



河南理工大学

《深度学习基础与实践》

课程设计报告

(2020—2021 学年第 二 学期)

题 目 嫌疑人物品检测

学生姓名 王荣胜、候一鸣、周乐、杨毛强

专业班级 计实验 1801 班

学生学号 311809000608

教师姓名 谷亚楠

成 绩:

评 语:

教师签名:

日期:

一、项目背景

涉案物品的登记与管理问题是公安机关在执法过程中遇到的最多问题之一，由于搜查过程没有统一的规范标准，各地存在对涉案财物管理不够规范，个别地方甚至出现有截留、挪用、调换、损毁或者擅自处理等问题，影响了公安机关执法办案的公信力。在处理过程中，应保证检查时，人民警察不得少于二人，并且在检查过程中只能凭借人民警察个人办案经验进行物品清查，其检查过程还会涉及个人财产安全问题。因此，在物品清查保持公正和不违反法律法规的前提下，应最大限度地保护被检查人员的个人财产安全，提高物品清查的程序化和正当性，此流程应更加的趋于标准化、规范化。

为保障公安执法工作顺利开展，优化公安系统工作中涉案物品管理的规范，本项目通过深度学习方法智能识别涉案物品并用于登记。

本项目通过对涉案物品的智能识别与登记的实现，为公安机关案件相关涉案人员的合法权益提供了有力支撑。其中所应用的背景差分 and 边缘检测技术是目前计算机视觉领域的重要研究课题之一，随着人工智能技术不断取得重大突破，背景差分 and 边缘检测技术准确率、误检率和漏检率等评价指标也在不断提升，如果能将此技术应用于公安系统中，对涉案物品进行智能识别与登记，可以有效提高工作效率，提升检查可靠性，保护个人隐私，同时其检测资料也可以永久留存，促进物品登记与留存的一体化。

此外，该技术不仅局限于应用在公安系统，还能拓展到医院手术室手术物品清点管理工作、后勤部物品管理登记等领域中，该技术的实现可以显著提高该类场景中的工作效率。

二、项目概述

2.1 简介

本项目基于百度 PaddlePaddle 训练模型，通过对包含单个或多个物品的图片进行识别检测，实现涉案物品智能识别与登记，完成公安系统中的物品清查工作。通过对采集到的图片进行自标、上传、训练、再标注及再训练的过程，达到了较高的物品识别精确度。考虑到实际应用中物品的多样性，对于无法识别的物品，设计了操作人员上传标签和图片的功能，对新的数据会进行再训练的操作，不断提高系统识别精确度、增加物品多样性，优化其管理系统。通过该系统的规范化，有效提高相关工作的效率和质量，减少了人力、财力的浪费，具有广阔的应用前景。

2.2 国内外研究现状

美国、英国及欧洲等一些发达国家，自从计算机应用和网络时代初期就开始研究对数字化、信息化方式管理商品销售、政务管理等工作。目前电子政务系统已经在许多发达国家广泛应用，各部门通过电子司法档案系统提供电子司法档

案，查阅，流转，移交，处理各种政务及司法工作事宜，大大提高政府管理效率和水平，效果显而易见。

2020 年，公安部在全国范围内部署各级公安机关开展对涉案财务管理问题专项整治行动，成效十分明显。为巩固集中行动取得成效，全国各地公安机关按照科技强警为指导，以服务实战为导向，借助网络信息技术软件、硬件建设为载体，根据公安部门出台的涉案财务管理办法这一依据，结合办理行政案件、刑事案件实际需要，围绕严格管理、落实责任，强化监督，促进规范，创新管理目标，开始设计研发涉案财物管理系统。该系统实现了涉案财物管理方面的程序化与正规化。

