Readme

一、主要任务
二、算法思路
2.1 核心方法
2.2 数据处理
2.3 数据增强
2.4 数据分配
三、复现流程
3.1 环境依赖
3.1.1 docker依赖
3.1.2 pip依赖
3.1.3 mmdetection依赖
3.1.4 Dockfile
3.2 代码结构
3.3 训练复现
3.3.1 训练参数
3.3.2 训练复现
3.4 预测复现
3.4.1 预测步骤
3.4.1 预测复现

一、主要任务

比赛采用的数据为大量伪造的证书文档类图像。任务是通过提供的训练集学习出有效的检测算法，对测试集的伪造图像进行篡改定位。

二、算法思路

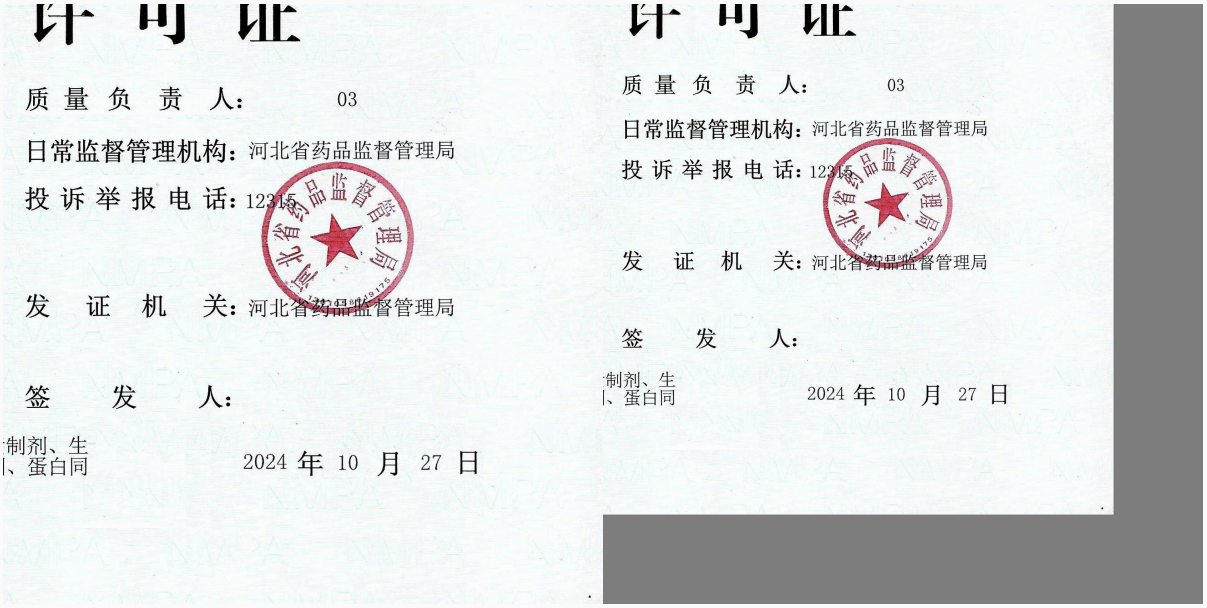
2.1 核心方法

转换为目标检测问题，将篡改区域定义目标类别class-cheat

2.2 数据处理

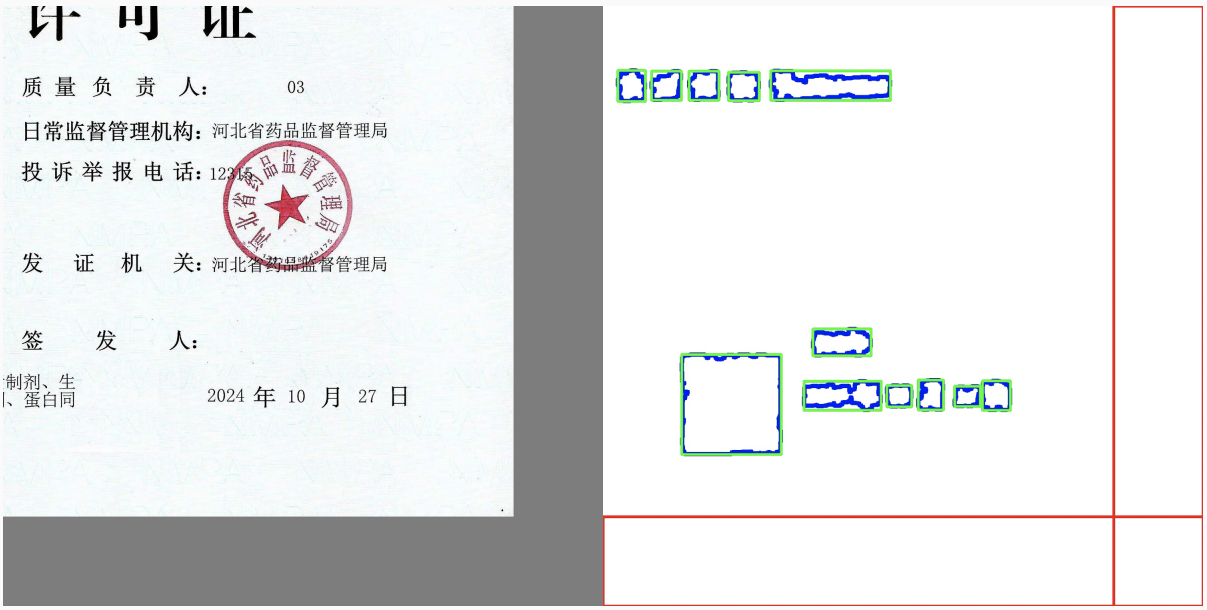
原始图像尺寸参差不齐，统一使用**颜色125**填充至1550*1550。

举例：



颜色125部分定义为第二类别class-pmiss

对mask进行解析，如下红色为class-pmiss，蓝绿色为class-cheat。

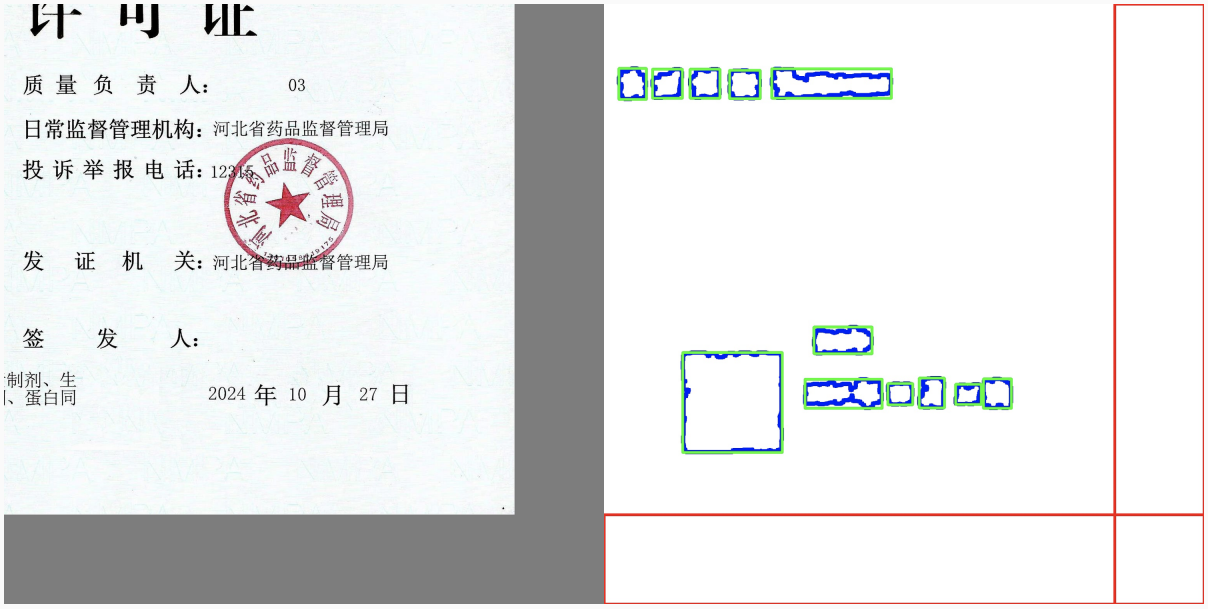


