

Midterm Project - Report

张千一 18302010032 王开阳 19307130069

2022 年 5 月 14 日

目录

1 在 CIFAR-100 数据集上训练 CNN	3
1.1 数据集介绍与划分	3
1.2 baseline 模型	4
1.2.1 模型基本结构	4
1.2.2 模型超参数调整	6
1.3 数据增强	6
1.3.1 随机裁剪和水平翻转	6
1.3.2 Cut out	7
1.3.3 Cut mix	9
1.3.4 Mix up	11
1.4 小结	12
2 在 VOC 数据集上训练 Faster R-CNN 和 YOLO V3	13
2.1 数据集介绍与划分	13
2.1.1 数据集总体概况	13
2.1.2 数据集划分	13
2.1.3 数据集可视化	15
2.1.4 读取 VOC 数据集	15
2.2 Faster R-CNN 模型	16
2.2.1 基本结构	16
2.2.2 特征金字塔网络 (FPN)	17
2.2.3 候选检测框生成网络 (RPN)	18
2.2.4 模型训练	19
2.2.5 模型评价	20
2.2.6 模型可视化	20
2.3 YOLO V3 模型	22
2.3.1 YOLO V3 基本结构	22
2.3.2 YOLO V3 的 anchor box	23
2.3.3 YOLO V3 损失函数	24
2.3.4 YOLO V3 SPP	25

2.3.5 模型训练	25
2.3.6 模型评价	26
2.3.7 模型可视化	27
2.4 小结	28
3 代码链接和模型下载地址	28

1 在 CIFAR-100 数据集上训练 CNN

1.1 数据集介绍与划分

CIFAR-100 数据集共有 60000 张带标签图像，这些图像被分为 100 个类，且类之间完全互斥。其中每个类有 600 张大小为 $32 \times 32 \times 3$ 的 RGB 彩色图像，500 张作为训练集，100 张作为测试集。对于每一张图像，它有 fine_labels（细粒度）和 coarse_labels（粗粒度）两个标签，对应图 1 中的 Classes 和 Superclass：

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

图 1: CIFAR-100 数据集的标签

下面展示了 CIFAR-100 训练集中的 9 张图像：

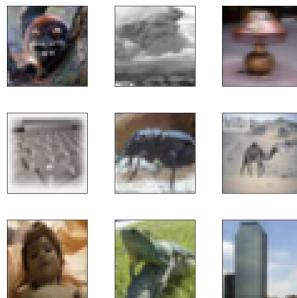


图 2: CIFAR-100 数据集的图像示例

1.2 baseline 模型

1.2.1 模型基本结构

残差网络 (ResNet) 由微软研究院的何恺明、张祥雨、任少卿、孙剑等人在 Deep Residual Learning for Image Recognition 一文中提出，在 2015 年的 ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 中取得了冠军。ResNet 认为，理论上，可以训练一个相对浅层的网络，然后在这个训练好的相对浅层的网络上堆几层恒等映射层，即输出等于输入的层，构建出一个更深的网络。这两个网络得到的结果应该是一模一样的，因而在训练集上，深层的网络不会比浅层的网络效果差。

ResNet 通过加入 shortcut connections，变得更加容易被优化。包含一个 shortcut connection 的几层网络被称为一个残差块 (residual block)，如图 3 所示。

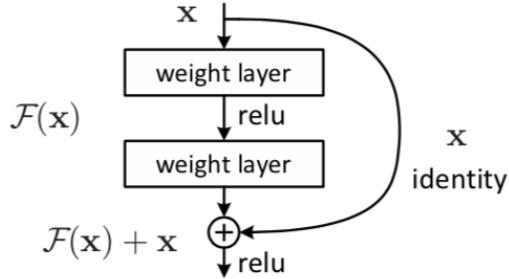


图 3: 残差块

如图 4 所示，展示了 ResNet 的几种结构：

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
3×3 max pool, stride 2						
conv2_x	56×56	$\left[\begin{array}{l} 3 \times 3, 64 \\ 3 \times 3, 64 \end{array} \right] \times 2$	$\left[\begin{array}{l} 3 \times 3, 64 \\ 3 \times 3, 64 \end{array} \right] \times 3$	$\left[\begin{array}{l} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{array} \right] \times 3$	$\left[\begin{array}{l} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{array} \right] \times 3$	$\left[\begin{array}{l} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{array} \right] \times 3$
conv3_x	28×28	$\left[\begin{array}{l} 3 \times 3, 128 \\ 3 \times 3, 128 \end{array} \right] \times 2$	$\left[\begin{array}{l} 3 \times 3, 128 \\ 3 \times 3, 128 \end{array} \right] \times 4$	$\left[\begin{array}{l} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{array} \right] \times 4$	$\left[\begin{array}{l} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{array} \right] \times 4$	$\left[\begin{array}{l} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{array} \right] \times 8$
conv4_x	14×14	$\left[\begin{array}{l} 3 \times 3, 256 \\ 3 \times 3, 256 \end{array} \right] \times 2$	$\left[\begin{array}{l} 3 \times 3, 256 \\ 3 \times 3, 256 \end{array} \right] \times 6$	$\left[\begin{array}{l} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{array} \right] \times 6$	$\left[\begin{array}{l} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{array} \right] \times 23$	$\left[\begin{array}{l} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{array} \right] \times 36$
conv5_x	7×7	$\left[\begin{array}{l} 3 \times 3, 512 \\ 3 \times 3, 512 \end{array} \right] \times 2$	$\left[\begin{array}{l} 3 \times 3, 512 \\ 3 \times 3, 512 \end{array} \right] \times 3$	$\left[\begin{array}{l} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{array} \right] \times 3$	$\left[\begin{array}{l} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{array} \right] \times 3$	$\left[\begin{array}{l} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{array} \right] \times 3$
average pool, 1000-d fc, softmax						
FLOPs	1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9	

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

图 4: ResNet 的几种结构

其中，ResNet-18 和 ResNet-34 使用两个 3×3 的卷积层作为一个块，而 ResNet-50，ResNet-101 和 ResNet-152 将其替换为 $1 \times 1 + 3 \times 3 + 1 \times 1$ 的卷积进行计算优化，这样的块即减少了计算量又能够保持精度，被称为 BottleNeck 块。

另外，为了使得输入和输出保持相同的维度，若特征图维度不同，对于卷积层的残差块，需要在 1×1 卷积后添加批归一化 (Batch Normalization) 处理，如图 5 所示。

考虑到训练的效率，我们使用 ResNet-18 作为 baseline 模型。由于 CIFAR-100 的图像较小，ResNet-18 中的第一层卷积使用 7×7 卷积核可能过大，难以提取 CIFAR-100 图像的特征，我们将其调整为 3×3 卷积核。此外，模型的其它的 baseline 如下：

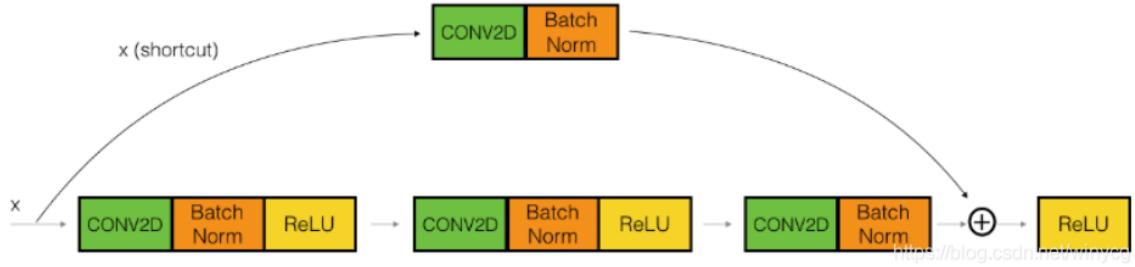


图 5: 批归一化

- epoch: 100
- batch size: 100
- drop out: 不使用
- 损失函数: 交叉熵损失
- 正则化: l_2 正则化, 正则化参数 $\lambda = 10^{-5}$
- 优化器: Adam
- 动量系数: $\beta_1 = 0.9, \beta_2 = 0.999$
- 初始学习率: $\alpha = 10^{-3}$
- 学习率衰减: 每个 epoch 后学习率衰减为上一个 epoch 的 0.98

我们得到 baseline 模型在测试集上的平均误差为 **2.455**, 分类精度为 **0.604**。模型的性能曲线如下所示:

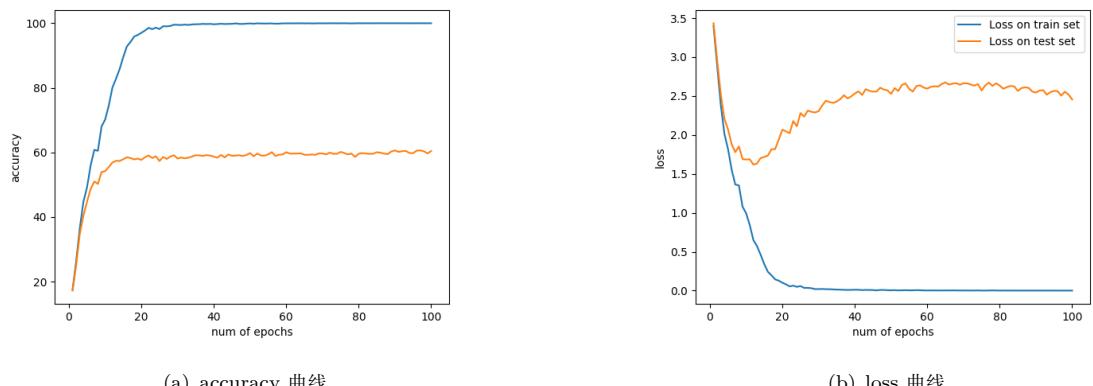


图 6: baseline 模型的性能曲线

可以发现, 模型的精度较低, 且模型存在较严重的振荡和过拟合现象。

1.2.2 模型超参数调整

为了缓解模型的过拟合现象，我们将 baseline 模型的超参数进行调整，以得到较高的精度。调整后模型的超参数如下：

- epoch: 100
- batch size: 100
- drop out: 使用
- 损失函数: 交叉熵损失
- 正则化: l_2 正则化, 正则化参数 $\lambda = 10^{-3}$
- 优化器: SGD
- 动量系数: $\beta = 0.9$
- 初始学习率: $\alpha = 10^{-2}$
- 学习率衰减: 每个 epoch 后学习率衰减为上一个 epoch 的 0.98

