技术方案

江舒 房梓晔 付孝德 沈飞

Docker 地址：

百度网盘：链接：https://pan.baidu.com/s/1aSInVLJyl_DZvrd_miQymw?pwd=zg6g

提取码：zg6g

1 算法结构

1.1 模型结构

本次大赛我们使用 Cascade R-CNN[1]架构，这是一种级联 R-CNN，使用级联回归作为重采样机制来提高 IoU，逐层增加 proposal 的 IoU 值，这样前一层的重采样建议就能适应下一个阶段的更高的阈值。

使用 CBSwinTransformertiny[2]和 ConvNeXtBase[3]两个 BackBone 提取。CBSwinTransfomer。Swin Transformer 是一种层次化的 transformer,采用移动窗口进行自注意力计算，还允许跨窗口连接来提高效率，相对于图像大小具有线性计算复杂度。transformer 在各个视觉任务上都取得了较好的成果。ConvNeXtBase。ConvNeXtBase 是一种大规模的图像识别检测模型，是依照 Transformer 网络的一些先进思想对现有的经典 ResNet50/200 网络做一些调整改进，将 Transforme r 网络的最新的部分思想和技术引入到 CNN 网络现有的模块中从而结合这两种网络的优势，是目前识别检测效果较好的 CNN 模型之一。

此外，我们使用 FPN 网络作为 Neck。在深度学习中，低层次的特征具有语义信息较少，但目标位置准确；高层次的特征有较多的语义信息，但目标位置粗糙。高层特征有更多的语义信息，但目标位置却很粗略。目标位置。使用 FPN 可以很好地解决这个问题。

1.2. 训练技巧

1.2.1 多尺度和大尺度训练

采用多尺度训练，在训练时，每隔几轮便改变模型输入尺寸。通过对不同尺度的图像进行训练，在一定程度上提高检测模型对海藻目标物体大小的鲁棒性。

大尺度图像通常会具有更复杂的语义信息，因此将输入的图像 reshape 为大尺度图像有更好的性能。

1.2.2 数据增强

数据增强主要采用随机翻转、MixUp 等方法。其中，MixUp 从每个 batch 中随机选择两张图像，并以一定比例混合生成新的图像。

主要起到以下两点作用：（1）增加噪声数据，提升模型的鲁棒性；（2）增加训练的数据量，提高模型的泛化能力。

1.2.3 模型结果平均

通常情况下，NMS 和 soft-NMS 是常用的候选框筛选方式。这些候选框仅能从单个模型推理出的预测框中，筛选出置信度和 IOU 值较高的候选框，而不能有效地结合多个模型的预测结果来产生一个平均的框。与单一模型相比，结合多个模型的预测会得到更好更准确的效果。

我们使用 WBF（weighted boxes fusion）完成 2 个模型的结果的平均。假设物体 n 有 T 个候选框，候选框左上角坐标为（X1,Y1），右下角坐标为（X2,Y2），候选框的置信度为 Ci。X 和 Y 的融合公式分别如（1）（2）所示。

$$X1,2 = \frac{\sum_{i=1}^{T} C_i * X1,2}{\sum_{i=1}^{T} C_i} \tag{1}$$

$$Y1,2 = \frac{\sum_{i=1}^{T} C_i * Y1,2}{\sum_{i=1}^{T} C_i} \tag{2}$$

使用每个框的得分作为权值，将两个框的坐标进行融合，得到一个新的框。因此，得分越高的框，权值越大，在生成新框的过程中，它的贡献更大。

1.3 数据集

大赛数据集训练集 4000 张，测试集 3806 张。我们对训练集做了如下分析。首先统计了 bbox 的宽高比，如图 1-1 所示。可以看到宽高比为 1:1 的 bbox 占据绝大比例，1:2 和 1:3 的次之，此外其他比例占比非常小。