据此构建COCO数据集，用于mmdetection目标检测。

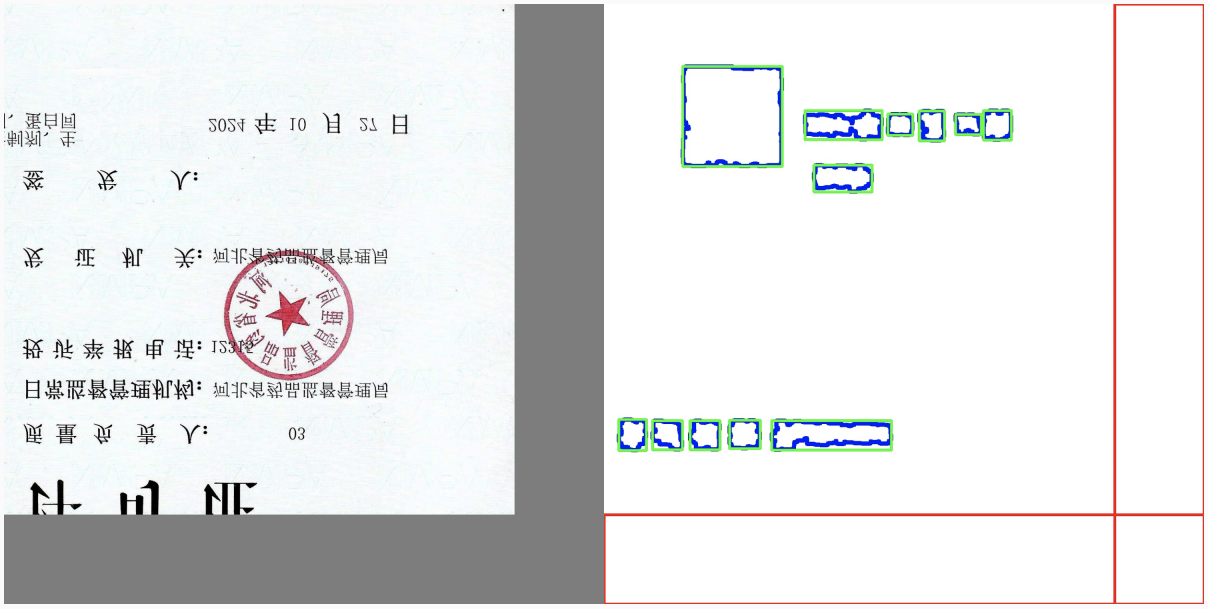
2.3 数据增强

原有训练数据集较少，且部分图像来自同一张图像，为避免过拟合。使用了以下方法以数据增强。

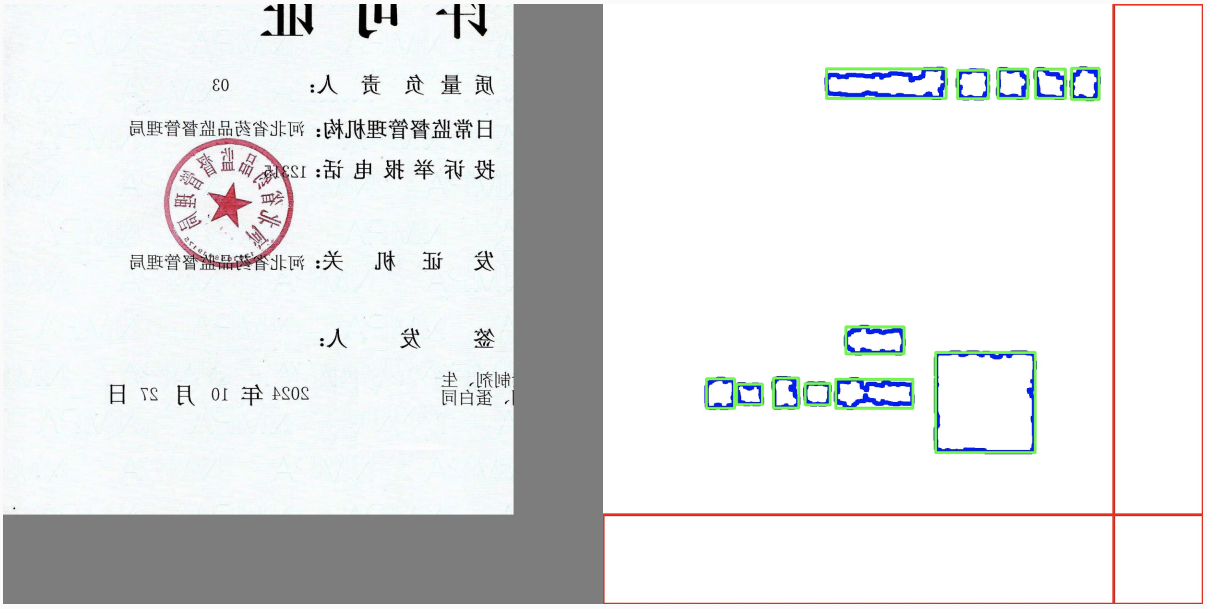
- 原始


质量负责人：03
日常监督管理机构：河北省药品监督管理局
投诉举报电话：12331
发证机关：河北省药品监督管理局
签发人：
2024 年 10 月 27 日

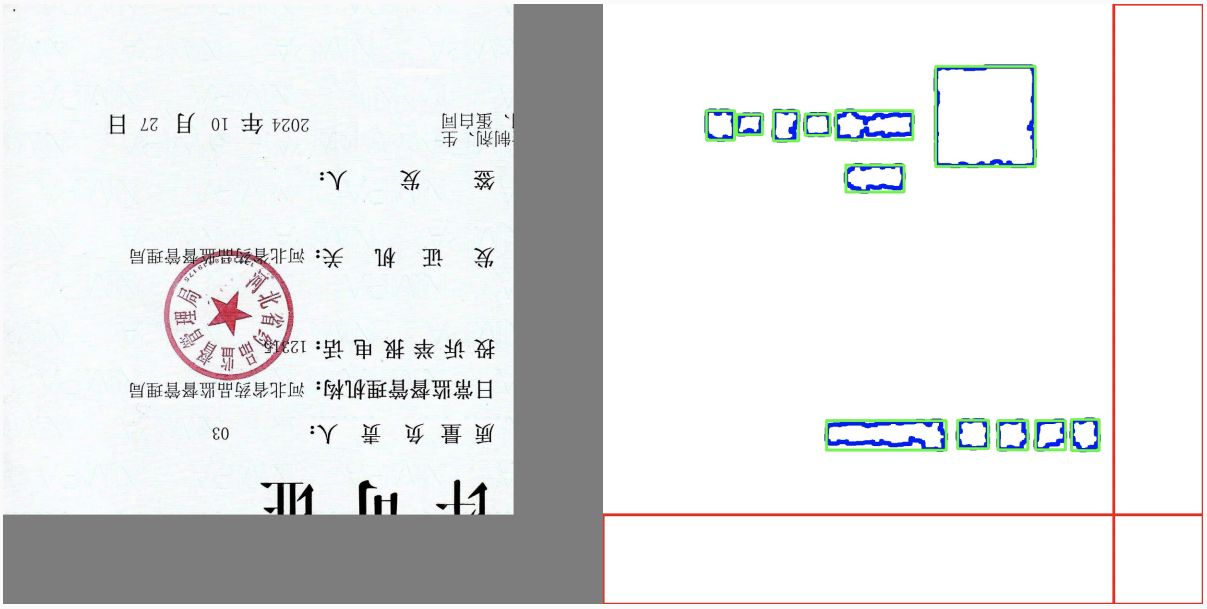
