飞桨

使用PaddleX快速完成深度学习的产业落地

百度飞桨产品经理　余志良

2021.12

目录

Contents

1 产品简介

飞桨产业级深度学习开源开放平台

PaddleX -- 飞桨全流程开发套件

图像分类 | **目标检测** | **实例分割** | **语义分割**

模块一：PaddleX 图形化开发界面

模块二：本地 API

数据准备

- 协议：兼容通用标注协议 COCO, VOC, 精灵标注
- 平台：智能数据服务平台 EasyData
- 算法：上百种增强策略 imgaug图像增强库

模型训练与优化

- 视觉开发套件：图像分类 PaddleClas；目标检测 PaddleDetection；图像分割 PaddleSeg
- 辅助工具组件：可视化分析 VisualDL；模型压缩 PaddleSlim；模型可解释性 AutoDL Interpretability

多端部署

- 端侧预测引擎 Paddle Lite
- 原生预测库 Paddle Inference
- 模型格式转换 Paddle Converter
- 模型加密 Model Encryption & Decryption

模块三：PaddleX Restful API

Manufacture SDK | **Deployment** | **OpenVINO加速** | **TensorRT加速**

本地部署 | **服务化部署** | **边缘端部署** | **ONNX部署** | **部署加密**

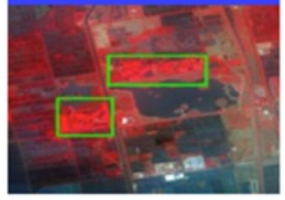
表计读数

工业质检

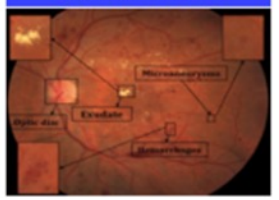
火情检测

安全帽检测

遥感影像检测



眼底病变筛查



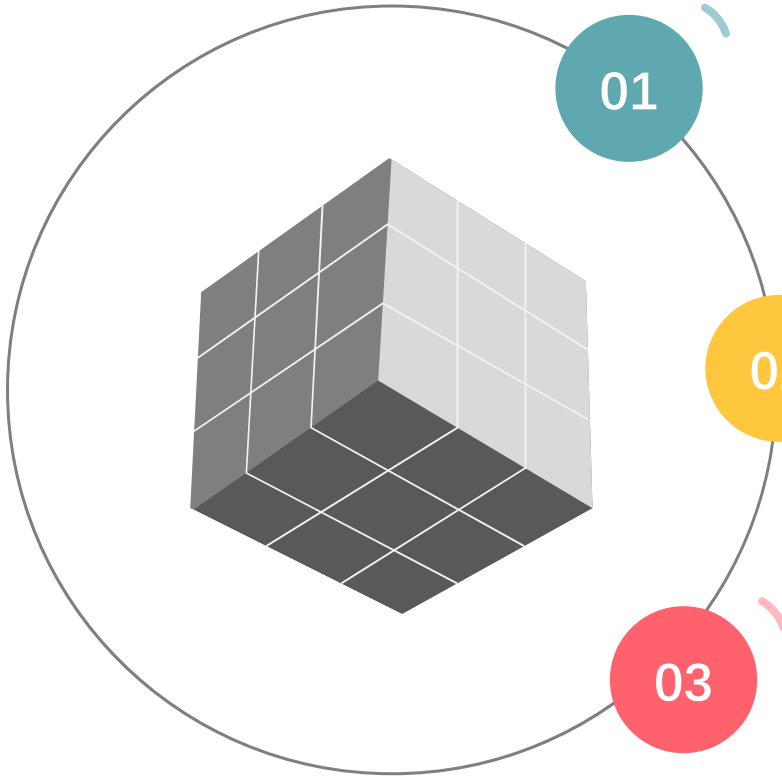
智慧交通

新零售商品检测

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

PaddleX 三大特点

为开发者带来产业应用最佳体验



01

全流程打通

打通数据准备、模型训练、模型调优、多端部署的深度学习全流程

02

融合产业实践经验

结合飞桨产业落地经验，精选高质量的视觉模型方案，开放详细产业案例

03

易用易集成

- 图形化开发界面：非专业算法人员可快速进行视觉模型开发
- Python 函数库：功能全面、开发灵活，易于被集成

2 案例分享

PaddleX提供经过产业验证的视觉解决方案

图像分类

公益
野生动物
ResNet50vd-10w

零售
商品分类
MobileNetV3-ssld

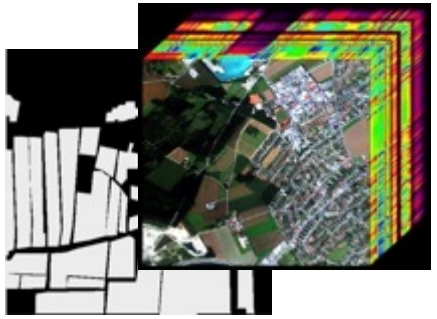
目标检测

安防巡检
电力表计检测
Faster-RCNN

工业生产
质量检测
YOLOv3

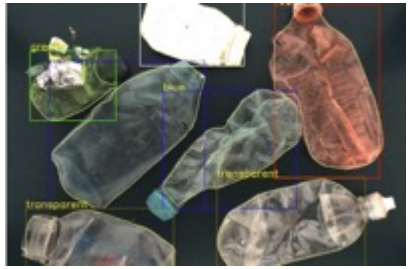
语义分割

短视频
实时人像分割
HRNet_w18_small

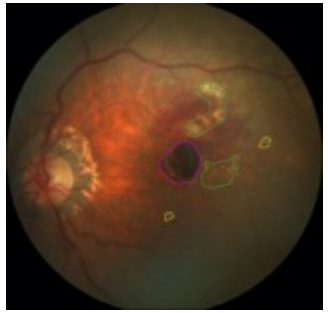


城市规划
地类分割
DeepLab-Xception65
DeepLab-MobileNet

实例分割

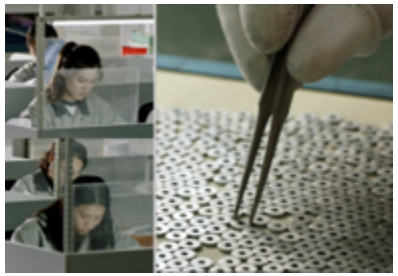


市政
垃圾分拣
Mask-RCNN-FPN



智慧医疗
眼底筛查
Mask-RCNN

2.1 工业质检



3C电子

印刷品

制药

汽车

纺织品

其他制造

行业特点：

- 缺陷种类数十数百；
- 每种缺陷变形众多；
- 缺陷位置不固定；

■ 人力检测：

评判标准不唯一，人员消耗大

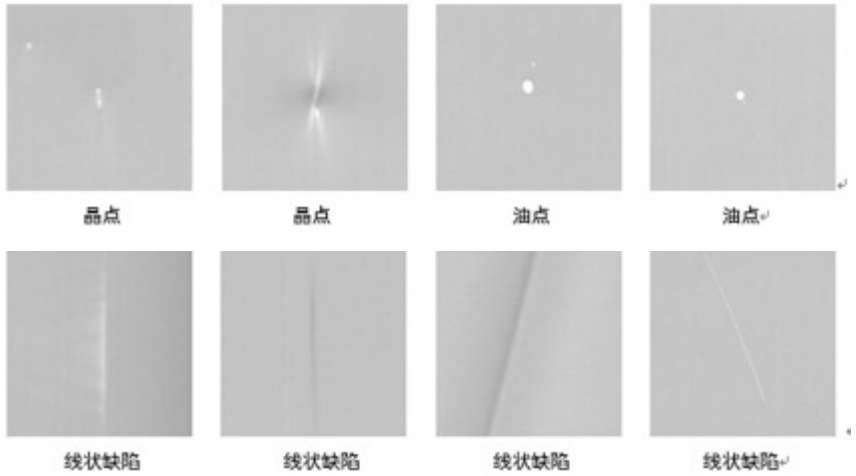
■ 传统视觉：

适应、迁移能力差，漏检、误检、过检难以平衡

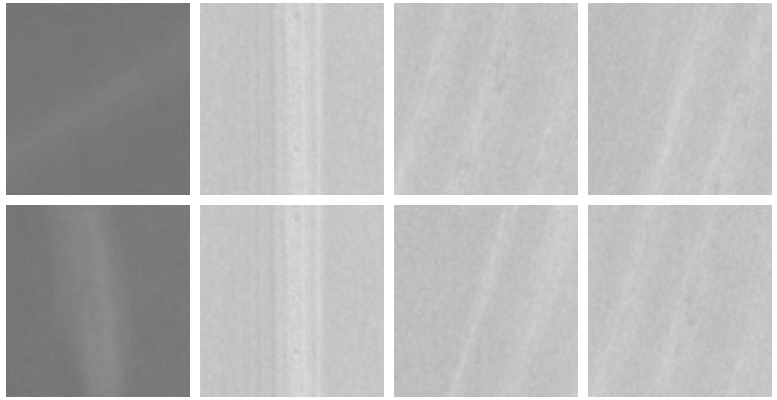
备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

