上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

书脊编号在移动端的实时批量读取

汇报人：周睿文

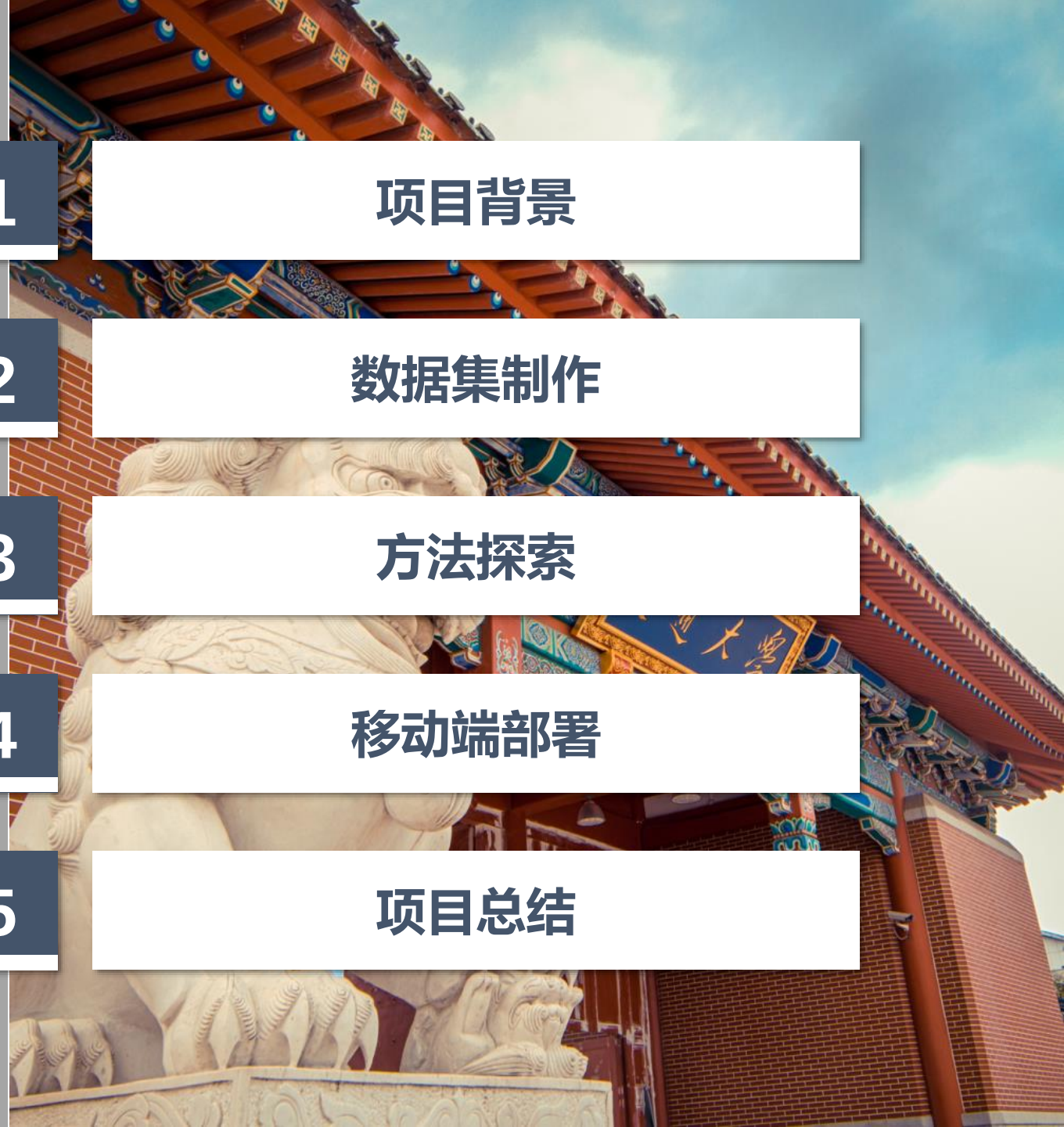
小组成员：丁立，陈梓煜，赵重印，王彦竣

2021年6月11日

饮 水 思 源 • 爱 国 荣 校

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录



1 项目背景

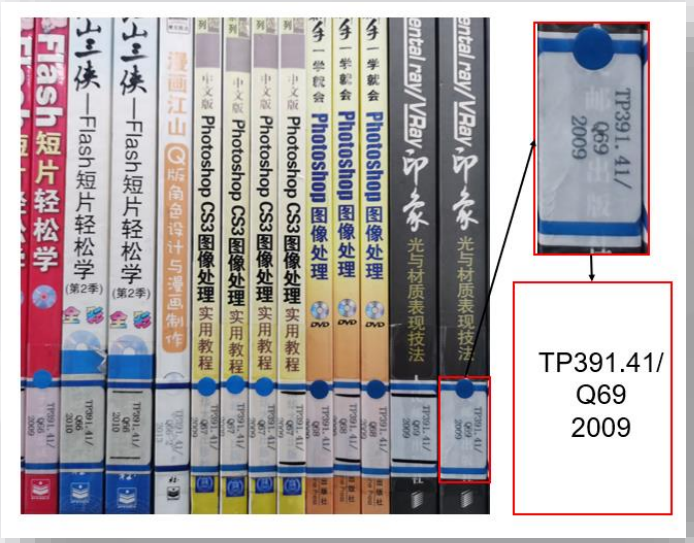