- cv旋转0



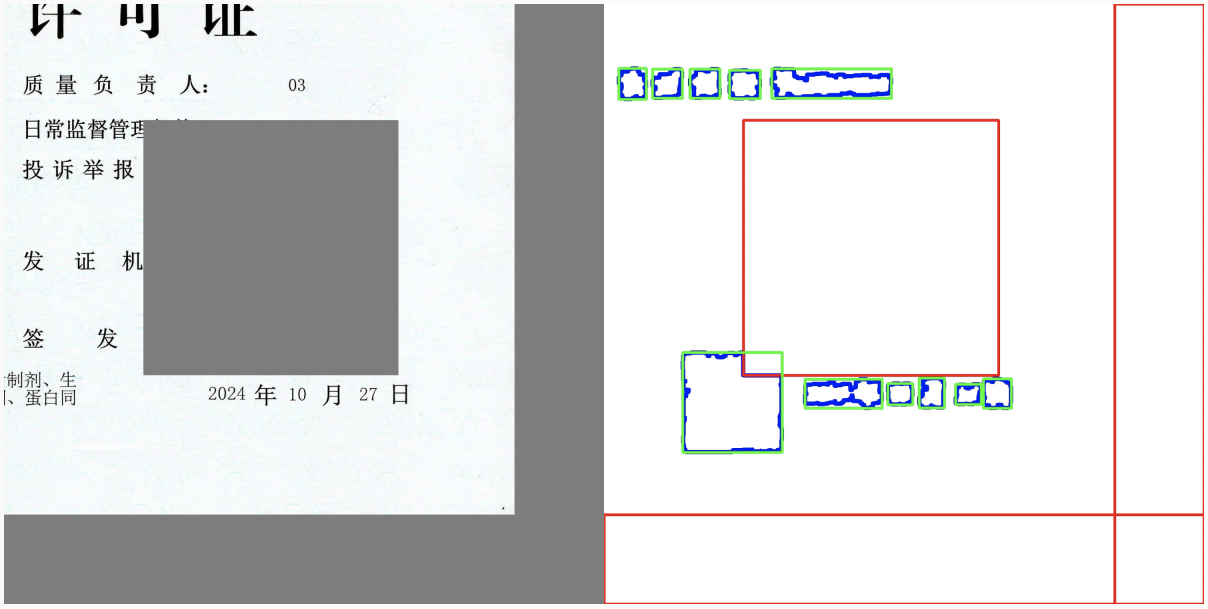
- cv旋转1



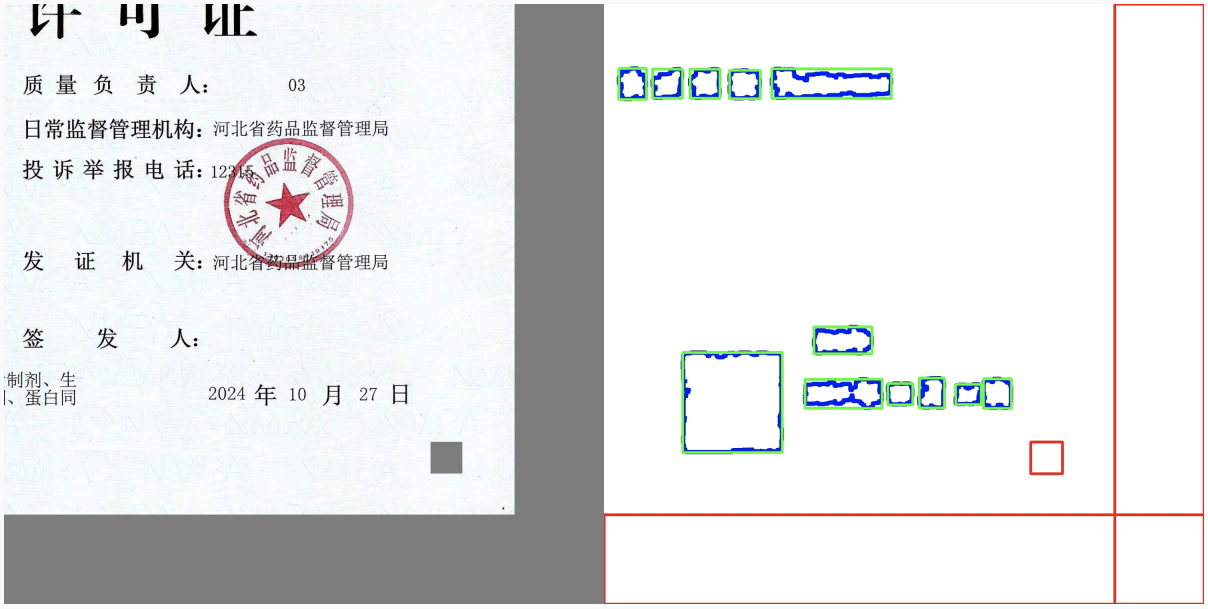
- cv旋转-1



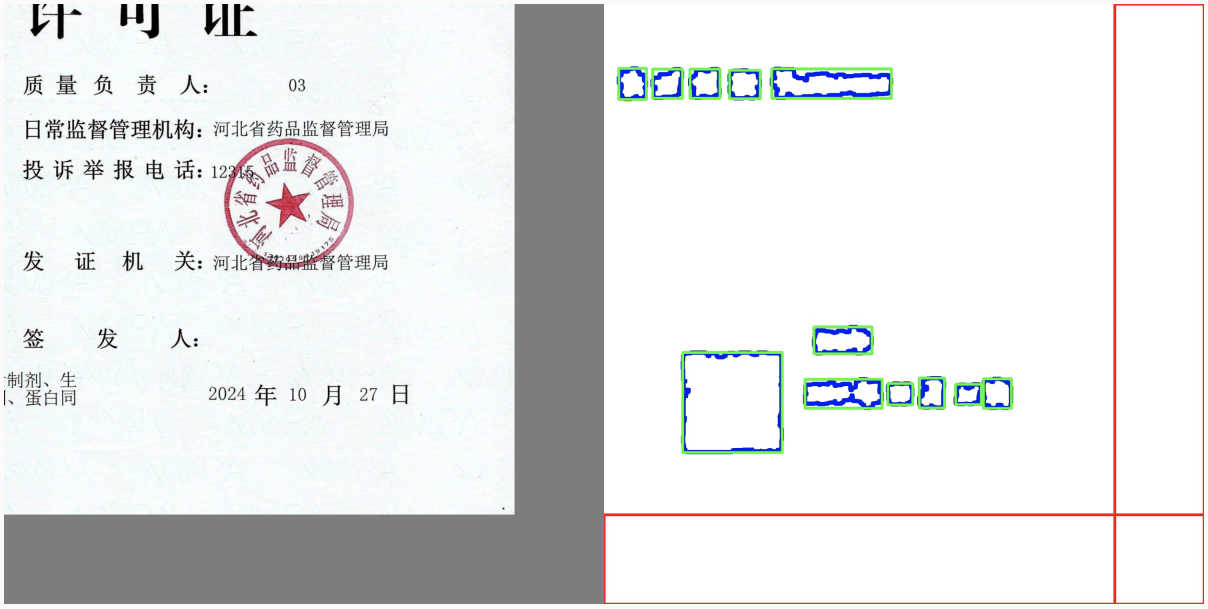
- M2（随机生成1/2边长正方形作为遮挡，遮挡部分定义为class-pmiss）



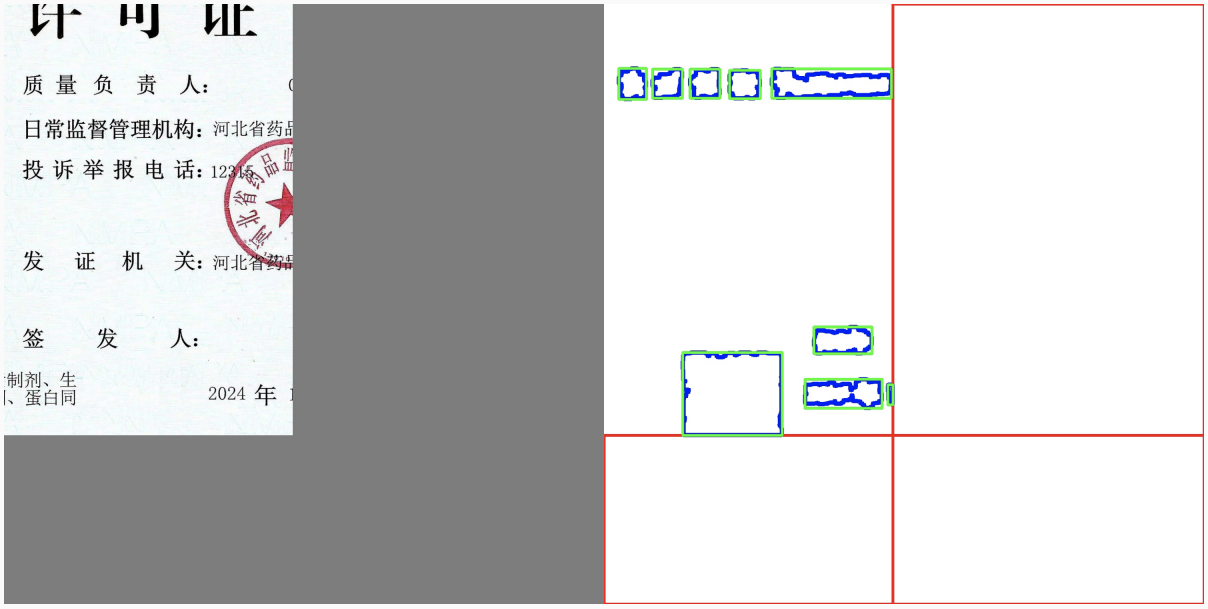
- MF（随机生成1/16边长正方形作为遮挡，遮挡部分定义为class-pmiss）