电池隔膜缺陷检测案例分析

- 缺陷类型：真空、漏涂、晶点、杂质、沙眼、划痕、油斑、亮斑、亮线、黑点、褶皱、蚊虫等十余种
- 上线要求：缺陷检测并分类分级



- 传统视觉容易漏检和误检的缺陷示例



融合深度学习技术的方案

传统算法

阈值分割

生成候选缺陷区域

深度学习算法

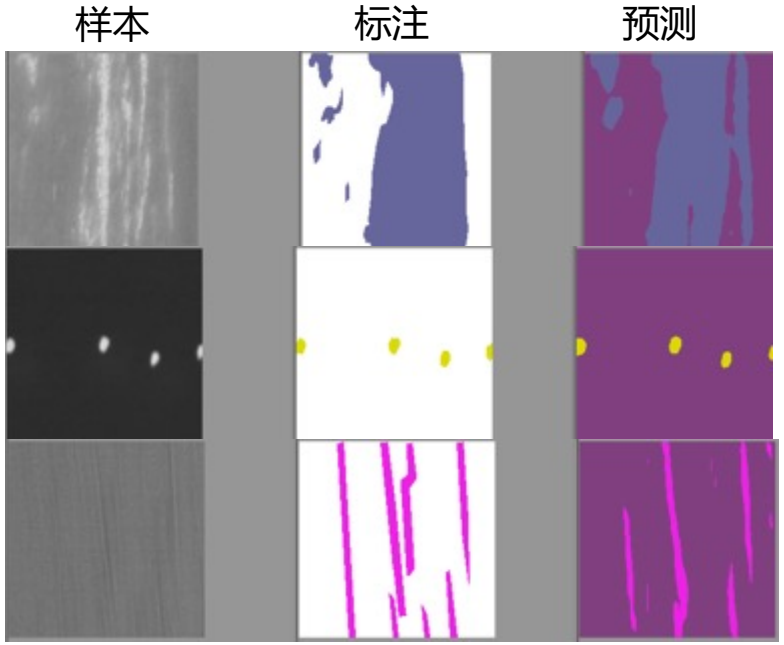
语义分割
Deeplabv3+
MobileNetV2-ssld

缺陷精细分割

图像分类
MobileNetV2-ssld

缺陷快速分类

实际效果



缺陷检出率提升 **30%**

预测速度提升 **50倍**

分割准确度：82%，
分类准确性：98%；
1080Ti 上200*200分割速度2ms

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

手机壳外观缺陷检测

问题难点

- 缺陷种类繁多：外观材质不同，背景复杂多变
- 缺陷数据少：偶发缺陷
- 检测速度快：单件产品，5面检测时间 **< 1s**

应用模型

- PP-YOLO（飞桨优化产业最优YOLO模型）

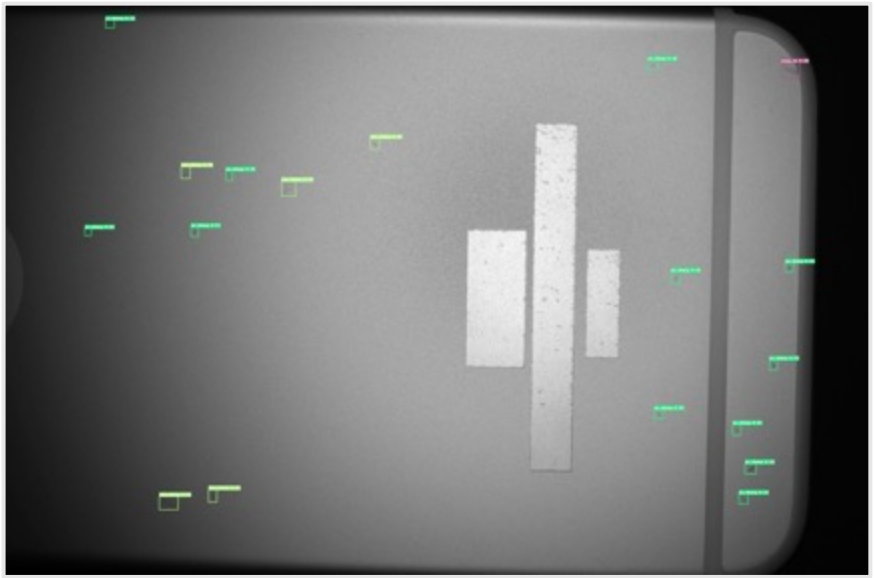
（COCO test-dev2017数据集 mAP 45.9%，
单卡V100 72.9FPS， TRT 下 FP16 推理速度 155.6 FPS）

优化方法

- 数据增强
- 更大的batch size
- 模型压缩

实现效果

内部评测数据	EdgeBoard检测	对比原有竞品同等算力下，
准确率>=99.55%	时间 <=750ms	预测速度提升200%+

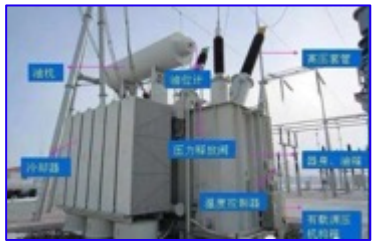


备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

2.2 工业巡检

设备巡检

表计巡检



设备状态监控

安全巡检

工服安全帽识别

火情检测

无人机本地巡检

鸟窝异物检测

绝缘子缺失

通道巡检

油气泄露

通道异物

■ **应用行业**：石油化工、燃气、自来水、热力、电力、电信、公路、铁路、桥隧、轨道交通、大型工厂等

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

工业仪表自动巡检

■ 【需求】

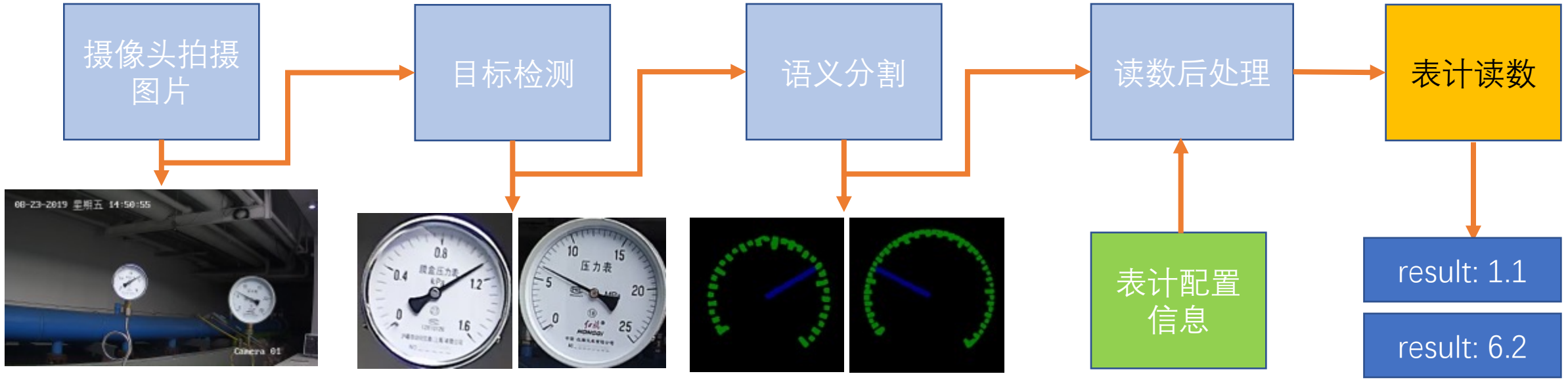
电力能源厂区需要定期监测表计读数，以保证设备正常运行及厂区安全；

■【现状】

厂区分布分散，表计种类多、散； 人工巡检耗时长，无法实时；

部分工作环境危险，且难以触达； 人工读表容易产生误差。

表计检测读数整体方案



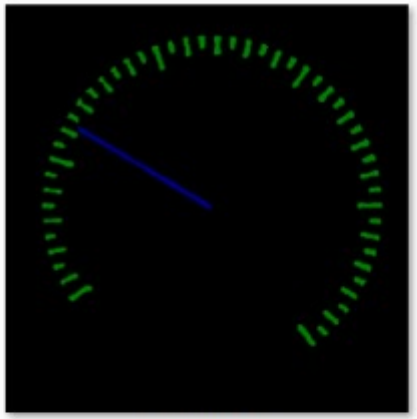
- 「**检测表计**」：平衡考虑算法的推理速度和检测效果，采用PaddleX 中集成的PP-YOLO模型。只做检测不做分类。
- 「**分割指针&刻度**」：使用经典语义分割模型DeepLapv3+ 将各细小的表计指针和刻度分割出来
- 「**计算读数策略**」：根据指针的相对位置和预知的量程计算出各表计的读数。

注：厂矿表计多为分散安装，大多数情况每个摄像头只拍摄一个表计。在表计集中安装的场景下，多个表计基本为同类表计。需要识别表计种类（量程和单位）的场景很少。表计的单位量程等信息通过配置的形式输入给推理进程，结合分割结果综合计算表计读数。）