2 数据集制作

3 方法探索

4 移动端部署

5 项目总结

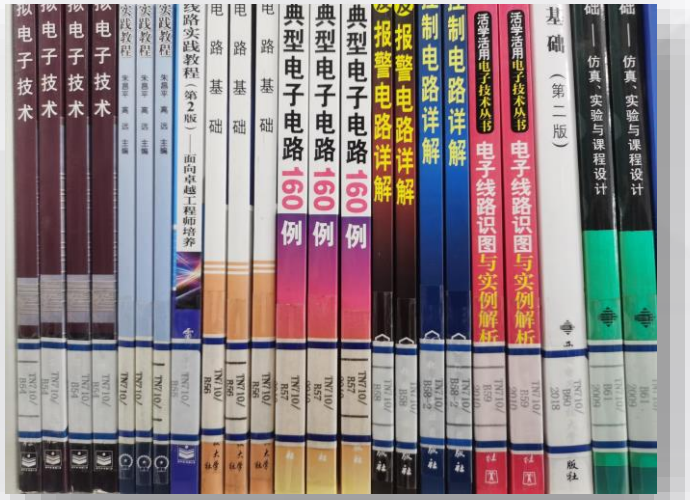
书脊编号识别项目背景



- 需求与应用场景： 图书馆的大批量书籍整理归位困难
- 问题定位：
  - 精确定位图片中书脊标签，并识别标签中的字符信息
  - 在移动端实现功能并保证流畅性
- 项目难点：
  - 移动端算力的限制 – 算法计算量级
  - 移动端流畅性的保证 – OCR集成方式
  - 移动端与pc端的差异 – 算法迁移部署