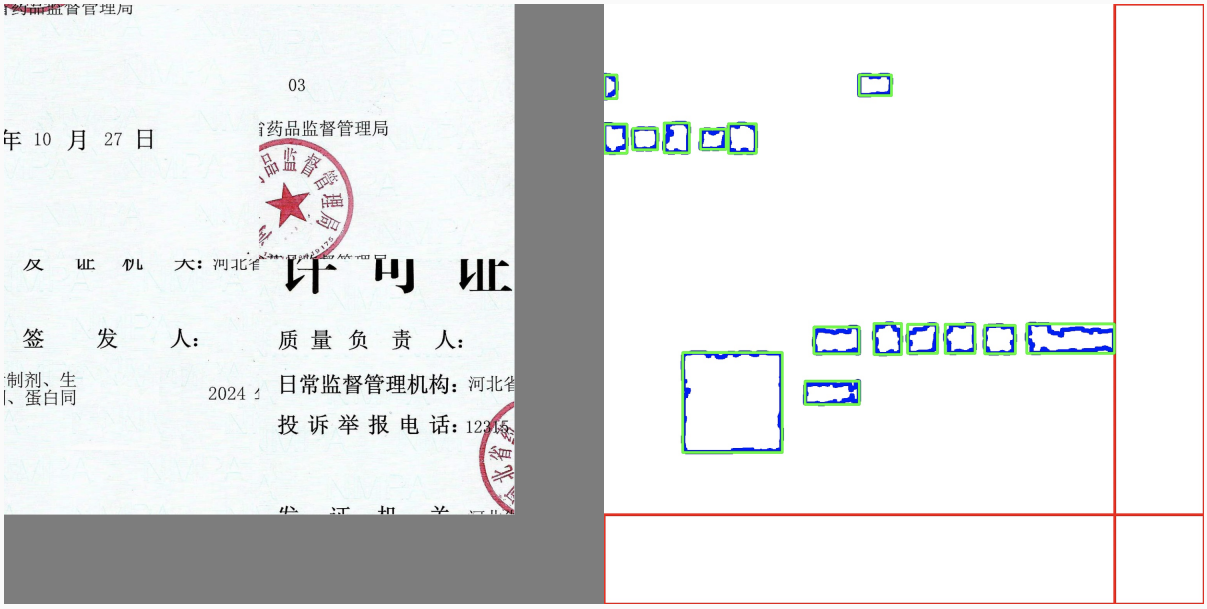
- ~~blur（调整蓝色色调，肉眼不可见）~~



- chicken（逛超市看到鸡块想到的鸡块法，也就是随机删减）



- jely（知名设计师jely提出的1/4随机拼凑）



时间有限，未逐一验证是否都有效，**但本方案最终使用了除blur外全部。**

> 详见 clean_data_code.py

2.4 数据分配

```
1 _n = random.choice(list("0000000012"))
```

随机分配，训练：测试：验证 = 8：1：1.

> 详见 clean_data_code.py

三、复现流程

3.1 环境依赖

3.1.1 docker依赖

```
1 FROM registry.cn-shanghai.aliyuncs.com/tcc-public/pytorch:1.1.0-cu
  da10.0-py3
```

- 核心参数：cuda10.0
- 其他参数同此docker镜像

3.1.2 pip依赖

```
##
RUN apt-get update
RUN apt-get install libglib2.0-dev libsm6 libxrender1 libxext-dev
 -y

## 在构建镜像时安装依赖包
RUN pip install -i https://mirrors.aliyun.com/pypi/simple -r requi
rements/build.txt
RUN pip install -i https://mirrors.aliyun.com/pypi/simple -r requi
rements/runtime.txt
```

- requirements/build.txt

```
pip==20.2.4
cython
pillow>=6.2.0
```

- requirements/runtime.txt

```
torch==1.3.1
torchvision==0.4.2
pandas
tqdm
opencv-python==4.2.0.34
pycocotools
```

3.1.3 mmdetection依赖

```
WORKDIR code/train/env/
RUN pip install mmcv_full-2.3.0+torch1.3.0+cu100-cp36-cp36m-manyli
nux1_x86_64.whl
WORKDIR mmdetection-master
RUN pip install -r requirements/build.txt
WORKDIR ../cocoapi-master/pycocotools/
RUN python setup.py build
RUN python setup.py install --user
WORKDIR ../../mmdetection-master
RUN pip install --no-cache-dir -e . --user
```

核心版本为mmcv_full-2.3.0+torch1.3.0+cu100-cp36-cp36m-manylinux1_x86_64.whl

3.1.4 Dockfile

完整Dockfile如下：

```
# Base Images
## 从天池基础镜像构建
FROM registry.cn-shanghai.aliyuncs.com/tcc-public/pytorch:1.1.0-cuda10.0-py3

## 把当前文件夹里的文件构建到镜像的根目录下
ADD . /

## 指定默认工作目录为根目录（需要把run.sh和生成的结果文件都放在该文件夹下，提交后才能运行）
WORKDIR /

##
RUN apt-get update
RUN apt-get install libglib2.0-dev libsm6 libxrender1 libxext-dev -y

## 在构建镜像时安装依赖包
RUN pip install -i https://mirrors.aliyun.com/pypi/simple -r requirements/build.txt
RUN pip install -i https://mirrors.aliyun.com/pypi/simple -r requirements/runtime.txt

##
WORKDIR code/train/env/
RUN pip install mmcv_full-2.3.0+torch1.3.0+cu100-cp36-cp36m-manylinux1_x86_64.whl
WORKDIR mmdetection-master
RUN pip install -r requirements/build.txt
WORKDIR ../cocoapi-master/pycocotools/
RUN python setup.py build
RUN python setup.py install --user
WORKDIR ../../mmdetection-master
RUN pip install --no-cache-dir -e . --user

```

```
WORKDIR /
## 镜像启动后统一执行 sh run.sh
CMD ["sh", "run.sh"]
```

3.2 代码结构

```
project
    |--README.md             # 解决方案及算法介绍文件，必选
    |--requirements.txt      # Python环境依赖，必选
    |--s2_data
        |--data
            |--test
                |--1.jpg
                |--2.jpg
                ...
                |--1500.jpg
            |--train
                |--1.jpg
                |--2.jpg
                ...
                |--1500.jpg
            |--train_mask
                |--1.png
                |--2.png
                ...
                |--1500.png
    |--user_data
        |--model_data        # 模型文件夹示例，可自行组织
            |--model.pth
        |--tmp_data          # 临时存储文件夹示例，可自行组织
            |--tmp.dat
        |-- 计算的中间文件
        |-- 临时存储目录
    |--prediction_result
        |-- images
            |-- 1.png
            |-- 2.png
            ...
            |--1500.png
```

```
    |--code
        |--train            # 训练代码文件夹示例，可自行组织
        |--test             # 预测代码文件夹示例，可自行组织
        |--run.sh           # 预测执行脚本，必选
        |--train.sh         # 训练执行脚本，必选
        |--clean_data_code.cpp(.py, .java) # 数据清洗代码，可自行取舍
```