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

检测及分割后的结果

6.jpg



6_label.png

35.jpg

35_label.png

238.jpg

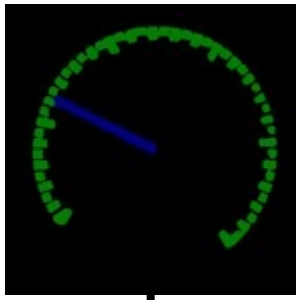
238_label.png

419.jpg

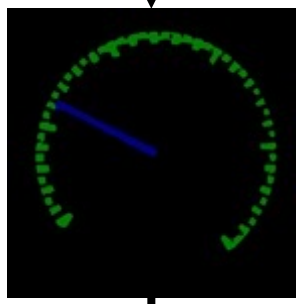
419_label.png

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

读数后处理



图像腐蚀



环形的表盘展开为矩形图像

二维图像转换为一维数组

对刻度用均值做二值化

定位指针相对刻度的位置

根据刻度的根数判断表盘类型
获取表盘量程

指针相对位置与量程做乘积

- 对语义分割的预测类别图进行图像腐蚀，细分刻度；
- 把环形的表盘展开为矩形图像，根据图像中类别信息生成一维的刻度数组和一维的指针数组
- 计算刻度数组的均值，用均值对刻度数组进行二值化操作
- 定位指针相对刻度的位置，依据刻度根数获取预知的量程，将指针相对位置与量程做乘积得到读数

整体方案效果

飞桨 与“广东电网”合作，
助力变电站智慧巡检，识别准确率高达**99.01%**

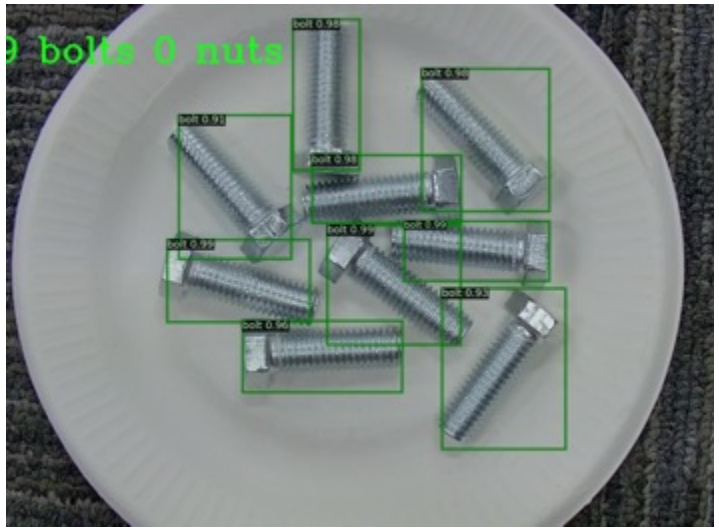
提供完整端到端的开发套件
支持企业实现需求的快速落地应用

需求场景	变电站数量众多，日常巡检常态化， 人工巡检工作**内容单调**，人力**投入大**，巡检**效率低**
解决方案	机器人拍照 → 表计检测 PaddleDetection；表盘分割 PaddleSeg；移动端部署 Paddle Lite → 返回读数
核心亮点	**效果优**：表面读数误差±2°占比 **99.01%** **端到端**：飞桨提供完整端到端的 **开发套件** **简单易用**：新表计适配 **10天完成**
最终效果	无人智能运维，**准实时**自动巡检

数据来源：内部测试评估结果，实际结果可能受测试环境影响而在一定范围内变化，仅供参考

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

2.3 工业零件计数



螺丝计数

圆木计数

人流统计

数据预处理

随机水平翻转
随机扩张图像
调整图像大小
随机裁剪
随机像素内容变换
MixupImage

算法训练及优化

目标检测算法
YOLOv3（骨架网络MobileNetv2）

预测计数

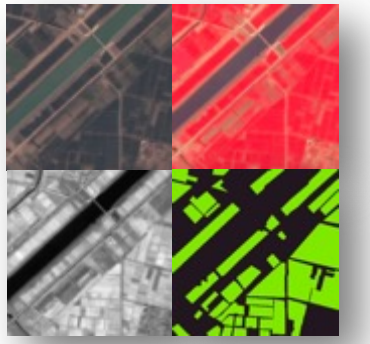
```
for r in results:
    if r["score"]>=threshold:
        filtered_results.append(r)
```

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

2.4 遥感图像解译

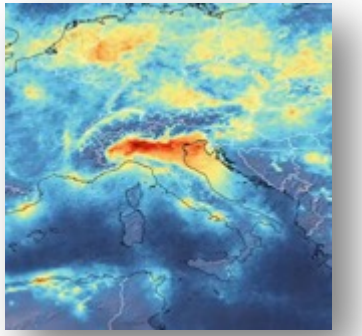
卫星遥感影像任务类型

特殊用地检测
（光伏发电站、高尔夫球场等）

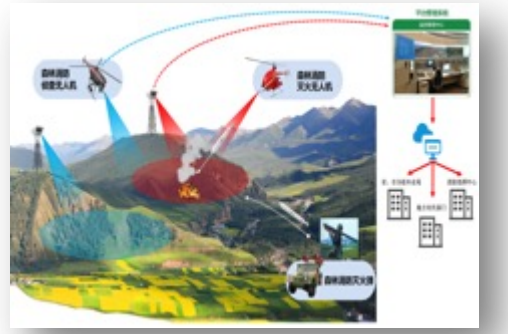


地类分割
（农田、街道、水域等）

地块变化检测
（2017~2019年新增建筑用地）



卫星、海洋气象
（气旋、积雪、污染物等复杂多样）



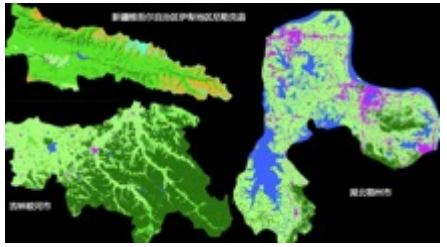
无人机遥感
（类安防巡检目标检测，森林火灾检测等）

遥感影像处理难点

目标多尺度

目标排列密集、多角度

图像分辨率低、高噪声



图像多空间、多时域

备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

重大工程用地检测

PaddleX 的独特设计

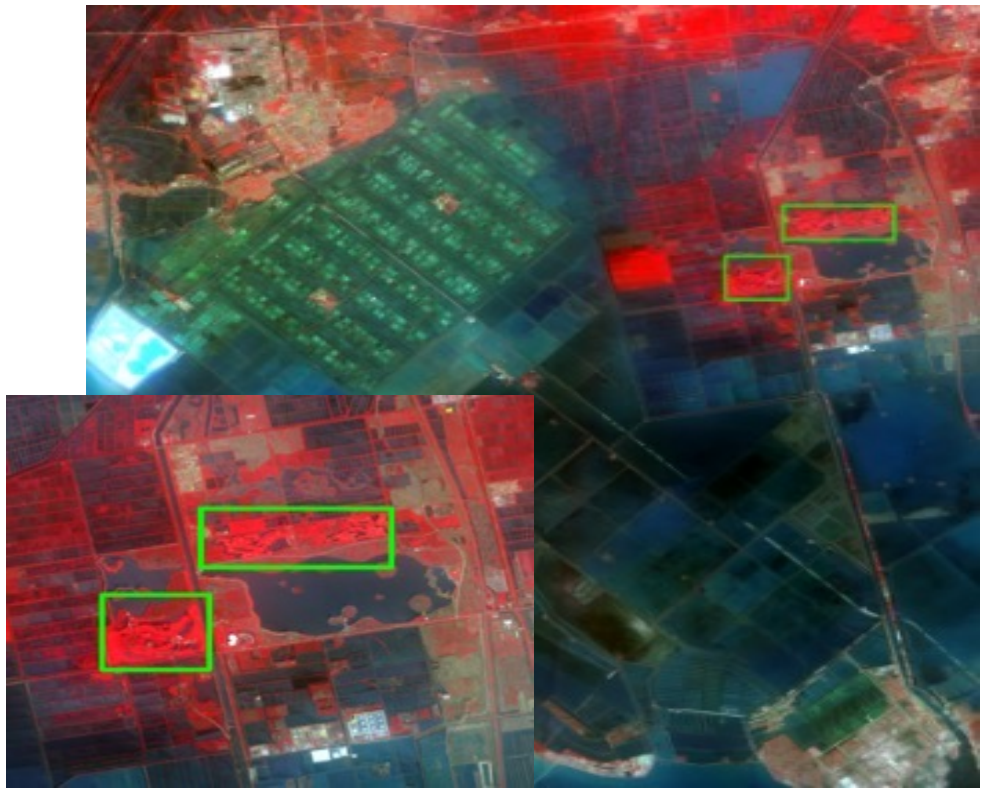
- **遥感数据格式兼容（RGB和多通道）：** tif，png, img, npy
- **数据分析**: 图片数量、图片通道数、图像尺寸极值、各通道像素值分布，通道归一化后的均值和方差，类别数量及比例