基于百度 PaddlePaddle 运用深度学习方法和数字图像处理技术，模型的识别精确率较高。该系统将改变目前人工识别登记的方式，实现智能登记。

2.3 设计目的和意义

该项目有望替代传统的人工登记模式，提高工作效率。具体来说，该系统的优势主要体现在以下几个方面：

(1) 物品清查规范化、标准化。

可以提高检测准确率以此来提高清查效率，保障工作安全，快速。

(2) 更好的保护个人财产安全。

物品的智能检测登记程序排除人为的对财物截留、挪用、调换、损毁或者擅自处理等操作，使财物的安全性显著提高。

(3) 物品识别登记可靠性高。

检测资料自动形成物品登记，减少人为有意或无意的失误，比人工登记更加可靠。

(4) 通过用户上传图片不断丰富数据库。

对于目前无法识别的物品，设计了操作人员自动上传图片 and 添加类别标签的功能，通过这样的方式不断丰富数据库。

(5) 自动标注锚框和模型更新。

根据用户上传的图片，利用数字图像处理技术检测物体所在位置，进行自动锚框标注。结合用户提供的类别标签，得到标注后的数据，并利用这些新的数据对模型进行更新。

(6) 识别登记过程智能化。

将物品识别与登记工作一并程序化，系统具有反馈功能，可以系统地监测物品识别与登记的过程。

三、模型设计与开发

3.1 数据部分

3.1.1 数据物品种类选择

本项目中我们决定对案件办理过程中常见的物品进行采集和识别。目前考虑了手机、钥匙、水杯、包（小包）、钱包和上述物品的组合。这些物品不仅在案件处理中较为常见，在校园中拍摄也容易获取相关素材。

考虑到个人隐私及安全问题，拟将身份证、银行卡、车票等常见物品图片的采集和识别放到后期处理。

3.1.2 数据采集与处理

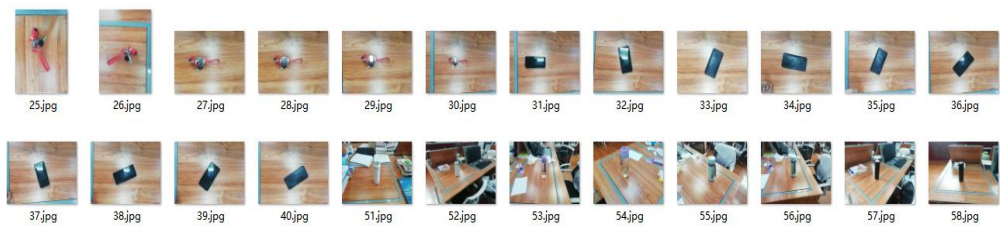


图 1. 数据图片

表 1: 各种类数据统计

| 物品名称 | 手机 | 包 | 钥匙 | 水杯 | 钱包 |
|-------|-----|-----|-----|-----|-----|
| 数量（张） | 292 | 249 | 372 | 336 | 333 |

本次我们共使用 1020 张数据集，各单个类别数据之间大致均衡，有利于后期模型的训练。数据采集具体信息：

- （1）拍摄人员：组织了 60 多个人员进行拍摄；
- （2）拍摄工具：手机拍摄，覆盖了多个手机品牌和型号；
- （3）拍摄要求：手机与物品距离远近适中，保证在 30-50cm；物品之间也要有一定距离，不可重叠；
- （4）拍摄对象：拍摄人员对自己拥有或收集到的物品进行拍摄，因此每个种类的差异较大。如在采集到的图片中，水杯包括塑料水杯、金属水杯、陶瓷水杯、有把手/无把手、有盖/无盖等多种类型。
- （5）拍摄背景：拍摄人员在教室、宿舍、地板等多个场所进行拍摄，单个图片背景相对固定，但数据集中的图片背景差异较大。