同此结构。

3.3 训练复现

3.3.1 训练参数

训练模型为FasterRCNN（resnet50-19c8e357.pth, faster_rcnn_r50_fpn_1x_coco）
获取官方文件后，我们做了部分修改，完整如下：

```
model = dict(
    type='FasterRCNN',
    pretrained='torchvision://resnet50',
    backbone=dict(
        type='ResNet',
        depth=50,
        num_stages=4,
        out_indices=(0, 1, 2, 3),
        frozen_stages=1,
        norm_cfg=dict(type='BN', requires_grad=True),
        norm_eval=True,
        style='pytorch'),
    neck=dict(
        type='FPN',
        in_channels=[256, 512, 1024, 2048],
        out_channels=256,
        num_outs=5),
    rpn_head=dict(
        type='RPNHead',
        in_channels=256,
        feat_channels=256,
        anchor_generator=dict(
            type='AnchorGenerator',
            scales=[8],
            ratios=[0.5, 1.0, 2.0],
```

```
            strides=[4, 8, 16, 32, 64]),
        bbox_coder=dict(
            type='DeltaXYWHBBoxCoder',
            target_means=[0.0, 0.0, 0.0, 0.0],
            target_stds=[1.0, 1.0, 1.0, 1.0]),
        loss_cls=dict(
            type='CrossEntropyLoss', use_sigmoid=True, loss_weig
ht=1.0),
        loss_bbox=dict(type='L1Loss', loss_weight=1.0)),
    roi_head=dict(
        type='StandardRoIHead',
        bbox_roi_extractor=dict(
            type='SingleRoIExtractor',
            roi_layer=dict(type='RoIAlign', output_size=7, sampl
ing_ratio=0),
            out_channels=256,
            featmap_strides=[4, 8, 16, 32]),
        bbox_head=dict(
            type='Shared2FCBBoxHead',
            in_channels=256,
            fc_out_channels=1024,
            roi_feat_size=7,
            num_classes=2,
            bbox_coder=dict(
                type='DeltaXYWHBBoxCoder',
                target_means=[0.0, 0.0, 0.0, 0.0],
                target_stds=[0.1, 0.1, 0.2, 0.2]),
            reg_class_agnostic=False,
            loss_cls=dict(
                type='CrossEntropyLoss', use_sigmoid=False, loss
_weight=1.0),
            loss_bbox=dict(type='L1Loss', loss_weight=1.0))))
train_cfg = dict(
    rpn=dict(
        assigner=dict(
            type='MaxIoUAssigner',
            pos_iou_thr=0.7,
            neg_iou_thr=0.3,
            min_pos_iou=0.3,
            match_low_quality=True,
```

```
            ignore_iof_thr=-1),
        sampler=dict(
            type='RandomSampler',
            num=256,
            pos_fraction=0.5,
            neg_pos_ub=-1,
            add_gt_as_proposals=False),
        allowed_border=-1,
        pos_weight=-1,
        debug=False),
    rpn_proposal=dict(
        nms_across_levels=False,
        nms_pre=2000,
        nms_post=1000,
        max_num=1000,
        nms_thr=0.7,
        min_bbox_size=0),
    rcnn=dict(
        assigner=dict(
            type='MaxIoUAssigner',
            pos_iou_thr=0.5,
            neg_iou_thr=0.5,
            min_pos_iou=0.5,
            match_low_quality=False,
            ignore_iof_thr=-1),
        sampler=dict(
            type='RandomSampler',
            num=512,
            pos_fraction=0.25,
            neg_pos_ub=-1,
            add_gt_as_proposals=True),
        pos_weight=-1,
        debug=False))
test_cfg = dict(
    rpn=dict(
        nms_across_levels=False,
        nms_pre=1000,
        nms_post=1000,
        max_num=1000,
        nms_thr=0.7,
```

```
        min_bbox_size=0),
    rcnn=dict(
        score_thr=0.05,
        nms=dict(type='nms', iou_threshold=0.5),
        max_per_img=100))
dataset_type = 'CocoDataset'
data_root = 'data/coco/'
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375],
to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True),
    dict(type='Resize', img_scale=(1550, 1550), keep_ratio=True)
,
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(
        type='Normalize',
        mean=[123.675, 116.28, 103.53],
        std=[58.395, 57.12, 57.375],
        to_rgb=True),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels'])
]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(1550, 1550),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(
                type='Normalize',
                mean=[123.675, 116.28, 103.53],
                std=[58.395, 57.12, 57.375],
                to_rgb=True),
            dict(type='Pad', size_divisor=32),
```

```
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img'])
        ])
]
data = dict(
    samples_per_gpu=2,
    workers_per_gpu=2,
    train=dict(
        type='CocoDataset',
        ann_file='code/train/env/mmdetection-master/data/coco/annotations/annotations_trai.json',
        img_prefix='code/train/env/mmdetection-master/data/coco/img/',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(type='LoadAnnotations', with_bbox=True),
            dict(type='Resize', img_scale=(1550, 1550), keep_ratio=True),
            dict(type='RandomFlip', flip_ratio=0.5),
            dict(
                type='Normalize',
                mean=[123.675, 116.28, 103.53],
                std=[58.395, 57.12, 57.375],
                to_rgb=True),
            dict(type='Pad', size_divisor=32),
            dict(type='DefaultFormatBundle'),
            dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels'])
        ]),
    val=dict(
        type='CocoDataset',
        ann_file='code/train/env/mmdetection-master/data/coco/annotations/annotations_test.json',
        img_prefix='code/train/env/mmdetection-master/data/coco/img/',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(
                type='MultiScaleFlipAug',
                img_scale=(1550, 1550),
```

```
                flip=False,
                transforms=[
                    dict(type='Resize', keep_ratio=True),
                    dict(type='RandomFlip'),
                    dict(
                        type='Normalize',
                        mean=[123.675, 116.28, 103.53],
                        std=[58.395, 57.12, 57.375],
                        to_rgb=True),
                    dict(type='Pad', size_divisor=32),
                    dict(type='ImageToTensor', keys=['img']),
                    dict(type='Collect', keys=['img'])
                ])
        ]),
    test=dict(
        type='CocoDataset',
        ann_file='code/train/env/mmdetection-master/data/coco/annotations/annotations_vals.json',
        img_prefix='code/train/env/mmdetection-master/data/coco/img/',
        pipeline=[
            dict(type='LoadImageFromFile'),
            dict(
                type='MultiScaleFlipAug',
                img_scale=(1550, 1550),
                flip=False,
                transforms=[
                    dict(type='Resize', keep_ratio=True),
                    dict(type='RandomFlip'),
                    dict(
                        type='Normalize',
                        mean=[123.675, 116.28, 103.53],
                        std=[58.395, 57.12, 57.375],
                        to_rgb=True),
                    dict(type='Pad', size_divisor=32),
                    dict(type='ImageToTensor', keys=['img']),
                    dict(type='Collect', keys=['img'])
                ])
        ]))
evaluation = dict(interval=1, metric='bbox')
```

```
optimizer = dict(type='SGD', lr=0.02, momentum=0.9, weight_decay=0.0001)
optimizer_config = dict(grad_clip=None)
lr_config = dict(
    policy='step',
    warmup='linear',
    warmup_iters=500,
    warmup_ratio=0.001,
    step=[8, 11])
total_epochs = 1
checkpoint_config = dict(interval=1)
log_config = dict(interval=50, hooks=[dict(type='TextLoggerHook')])
dist_params = dict(backend='nccl')
log_level = 'INFO'
load_from = None
resume_from = None
workflow = [('train', 2), ('val', 1)]
work_dir = 'code/train/env/mmdetection-master/work_dirs'
gpu_ids = range(0, 1)
```