方案及效果

影像预处理 → 球场标注 影像裁剪 → 模型训练&调优 → 样本增强 → 上线部署

构建共409景影像的 865个多时相球场样本的全国标准高尔夫球场遥感数据集

Faster R-CNN	目标检测模型
88%	检出准确率
15分/景 VS 10秒/景	较传统人工，作业识别效率提升90倍

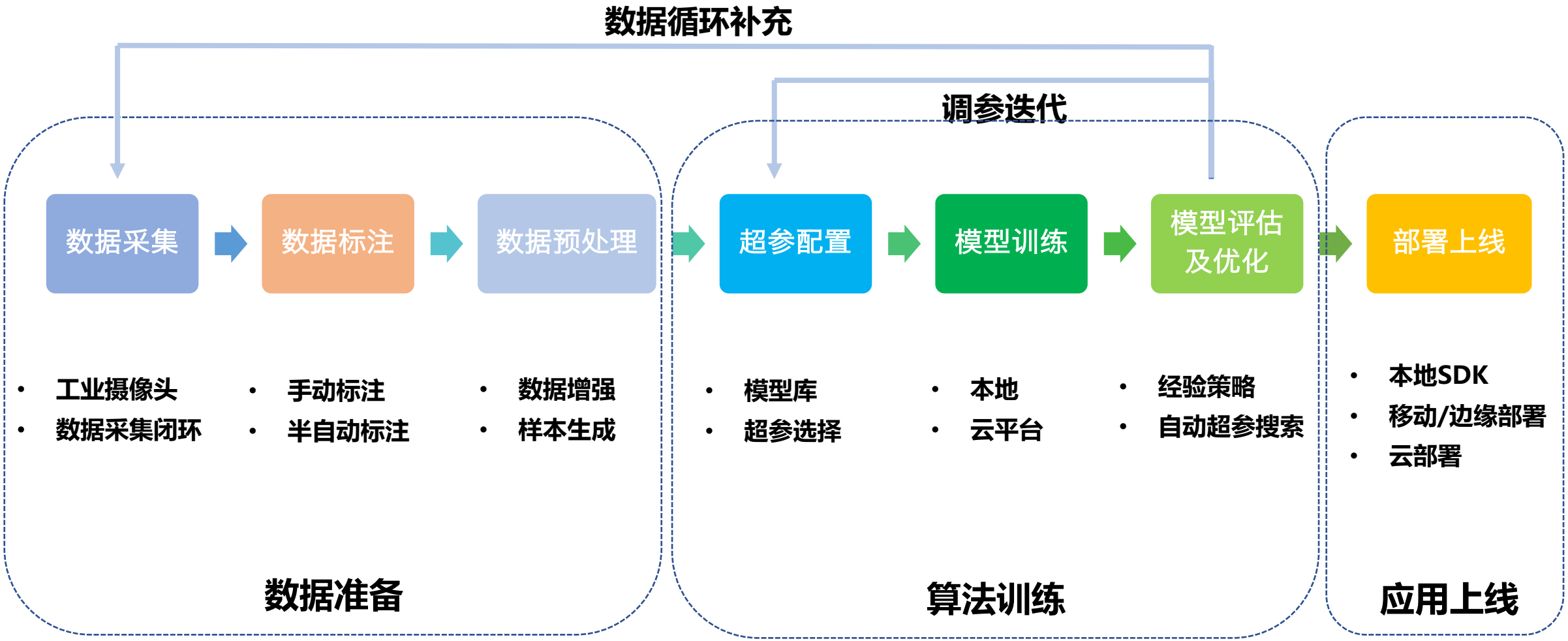


备注说明：图片来源于网络/公开发表论文/合作伙伴提供，如有侵权，请告知

3 使用方法

3.1 数据标注及预处理
3.2 模型训练
    3.2.1 函数库方式（本地API）
    3.2.2 进阶应用—模型裁剪、可解释性
3.3 模型部署

深度学习产业应用落地全流程



3.1 数据标注

PaddleX 适配产业标准数据标注格式及EasyData智能数据标注平台

数据标注工具

支持常用标注工具

原生匹配数据格式转换方法

Labelme

目标检测、实例分割、语义分割

pip install labelme (本地数据标注)

精灵标注

图像分类、目标检测、实例分割、语义分割

客户端下载（本地数据标注）

EasyData

图像分类、目标检测、实例分割、语义分割

平台标注，自动数据清洗及预标注

数据格式说明

支持业界标准数据格式读取

ImageNet
图像分类

PascalVOC
目标检测

MSCOCO
语义分割、实例分割、目标检测

数据格式转换

将标注数据导出为需要的数据格式

参数说明

--source	*--to*	*--pics*	*--annotations*
labelme jingling easydata	ImageNet PascalVOC MACOCO	原图所在 目录路径	标注文件 目录路径
.json			

转换脚本

```
paddlex --data_conversion --source labelme --to PascalVOC --pics ./pics --
annotations ./annotations --save_dir ./converted_dataset_dir
```

数据标注工具在线文档说明：
https://paddlex.readthedocs.io/zh_CN/develop/data/annotation.html
精灵标注下载链接：http://www.jinglingbiaozhu.com/
EasyData网页：https://ai.baidu.com/easydata/

3.1 数据格式完整说明

	图像分类ImageNet	目标检测PascalVOC	实例分割MSCOCO	语义分割Seg
文件结构	MyDataset/ *#数据集根目录* |--dog/ *#所有dog类别图像的文件夹* | |--dog1.jpg | |--dog2.jpg | |--…| |--… | |--cat/ *#所有cat类别图像的文件夹* | |--cat1.jpg | |--cat2.jpg | |--… |--snake/ | |--…	MyDataset/ *#数据集根目录* |--JPGImages/ *#所有原图文件* | |--image1.jpg | |--image2.jpg *#图2原图* | |--…| |--… | |--Annotations/ *#所有标注文件所在目录* | |--image1.xml | |--image2.xml *#图2对应的标注文件* | |--…	MyDataset/ *#数据集根目录* |--JPGImages/ *#所有原图文件* | |--image1.jpg | |--image2.jpg | |--…| |--… | |--annotations.json *#所有标注信息存在的文件*	MyDataset/ *#数据集根目录* |--JPGImages/ *#所有原图文件* | |--image1.jpg | |--image2.jpg *#图2原图* | |--…| |--… | |--Annotations/ *#所有标注文件所在目录* | |--image1.png | |--image2.png *#图2对应的标注文件* | |--…
数据集划分（可视化前端可自动切分）	--train_list.txt *#训练集列表文件* dog/dog1.jpg 0 *#0代表dog的类别ID* dog/dog2.jpg 0 cat/cat1.jpg 1 *#1代表cat的类别ID* … … snake/snake1.jpg 2 --val_list.txt *#验证集列表文件,与train格式一致* --labels.txt *#类别标签列表* dog cat snake	--train_list.txt *#训练集列表文件* JPGImages/image1.jpg Annotations/image1.xml *#原图和标注一一对应* JPGImages/image2.jpg Annotations/image2.xml … … --val_list.txt *#验证集列表文件,与train格式一致* --labels.txt *#类别标签列表,与图像分类一致*	--train.json *#训练数据标注信息说明文件* --test.json --val.json 切分脚本 *paddlex --split_dataset --format COCO --dataset_dir MyDataset --val_value 0.2 --test_value 0.1*	--train_list.txt *#训练集列表文件* JPGImages/image1.jpg Annotations/image1.png*#原图和标注对应* JPGImages/image2.jpg Annotations/image2.png … … --val_list.txt *#验证集列表文件,与train格式一致* --labels.txt background *#背景类别* human car

完整说明及切分文档： https://paddlex.readthedocs.io/zh_CN/develop/data/format/instance_segmentation.html

注：原图的数据格式支持.jpg, .jpng, .png, .bmp

3 使用方法

3.2.1 模型训练—函数库方式

PaddleX 函数库与【图形化开发界面】开发流程相同，提供极简的API，**功能更丰富、开发更灵活、开源易集成**

10分钟快速上手使用—昆虫检测（目标检测）项目示例 飞桨明星模型PP YOLO

PaddleX 函数库模型训练通用流程

注：需要在已安装PaddlePaddle和PaddleX的前提下
完整文档说明：https://paddlex.readthedocs.io/zh_CN/develop/train/index.html
完整AI Studio项目示例：https://aistudio.baidu.com/aistudio/projectdetail/442375

3.2.1 模型训练—函数库方式

1 定义数据预处理

transforms 是用来对数据进行裁剪、旋转等数据增强等预处理工作的。
PaddleX 原生提供针对分类、检测、分割常用的数据预处理方法，并可直接支持调用imgaug数据增强库

 # 在数据标注完成、开始训练前，需要定义训练和验证时的transforms
 # 完整API说明 https://paddlex.readthedocs.io/zh_CN/develop/apis/transforms/det_transforms.html

```
from paddlex.det import transforms

train_transforms = transforms.Compose([    #训练数据预处理模块
    transforms.MixupImage(mixup_epoch=250),
    transforms.RandomDistort(),
    transforms.RandomExpand(),
    transforms.RandomCrop(),
    transforms.Resize(target_size=608, interp='RANDOM'),
    transforms.RandomHorizontalFlip(),
    transforms.Normalize()
])

eval_transforms = transforms.Compose([    #评估数据预处理模块
    transforms.Resize(target_size=608, interp='CUBIC'),
    transforms.Normalize()
])
```