书脊编号数据集制作

- 在新图 B 区共拍摄了近 650 张含书脊编号的照片



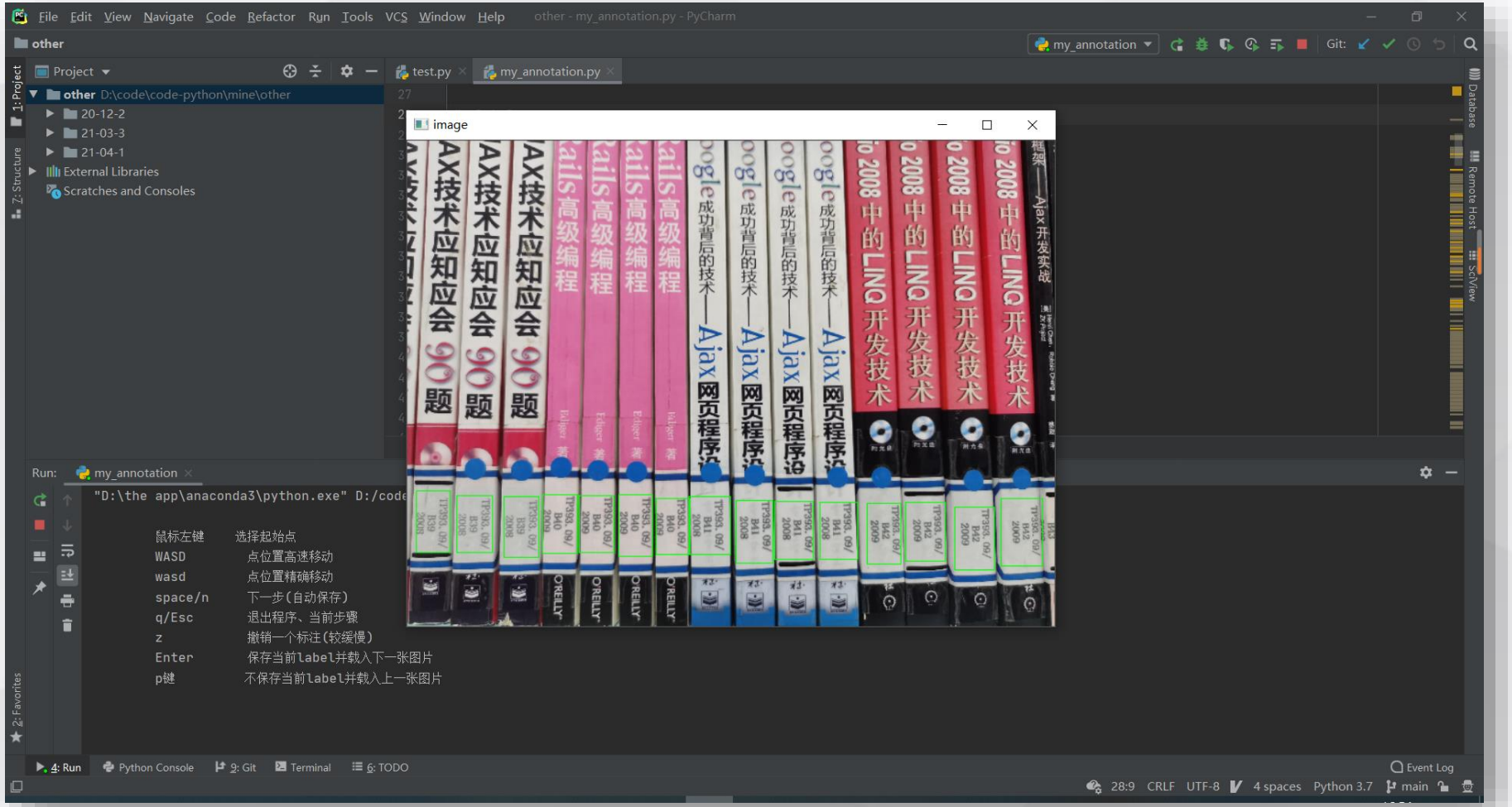
宽视野图片
约500张

窄视野图片
约150张

数据集中干扰
较大的图片

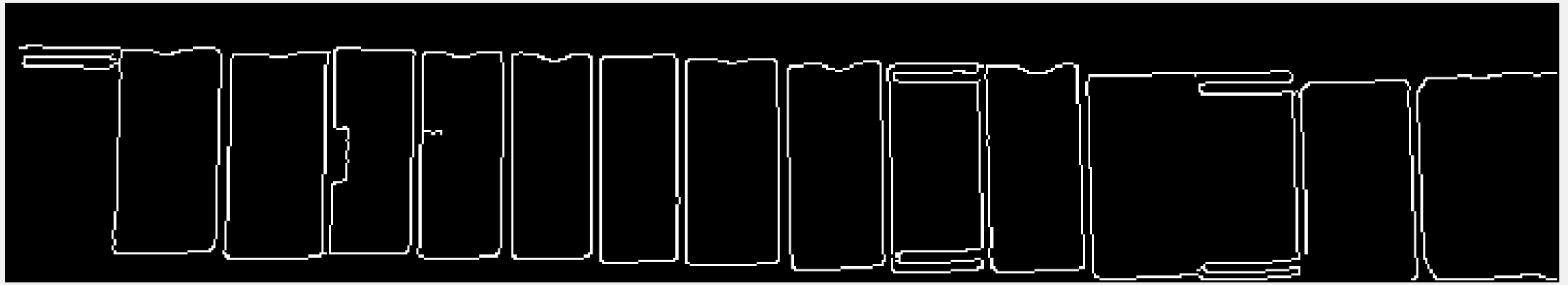
书脊编号数据集制作

- 人工标注约 510 张照片的书脊编号定位框



传统图像处理方法的缺陷

- 检测定位书脊标签边缘的精度差
- 许多书脊标签有破损、模糊，难以识别
- 对其他区域的边缘（如浅色基底的书脊）没有抵抗性
- 在**含有书脊标签编号的区域**得到的效果