- [ ] 可以换用其他更高版本。

完整训练代码见`code/train.sh`，一共分三步：

（1）clean_data_code训练数据处理；

（2）mmdetection初始化，主要导入模型、创建路径、修改文件；

（3）开始训练；

```
# Train
# data
cd /
python code/clean_data_code.py train

# model
mmdetn=1
mmdet=code/train/env/mmdetection-master

mkdir -p /root/.cache/torch/checkpoints
cp user_data/model_data/resnet50-19c8e357.pth /root/.cache/torch/
```

```
checkpoints/resnet50-19c8e357.pth

mkdir -p $mmdet/data/coco
ln -s /user_data/tmp_data/work $mmdet/data/coco/annotations
ln -s /user_data/tmp_data/work $mmdet/work_dirs

cp code/train/coco$mmdetn.py $mmdet/mmdet/datasets/coco.py
cp code/train/class_names$mmdetn.py $mmdet/mmdet/core/evaluation/class_names.py

# 示例total_epochs = 1, 复现需修改
python $mmdet/tools/train.py code/train/faster_rcnn_r50_fpn_1x_coco.py --gpus 1 --work-dir $mmdet/work_dirs
```

其中，

- coco.py需修改：

```
@DATASETS.register_module()
class CocoDataset(CustomDataset):

    # CHANGE BY IVAN
    CLASSES = ('000', '001')
    # CHANGE BY IVAN
```

- class_names.py需修改：

```
# CHANGE BY IVAN
def coco_classes():
    return [
        '000',
        '001'
    ]
# CHANGE BY IVAN
```

3.3.2 训练复现

```
cd /
```

```
sh code/train.sh
```

但应修改`code/train/faster_rcnn_r50_fpn_1x_coco.py`内训练次数（现为1）

3.4 预测复现

3.4.1 预测步骤

（1）clean_data_code预测数据处理，主要修改尺寸；

```
# test
nnn = 0
test_ot = f"{ot}/temp3"
for i in tqdm(os.listdir(f"{dt}/test")):
    if ".jpg" in i:
        i = i.replace(".jpg", "")
        img0 = np.random.randint(0, 1, size=(N,N,3), dtype=np.uint8)
        img0[:, :] = P

        imgb = cv.imread(f"{dt}/test/{i}.jpg")
        i_y, i_x, _ = imgb.shape

        img0[:i_y, :i_x] = imgb
        cv.imwrite(f"{test_ot}/_{i}.jpg", img0)

        nnn += 1
        if nnn > 9999:
            break
```

（2）导入模型-S499、S488、S477，这里的4XX值模型线上分数；

```
#
import os
import cv2 as cv
import numpy as np
import pandas as pd
from tqdm import tqdm
from mmdet.apis import init_detector, inference_detector
```

```
8
9  config_file = "code/train/faster_rcnn_r50_fpn_1x_coco.py"
10 checkpoint_file = "user_data/model_data/S499.pth"
11 model1 = init_detector(config_file, checkpoint_file, device="cud
   a:0")
12 print("M1", model1.CLASSES)
13
14 checkpoint_file = "user_data/model_data/S488.pth"
15 model2 = init_detector(config_file, checkpoint_file, device="cud
   a:0")
16 print("M2", model2.CLASSES)
17
18 checkpoint_file = "user_data/model_data/S477.pth"
19 model3 = init_detector(config_file, checkpoint_file, device="cud
   a:0")
20 print("M3", model3.CLASSES)
```

（3）调用模型，分别获取预测结果；

```
1  #
2  N = 9999
3  p5 = "user_data/tmp_data/temp3/"
4  p6 = "s2_data/data/test/"
5  p7 = "prediction_result/images/"
6
7  dresult1, dresult2, dresult3 = {}, {}, {}
8
9  nnn = 0
10 for _img in tqdm(os.listdir(f"{p5}")):
11     # print(_img)
12     if ".jpg" in _img:
13         _img = _img.replace(".jpg", "").replace("_", "")
14
15         result = inference_detector(model1, f"{p5}_{_img}.jpg")
16         _result = [ij for i, j in enumerate(result) if i == 0 and
   j.shape != (0,5) for ij in j]
17         dresult1[_img] = _result
18
19         result = inference_detector(model2, f"{p5}_{_img}.jpg")
```

```
        _result = [ij for i, j in enumerate(result) if i == 0 and
    j.shape != (0,5) for ij in j]
        dresult2[_img] = _result

        result = inference_detector(model3, f"{p5}_{_img}.jpg")
        _result = [ij for i, j in enumerate(result) if i == 0 and
    j.shape != (0,5) for ij in j]
        dresult3[_img] = _result

        nnn += 1
        if nnn > N:
            break
```