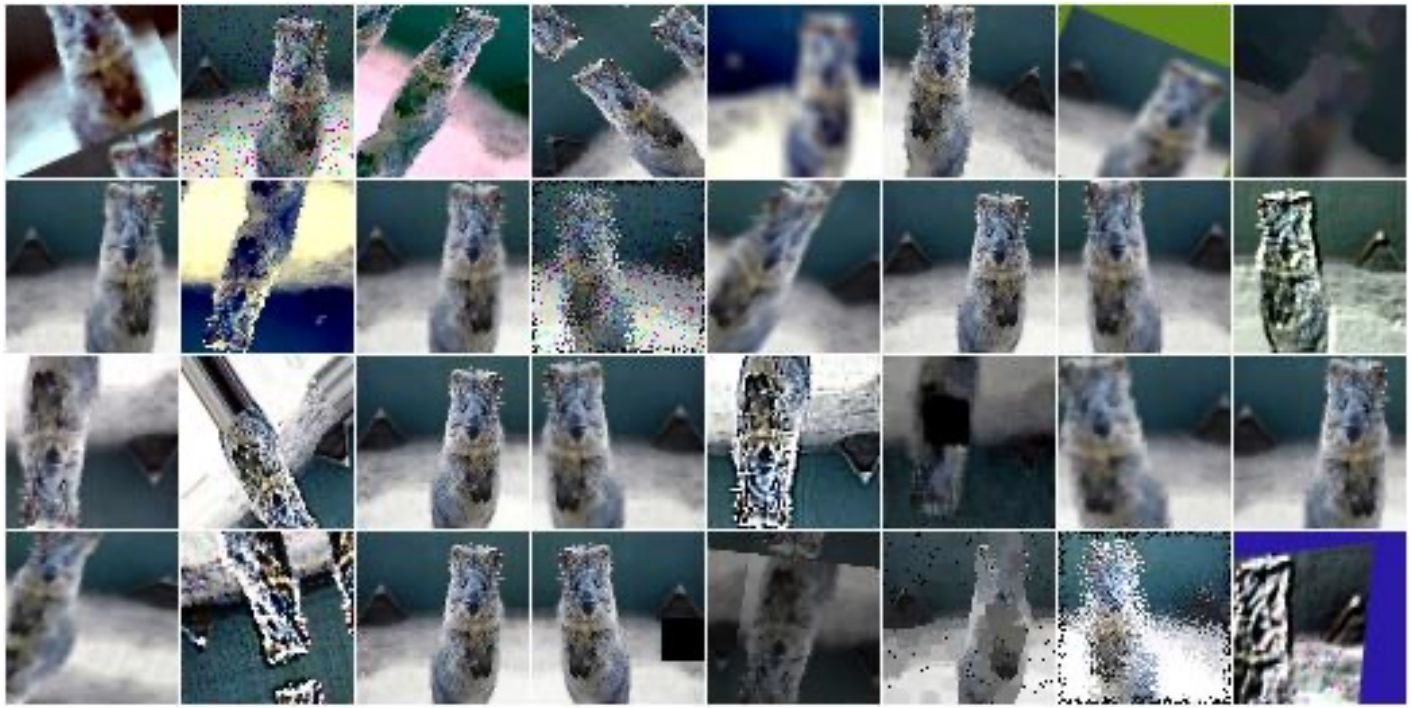
针对不同的任务类型，往往需要采用不同的数据增强方法，PaddleX都提供相应的有效策略供开发者选择。

3.2.1 模型训练—函数库方式

- 适配imgaug图像增强库，支持上百种数据增强操作

```
from paddlex.cls import transforms
import imgaug.augmenters as iaa

custom_augmenters1 = iaa.Sometimes(0.3, [
    iaa.blur.GaussianBlur(),
    iaa.arithmetic.Dropout()
])
custom_augmenters2 = iaa.SomeOf(2, [
    iaa.color.AddToBrightness(),
    iaa.contrast.GammaContrast()
])
train_transforms = transforms.Compose([
    iaa.flip.Fliplr(p=0.5),
    custom_augmenters1,
    custom_augmenters2,
    transforms.RandomCrop(crop_size=224),
    transforms.Normalize()
])
```



3.2.1 模型训练—函数库方式

2 定义加载数据集

- 定义传入到模型中的数据集位置及格式。

```
train_dataset = pdx.datasets.VOCDetection( #表示VOC格式的目标检测数据集
    data_dir='insect_det',
    file_list='insect_det/train_list.txt',
    label_list='insect_det/labels.txt',
    transforms=train_transforms,
    shuffle=True)

eval_dataset = pdx.datasets.VOCDetection( #评估所用数据集
    data_dir='insect_det',
    file_list='insect_det/val_list.txt',
    label_list='insect_det/labels.txt',
    transforms=eval_transforms)
```

- PaddleX针对各类数据集均提供极简读取方式

数据集读取

- paddlex.datasets.ImageNet
- paddlex.datasets.VOCDetection
- paddlex.datasets.CocoDetection
- paddlex.datasets.SegDataset
- paddlex.datasets.EasyDataCls
- paddlex.datasets.EasyDataDet
- paddlex.datasets.EasyDataSeg

3.2.1 模型训练—函数库方式

3 超参配置开始训练

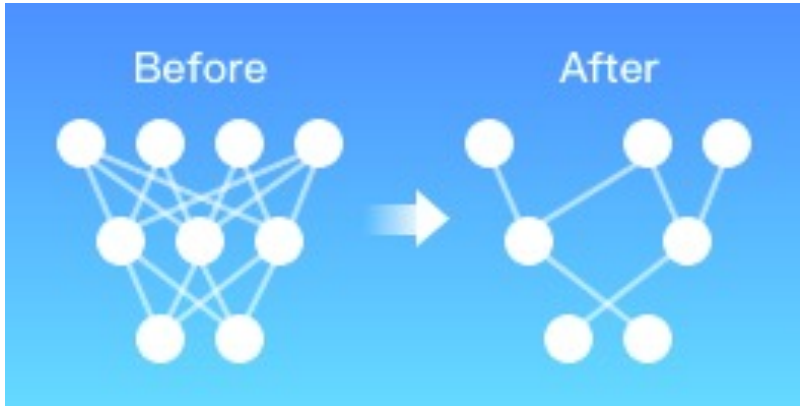
PaddleX提供图像分类、目标检测、语义分割、实例分割四个方向经过产业验证的、精选的实用模型并提供如下展示的统一的任务接口，使开发者可以采用同一套编程逻辑快速选用模型开始开发。

```
num_classes = len(train_dataset.labels) #获取标签类别
model = pdx.det.YOLOv3(num_classes=num_classes, backbone='DarkNet53')
#使用YOLOv3网络，选择backbone DarkNet53
model.train(
    num_epochs=270, #迭代轮次
    train_dataset=train_dataset, #训练数据集
    train_batch_size=8, #批大小
    eval_dataset=eval_dataset, #评估数据集
    learning_rate=0.000125, #学习率大小
    lr_decay_epochs=[210, 240], #定义衰减轮次
    save_interval_epochs=20, #定义模型保存间隔
    save_dir='output/yolov3_darknet53',
    use_vdl=True)  #可视化分析工具的采用
```

3.2.2 模型训练进阶应用—模型压缩

1个API完成模型量化

```
pdx.slim.export_quant_model(
    model='/PATH/TO/MODEL',
    test_dataset=test_dataset,
    save_dir='/PATH/TO/QUANT_MODEL')
```



2个API完成模型裁剪

1. 分析并保存模型敏感度

```
pdx.slim.cal_params_sensitivities(
    model='/PATH/TO/MODEL/',
    save_file='./sensitivities.data',
    eval_dataset=eval_dataset)
```