深度学习框架的选择

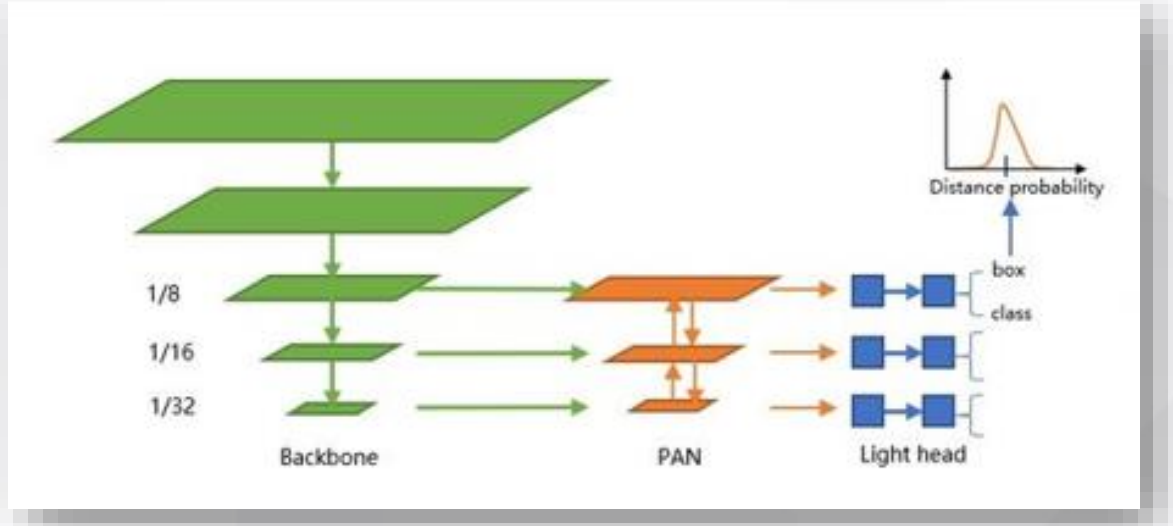
NanoDet 相比于 Yolo, RCNN 等的优势.

- 模型轻量：模型文件只有 980KB (INT8 dtype) 或 1.8MB (FP16 dtype)
- 推理高效：在移动 ARM CPU 上为 97 FPS (10.23 ms / frame)
- 易于部署：提供后端的 C++ 实现。

Model	Resolution	COCO mAP	Latency	FLOPs	Params	Model Size
NanoDet-m	416*416	**26.8**	**21.53ms**	**2.42G**	**2.08M**	**3.9MB**
YoloV3-Tiny	416*416	16.6	37.60ms	5.62G	8.86M	33.7MB
YoloV4-Tiny	416*416	21.7	32.81ms	6.96G	6.06M	23.0MB

NanoDet 与迁移学习方案

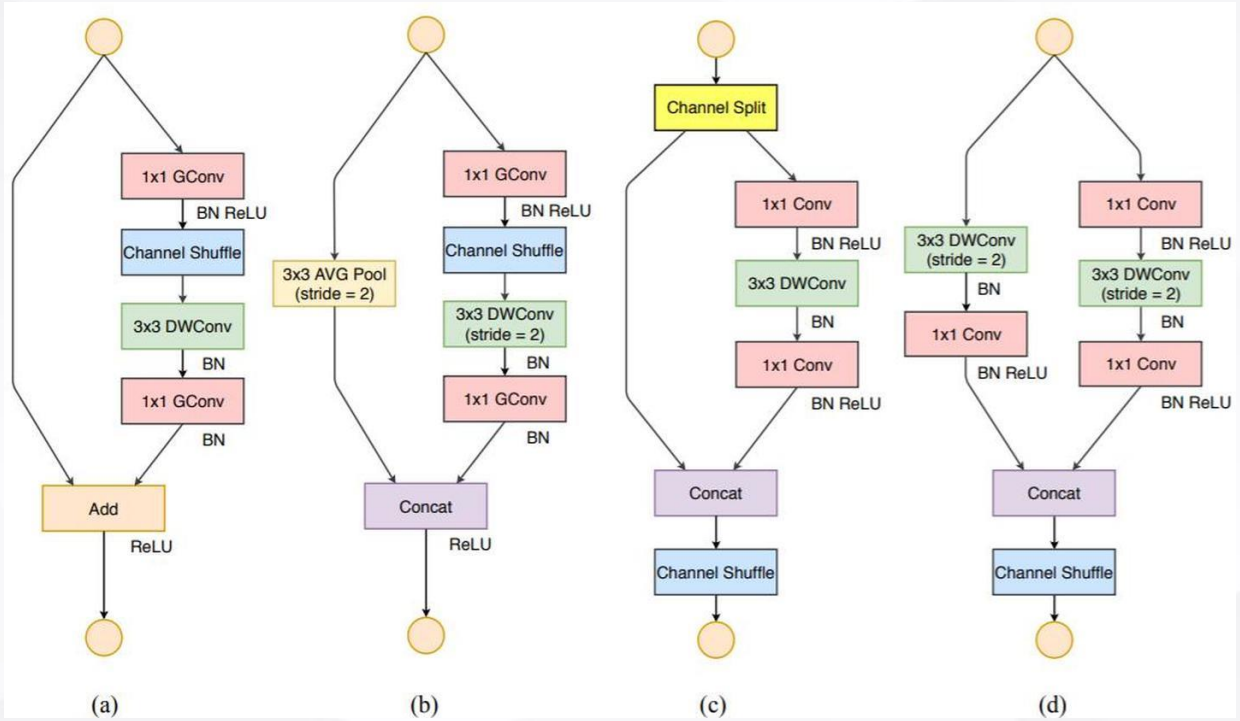
- NanoDet 移动端物体识别：
  - Model
    - Backbone – ShuffleNet
    - Neck(FPN) – PAN
    - Head – FCOS
  - Loss
    - Generalized Focal Loss