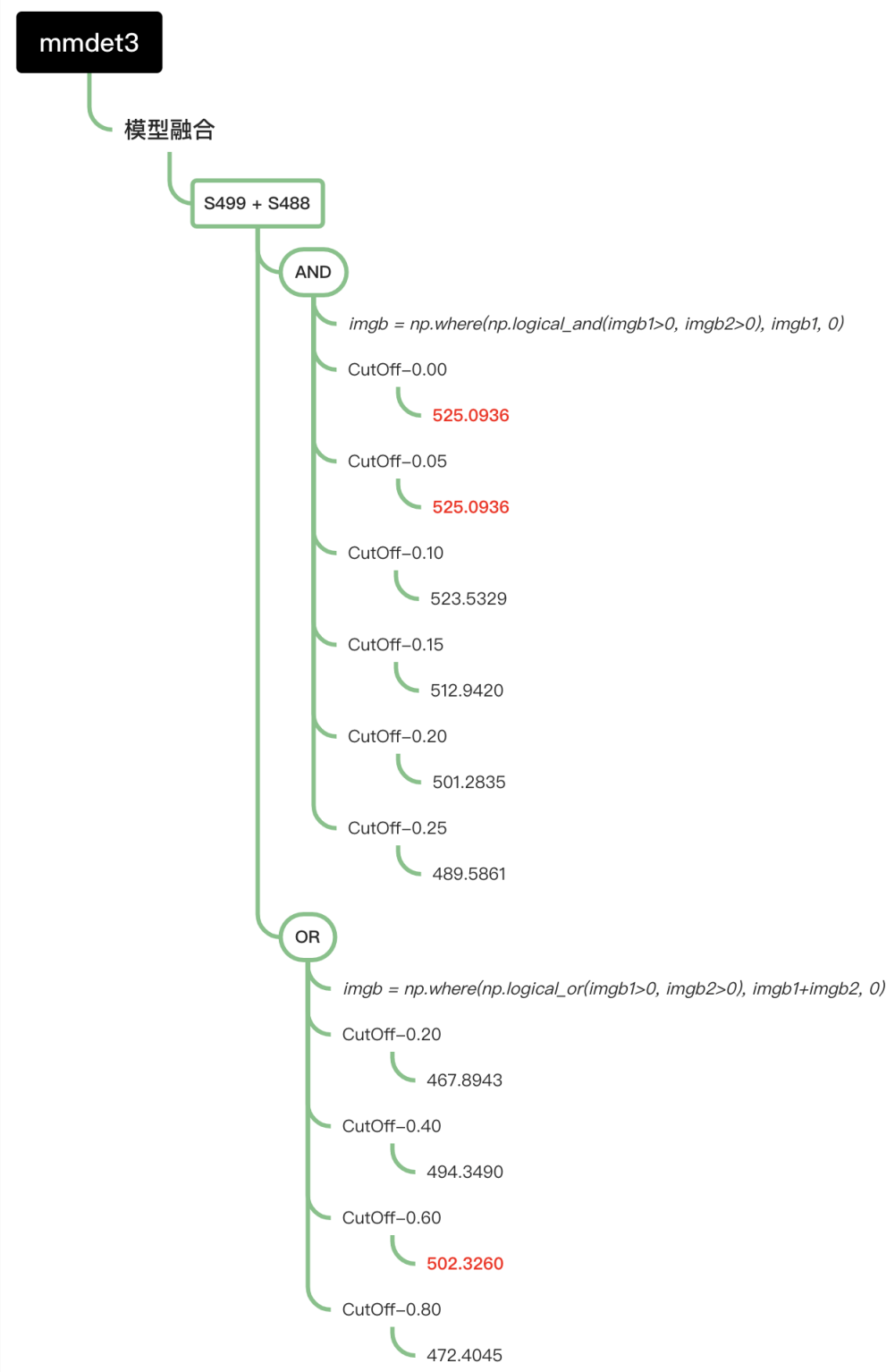
（4）结果输出

这里我们在训练集上做了大量探索，引入模型融合概念。

调整方法有三：

- 预测概率阈值，小icut即放弃；
- 大小限制，预测框过大或过小即放弃；
- 不同合并方式，取三个模型预测mask的并集或交集或其他；

对不同阈值、不同合并方式，线上情况如下：



可提高50分左右。

最终选定预测概率阈值为0，限定预测框大小300-90000，并取三个模型预测mask交集（**最小化、最精确化**）。

```
#
icut = 0.00
```

```
high1, high2 = 90000, 300 # 90000, 300
logs = "and"
r1, r2, r3 = [], [], []

nnn = 0
for _img in tqdm(os.listdir(f"{p5}")):
    if ".jpg" in _img:
        _img = _img.replace(".jpg", "").replace("_", "")



        #
        imgb1 = cv.imread(f"{p6}{_img}.jpg")
        imgb1[:] = 0
        for _iresult in dresult1[_img]:
            ymin, ymax, xmin, xmax = int(_iresult[1]), int(_iresu
lt[3]), int(_iresult[0]), int(_iresult[2])
            if _iresult[4] >= icut and (high1 >= (ymax-ymin)*(xma
x-xmin) >= high2):
                r1.append((ymax-ymin)*(xmax-xmin))
                imgb1[ymin:ymax, xmin:xmax] = bads

        #
        imgb2 = cv.imread(f"{p6}{_img}.jpg")
        imgb2[:] = 0
        for _iresult in dresult2[_img]:
            ymin, ymax, xmin, xmax = int(_iresult[1]), int(_iresu
lt[3]), int(_iresult[0]), int(_iresult[2])
            if _iresult[4] >= icut and (high1 >= (ymax-ymin)*(xma
x-xmin) >= high2):
                r2.append((ymax-ymin)*(xmax-xmin))
                imgb2[ymin:ymax, xmin:xmax] = bads

        #
        imgb3 = cv.imread(f"{p6}{_img}.jpg")
        imgb3[:] = 0
        for _iresult in dresult3[_img]:
            iymin, ymax, xmin, xmax = int(_iresult[1]), int(_ires
ult[3]), int(_iresult[0]), int(_iresult[2])
            if _iresult[4] >= icut and (high1 >= (ymax-ymin)*(xma
x-xmin) >= high2):
```

```
                r3.append((ymax-ymin)*(xmax-xmin))
                imgb3[ymin:ymax, xmin:xmax] = bads


        if logs == "and":
            imgb = np.where(np.logical_and(imgb1>0, imgb2>0, imgb3>0), imgb1, 0)
        if logs == "or":
            imgb = np.where(np.logical_or(imgb1>0, imgb2>0, imgb3>0), imgb1+imgb2+imgb3, 0)

        cv.imwrite(f"{p7}{_img}.png", imgb)

        nnn += 1
        if nnn > N:
            break
```

3.4.1 预测复现

```
# Tests
# data
cd /
python code/clean_data_code.py test

# model
python code/test/result.py
```

```
cd /
sh code/run.sh
```