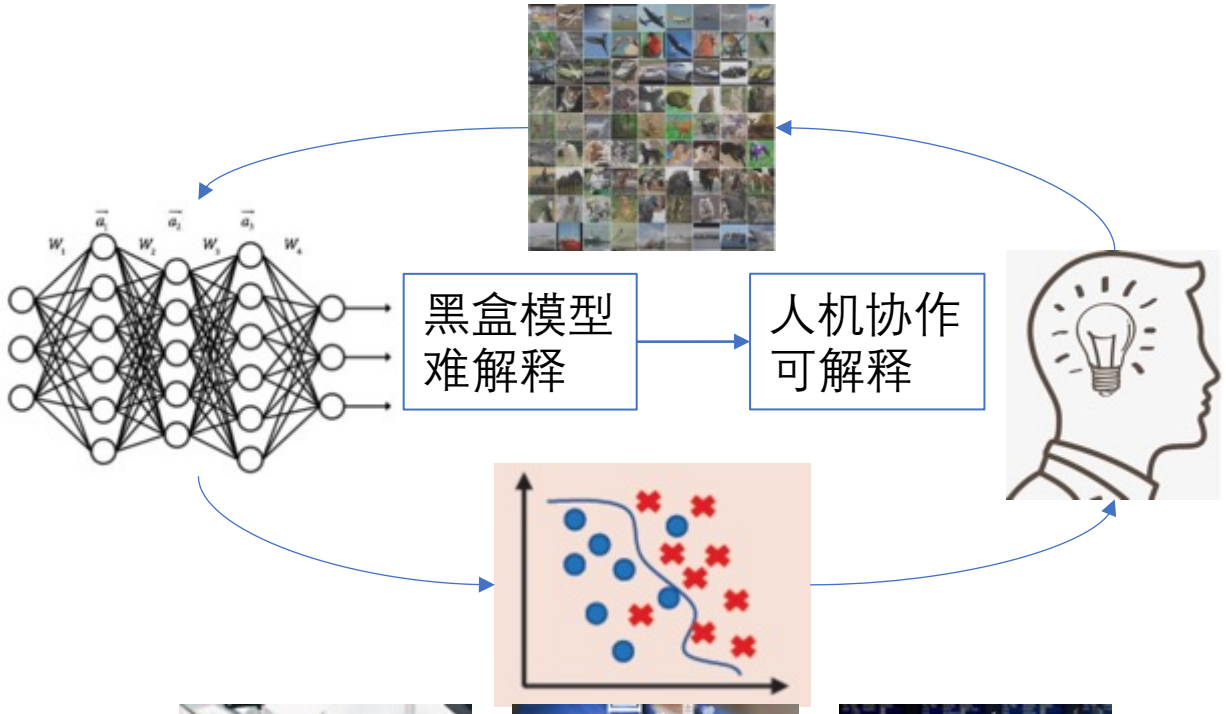
2. 对模型进行重新训练

```
mode.train(num_epochs=10,
           train_dataset=train_dataset,
           train_batch_size=32,
           lr_decay_epochs=[4, 6, 8],
           learning_rate=0.025,
           pretrain_weights='/PATH/TO/MODEL/',
           save_dir='/PATH/TO/PRUNED_MODEL/',
           sensitivities_file='./sensitivities.data',
           eval_metric_loss=0.05,
           early_stop=True)
```

模型	压缩策略	GFLOPs	模型大小(MB)	精度(%)
MobileNetV1 on ImageNet	baseline	1.11	17	70.99
	量化	-	4.4(-74%)	70.18(-0.81)
	剪裁	0.74(-33%)	12(-29%)	70.4(-0.59)
YOLOV3-MobileNetV1 on COCO	baseline	20.64	95	29.3
	量化	-	25(-74%)	27.9(-1.4)
	剪裁	13.57(-34%)	67.60(-29%)	30.2(+0.9)

3.2.2 模型训练进阶应用—模型可解释性

模型可解释性的意义



无人驾驶

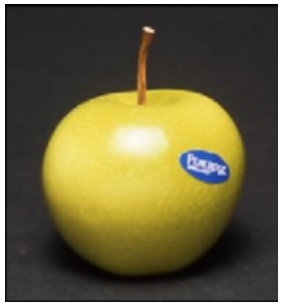
AI医疗

AI金融

一行代码完成模型可解释性--LIME算法工具

```
import paddlex as pdx
from paddlex.cls import transforms

test_transforms = transforms.Compose([
    transforms.ResizeByShort(short_size=256),
    transforms.CenterCrop(crop_size=224),
    transforms.Normalize()
])
test_dataset = pdx.datasets.ImageNet(
    data_dir='mini_imagenet_veg',
    file_list='mini_imagenet_veg/test_list.txt',
    label_list='mini_imagenet_veg/labels.txt',
    transforms=test_transforms)
model = pdx.load_model('output/resnet50/best_model')
pdx.interpret.visualize(
    img_file='mini_imagenet_veg/apple/1106.JPEG',
    model=model,
    dataset=test_dataset,
    algo='normlime',
    save_dir='normline_result'
```



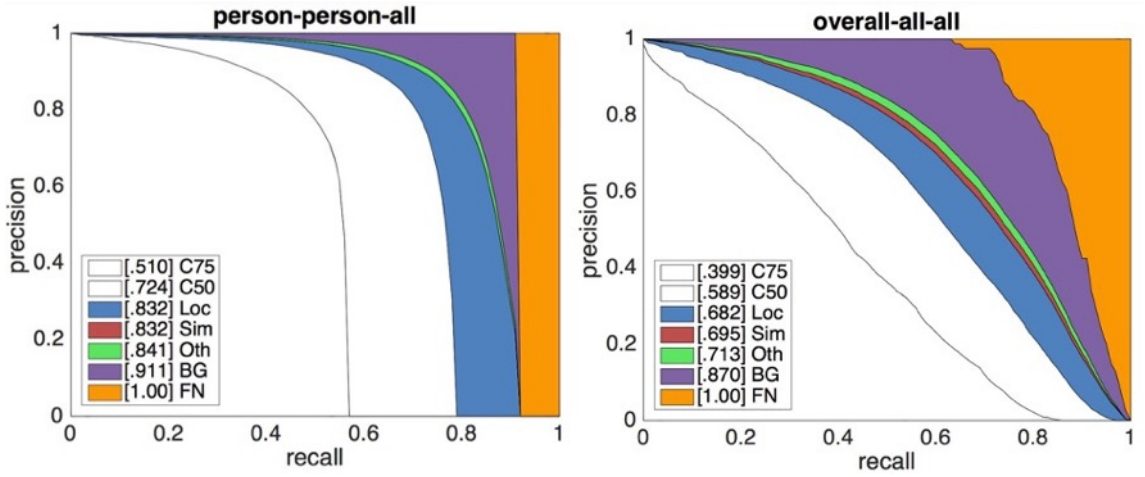
3.2.2 coco模型分析工具

PaddleX提供的接口paddlex.det.coco_error_analysis，只需要简单的代码便可实现数据的复核

```
import os
import os.path as osp
import paddlex as pdx

model_dir = 'output/faster_rcnn_r50_vd_dcn/best_model/'
save_dir = 'visualize/faster_rcnn_r50_vd_dcn'
if not osp.exists(save_dir):
    os.makedirs(save_dir)

eval_details_file = osp.join(model_dir, 'eval_details.json')
pdx.det.coco_error_analysis(eval_details_file, save_dir=save_dir)
```



逐个分析模型预测错误的原因，并将分析结果以图表的形式展示。分析结果右图所示，分别会展示出整体的map和单个类别的map变化情况

3 使用方法

3.3 模型部署

PaddleX 支持用户快速将模型导出，并部署到实际应用场景进行应用

1

部署模型导出

（图形化开发界面可一键实现）

#原生模型文件结构

OriginalModel/
|--model.yml
|--model.pdparams *#模型参数*
|--model.pdmodel*#整个训练网络*
|--model.opt *#优化器相关参数，支持恢复训练*

#预测模型文件结构

InferenceModel/
|--model.yml
|--__params__ *#预测模型参数*
|--__model__ *#预测模型结构*

预测时，将会把反向运算的参数、网络裁减掉得到预测网络模型，用于C++、Python等语言进行调用预测

「图形化开发界面」可直接将模型发布为预测格式
「函数库」可以通过一行代码实现模型转化

```
paddlex --export_inference --model_dir=./original_model --save_dir=./inference_model
```

2

轻量级服务化部署

（在线预测）

2

服务端部署

（本地、服务器，也可打包为服务）

2

移动端部署

（轻量化终端）

部署语言

- Python部署
- C++部署

操作系统

- Windows系统
- Linux系统
- Android系统

硬件架构

- X86（PC CPU）
- ARM （嵌入式）
- Nvidia GPU（Jetson等）
- EdgeBoard（FPGA）
- 树莓派
- …

Rockchip 瑞芯微电子 Cambricon 寒武纪科技 昆仑 MEDIATEK BITMAIN HISILICON

加速支持

- OpenVINO
- TensorRT

3.3.1 轻量集服务化部署

用于快速实现在线预测

1 模型转换

PaddleX的轻量级服务化部署借助于PaddleHub-Serving的能力，因此需安装Paddlehub并将PaddleX模型转化为hub模型

```
$ hub convert --model_dir XXXX \
              --module_name XXXX \
              --module_version XXXX \
              --output_dir XXXX
```

2 模型安装

将转换得到的 `.tar.gz` 格式的预训练模型压缩包，在进行部署前先安装到本机。其中 `${MODULE}` 为要安装的预训练模型文件路径。

```
$ hub install ${MODULE}
```

3 模型部署

使用 hub serving命令完成模型的一键部署

```
$ hub serving start --modules/-m
                    [Module1==Version1, Module2==Version2, ...] \
                    --port/-p XXXX
                    --config/-c XXXX
```

4 模型测试

在第2步模型安装的同时，会生成一个客户端请求示例，存放地址默认为*${HUB_HOME}/.paddlehub/modules*，我们提供了客户端示例代码*serving_client_demo.py*
将代码中的*IMAGE_PATH1*改成想要进行预测的图片路径后，在命令行执行以下代码，即可收到预测结果

```
python ~/.paddlehub/module/yolov3_hub/serving_client_demo.py
```