- 迁移学习方案：
  - 使用在大数据集（ImageNet, Coco Dataset）上预训练的模型参数
  - 在人工标注的书脊编号小数据集上进行参数调优

ShuffleNet：图像特征提取

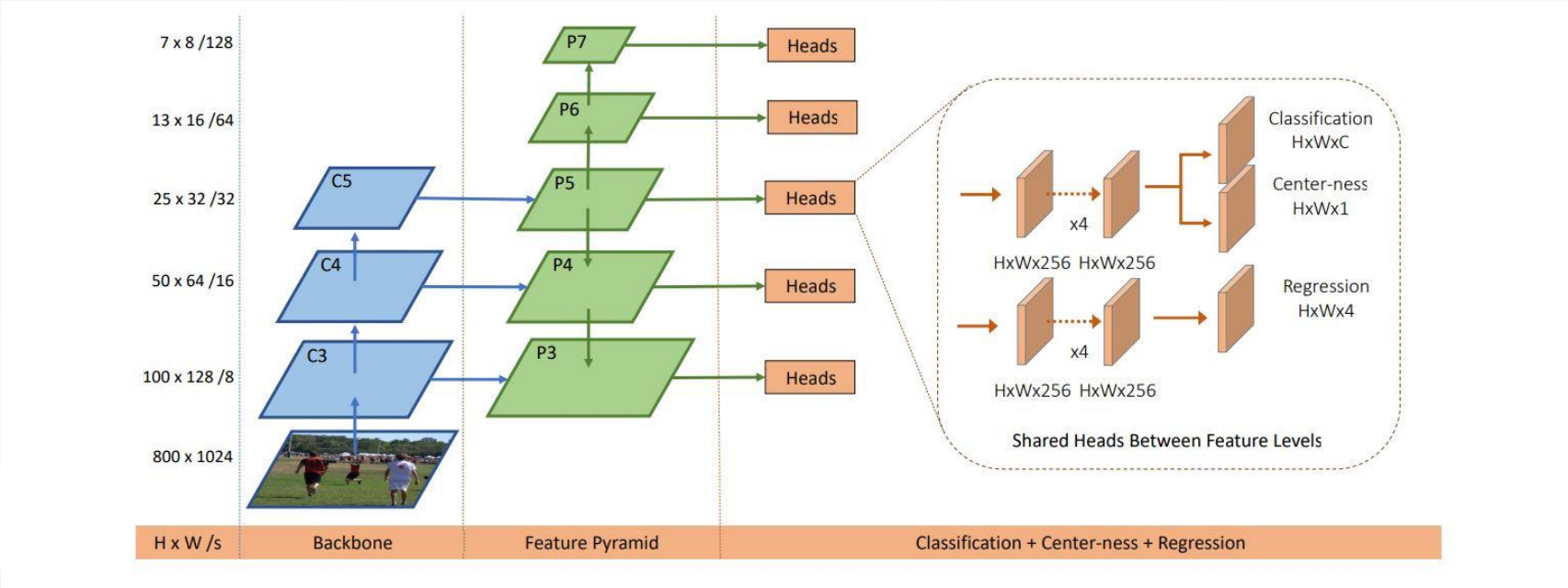
- 适合在 CPU（移动端）上进行计算的卷积神经网络
- 分组卷积 (Group Convolution)， 通道随机重排 (Channel Shuffle)



- Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, et al. *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.* CVPR 2018.