经过调整后，我们得到模型在测试集上的平均误差为 **1.317**，分类精度为 **0.672**。模型的性能曲线如下所示：

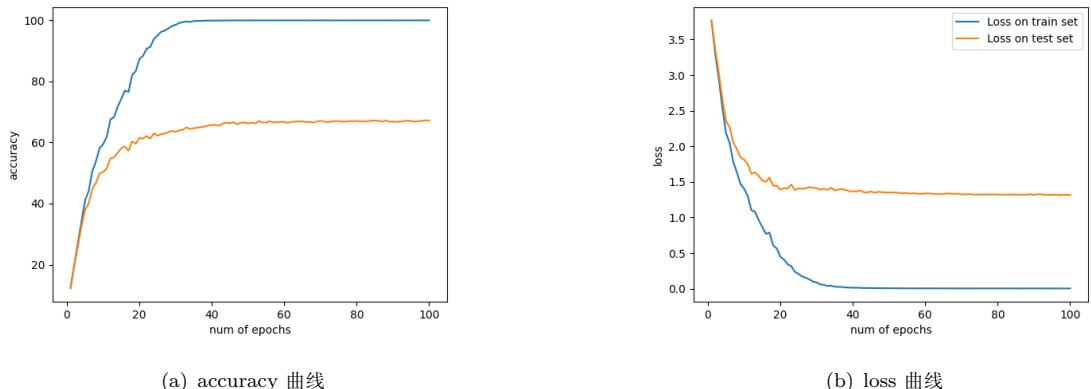


图 7: 调整后的性能曲线

可以发现，模型的过拟合现象得到了一定程度的改善，精度得到了提高，且不再出现测试集上 loss 随训练过程增加的现象。

1.3 数据增强

1.3.1 随机裁剪和水平翻转

我们可以对模型的数据作一些简单的数据增强。基于 torchvision 自带的 transforms.RandomCrop 和 transforms.RandomHorizontalFlip 函数，我们可以简单地实现对训练集图像进行向四周填充后随机裁剪以及随机翻转的操作。这两种操作不会对图像的基本信息造成太大影响。

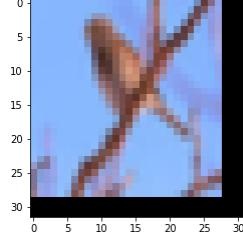
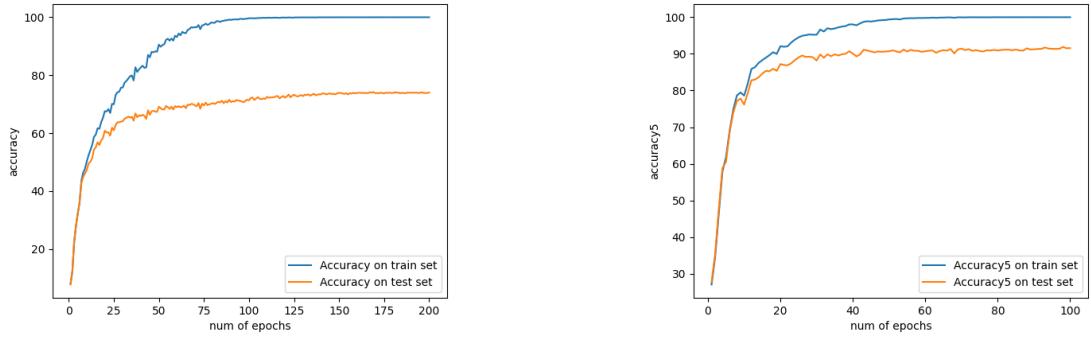
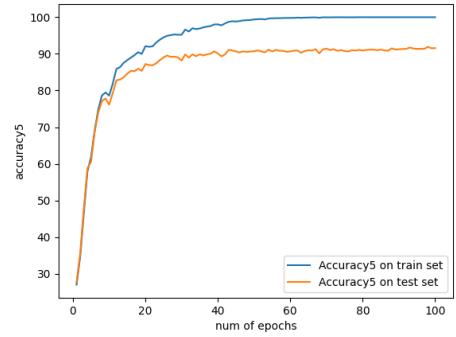


图 8: 向四周填充后随机裁剪的效果

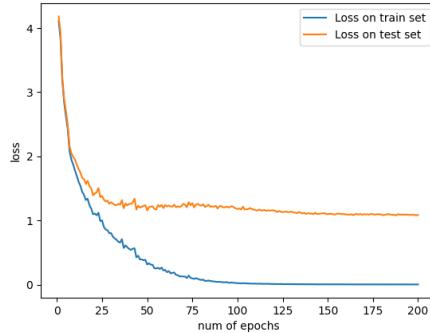
经过随机裁剪和水平翻转后，我们得到模型在测试集上的平均误差为 **1.154**，分类 Top-1 精度为 **72.2%**，Top-5 精度为 **91.9%**。模型的性能曲线如下所示：



(a) Top-1 accuracy 曲线



(b) Top-5 accuracy 曲线



(c) loss 曲线

图 9: 增强后模型的性能曲线

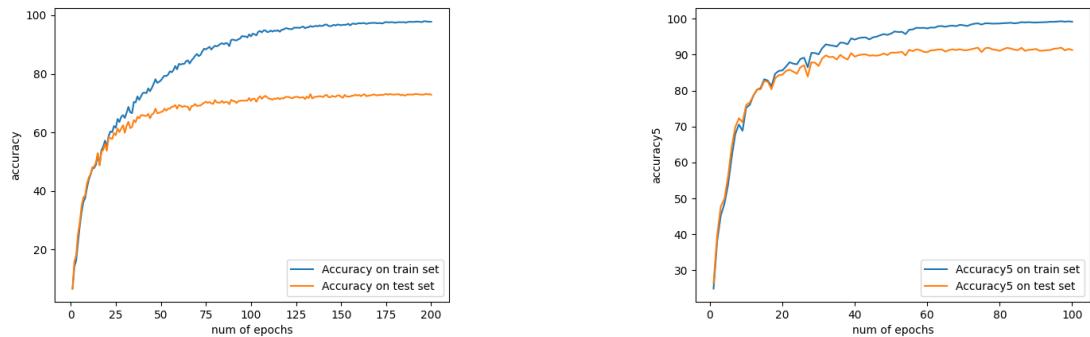
1.3.2 Cut out

Cut out 的原理是：随机将样本中的部分区域去掉，然后填充 0 像素值，分类的结果不变。如图 10 所示，展示了三张训练集样本 Cut out 后的图像。

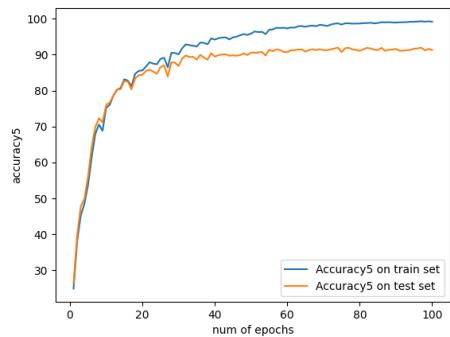
基于 torchtoolbox 包自带的 Cutout 方法，我们在数据增强时增加 Cut out 操作，设置对图像进行 Cutout 的概率为 0.5。进行 Cut out 后，得到模型在测试集上的平均误差为 **1.107**，分类 Top-1 精度为 **72.3%**，Top-5 精度为 **91.6%**。模型的性能曲线如图 11 所示。



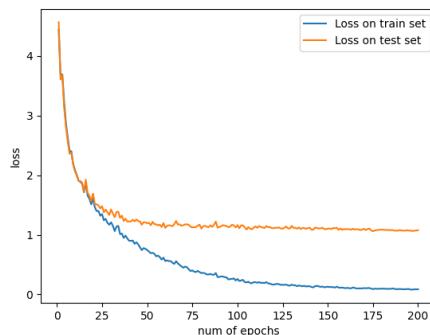
图 10: cutout 的效果



(a) Top-1 accuracy 曲线



(b) Top-5 accuracy 曲线



(c) loss 曲线

图 11: cutout 后模型的性能曲线

我们可以可视化五个中间 block 各一个特征通道的输出，如图 12 所示：

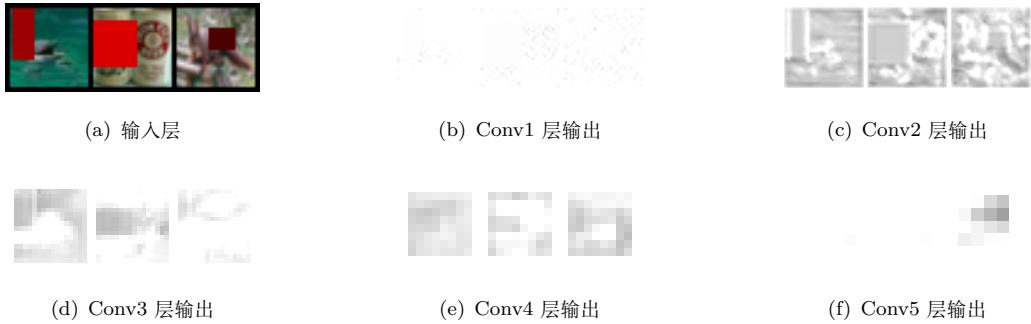


图 12: 隐藏层输出可视化

1.3.3 Cut mix

Cutmix 的原理是：将一部分区域去掉，但不是填充随机的固定像素值，而是随机填充训练集中的其他数据的区域像素值，分类结果按一定的比例分配。

Cutmix 可以用公式表示：

$$\begin{aligned}\tilde{x} &= \mathbf{M} \odot x_A + (\mathbf{1} - \mathbf{M}) \odot x_B \\ \tilde{y} &= \lambda y_A + (1 - \lambda) y_B\end{aligned}$$