数据的多样性和采集的灵活性将赋予模型更好的泛化能力，使之适用于更加广泛的使用场景。

为了减轻后期模型开发的庞大负担任务，我们对采集到的图像数据进行了降低分辨率的批量处理。

3.1.3 数据标注

对于采集到的图像数据，我们在本地环境使用 Labelimg 对图像中的物体进行 voc 格式的标注。其标注过程为：

- (1) 预先导入标签到 Labelimg 下的 txt 文件。
- (2) 利用 Labelimg 进行标注并进行保存。

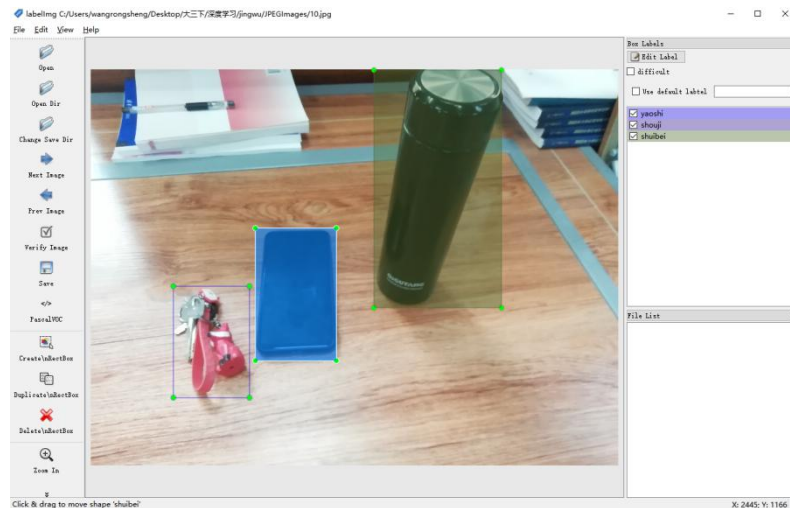
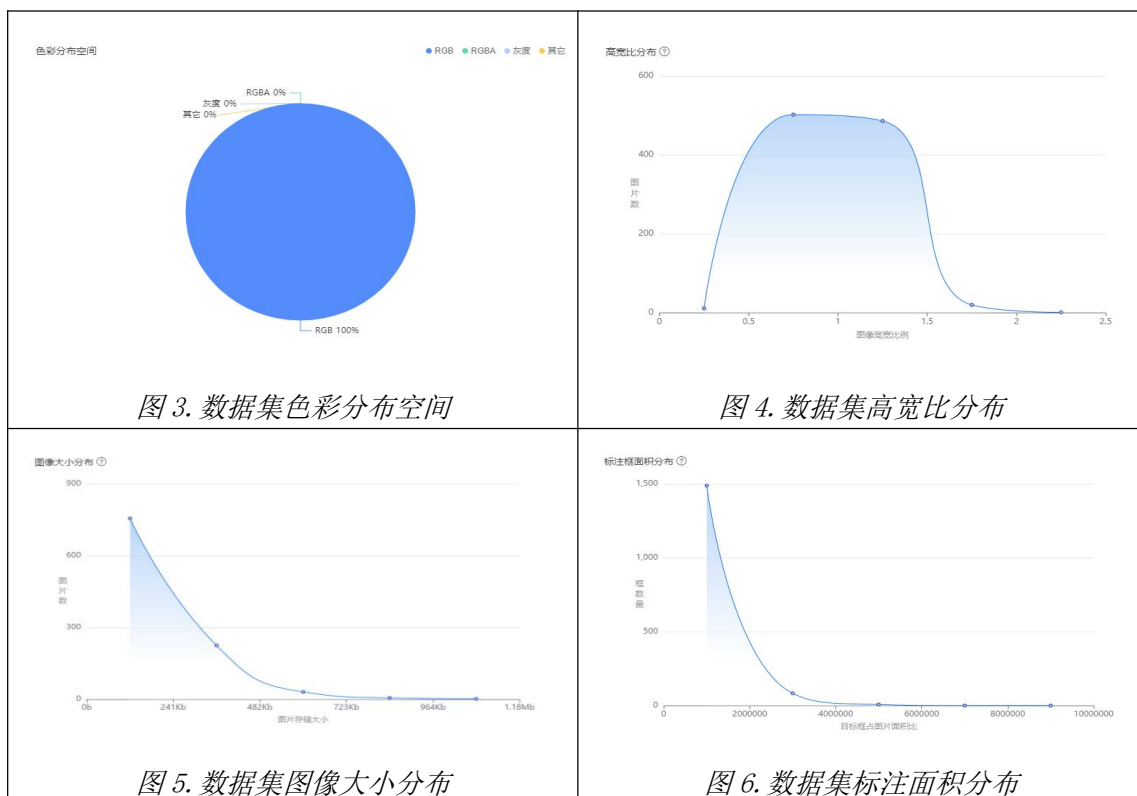
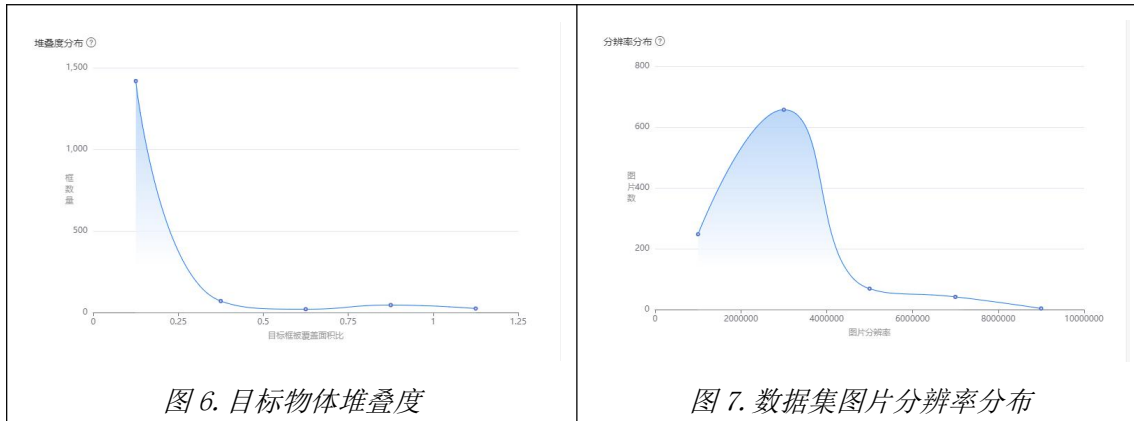