据此，我们设计了 anchor 的比例为[0.25, 0.5, 1.0, 2.0, 4]。

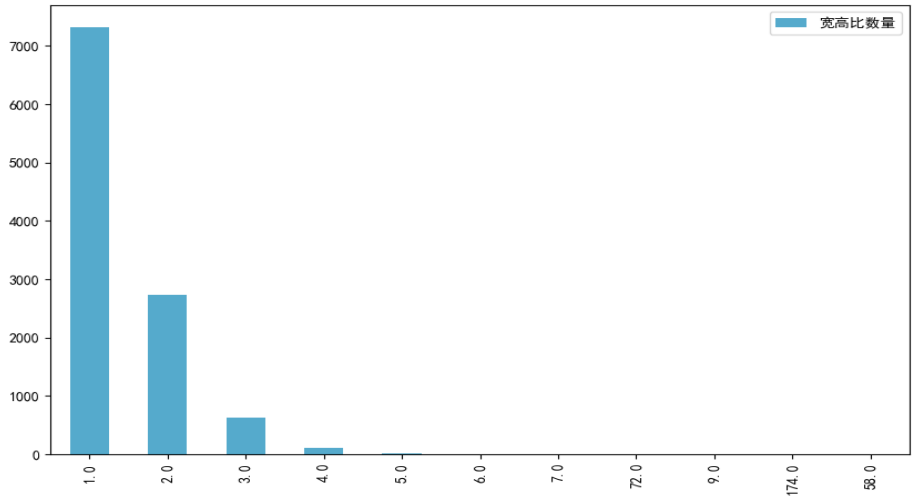

图 1-1 bbox 的宽高比

训练集的类别数目如表 1-1 所示，每类占比示意图如图 1-2 所示。paper_trash，snakeskin_bag 和 carton 三个类别占据的数目最多。但由于 paper_trash 目标很小，图像主要为生活场景图，细节多，因此 paper_trash 的检测效果很差。为解决此问题，我们设计了专家模型，对 paper_trash 进行检测，未经精细调参时，结果可以提升 0.001。考虑到专家模型训练时间长，分数提升幅度小，我们最终没有采用本方案。

表 1-1 类别数目

ID	类别	数目	ID	类别	数目
0	paper_trash	3629	5	stone_waste	381
1	packed_trash	1491	6	sand_waste	105
2	snakeskin_bag	3681	7	foam_trash	216
3	plastic_trash	1183	8	metal_waste	127
4	carton	3592	9	wood_waste	39

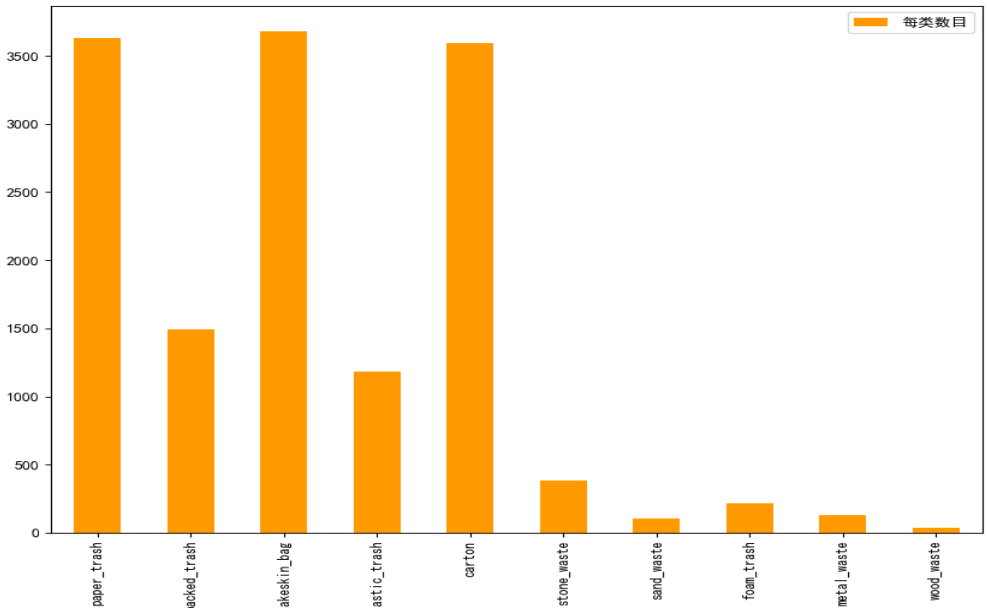

图 1-2 每类数目

图像尺度统计如图 1-3 所示，可以看到(1920, 1080)的尺度占据了图像的绝大多数，有 2719 张，因此，我们将图像放到 2048 进行训练。然而其他尺度的图像数目也很多，因此我们设计了多尺度训练策略，ConvNeXtBase 以(2048, 1200)到(2048, 1600)尺度进行训练。考虑网络模型参数规模和训练测试数据的数量，CBSwinTransfomer 模型权重以(4096, 1200)到(4096, 1600)的尺度范围进行训练。