其中 $\lambda \sim Beta(\alpha, \alpha)$, $\mathbf{M} \in \{0, 1\}^{W \times H}$ 表示选择来自哪张图片的掩码， \odot 表示逐元素相乘。为此，我们还需要确定一个随机采样得到的 bounding box，即 $B = (r_x, r_y, r_w, r_h)$ ，采样方法为：

$$\begin{aligned}r_x &\sim \text{Unif}(0, W), & r_w &= W\sqrt{1 - \lambda} \\ r_y &\sim \text{Unif}(0, H), & r_h &= H\sqrt{1 - \lambda}\end{aligned}$$

这样即可保证裁剪区域的大小为原图像的 $1 - \lambda$ 倍。

我们取 $\alpha = 1$ ，此时 $\lambda \sim \text{Unif}(0, 1)$ 。图 13 展示了以三张训练集样本作为一个 batch，经过 cutmix 后的图像：

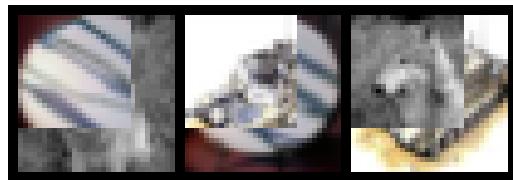


图 13: cutmix 的效果

进行 cutmix 后，我们得到模型在测试集上的平均误差为 **1.000**，分类 Top-1 精度为 **72.9%**，Top-5 精度为 **92.9%**。模型的性能曲线如图 14 所示。

可视化五个中间 block 各一个特征通道的输出，如图 15 所示。

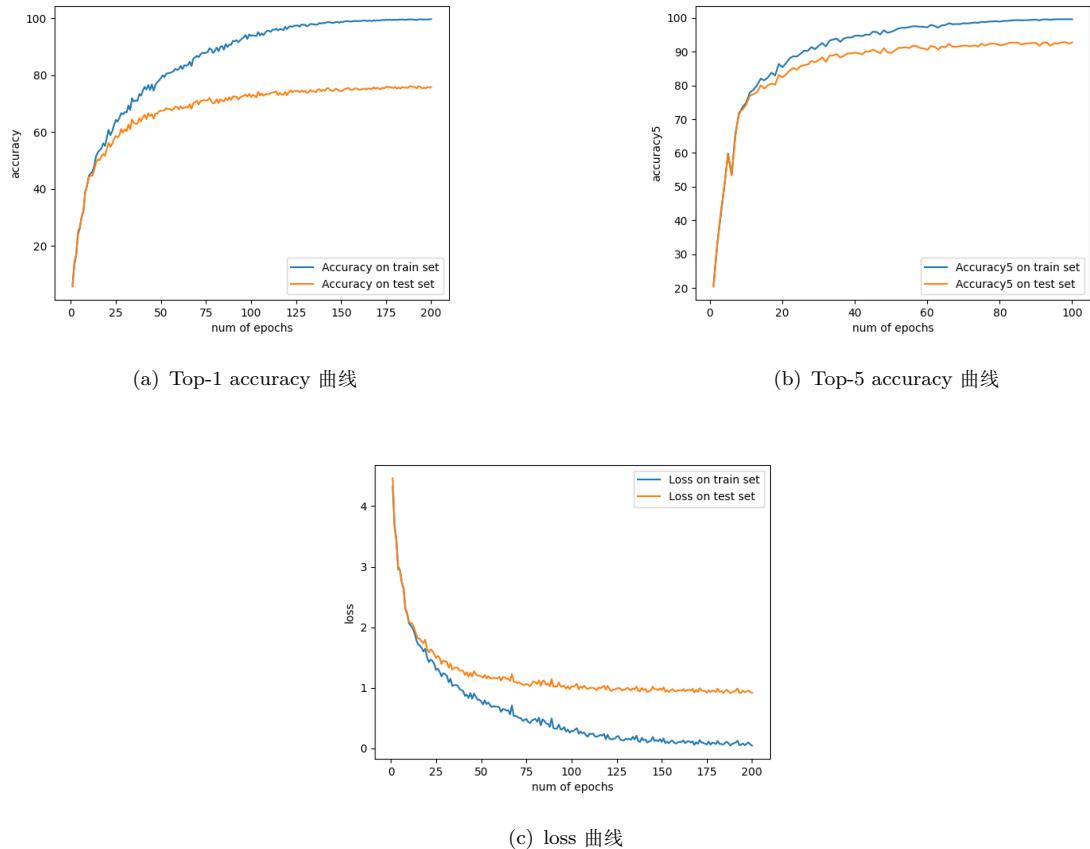


图 14: cutmix 后模型的性能曲线

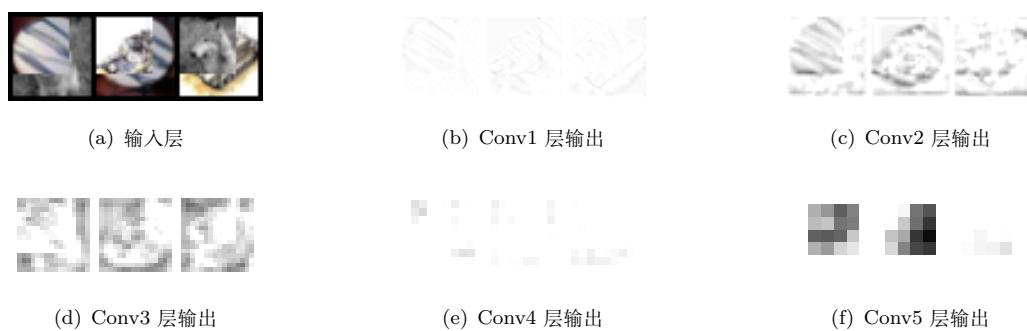


图 15: 隐藏层输出可视化

1.3.4 Mix up

Mixup 的原理是：将两张图按比例进行插值来混合样本。

Mixup 可以用公式表示：

$$\tilde{x} = \lambda x_A + (1 - \lambda)x_B$$

$$\tilde{y} = \lambda y_A + (1 - \lambda)y_B$$

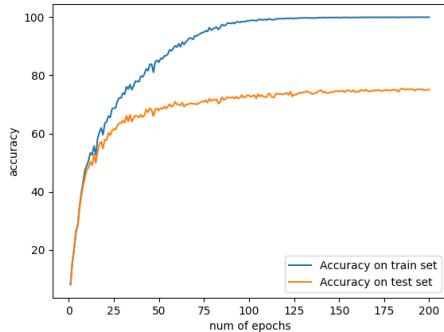
其中 $\lambda \sim Beta(\alpha, \alpha)$

我们取 $\alpha = 0.2$, 如图 16 所示, 展示了以三张训练集样本作为一个 batch, 经过 Mix up 后的图像。

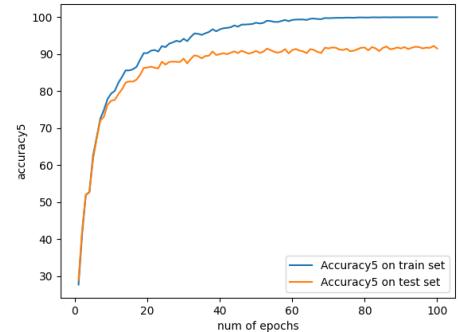


图 16: mixup 的效果

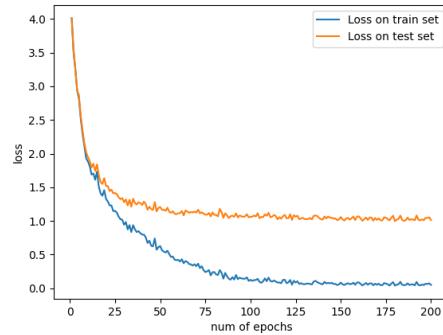
进行 mixup 后, 我们得到模型在测试集上的平均误差为 **1.036**, 分类 Top-1 精度为 **72.5%**, Top-5 精度为 **92.2%**。模型的性能曲线如图 17 所示。



(a) Top-1 accuracy 曲线



(b) Top-5 accuracy 曲线



(c) loss 曲线

图 17: mixup 后模型的性能曲线

可视化五个中间 block 各一个特征通道的输出, 如图 18 所示。

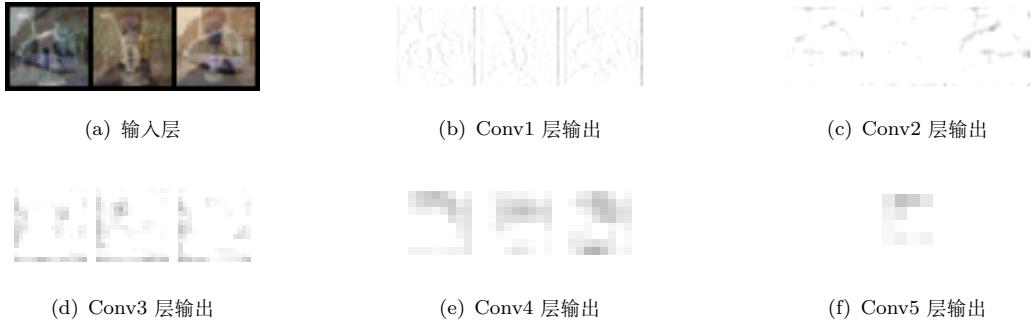


图 18: 隐藏层输出可视化

1.4 小结

我们总共得出下面四个模型:

- baseline 模型 (ResNet-18, 经过超参数调整)
- baseline 模型 + cutout
- baseline 模型 + cutmix
- baseline 模型 + mixup

四个模型在测试集上的 loss 和 accuracy 如下表所示:

模型	测试集误差	测试集 Top-1 精度	测试集 Top-5 精度
baseline	1.154	72.2%	91.9%
baseline + cutout	1.107	72.3%	91.6%
baseline + cutmix	1.000	72.9%	92.9%
baseline + mixup	1.036	72.5%	92.2%

表 1: 不同数据增强方式对模型性能的影响

我们发现, 在 baseline 模型上, 增加 cutmix 方法对数据进行增强, 相对于 cutout 和 mixup 两种方法达到的测试集上的误差最小, 精度最高。可以总结出 cutmix 的以下几个优点:

- 在训练过程中不会出现非信息像素, 从而能够提高训练效率;
- 保留了 cutout 方法 regional dropout 的优势, 能够关注目标的 non-discriminative parts;
- 通过要求模型从局部视图识别对象, 对 cut 区域中添加其他样本的信息, 能够进一步增强模型的定位能力;
- 相较于 mixup 方法, 不会有图像混合后不自然的情形, 能够提升模型分类的表现;
- 训练和推理的代价保持不变。

2 在 VOC 数据集上训练 Faster R-CNN 和 YOLO V3

2.1 数据集介绍与划分

2.1.1 数据集总体概况

PASCAL 对于目标检测或分割类型来说属于先驱者的地位。对于现在的研究者来说比较重要的两个年份的数据集是 **PASCAL VOC 2007** 与 **PASCAL VOC 2012**，这两个数据集频频在现在的一些检测或分割类的论文当中出现。

由于从 2007 年之后，PASCAL VOC 的测试集均不再公布，因此我们选择 PASCAL VOC 2007 作为我们的数据集。下面展示了 PASCAL VOC 2007 数据集的 20 个类别及其层级结构：

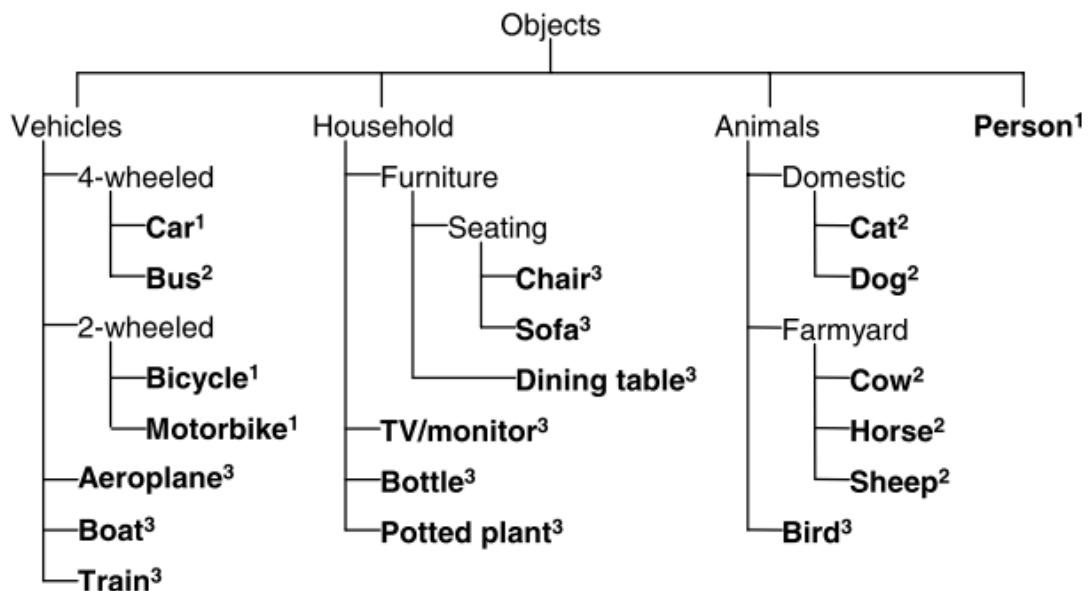


图 19: VOC 2007 数据集的标签

其中：

- 分为 4 个大类，20 个小类。预测时主要输出的小类。
- 数据集主要关注分类和检测，也就是分类和检测用到的数据集相对规模较大。关于其他任务比如分割，动作识别等，其数据集一般是分类和检测数据集的子集。

如图 20 所示，展示了 PASCAL VOC 2007 数据集总体的统计情况。

2.1.2 数据集划分

PASCAL VOC 2007 数据集分为两部分：训练和验证集 trainval，测试集 test，两部分各占数据总量的约 50%。其中 trainval 又分为训练集和测试集，二者分别各占 trainval 的 50%。每张图片中有可能包含不只一个目标 object。

如图 21 所示，展示了数据集在训练集，验证集，测试集上的划分情况。

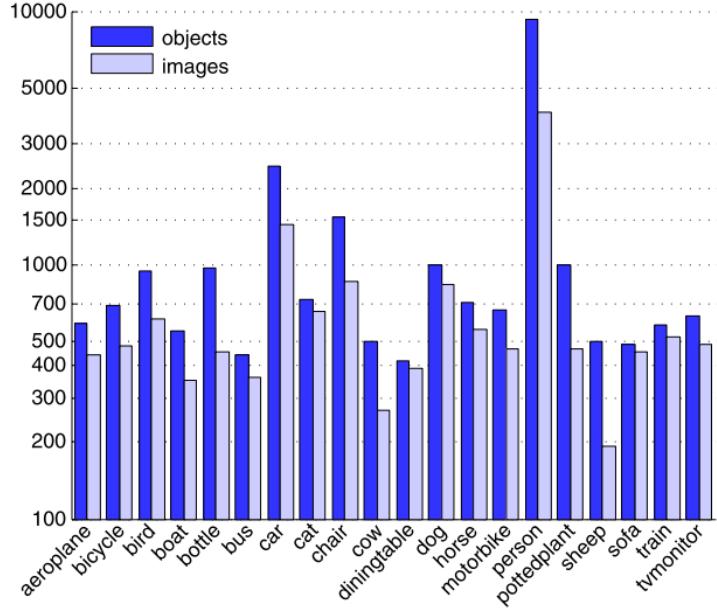


图 20: VOC 2007 数据集的统计情况

Table 2 Statistics of the VOC2007 dataset. The data is divided into two main subsets: training/validation data (`trainval`), and test data (`test`), with the `trainval` data further divided into suggested training (`train`) and validation (`val`) sets. For each subset and class, the

number of images (containing at least one object of the corresponding class) and number of object instances are shown. Note that because images may contain objects of several classes, the totals shown in the image columns are not simply the sum of the corresponding column

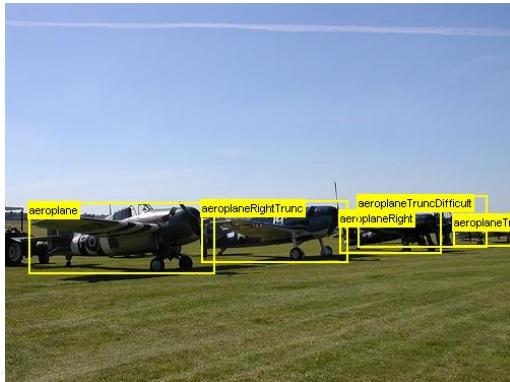
	train		val		trainval		test	
	images	objects	images	objects	images	objects	images	objects
Aeroplane	112	151	126	155	238	306	204	285
Bicycle	116	176	127	177	243	353	239	337
Bird	180	243	150	243	330	486	282	459
Boat	81	140	100	150	181	290	172	263
Bottle	139	253	105	252	244	505	212	469
Bus	97	115	89	114	186	229	174	213
Car	376	625	337	625	713	1,250	721	1,201
Cat	163	186	174	190	337	376	322	358
Chair	224	400	221	398	445	798	417	756
Cow	69	136	72	123	141	259	127	244
Dining table	97	103	103	112	200	215	190	206
Dog	203	253	218	257	421	510	418	489
Horse	139	182	148	180	287	362	274	348
Motorbike	120	167	125	172	245	339	222	325
Person	1,025	2,358	983	2,332	2,008	4,690	2,007	4,528
Potted plant	133	248	112	266	245	514	224	480
Sheep	48	130	48	127	96	257	97	242
Sofa	111	124	118	124	229	248	223	239
Train	127	145	134	152	261	297	259	282
Tv/monitor	128	166	128	158	256	324	229	308
Total	2,501	6,301	2,510	6,307	5,011	12,608	4,952	12,032

图 21: VOC 2007 数据集的具体划分情况

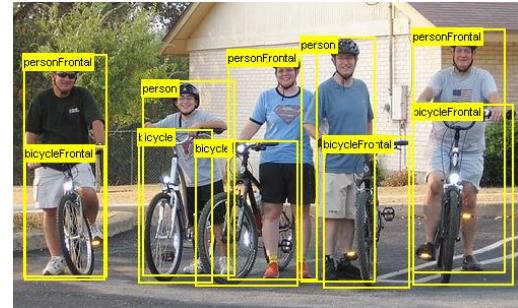
2.1.3 数据集可视化

<http://host.robots.ox.ac.uk/pascal/VOC/voc2007/examples/index.html>

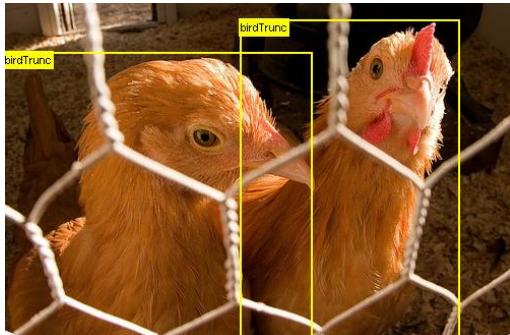
网站中展示了 VOC-2007 中至少包含每个类一个目标的 8 张图像及其目标的 ground truth 位置。下面展示一部分：