图 2. 标注工具 Labelimg

3.1.4 数据可视化统计





3.1.5 标注文件修正

由于本地标注数据的 path 和采用 AIStudio 训练的路径不同，因此需要对标注文件进行简单的 path 修正：

```
1. !mkdir dataset/Annotations1
2. import xml.dom.minidom
3. import os
4.
5. path = r'dataset/Annotations'
6. sv_path = r'dataset/Annotations1'
7. files = os.listdir(path)
8. cnt = 1
9.
10. for xmlFile in files:
11.     dom = xml.dom.minidom.parse(os.path.join(path, xmlFile))
12.     root = dom.documentElement
13.     item = root.getElementsByTagName('path')
14.     for i in item:
15.         i.firstChild.data = 'dataset/JPEGImages/' + str(cnt).zfill(6) + '.jpg'
16.
17.     with open(os.path.join(sv_path, xmlFile), 'w') as fh:
18.         dom.writexml(fh)
19.     cnt += 1
```

3.1.6 数据匹配修正

此情况对应解决的是数据图片和数据标注文件不匹配问题，对于检出的未匹配数据进行删除，以保证训练的正常进行：

```
1. import os,shutil
2.
3. jpeg = 'dataset/JPEGImages'
```

```

4. jpeg_list = os.listdir(jpeg)
5.
6. anno = 'dataset/Annotations'
7. anno_list = os.listdir(anno)
8.
9. for pic in jpeg_list:
10.     name = pic.split('.')[0]
11.     anno_name = name + '.xml'
12.     #print(anno_name)
13.     if anno_name not in anno_list:
14.         os.remove(os.path.join(jpeg,pic))

```

3.1.7 数据划分

为了简化模型开发，我们采用 PaddleX 的数据集分割工具进行划分：

```
1. paddlex --split_dataset --format VOC --dataset_dir dataset --val_value 0.2 --test_value 0.0
```

在这里，我们将 1020 张数据集划分成了 816 张训练集和 204 张验证集。对于测试集，我们则在后期进行相同拍摄环境但未标注的图片数据和爬虫得来的随机图像数据进行模型的测试工作。

3.2 模型部分

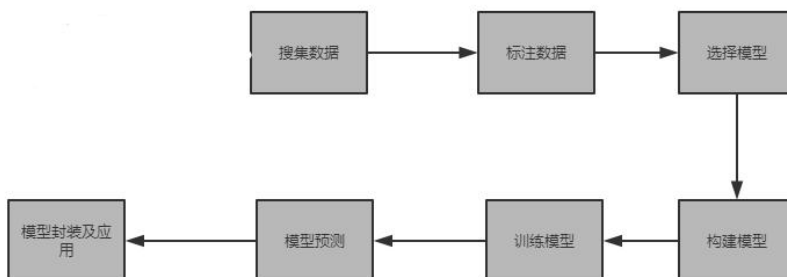


图 8. 模型全流程开发过程

3.2.1 模型选择

常用的目标检测模型有如下表 2：

| 模型类型 | | | | |
|-----------|-------------|--------|---------|-----|
| One-Stage | YOLOv3 | YOLOV4 | PP-YOLO | SSD |
| Two-Stage | Faster-RCNN | | | |

常用模型主干网络：

- ResNet (&vd)
- ResNeXt (&vd)
- SENet
- DarkNet
- VGG
- MobileNetv1/v3
- Efficientnet

One-Stage 类型的目标检测模型可以直接回归物体的类别概率和位置坐标值（无 region proposal），但准确度低，速度相比 Two-Stage 快。但是精度相对 Two-Stage 目标检测网络略低。作为 One-Stage 类型的经典代表之一的就是 YOLOv3 网络，YOLOv3 是整个 YOLO 系列模型的最经典，应用最广泛的网络。YOLOv3 在 v1, v2 基础上没有太多的创新，主要是借鉴一些好的方案融合到 YOLO 里面。不过效果还是不错的，在保持速度优势的前提下，提升了预测精度，尤其是加强了对小物体的识别能力。因此为了尽可能保证精度和速度，我们本次选用 YOLOv3 模型。