3.3.2 本地化部署

Python部署

■ 单张预测：

```
import paddlex as pdx
predictor = pdx.deploy.Predictor('./inference_model')
result = predictor.predict(image='./test_image.jpeg')
```

■ 批量预测：

```
import paddlex as pdx
predictor = pdx.deploy.Predictor('./inference_model')
image_list = ['./test_image1.jpeg',
    './test_image2.jpeg']
result = predictor.batch_predict(image_list=image_list)
```

■ 视频流预测：

```
import cv2
import paddlex as pdx
predictor = pdx.deploy.Predictor('./inference_model')
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        result = predictor.predict(frame)
        vis_img = pdx.det.visualize(frame, result, threshold=0.6, save_dir=None)
        cv2.imshow('test', vis_img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
```

3.3.2 本地化部署

C++部署

■ Windows平台：

Step1：下载PaddleX预测代码

```
d:
mkdir projects
cd projects
git clone https://github.com/PaddlePaddle/PaddleX.git
```

Step2：下载Paddle C++预测库 paddle_inference

PaddlePaddle C++ 预测库针对是否使用GPU、是否支持TensorRT、以及不同的CUDA版本提供了已经编译好的预测库，目前PaddleX依赖于Paddle 1.8.4，基于Paddle 1.8.4的Paddle预测库下载清单如下所示

版本说明	预测库(1.8.4版本)
cpu_avx_mkl	paddle_inference
cpu_avx_openblas	paddle_inference
cuda9.0_cudnn7_avx_mkl	paddle_inference
cuda9.0_cudnn7_avx_openblas	paddle_inference
cuda10.0_cudnn7_avx_mkl	paddle_inference

Step3：安装配置OpenCV

1. 在OpenCV官网下载适用于Windows平台的3.4.6版本， 下载地址
2. 运行下载的可执行文件，将OpenCV解压至指定目录，例如D:\projects\opencv
3. 配置环境变量，如下流程所示
   我的电脑->属性->高级系统设置->环境变量
   在系统变量中找到Path（如没有，自行创建），并双击编辑
   新建，将opencv路径填入并保存，如D:\projects\opencv\build\x64\vc14\bin

Step4：使用Visual Studio 2019直接编译CMake

Step5：预测及可视化

编译成功后，图片预测demo的入口程序为paddlex_inference\detector.exe，paddlex_inference\classifier.exe，paddlex_inference\segmenter.exe，用户可根据自己的模型类型选择

完整预测使用样例请参照：
https://paddlex.readthedocs.io/zh_CN/develop/deploy/server/cpp/windows.html#id1

3.3.2 本地化部署

C++部署

- Linux平台：

Step1：下载PaddleX预测代码

```
git clone https://github.com/PaddlePaddle/PaddleX.git
```

Step2：下载Paddle C++预测库 paddle_inference

类似windows平台，提供不同版本的ubuntu系统预测库

版本说明	预测库(1.8.4版本)
ubuntu14.04_cpu_avx_mkl	paddle_inference
ubuntu14.04_cpu_avx_openblas	paddle_inference
ubuntu14.04_cpu_noavx_openblas	paddle_inference
ubuntu14.04_cuda9.0_cudnn7_avx_mkl	paddle_inference
ubuntu14.04_cuda10.0_cudnn7_avx_mkl	paddle_inference

Step3：编译

Paddlex为大家提供编译cmake命令示例：scripts/build.sh

Step4：预测及可视化

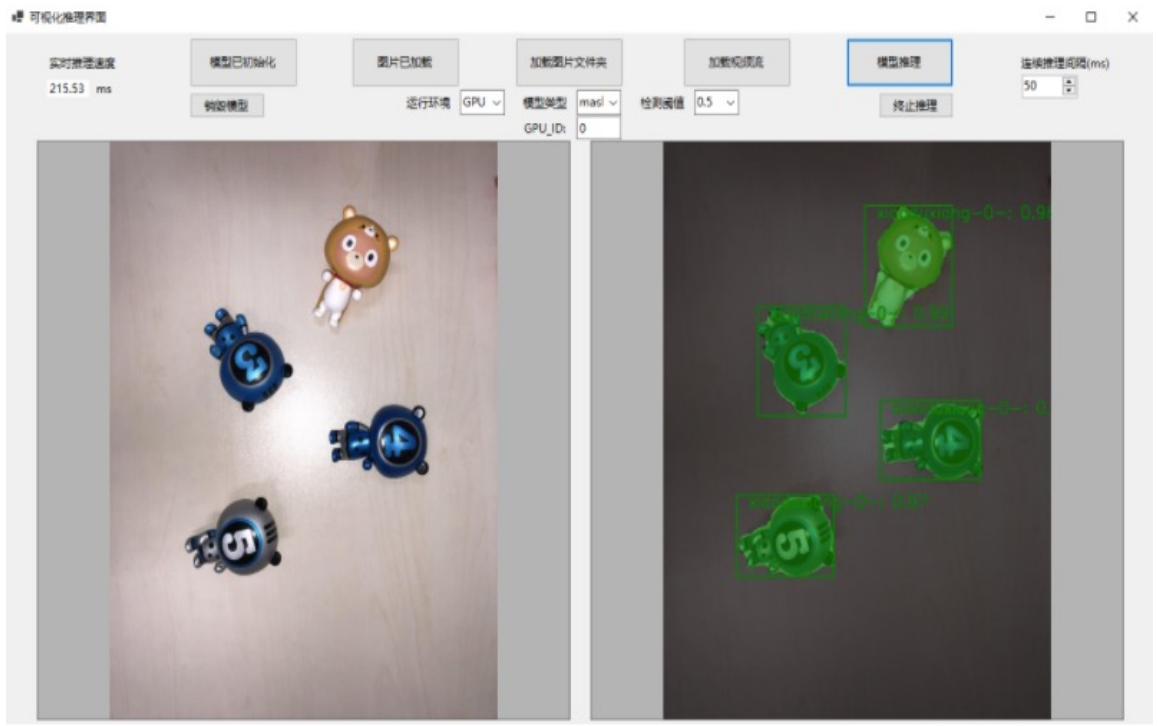
编译成功后，图片预测demo的可执行程序分别为
build/demo/detector，build/demo/classifier，
build/demo/segmenter，用户可根据自己的模型类型选择

完整编译参数配置说明及预测样例请参照：
https://paddlex.readthedocs.io/zh_CN/develop/deploy/server/cpp/linux.html

3.3.2 本地化部署

C++部署之 Deployment部署工具

PaddleX-Deployment提供了强大的部署性能，可同时兼容飞桨视觉套件**PaddleDetection、PaddleClas、PaddleSeg、PaddleX**统一部署，支持Windows、Linux等多种系统。同时提供了工业级别的**C#（Win系统）、QT（Win Linux）**工程示例。