FCOS：单阶段物体检测

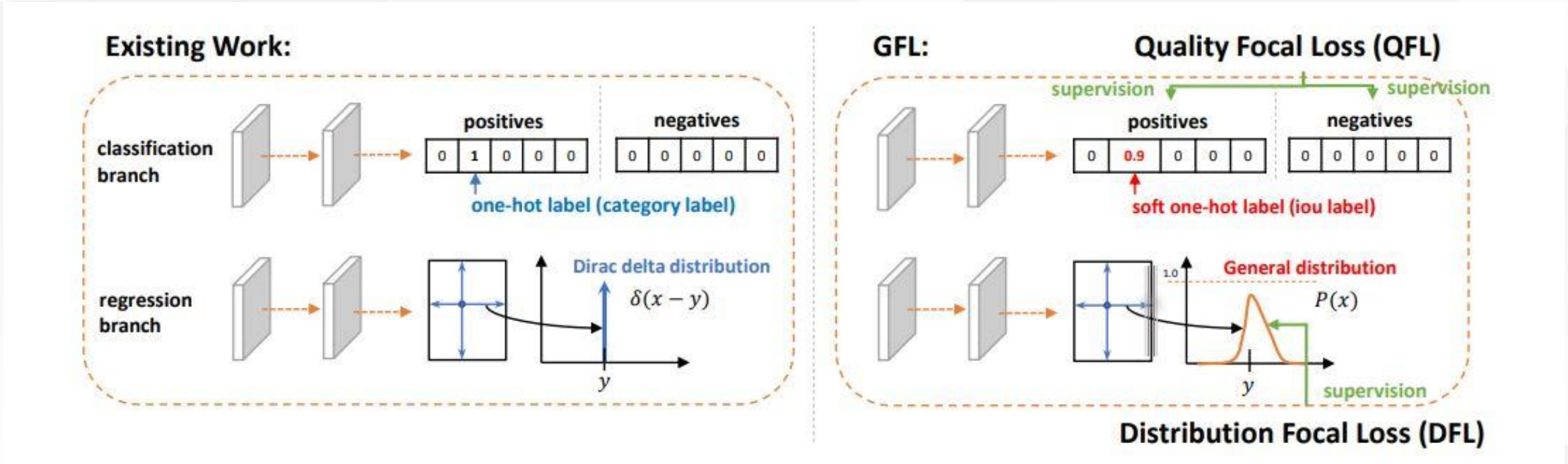
- 单阶段物体检测网络。
- 卷积共享权重，每一层使用一个可学习的 Scale 值作为系数。
- BN 替换 GN，在推理时能够将其归一化的参数直接融合进卷积中。



- Zhi Tian, Chunhua Shen, Hao Chen, et al. *FCOS: Fully Convolutional One-Stage Object Detection.* ICCV 2019.

GFL：改进的损失函数

- 优化 FCOS 的结构，去除 Centerness 分支
- 降低训练难度，省去了一部分计算开销



- Xiang Li, Wenhai Wang, Lijun Wu, et al. *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection.* NeurIPS, 2020