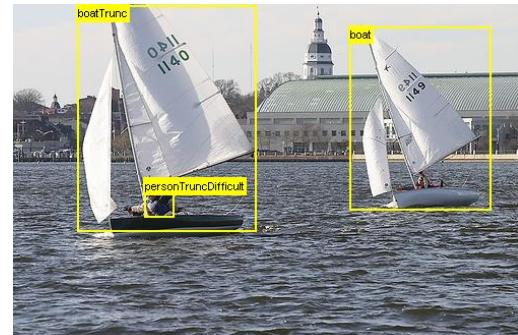
(a) Aeroplanes



(b) Bicycles



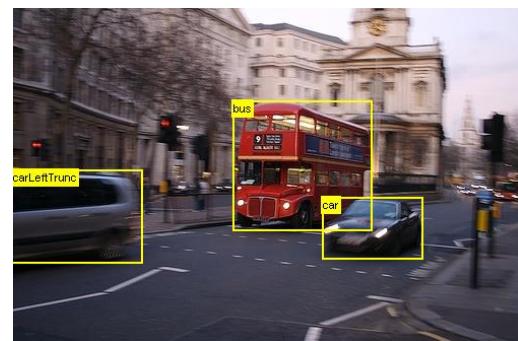
(c) Birds



(d) Boats



(e) Bottles



(f) Buses

图 22: VOC 2007 数据集示例

2.1.4 读取 VOC 数据集

使用 `torchvision.datasets` 包中的 `VOCDetection` 下载数据集，在 `VOC2007` 文件夹中，得到下面几个部分：

- Annotations: 包含了记录图像和标签信息的 xml 文件
- ImageSets: 主要包含划分的训练集和测试集中图片的名称
- JPEGImages: 包含数据集的原始图片
- SegmentationClass: 按类别分割的 ground truth 位置
- SegmentationObject: 按对象分割的 ground truth 位置

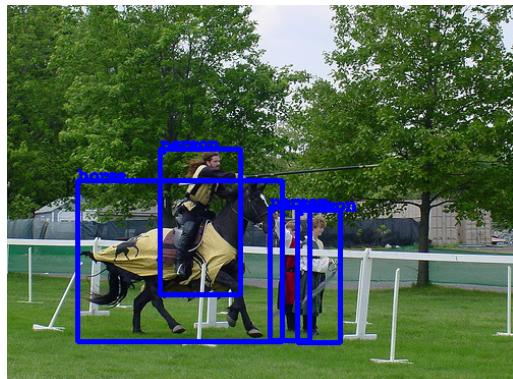
我们读取了四张训练集的图像并对其目标的 ground truth 进行了标注：



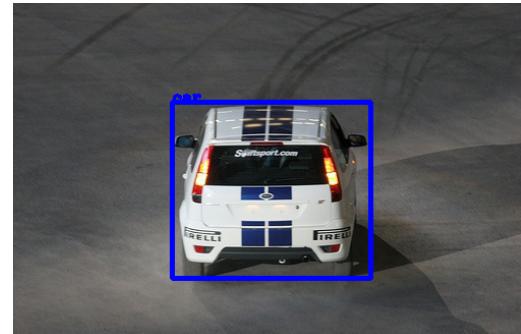
(a) 图片 1



(b) 图片 2



(c) 图片 3



(d) 图片 4

图 23: VOC 2007 训练集图像 ground truth 标注示例

2.2 Faster R-CNN 模型

2.2.1 基本结构

经过 R-CNN 和 Fast R-CNN 的积淀，Ross B. Girshick 在 2016 年提出了新的 Faster R-CNN，在结构上，Faster R-CNN 已经将特征抽取 (feature extraction)，proposal 提取，bounding box regression(rect refine)，classification 都整合在了一个网络中，使得综合性能有较大提高，在检测速度方面尤为明显。

整个 Faster R-CNN 分为 4 大部分：共享卷积网络，候选检测框生成网络 RPN (Region Proposal Networks)，感兴趣区域池化 RoI (Region of Interest) Pooling 和分类器。如图 24 所示。

其中：

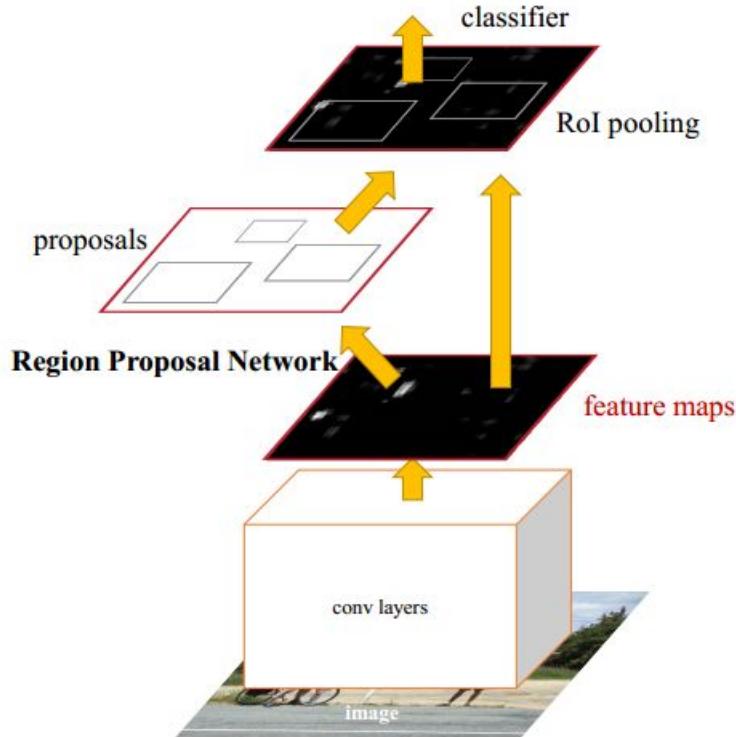


图 24: Faster R-CNN 基本结构

- 共享卷积网络 (Conv layers): 提取 image 的 feature maps。该 feature maps 被共享用于后续 RPN 层和全连接层。
- 候选检测框生成网络 (RPN): RPN 网络用于生成 region proposals。该层通过 softmax 判断 anchors 属于 positive 或者 negative，再利用 bounding box regression 修正 anchors 获得精确的 proposals。
- 感兴趣区域池化 (RoI Pooling): 该层收集输入的 feature maps 和 proposals，综合这些信息后提取 proposal feature maps，送入后续全连接层判定目标类别。
- 分类器 (classifier): 利用 proposal feature maps 计算 proposal 的类别，同时再次 bounding box regression 获得检测框最终的精确位置。

2.2.2 特征金字塔网络 (FPN)

注意到，上面我们提到的最基本的 Faster R-CNN 有一个明显的缺陷，即小物体本身具有的像素信息较少，在下采样的过程中极易被丢失。FPN (Feature Pyramid Network) 算法可以同时利用低层特征高分辨率和高层特征的高语义信息，通过融合这些不同层的特征达到很好的预测效果。

如图 25 所示，左侧模型叫 bottom-up，右侧模型叫 top-down，横向的箭头叫横向连接 lateral connections。特征金字塔网络相当于先进行传统的自上而下的特征卷积，然后 FPN 试图融合左侧特征图的相邻的特征图。具体做法是两个特征层的较高层特征 2 倍上采样（一般采用内插值方法，即在原有图像像素的基础上在像素点之间采用合适的插值算法插入新的元素），较低层特征通过 1×1 卷积改变一下低层特征的通道数，然后简单地把将上采样和 1×1 卷积后的结果对应元素相加。

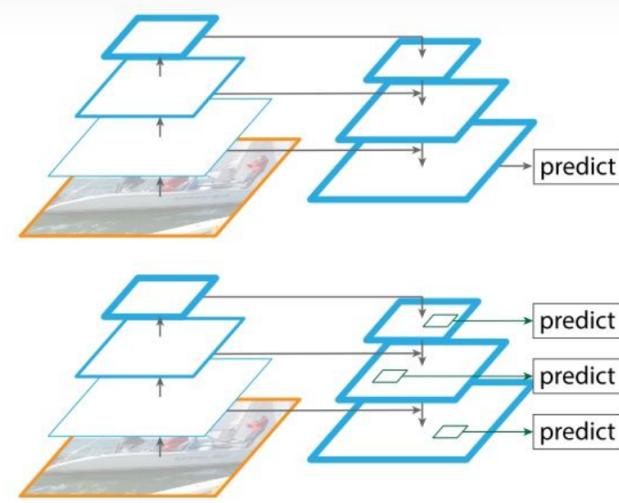


图 25: FPN 基本结构

2.2.3 候选检测框生成网络 (RPN)

下面对 RPN 部分进行详细分析，网络的详细结构如图 26 所示。

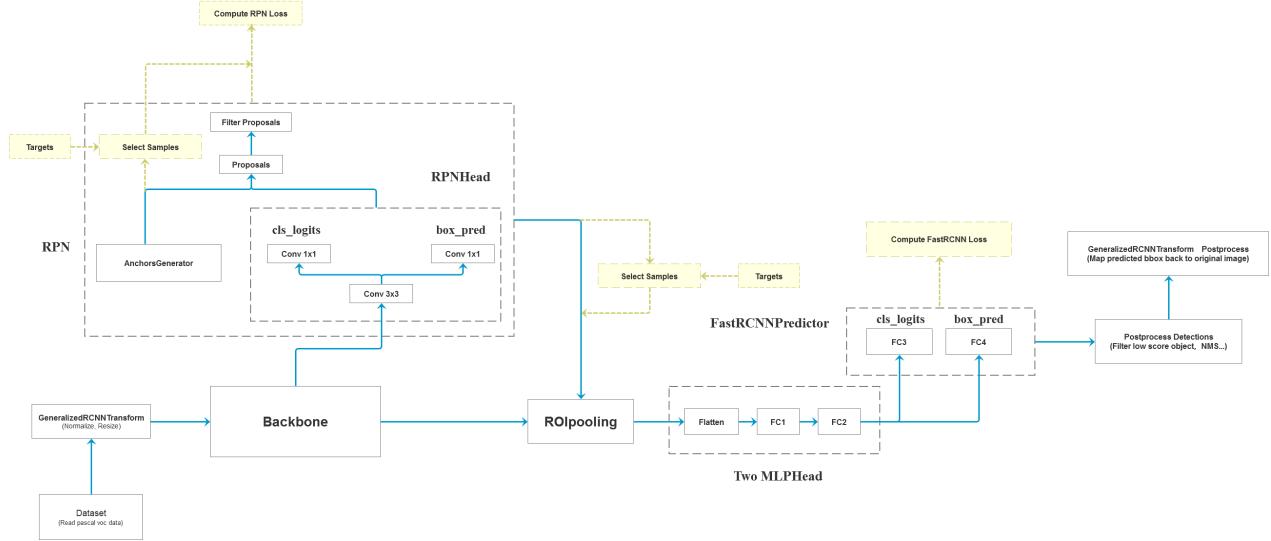


图 26: Faster R-CNN 结构

- 特征图上每个位置都对应生成五种面积 ($32^2, 64^2, 128^2, 256^2, 512^2$) 以及三种比例 (1:1,1:2,2:1) 的 anchor, 每个 anchor 有对应的前景概率及左上、右下角坐标。
- 对于一张 $1000*60*3$ 的图像, 大约有 20k 个 anchor, 忽略跨越边界的 anchor 以后, 剩下大约 6k 个 anchor。对于 RPN 生成的候选框之间存在大量的重叠, 基于候选框的 cls 得分, 采用非极大值抑制, IoU 设为 0.7, 这样每张图片只剩 2k 个候选框。