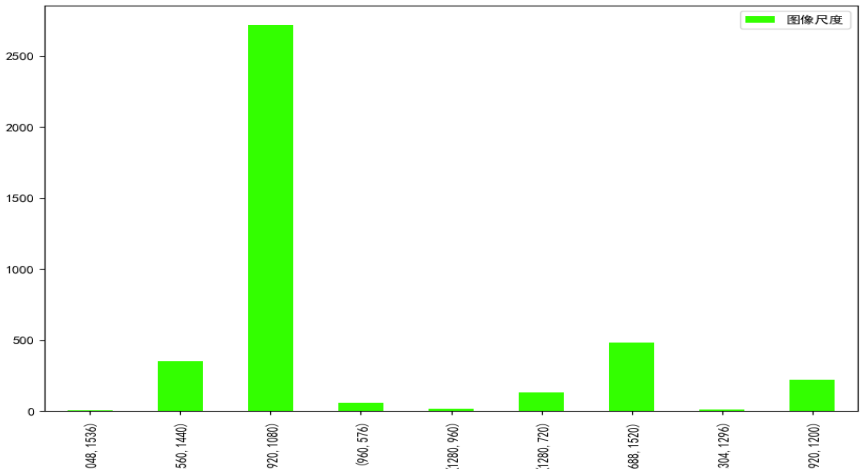

图 1-3 图像尺度

1.4 算法介绍

常见的二阶段检测器如 Fast RCNN 等模型只有 proposal 自身的阈值和训练器训练用的阈值较为接近的时候，训练器的性能才最好。如果两个阈值相距比较远，效果就会很差。此外，单一阈值训练出的检测器效果非常有限，不能对所有的 Proposals 都有很好的优化作用。

Cascade R-CNN 会逐 stage 提高正样本的 iou 阈值，每一次的回归后的框重新采样输入到下一阶段。图 1-4 展示了 Cascade R-CNN 的网络结构，其中“I”是输入图像，“conv”是 BackBone，“pool”是区域特征提取，“H”是网络头，“B”是 b box，“C”是分类。“B0”是所有架构中的 proposal。

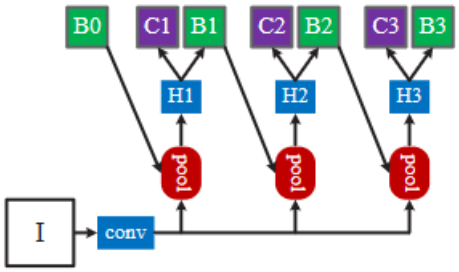

图 1-4 Cascade R-CNN

Swin Transformer 的结构如图 1-5 所示。Patch Merging 顾名思义就是把临近的小 patch 合并成一个大 patch，起到下采样的效果。Swin Transformer Block 将每次输入做 Layernorm 操作，然后做窗口的多头自注意力运算，通过 Layernorm 和 MLP，第一个 block 就结束了，紧接着做一次 Shifted window，也就是基于移动窗口的多头自注意力，然后再过 MLP 得到输出。

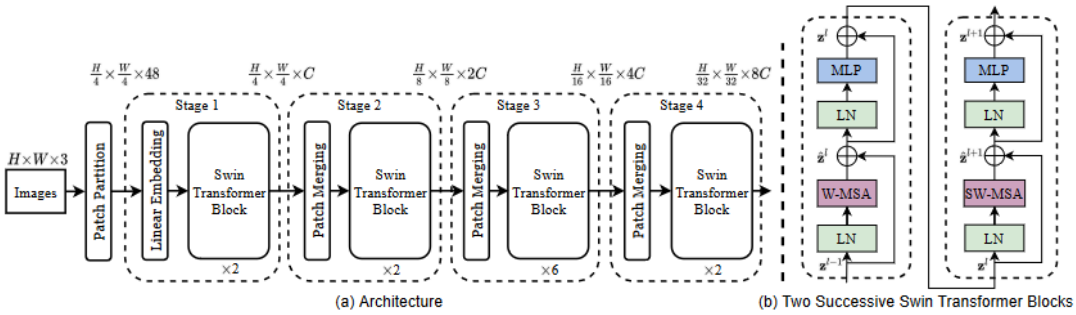

图 1-5 Swin Transformer

ConvNeXt 网络本身没有什么亮点，是应用的现有的方法来进行网络的调整，特别是大量细节的设计都是参考了 swin transformer 的网络结构的。结构如图 1-6 所示。