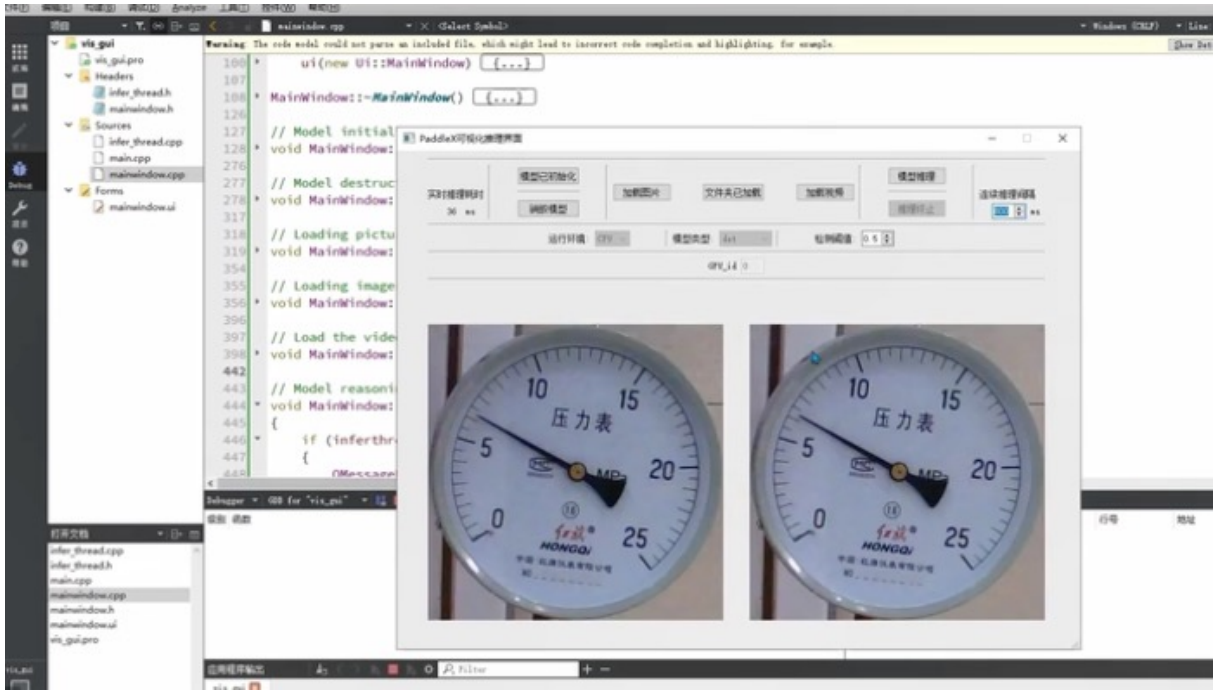


3.3.2 在NV-Jetson部署方案

Jetson 编译依赖说明

SO文件生成演示

QT移植部署演示



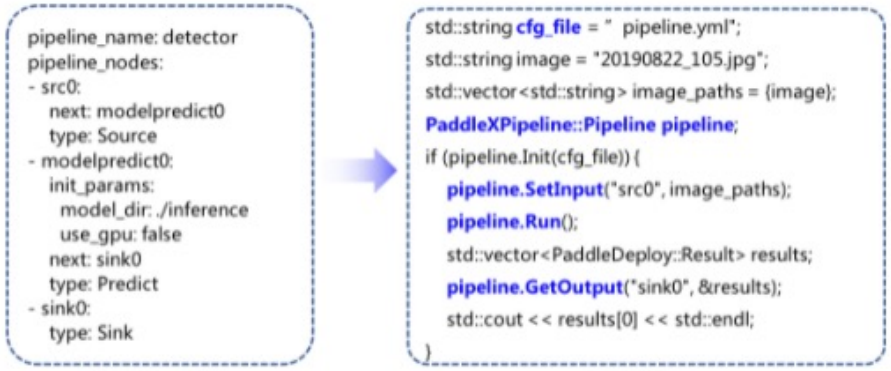
B站地址：https://www.bilibili.com/video/BV1vS4y1R7wT?p=2

3.3.2 本地化部署

C++部署之 Manufacture SDK部署工具

为更进一步地提升部署效率，PaddleX部署发布Manufacture SDK，提供工业级多端多平台部署加速的预编译飞桨部署开发包

- 通过配置业务逻辑流程文件即可以**低代码**方式快速完成推理部署。



- 通过配置文件，提供了**多模型串联**部署的方式，满足更多生产环境要求.

```
img = self.decode(img_file)
det_results = self.detector.predict(img)
filtered_results = self.filter_bboxes(det_results, score_threshold)
sub_imgs = self.roi_crop(img, filtered_results)
sub_imgs = self.resize(sub_imgs, METER_SHAPE)
seg_results = self.seg_predict(self.segmenter, sub_imgs, seg_batch_size)
seg_results = self.erode(seg_results, erode_kernel)
meter_readings = self.get_meter_reading(seg_results)
self.print_meter_readings(meter_readings)
self.visualize(img, filtered_results, meter_readings, save_dir)
```

图像解码 —> 检测表计 -> 过滤检测框 -> 提取检测框所在图像区域 -> 图像缩放 -> 指针和刻度分割 -> 读数后处理 -> 打印读数 -> 可视化预测

3.3.3 移动端部署

Paddle Lite 多平台高性能深度学习预测引擎

01 使用Paddle Lite OPT 对预测模型进行优化

下载 export_lite.py , 将模型一键转化

```
pip install paddlelite
python export_lite.py --model_dir /path/to/inference_model
       --save_file /path/to/lite_model_name --place place/to/run
```

开发者也可以使用bin文件优化模型（linux）

```
./opt --model_file=<model_path> \
      --param_file=<param_path> \
      --valid_targets=arm \
      --optimize_out_type=naive_buffer \
      --optimize_out=model_output_name
```

02 基于PaddleX Android SDK 快速部署模型

① 导入PaddleX Demo工程并直接运行，进行预测。

demo存放在：

```
/PaddleX/deploy/lite/android/demo
```

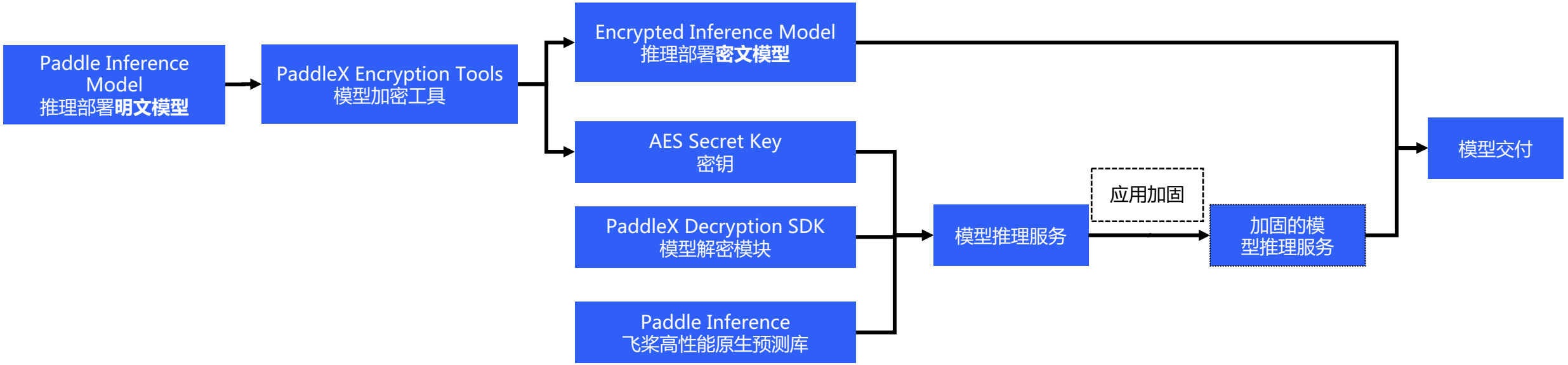
② 部署自定义模型

在Android Studio的project视图中根据文档修改 MODEL_PATH_DEFAULT 和 YAML_PATH_DEFAULT

③ 开发者也可基于PaddleX Android SDK进行二次开发

https://paddlex.readthedocs.io/zh_CN/develop/deploy/paddlelite/android.html#inferencepaddle-lite

3.3.4 模型加密部署



模型AES加密

一行命令加密模型，生成密钥

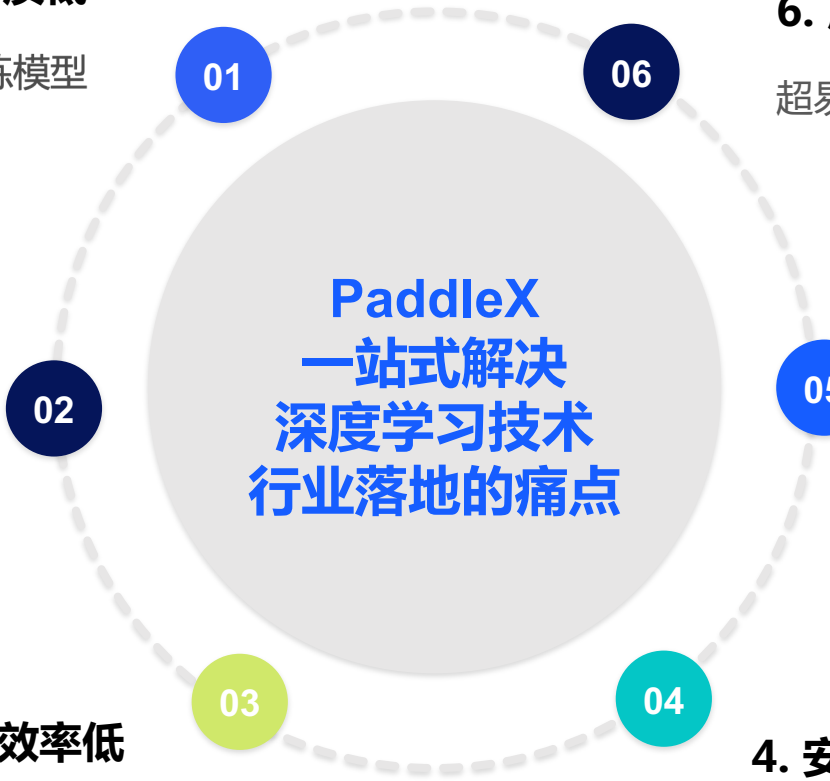
```
Command: ./paddlex_encrypt_tool -model_dir epoch_12 -save_dir encrypted_model
Output:  Encryption key:
         nNh3ryH53eZABbqi9ounbqRx7cv83aLYgW3WPSXSj7c=
```

密文模型解密

模型初始化接口传入密钥即可解密加载

```
PaddleX::Model model;
std::string key = "nNh3ryH53eZABbqi9ounbqRx7cv83aLYgW3WPSXSj7c=";
model.Init("encrypted_model", true, false, 0, key);
```

PaddleX一站式解决深度学习产业落地痛点



1. 标注数据量少质低

内置上百种数据增强策略、高质量预训练模型

2. 上手难

极简API，GUI模式开发

集成丰富企业案例及Demo

3.效率低

可视化分析工具VisualDL，模型可解释性算法

一键模型剪枝、量化

4. 安全性要求高

全本地开发、模型加密SDK

5. 部署难

移动端、服务端一键发布

附带部署示例

6. 成本高

超易用，方便集成/再开发

飞桨

END

2021.12