- 对于 $\text{IoU} \geq 0.7$ 或和某个 ground truth 重叠最大的 anchor 设置为正样本, $\text{IoU} \leq 0.3$ 作为负样本, 其余样本舍去。

计算 RPN 损失:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

$$L_{cls} = -[p_i^* \log(p_i) + (1 - p_i^*) \log(1 - p_i)]$$

$$L_{reg} = \sum_i \text{smooth}_{L_1}(t_i - t_i^*)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2, & |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \quad (1)$$

- p_i 表示第 i 个 anchor 预测为真实标签的概率
- p_i^* 当正样本时为 1, 负样本时为 0
- t_i 表示第 i 个 anchor 预测的边界框回归参数
- t_i^* 表示第 i 个 anchor 对应的 GT Box

2.2.4 模型训练

我们采用 RPN Loss+Fast R-CNN Loss 联合训练方法:

- 利用 ImageNet 预训练分类模型初始化前置卷积网络层参数, 并单独训练 RPN 网络参数
- 固定 RPN 网络的卷积层和全连接层参数, 再利用预训练分类模型初始化前置卷积网络参数, 并利用 RPN 网络生成的目标建议框去训练 Fast R-CNN 网络参数。
- 固定利用 Fast R-CNN 训练好的前置卷积网络层参数, 微调 RPN 网络卷积层及全连接层参数。
- 固定前置卷积网络层参数, 微调 Fast R-CNN 全连接层参数。最后 RPN 网络和 Fast R-CNN 构成统一网络。

训练细节:

- epoch=15, 由于使用了预训练分类模型, 收敛较快
- batch size=4, 过大会导致报错
- 采用学习率下降策略, 每 3 个 epochs 下降一次
- 优化器使用 SGD, loss function 上文中已经介绍

2.2.5 模型评价

下面展示了训练过程中的 mAP 图像，以及学习率和 loss 图像：

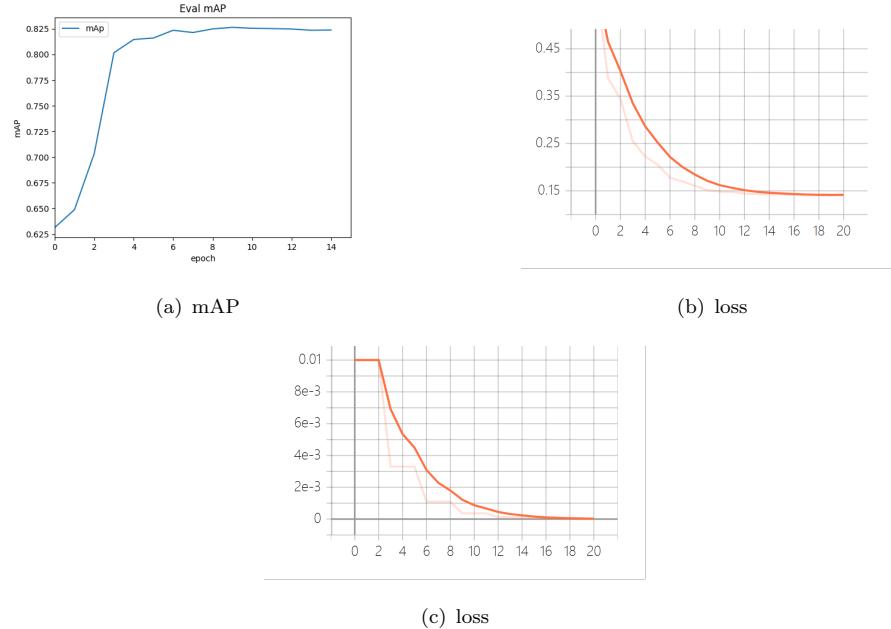


图 27: 训练过程的 mAP, loss 和学习率

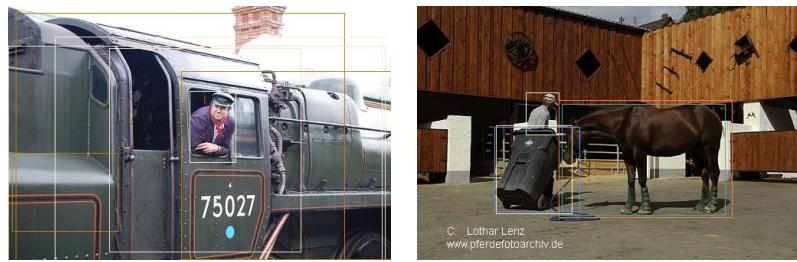
我们得到最终模型的性能指标如下：

mAP@IoU=0.5	mAP@IoU=0.5:0.95	mAR@IoU=0.5:0.95
0.828	0.526	0.640

表 2: 模型性能指标

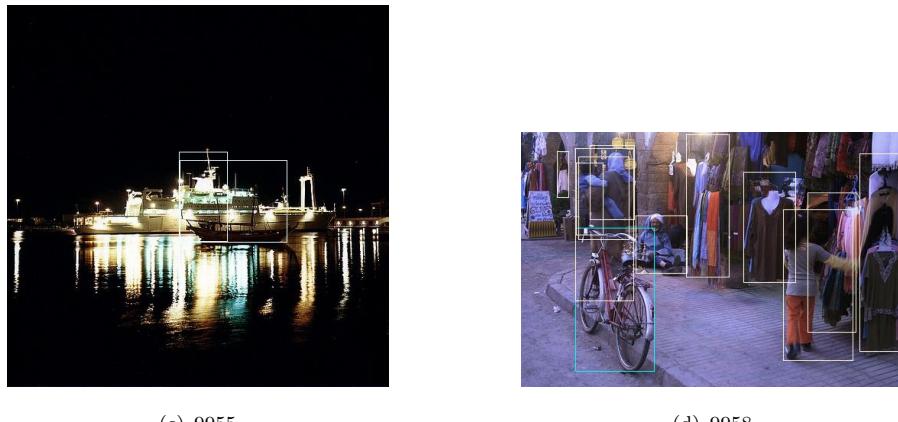
2.2.6 模型可视化

我们在四张测试图像上可视化 Faster R-CNN 第一阶段的 proposal box，并在网上寻找了三张不在 VOC 数据集中的图片，并使用训练好的模型进行标注：



(a) 9950

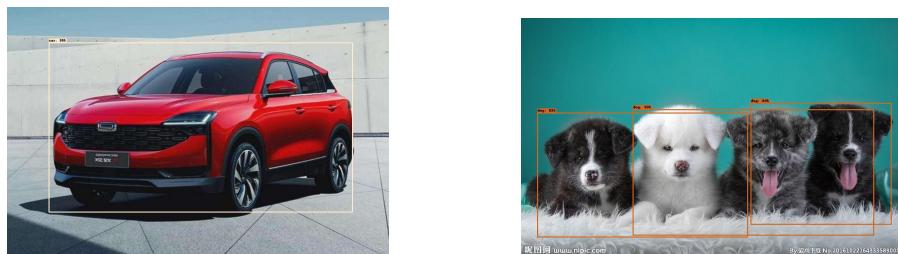
(b) 9954



(c) 9955

(d) 9958

图 28: proposal box



(a) 图片 1

(b) 图片 2



(c) 图片 3

图 29: 网上搜索的图像标注结果

2.3 YOLO V3 模型

2.3.1 YOLO V3 基本结构

相较于两阶段的检测模型 Faster R-CNN, Joseph Redmon 和 Ali Farhadi 等人的 YOLO 提供了另一种更为直接的思路：直接在输出层回归 bounding box 的位置和 bounding box 所属的类别（整张图作为网络的输入，把目标的问题转化成一个回归问题）。