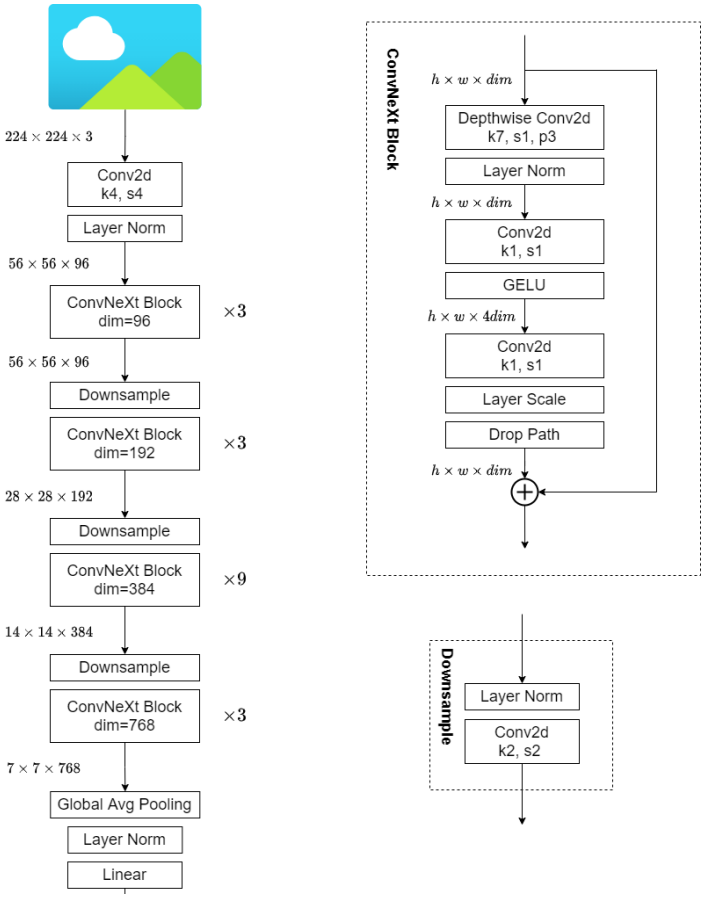

图 1-6 ConvNeXt[4]

2 算法运行说明

2.1 算法相关文件结构

2.1.1 训练脚本

train_grabge.sh 设置相应的训练模型 config 及分布式训练所用的 GPU 数目，调用 train.py 进行训练

2.1.2 推理脚本

test_ensemble_gra.py，推理并融合两个模型的预测结果

2.1.3 训练数据位置

/home/js/train_data

2.1.4 测试数据位置

/home/js/train_data

2.1.5 其他文件

（1）cascade_rcnn_convnext_base_fpn.py ConvNeXtBase 模型配置文件，包括网络模型，训练流程，测试流程，数据加载，数据预处理，损失函数，优化方

式等。

（2）cascade_rcnn_cbnet_tiny_nozeng.py　CBSwinTransformertiny 模型配置文件，其余同上

（3）/home/js/mmdetection/文件夹为 mmdetection 框架，训练测试模式与 mmdection 保持一致。

（4）/home/js/cascade_rcnn_convnext_base_fpn/存放 ConvNeXtBase 运行结果、配置文件、运行过程中的日志。

（5）/home/js/cbswin/存放 CBSwinTransformertiny 运行结果、配置文件、运行过程中的日志。

（6）/home/js/cascade_rcnn_convnext_base_fpn/epoch_10.pth 是我们已经训练好的 ConvNeXtBase 模型，可直接用于推理。