在目标检测中，目标检测器通常使用主干网络来提取基本特征，进而用于目标检测，而这些主干网络通常是图像分类任务设计，并在 ImageNet 数据集上进行预训练。因此，如果主干网络能够提取更具表示性的特征，则检测器将获得更好的性能。也就是说，主干网络越强大，目标检测性能越好。在项目中，我们选择了 YOLOv3 的默认主干网络 DarkNet53，对于其它主干网络，例如 ResNet 等，它们在图像分类任务中表现已经非常出色，但是作为原配的 DarkNet53 主干网络是 YOLOv3 的核心组件，DarkNet53 每一个卷积使用了特有的 DarkNetConv2D 结构，这是其它主干网络不能提供的。

3.2.2 模型训练

对于模型训练，我们选择百度 PaddlePaddle 团队开发的目标检测套件 PaddleDetection 进行简单配置训练，除了 PaddleDetection 套件，也可以选择全流程开发工具 PaddleX 作为模型训练的选择，但是 PaddleX 作为一个代码级别的工具，所必须要的是超参数的调整，参数的调整很大程度上会影响模型的最终识别效果，鉴于我们调参经验缺乏，所以 PaddleDetection 是我们最好的选择。

(1) 下载套件：

```
1. git clone https://gitee.com/paddlepaddle/PaddleDetection.git
```

(2) 安装环境：

```
1. pip install -r requirements.txt -q
```

(3) 训练配置：

作为一个国内深度学习框架的最前排团队，PaddleDetection 除了代码的解耦优化，PaddleDetection 也进行了配置文件的解耦，解耦后的配置文件可以更加便捷快速的进行组建、训练、优化及部署等全开发流程。我们只需要配置我们核心的文件就可以：

①数据配置文件：

```
1. metric: COCO # 默认 COCO 类型数据
```



```

2. num_classes: 5 # 配置, 参考Label 数量填写类别数量
3.
4. TrainDataset: # 训练集
5. !COCODataset
6. image_dir: /home/aistudio/work/dataset/JPEGImages
7. anno_path: /home/aistudio/work/dataset/voc_train.json
8. dataset_dir: /home/aistudio/work/dataset
9. data_fields: ['image', 'gt_bbox', 'gt_class', 'is_crowd']
10.
11. EvalDataset: # 验证集
12. !COCODataset
13. image_dir: /home/aistudio/work/dataset/JPEGImages
14. anno_path: /home/aistudio/work/dataset/voc_val.json
15. dataset_dir: /home/aistudio/work/dataset
16.
17. TestDataset: # 测试集
18. !ImageFolder
19. anno_path: /home/aistudio/work/dataset/voc_test.json

```

②runtime 配置:

```

1. use_gpu: true # gpu 使用
2. log_iter: 10 # 训练日志文件
3. save_dir: output # 模型保存
4. snapshot_epoch: 30 # 模型保存轮次

```

③其它配置:

除了上述的配置, 其它配置我们保持默认, YOLOv3 默认的配置文件的在在经过百万数据和多次测试获得的优秀配置数据, 在非专业情况下, 建议不做修改。当然, 在一些特殊任务中(如: 小目标物体检测)可以修改这些配置文件, 增加下游网络提取的层数等, 以便更加符合开发任务。

(4) 训练:

```

1. python -u tools/train.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml --eval