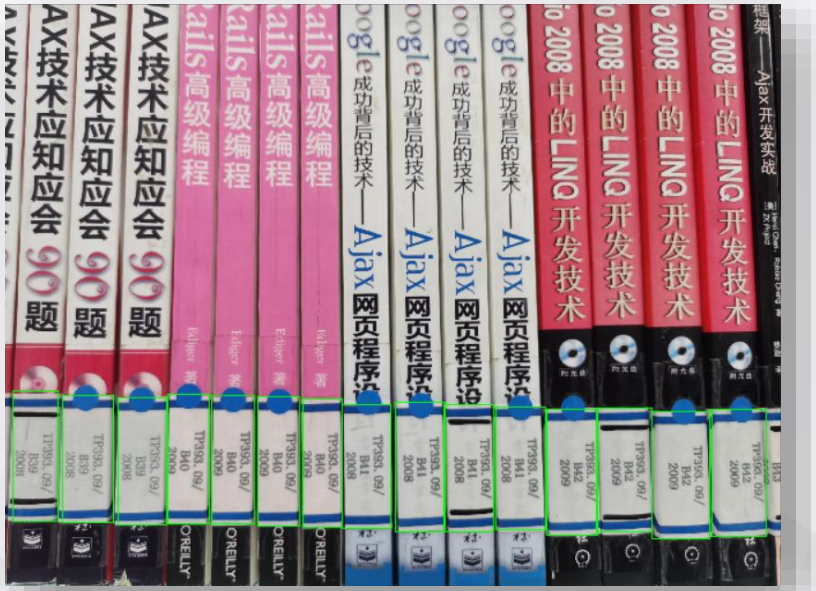
中期的检测效果


B39
2008
B39
1P393.09
/60''Ccdi
B39
2008
1393.09
2009
1TP393.09/
B41
Tp393.09/
TP393.09/
B41
2008
TP393.09/
B2
2009
/60''868d
TP393.09
842
SHEA

数据集标注方法探索

原始照片

改进前



改进后

检测结果对比

改进前

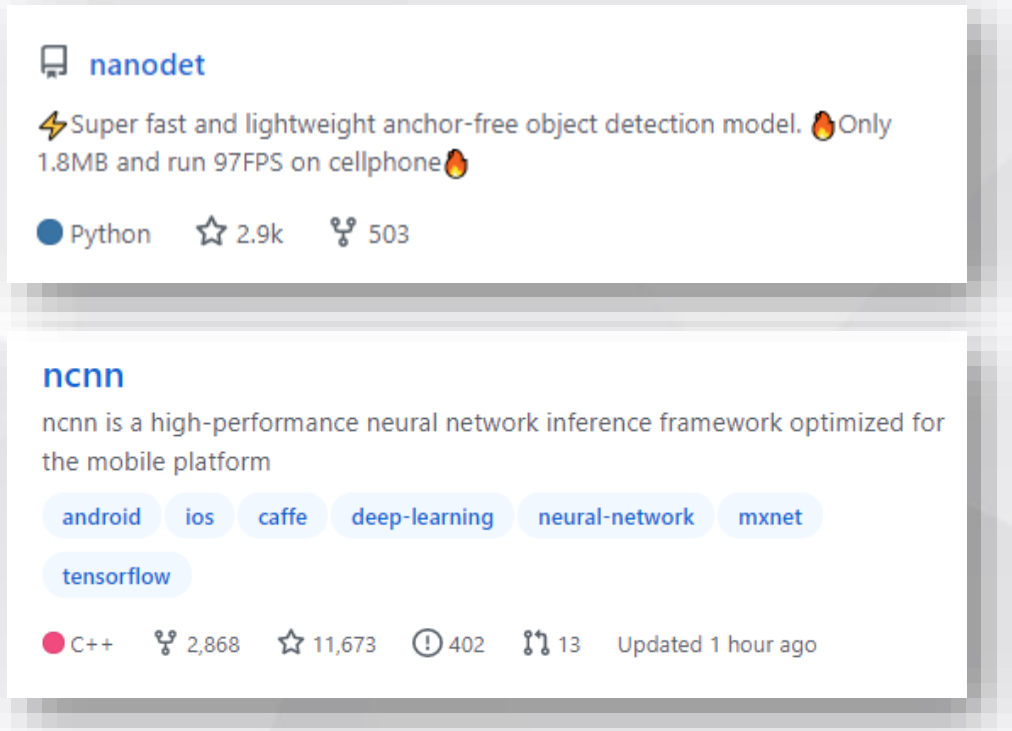
改进后

移动端模型部署

- 基于 Tencent ncnn 框架将 PC 端模型部署到移动端
  - ncnn 是安卓端**已知速度最快的开源前向神经网络计算框架**
  - ncnn 为 PC 端训练模型的迁移部署提供了良好、易用的支持

