之间转换关系如下图所示。经转换后的最终结果见上方测试结果部分。

$$\begin{aligned} \hat{G}_x &= P_w d_x(P) + P_x & \hat{G}_w &= P_w \exp(d_w(P)) \\ \hat{G}_y &= P_h d_y(P) + P_y & \hat{G}_h &= P_h \exp(d_h(P)) \end{aligned}$$

图 12: proposal box 调整

利用 `Tensorboard` 可可视化测试 mAP 曲线如下。在第 10 个回合处，训练模型回溯至此前各个回合训练结束后 mAP 最高的模型继续训练，因此此处 mAP 曲线呈现明显的下降-上升趋势。

测试损失函数曲线如下，因为测试样本较小，所以误差下降不明显，但从模型在测试集的 mAP 表现来看，训练效果对模型改进仍比较显著。

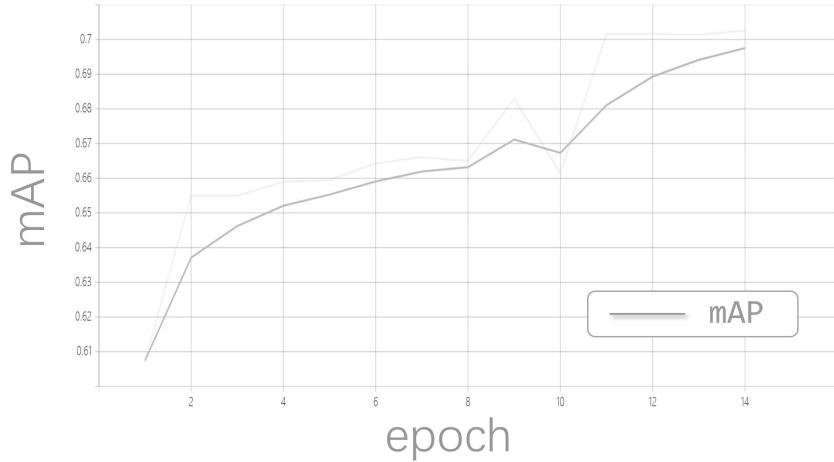


图 13: Faster R-CNN 测试 mAP 曲线

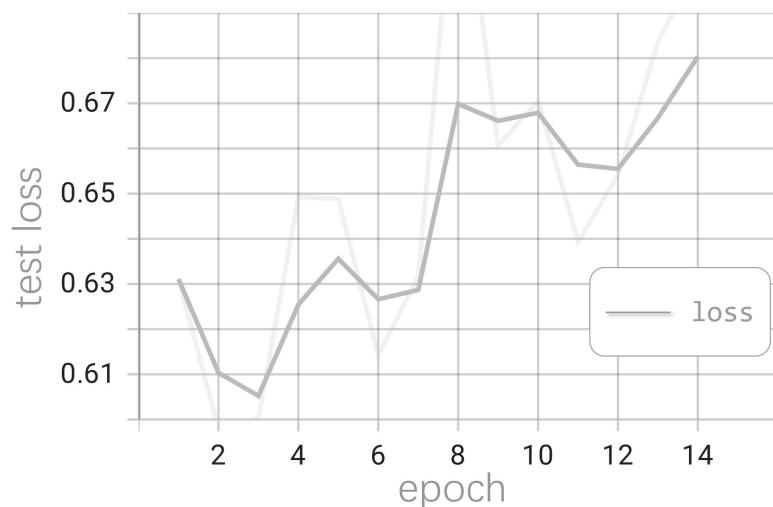


图 14: Faster R-CNN 测试损失函数曲线

2.5.2 YOLOv3

通过`valid`命令对整个测试集进行测试,所得结果储存在`results`文件夹下,调用`compute/compute_mAP.py`文件可得各类别 AP 值和 mAP 值。

通过`recall`命令对测试集进行测试, 可得测试集 mIoU 值。

利用Tensorboard可视化测试 mAP、mIoU 曲线如下。

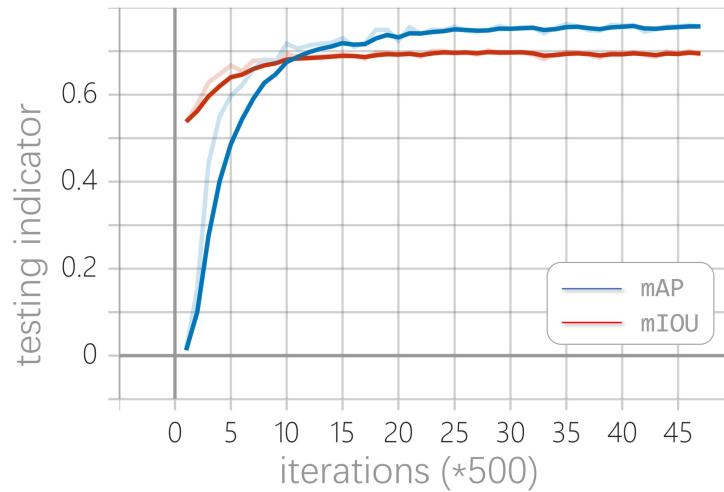


图 15: YOLOv3 测试曲线