```

```

4783 data_coco: 2.2477 lpr: 0.2247 lmaper/a
[04/24 18:12:42] ppnet-engine INFO: Epoch: (115) [ 0/96] learning_rate: 0.000000 loss_pty: 2.098602 loss_why: 0.766500 loss_hm2: 2.920420 loss_cls: 1.229204 loss: 7.455565 eta: 6:28:15 batch_count: 2,
5645 data_coco: 2.1245 lpr: 0.3750 lmaper/a
[04/24 18:14:08] ppnet-engine INFO: Epoch: (115) [10/96] learning_rate: 0.000000 loss_pty: 2.715849 loss_why: 0.782024 loss_hm2: 0.554176 loss_cls: 1.009719 loss: 6.962740 eta: 6:27:46 batch_count: 2,
6622 data_coco: 2.4022 lpr: 0.0000 lmaper/a
[04/24 18:16:39] ppnet-engine INFO: Epoch: (115) [20/96] learning_rate: 0.000000 loss_pty: 2.259132 loss_why: 0.583020 loss_hm2: 2.996617 loss_cls: 0.872170 loss: 6.893294 eta: 6:27:23 batch_count: 2,
7600 data_coco: 2.4752 lpr: 0.4510 lmaper/a
[04/24 18:19:03] ppnet-engine INFO: Epoch: (115) [30/96] learning_rate: 0.000000 loss_pty: 2.139613 loss_why: 0.529620 loss_hm2: 3.580087 loss_cls: 1.111332 loss: 6.095566 eta: 6:26:59 batch_count: 2,
8578 data_coco: 2.6472 lpr: 2.4030 lmaper/a
[04/24 18:21:38] ppnet-engine INFO: Epoch: (115) [40/96] learning_rate: 0.000000 loss_pty: 2.418381 loss_why: 0.719719 loss_hm2: 2.784931 loss_cls: 0.645907 loss: 6.472565 eta: 6:26:39 batch_count: 2,
9556 data_coco: 2.4704 lpr: 0.0000 lmaper/a
[04/24 18:24:09] ppnet-engine INFO: Epoch: (115) [50/96] learning_rate: 0.000000 loss_pty: 2.097045 loss_why: 0.477723 loss_hm2: 2.896147 loss_cls: 0.834173 loss: 6.023566 eta: 6:26:12 batch_count: 2,
10534 data_coco: 2.2780 lpr: 0.2245 lmaper/a
[04/24 18:26:42] ppnet-engine INFO: Epoch: (120) [ 0/96] learning_rate: 0.000000 loss_pty: 2.409360 loss_why: 0.949745 loss_hm2: 2.920886 loss_cls: 0.898950 loss: 7.348740 eta: 6:25:48 batch_count: 2,
11512 data_coco: 2.1588 lpr: 0.3587 lmaper/a
[04/24 18:29:13] ppnet-engine INFO: Epoch: (120) [10/96] learning_rate: 0.000000 loss_pty: 2.495108 loss_why: 0.894955 loss_hm2: 0.526069 loss_cls: 0.943147 loss: 7.331877 eta: 6:25:19 batch_count: 2,
12490 data_coco: 2.4070 lpr: 0.1016 lmaper/a
[04/24 18:31:43] ppnet-engine INFO: Epoch: (120) [20/96] learning_rate: 0.000000 loss_pty: 2.743556 loss_why: 0.881951 loss_hm2: 2.783551 loss_cls: 0.849232 loss: 6.899310 eta: 6:24:45 batch_count: 2,
13468 data_coco: 2.1550 lpr: 0.1679 lmaper/a
[04/24 18:34:15] ppnet-engine INFO: Epoch: (120) [30/96] learning_rate: 0.000000 loss_pty: 2.472626 loss_why: 0.870377 loss_hm2: 2.897270 loss_cls: 0.872353 loss: 7.484000 eta: 6:24:13 batch_count: 2,
14446 data_coco: 2.1046 lpr: 0.1245 lmaper/a
[04/24 18:36:46] ppnet-engine INFO: Epoch: (120) [40/96] learning_rate: 0.000000 loss_pty: 2.398666 loss_why: 0.583208 loss_hm2: 2.498658 loss_cls: 0.644929 loss: 6.391897 eta: 6:23:44 batch_count: 2,
15424 data_coco: 2.3780 lpr: 0.3787 lmaper/a
[04/24 18:39:18] ppnet-engine INFO: Epoch: (120) [50/96] learning_rate: 0.000000 loss_pty: 2.535393 loss_why: 0.582285 loss_hm2: 2.748317 loss_cls: 0.878383 loss: 6.931439 eta: 6:23:15 batch_count: 2,
16402 data_coco: 2.4145 lpr: 0.5247 lmaper/a

```

图 9. 训练日志输出

在本次的嫌疑人物品检测识别任务中, 我们的模型共训练了 270 个轮次, Batch_Size 为 24, 模型每 30 轮次评估一次并保存, 共用时 11 小时 30 分钟。在训练过程中, 我们配置了边训练边评估, 训练中最好的评估 ap 为 0.792, 也就是在第 270 个轮次出现的评估结果。目前该模型还可以继续通过训练继续提升 ap 分, 在实际开发中 ap 达到 0.80 就基本可以满足需求, 所以当前我们的模型效果是相当可观的。

(5) 评估:

模型的评估除了在训练过程中的边训练便评估我们还可以对于训练好的模型进行简单的评估,此项目中,我们借助 PaddleDetection 的便捷开发进行了我们项目模型的评估:

```
1. python tools/eval.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml -o weights=output/best792/best_model.pdparams

Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.795
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.995
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.970
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.622
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.802
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.803
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.826
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.826
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.635
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.833
[06/28 14:21:48] ppdet.engine INFO: Total sample number: 204, averge FPS: 25.517924068939113
```

图 10. 模型评估结果

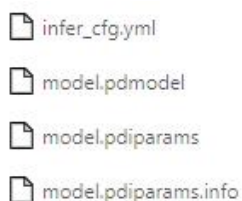
评估指标基本合理且表现较好。

(6) 导出:

当前为了模型的开发,各大深度学习框架,例如 Tensorflow 等都提供了动态图机制,所谓动态图机制,就是基于图进行模型的训练,这是非常有利于机器理解 and 学习的,但是对人或者开发者不是很友好。百度 Paddle 具有高瞻远瞩的见识,也建立有动态图和静态图机制,为了后期更便利的模型测试等工作,我们将训练好的动态图转化为静态图:

```
1. python tools/export_model.py -c configs/yolov3/yolov3_darknet53_270e_coco.yml -o weights=output/best792/best_model.pdparams
```

模型转化生成的静态图结果为:



- infer_cfg.yml
- model.pdmodel
- model.pdiparams
- model.pdiparams.info

图 11. 模型导出结果

- Infer_cfg.yml: 模型的配置文件,包括输入图片的大小等;
- Model_pdmodel: 模型架构文件;
- Model_pdiparams: 模型参数文件;
- Model_pdiparams.info: 额外的模型参数文件,可以忽略;

3.2.3 模型测试

模型测试是将模型应用到工业生产的最后一步，也是非常必要的一步，它可以检验我们模型的泛化性能，基于我们最初的想法，我们进行了单一物品和复杂物品的检验工作。

单一环境模型测试：

| 原始图片 | 检测结果图片 |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
|  |  正确识别 shuibei |
|  |  正确识别 yaoshi |
|  |  正确识别 shouji |

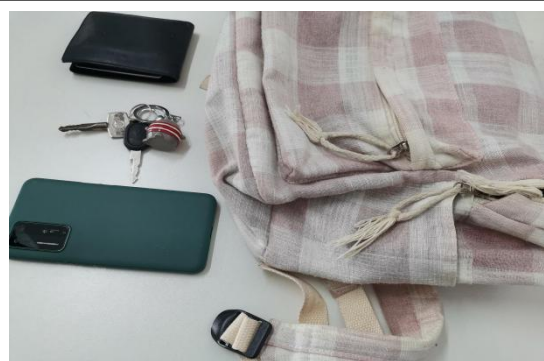
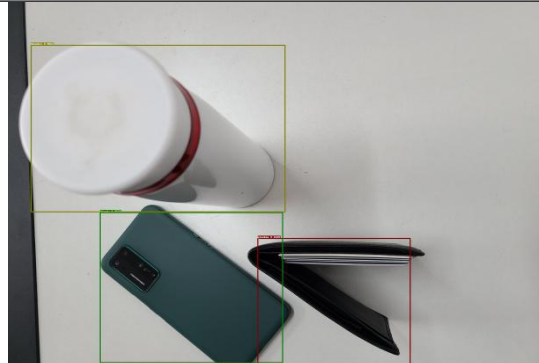


正确识别 bao

复杂环境模型测试（与训练集相同拍摄环境但未标注的图片数据）：

原始图片

检测结果图片





复杂环境模型测试（随机图片数据）：

原始图片

检测结果图片




右侧 shuibeizi 识别为 bao



未识别 shuibeizi、bao



| | |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
|  | <p>未识别 yaoshi</p>  <p>未识别 bao</p> |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|

由以上单一物品、两种不同环境下复杂环境物品检测，我们可以看出：

1. 单一物品识别检测并正确表现非常好；
2. 在相同环境拍摄的图像复杂情况下，识别效果也可以大多表现正确；
3. 在随机图像的复杂情况下，部分物品未识别检出，也存在部分识别错误；