（7）/home/js/cbswin/epoch_20.pth 是我们已经训练好的 CBSwinTransformertiny 模型，可直接用于推理。

（8）/home/js/result.json 是我们已经推理好的最终结果。

2.2 模型训练环境，运行参。训练时间，预测时间

（1）训练环境

单张 RTX3090

MMDetection

cuda11.1

python3.6.9

pytorch1.10

（2）运行参数

包括 Config 文件路径和 GPU 数目

```
CONFIG=$1
GPUS=$2
NNODES=${NNODES:-1}
NODE_RANK=${NODE_RANK:-0}
PORT=${PORT:-29500}
MASTER_ADDR=${MASTER_ADDR:-"127.0.0.1"}

PYTHONPATH="$(dirname $0)/..":$PYTHONPATH \
python3 -m torch.distributed.launch \
    --nnodes=$NNODES \
    --node_rank=$NODE_RANK \
    --master_addr=$MASTER_ADDR \
    --nproc_per_node=$GPUS \
    --master_port=$PORT \
    $(dirname "$0")/train.py \
    $CONFIG \
    --seed 0 \
    --launcher pytorch ${@:3}
```

train_grabge.sh

```
#第一个模型
/home/js/mmdetection/tools/train_grabge.sh /home/js/mmdetection/configs/GarbageExposure/cascade_rcnn_cbnet_tiny_nozeng.py 4
#第二个模型
/home/js/mmdetection/tools/train_grabge.sh /home/js/mmdetection/configs/GarbageExposure/cascade_rcnn_convnext_base_fpn.py 4
```

General.sh

（3）训练时间

CBSwinTransformertiny 模型 10 小时左右

ConvNeXtBase 模型 5 小时左右

（4）测试时间

20000s 左右

```
[                                                  ] 1/3806, 0.2 task/s, elapsed: 6s, ETA: 21873s20210513174630517_0.jpg
```

2.3 模型训练和测试说明

（1）end-to-end 训练和测试模型，运行/home/js/general.sh，即可训练和测试一体化进行，如图下所示。训练好的模型和配置文件分别在/home/js/cascade_rcnn_convnext_base_fpn/和/home/js/cbswin/目录下，最终推理结果为/home/js/result1.json。

```
source activate mmdet
#训练[sh,config,GPU]
#第一个模型
/home/js/mmdetection/tools/train_grabge.sh /home/js/mmdetection/configs/GarbageExposure/cascade_rcnn_cbnet_tiny_nozeng.py 4
#第二个模型
/home/js/mmdetection/tools/train_grabge.sh /home/js/mmdetection/configs/GarbageExposure/cascade_rcnn_convnext_base_fpn.py 4
#融合 第一个模型第20轮结果与第二个模型第十轮结果融合
python3 /home/js/mmdetection/test_ensemble_gra.py
#结果在/home/js/result1.json
```

General.sh

（2）我们训练的 CBSwinTransformertiny 以及 ConvNeXtBase 模型也分别在/home/js/cascade_rcnn_convnext_base_fpn/和/home/js/cbswin/，可以跳过训练直接推理。

```
source activate mmdet
#训练[sh,config,GPU]
#第一个模型
# /home/js/mmdetection/tools/train_grabge.sh /home/js/mmdetection/configs/GarbageExposure/cascade_rcnn_cbnet_tiny_nozeng.py 4
#第二个模型
# /home/js/mmdetection/tools/train_grabge.sh /home/js/mmdetection/configs/GarbageExposure/cascade_rcnn_convnext_base_fpn.py 4
#融合 第一个模型第20轮结果与第二个模型第十轮结果融合
python3 /home/js/mmdetection/test_ensemble_gra.py
#结果在/home/js/result1.json
```

3 未来的优化方向

未来我们希望可以对以下几点进行优化：

（1）优化专家模型，提升对 paper_trash 的检测精度。

（2）目前检测出的目标框冗余度较高，这是由于有的目标重合程度高，且背景复杂较为密集，未来希望对目标框进行过滤。

参考文献

[1] Z. Cai and N. Vasconcelos, “Cascade R-cnn: Delving into High Quality Object Detection,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition, Aug. 2018, pp. 6154-6162.

[2] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang et al., “SwinTransformer: Hierarchical Vision Transformer Using Shifted Windows,”in Proc. IEEE/CVF Int. Conf. Computer Vision, October, 2021, pp.10012-10022.

[3] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A Convnet for the 2020s,” in Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition, June. 2022, pp. 11976-11986.

[4] https://blog.csdn.net/qq_37541097/article/details/122556545