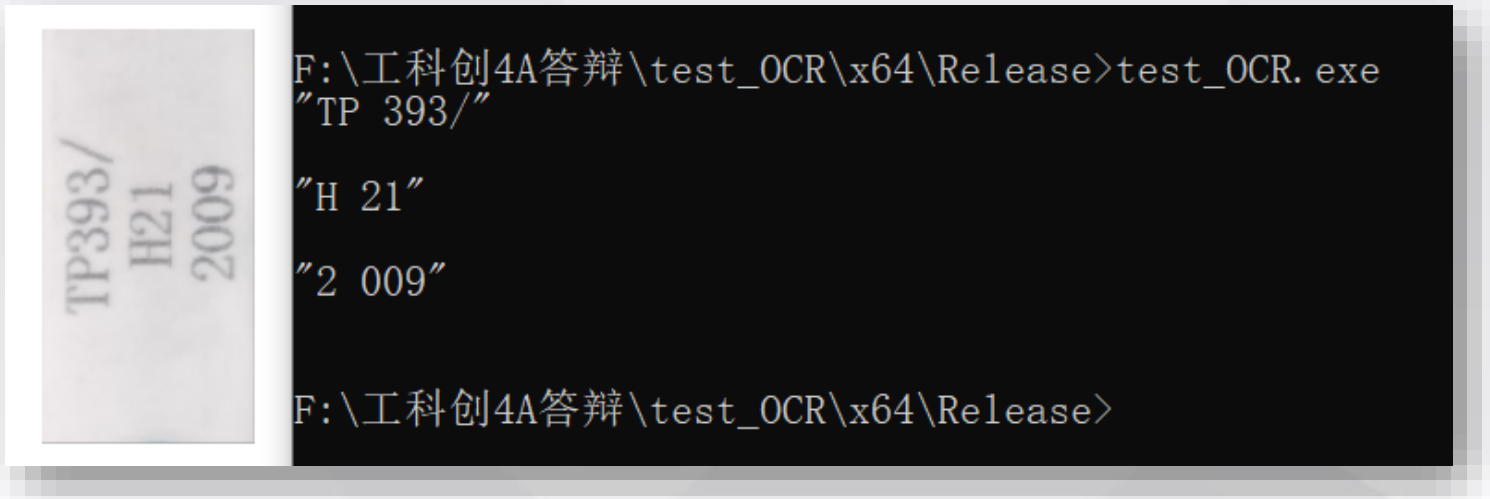
- 使用 ncnn 迁移模型到移动端的过程：
  - 在本地基于 CMake 编译 ncnn 项目
  - 将 PC 端保存的模型参数导出为 .onnx 文件
  - 使用 onnx-simplifier 简化 onnx 模型
  - 将 onnx 模型导出为 .bin 和 .param 文件
  - 使用 ncnnoptimize 优化 ncnn 模型

- NanoDet 模型生成的 ncnn 模型**仅有 1.8 MB**



移动端 OCR 部署

- 基于 **百度 OCR API** 的 PC 端 OCR 部署尝试
- 使用 **OpenCV** 与百度 **OCR C++** 实现文字识别



- **延时过大，不适用于移动端（每次约 200 ms）**

移动端 OCR 部署

- 本地 OCR 对实时计算性能的要求：
  - 假定需要达到的帧率为 20 FPS，每帧有 10 个书脊编号
  - 主程序进行 OCR 要求的最慢识别速度为**每个编号 5 ms**
- 按照目前主流 OCR 框架的性能，**几乎不可能达到**
- 另一问题：OCR 识别速度与精度的 Tradeoff

实时 OCR 真的完全不可实现吗?

移动端 OCR 部署

- 基于 Google Tesseract 的**多线程**移动端 OCR 方案
  - 静态探测的条件下，准确度较高
  - 镜头移动的条件下，画面模糊，效果一般
  - **远端只进行识别，近端进行 OCR 识别**
- 分两类情况处理：
  - 在图中出现标签不大于 5 张时，即返回 OCR 结果
  - 在标签多于 5 张时，不返回 OCR 结果，只显示框
- 动态识别有少许延迟，但**只要短暂停止移动，即可快速恢复**

移动端效果展示

- 精准识别多排书脊编号
- 具有较为良好的鲁棒性
- 在未加 OCR 的条件下达到 **40 FPS**
- 在加入 OCR 的条件下达到 **20 FPS**
- 可以动态调节 NMS 参数及概率阈值

项目总结

成果一览：

- 标签检测算法准确度高（定位准确度高于 95%）
- 标签检测算法已移植到移动端，识别实时性已达到要求（40 FPS >> 8 FPS）
- 标签 OCR 在安卓端实现（实时 & 静态），选定图片 OCR 效果好
- 设计了兼顾实时性（20 FPS >> 8 FPS ）与精度的 OCR 应用方案
- 整体工作完整，效果较好，基本达到预期要求

改进空间：

- 受限于现有免费开源 OCR 的性能，目前难以使 OCR 完全跟上标签检测
- 对接图书馆数据库，真正形成图书馆书籍定位的完整产品

感谢

饮 水 思 源　爱 国 荣 校