四、项目改进

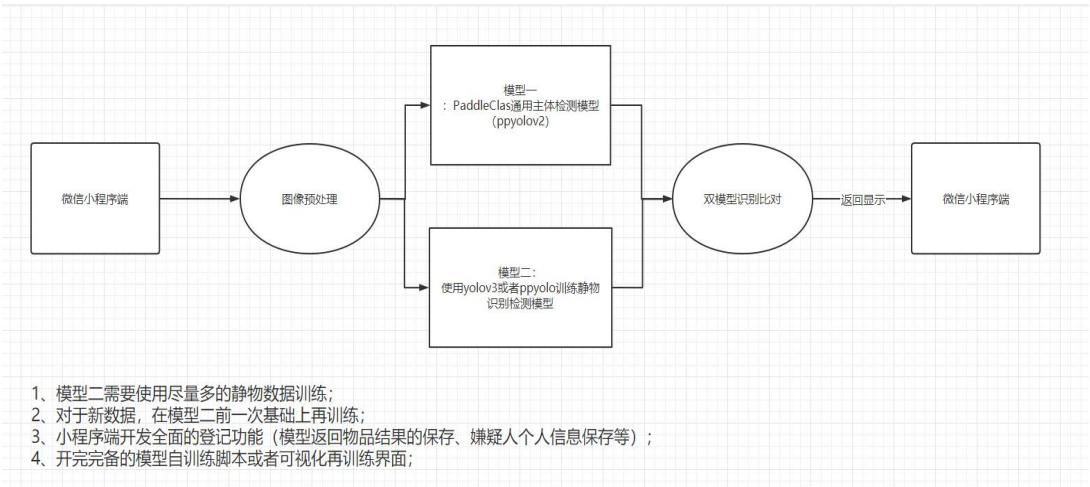
4.1 数据改进

在前面，我们已经提到，为了减少后期模型训练的压力，我们对数据图片进行了分辨率降低处理。在初代模型训练完成后，我们利用测试图对模型进行了测试（见 3.2.3 中复杂环境检验第二部分），从测试结果中，有检测识别不到物体情况，我们猜测大概率是因为分辨率降低，使得模型的泛化能力对于低分辨率图片中物体不再敏感所致。后期考虑使用原图数据或者采用不大规模降低分辨率方法改进模型精度。

4.2 模型改进

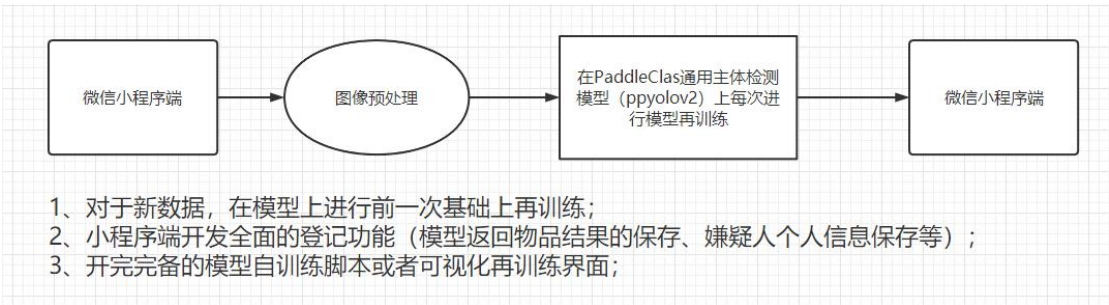
对于模型方面的改进策略，我们提出了两种新的思路：第一种思路是双模型方法；第二种是单模型在上次基础上再训练方法。

4.2.1 双模型方法



借助 PaddleClas 通用识别模型与我们自构建的模型进行静物识别, 对于两个模型识别数量相等和标签相同情况下, 模型对结果进行正确返回。对于识别数量相等但标签不同情况下, 模型输出相同结果, 并要求人工介入。对于识别相同但数量不同情况, 模型输出相同结果, 并要求人工介入。依托双模型的优秀识别能力, 再加上全面的微信小程序端开发, 可以在此基础上拓展完备的嫌疑人信息记录程序。

4.2.2 单模型再训练方法



仍然依托 PaddleClas 通用识别模型, 我们采用迁移学习思想, 每次在新数据投入之后, 在上次模型基础上进行再次训练迭代。除了有完备的模型模型训练策略。

我们还设想了, 利用爬虫或者模拟操作方法, 设计友好交互的开发界面或者利用脚本, 设置定时任务, 对于每次新数据数量达到阈值后模型开启自训练进程, 打通模型的自我训练, 部署等全系列流程。

五、项目分工

王荣胜: 控制总体任务进度, 模型训练与报告撰写;

候一鸣: 数据标注、数据预处理;

周乐：数据标注、模型测试；

杨毛强：数据标注、数据纠正、项目成果整理；