YOLO V3 模型（基于 COCO 数据集，检测物体分为 80 类）的基本结构如下所示：

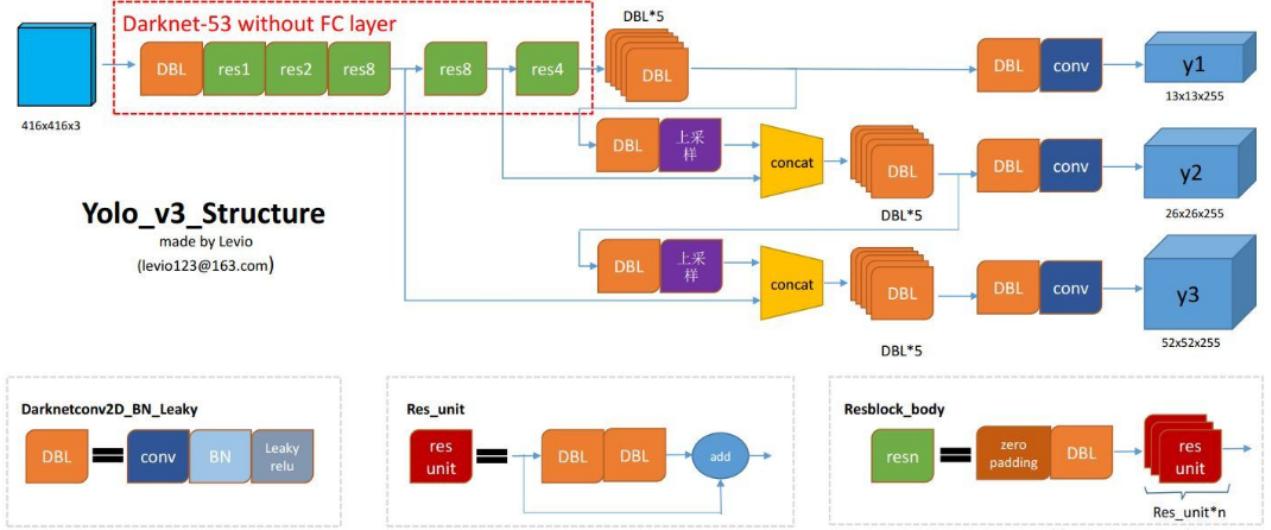


图 30: YOLO V3 基本结构

其中：

- 输入：把大小为 416×416 的输入图像划分成 $S \times S$ 大小的 grid cell，然后对每个 cell 都预测 $B = 3$ 个 bounding boxes，每个 bounding box 都包含 5 个预测值： $(x, y, w, h, \text{confidence})$ 。
 - (x, y) : bounding box 的中心坐标
 - (w, h) : 相对于 cell 大小进行归一化后 bounding box 的宽和高，注意这里使用了 sigmoid 函数
 - confidence: 代表了所预测的 box 中含有目标的置信度和这个 box 预测有多准两重信息
- backbone 网络：DarkNet-53，其中包括若干个卷积层和批归一化层，但不包括池化层（用一定步长的卷积代替）和全连接层，使用 Leaky-ReLU 作为激活函数。
 - DBL: DarkNet 的基本块，包括卷积、批归一化和 Leaky ReLU
 - resn: 使用了 n 个残差连接的块，其中 res_unit 表示将残差连接
 - concat: 张量拼接，将 DarkNet 中间层和后面的某一层的上采样进行拼接。
- 支路预测：为了加强算法对小目标检测的精确度，YOLO V3 中采用类似 FPN 的上采样和融合做法，在多个尺度 ($13 \times 13, 26 \times 26, 52 \times 52$) 的特征图上做检测，这相当于分别对原图像进行了 32, 16, 8 倍的降采样。
- 输出：针对 80 类，每个 cell 对应的输出应为 $3 \times (80 + 4 + 1) = 255$ ，因此有上图所示的三种输出

Type	Filters	Size	Output
Convolutional	32	3×3	256×256
Convolutional	64	$3 \times 3 / 2$	128×128
1x	32	1×1	
	64	3×3	
	Residual		128×128
Convolutional	128	$3 \times 3 / 2$	64×64
2x	64	1×1	
	128	3×3	
	Residual		64×64
Convolutional	256	$3 \times 3 / 2$	32×32
8x	128	1×1	
	256	3×3	
	Residual		32×32
Convolutional	512	$3 \times 3 / 2$	16×16
8x	256	1×1	
	512	3×3	
	Residual		16×16
Convolutional	1024	$3 \times 3 / 2$	8×8
4x	512	1×1	
	1024	3×3	
	Residual		8×8
Avgpool		Global	
Connected		1000	
Softmax			

图 31: DarkNet-53

2.3.2 YOLO V3 的 anchor box

从 YOLO V2 开始，引入了 Faster R-CNN 中人工设计的 anchor box。这存在一个弊端，就是并不能保证它们一定能很好的适合数据集，YOLO V2 中提到了这个问题，并使用 K-means 聚类来代替人工设计，通过对训练集的 bounding box 进行聚类，自动生成一组更加适合数据集的 anchor，可以使网络的检测效果更好。如果 anchor 的尺寸和目标的尺寸差异较大，则会影响模型的检测效果。

在 YOLO V3 中，32 倍降采样的感受野最大，适合检测大的目标，所以在输入为 416×416 时，每个 cell 的三个 anchor box 为 $(116, 90), (156, 198), (373, 326)$ ；16 倍适合一般大小的物体，anchor box 为 $(30, 61), (62, 45), (59, 119)$ ；8 倍的感受野最小，适合检测小目标，anchor box 为 $(10, 13), (16, 30), (33, 23)$ 。所以当输入为 416×416 时，实际总共有 $(52 \times 52 + 26 \times 26 + 13 \times 13) \times 3 = 10647$ 个 proposal box。

引入 anchor box 的时候遇到的另一个问题是：模型不稳定，尤其是在训练刚开始的时候。因此，相较于 Faster R-CNN 的无约束预测

$$\begin{aligned} x &= (t_x \times w_a) + x_a \\ y &= (t_y \times h_a) + y_a \end{aligned}$$

YOLO V3 预测的是边界框中心点相对于对应 cell 左上角位置的相对偏移值：

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w \exp(t_w) \\ b_h &= p_h \exp(t_h) \end{aligned}$$

其中 p_w, p_h 指 anchor box 的宽和高， c_x, c_y 指 cell 左上角相对于 image 左上角的距离， t_x, t_y, t_w, t_h 为每个 bounding box 预测的值。

2.3.3 YOLO V3 损失函数

为了计算损失函数，我们首先需要选择正负样本。其中，正样本表示负责预测目标的样本，负样本表示负责预测背景的样本。正负样本是按照以下规则决定的：如果一个预测框与所有的 Ground Truth 的最大 IoU < ignore thresh 时，那这个预测框就是负样本。如果 Ground Truth 的中心点落在一个区域中，该区域就负责检测该物体。将与该物体有最大 IoU 的预测框作为正样本（注意这里没有用到 ignore thresh，即使该最大 IoU < ignore thresh 也不会影响该预测框为正样本）

在 YOLOv3 中，Loss 分为三个部分：

- 定位误差，也即计算 bounding box 带来的 loss
- 置信度误差，bounding box 中是否含有物体的概率，也就是 obj 带来的 loss
- 分类误差，也就是 class 带来的 loss

用公式表示即：

$$\begin{aligned} loss &= l_{\text{box}} + l_{\text{obj}} + l_{\text{cls}} \\ l_{\text{box}} &= \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{\text{obj}} (2 - w_i h_i) [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2] \\ l_{\text{cls}} &= \lambda_{\text{class}} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{\text{obj}} \sum_{c \in \text{classes}} p_i(c) \log(\hat{p}_i(c)) \\ l_{\text{obj}} &= \lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{\text{obj}} (c_i - \hat{c}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{\text{noobj}} (c_i - \hat{c}_i)^2 \end{aligned}$$

其中：

- S : grid cell 的大小，分为三种（见 2.3.2）
- B : bounding box 的个数
- $1_{i,j}^{\text{obj}}$: 若在 (i, j) 的 box 有目标，则为 1，否则为 0
- $1_{i,j}^{\text{noobj}}$: 若在 (i, j) 的 box 没有目标，则为 1，否则为 0

需要注意的是，定位误差中，我们使用了 GIoU 代替 IoU，其计算方法为

$$GIoU = IoU - \frac{|A_c - U|}{|A_c|}$$

其中：

- A_c : 两个框构成的最小闭包面积，也即同时包含了预测框和真实框的最小框的面积
- U : 两个框的并集面积

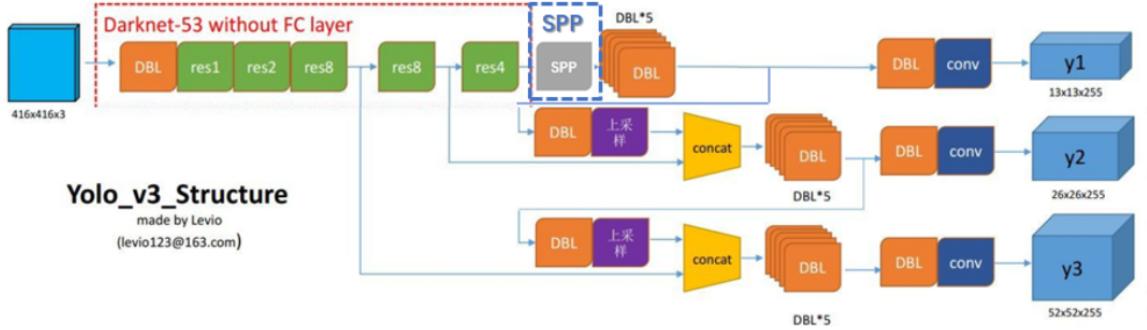


图 32: YOLO V3-SPP 基本结构

2.3.4 YOLO V3 SPP

SPP 全称为 Spatial Pyramid Pooling (空间金字塔池化结构)，由何恺明提出，有效避免了对图像区域剪裁、缩放操作导致的图像失真等问题；同时解决了卷积神经网络对图像重复特征提取的问题，大大提高了产生候选框的速度，且节省了计算成本。

在 YOLOv3-SPP 中，SPP module 由四个并行的分支构成，分别是 kernel size 为 5×5 , 9×9 , 13×13 的最大池化和一个跳跃连接。特征图经过局部特征与全局特征相融合后，丰富了特征图的表达能力，有利于待检测图像中目标大小差异较大的情况，尤其是对于 YOLOv3 这种复杂的多目标检测，所以对检测的精度上有了很大的提升。

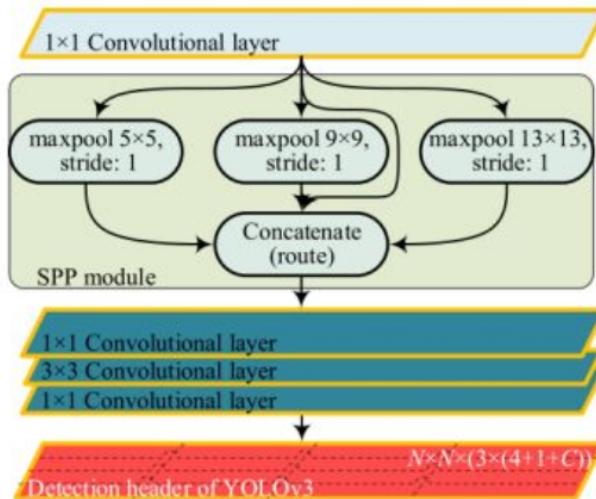


图 33: SPP

2.3.5 模型训练

我们在官方提供的预训练模型下训练，经过超参数调整，我们设置的模型超参数如下：

- epoch: 30
- 优化器: SGD+momentum，设置动量系数为 0.937

- 损失函数：前面已给出，包含三项（定位误差、置信度误差和分类误差）
 - 损失函数权重：
 - $\lambda_{\text{coord}} = 3.54$
 - $\lambda_{\text{class}} = 37.4$
 - $\lambda_{\text{obj}} = 64.3$
 - 正则化： l_2 ，设置正则化参数为 0.0005
 - 学习率：初始学习率为 $1e-3$ ，在训练过程中以余弦方式衰减至 $1e-5$
 - 样本选取 IoU 阈值：0.2
- 同时，我们使用了 HSV 转换的数据增强方法，对训练图像的色相、饱和度和灰度作随机抖动。

2.3.6 模型评价

下面给出 Tensorboard 中显示的学习率以及各项损失函数的曲线：

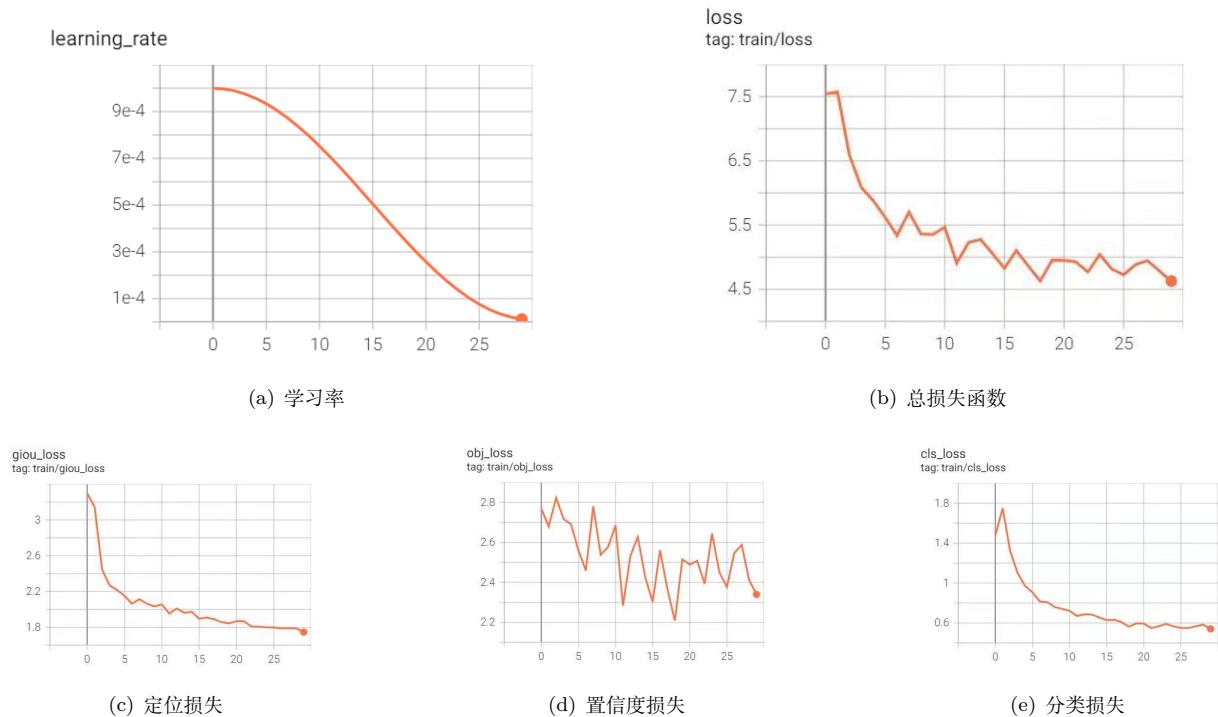


图 34: 学习率和损失函数曲线

我们得到最终模型的性能指标如下：

mAP@IoU=0.5	mAP@IoU=0.5:0.95	mAR@IoU=0.5:0.95
0.806	0.565	0.682

表 3: 模型性能指标

对应训练过程中的 mAP 和 mAR 曲线如图 32 所示。

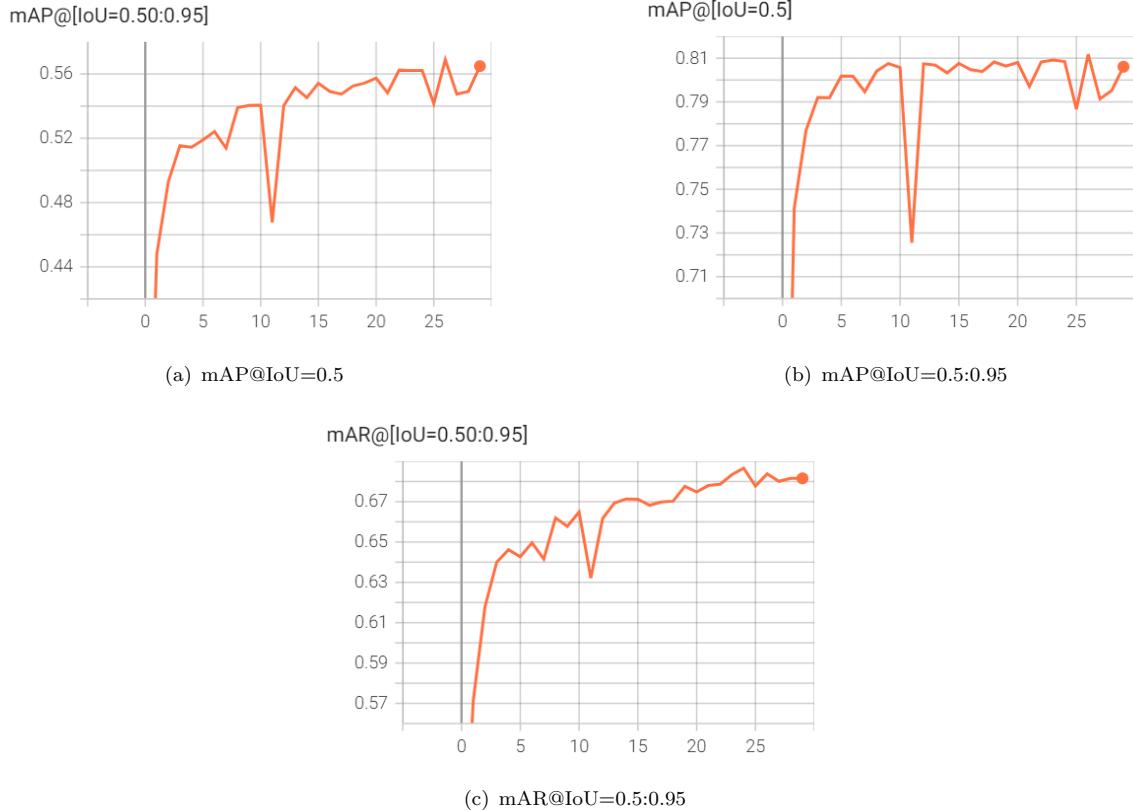


图 35: 训练过程曲线

2.3.7 模型可视化

我们使用可视化 Faster R-CNN 中三张不在 VOC 数据集中的图片，并使用训练好的模型进行标注。



图 36: 网上搜索的图像标注结果

值得注意的是，图片 3 中，有一个人未被检测出，这是 Faster R-CNN 不曾表现出的误差。

2.4 小结

在本节中，我们分别使用目标检测模型：两阶段的 **Faster R-CNN** 和单阶段的 **YOLOv3-SPP** 在 VOC2007 数据集上训练。

Faster R-CNN 通过将 region proposal 整合到 CNN 模型来提高速度：构建由 RPN 和具有共享卷积特征层的 Fast R-CNN 组成的统一模型；而 YOLO 模型将目标检测问题看成一个回归问题而不是分类问题，将损失函数表示为定位误差、置信度误差和分类误差的加权求和。

下面是两个模型的性能对比：

	mAP@IoU=0.5	mAP@IoU=0.5:0.95	mAR@IoU=0.5:0.95
Faster R-CNN	0.828	0.526	0.640
YOLO V3-SPP	0.806	0.565	0.682

表 4: 模型性能指标

可以发现，两个模型在目标检测上的表现均较好，其中 YOLO-v3 在全面了解背景的情况下可以更好地识别背景，且检测较快，但相对于 Faster R-CNN，其对小物体和复杂物体识别率低。

3 代码链接和模型下载地址

Github repo:

模型网盘:

- 任务一, baseline: <https://pan.baidu.com/s/1Lx8wc3jproLMwCHYXuNjTg?pwd=4q0k>, 提取码 4q0k
- 任务一, cutout: <https://pan.baidu.com/s/1F8mqFeKTgD6aAZtm5savng?pwd=4btp>, 提取码 4btp
- 任务一, cutmix: <https://pan.baidu.com/s/1mBoSj5ixxZduGGvmJUeFLQ?pwd=oid3>, 提取码: oid3
- 任务一, mixup: <https://pan.baidu.com/s/1jL60qw1B-T3JHGsoWuY-vw?pwd=u583>, 提取码 u583
- 任务二, Faster R-CNN:
- 任务二, YOLO v3: <https://pan.baidu.com/s/19vekigVva1uzdZOMkLlAJA?pwd=sh33>, 提取码 sh33