PaddleX

Dec 03, 2020

PaddleX 使用文档目录

PaddleX 是基于飞桨核心框架、开发套件和工具组件的深度学习全流程开发工具。具备 **全流程打通**、**融合产业实践**、**易用易集成**三大特点。

- 项目官网: http://www.paddlepaddle.org.cn/paddle/paddlex
- 项目 GitHub: https://github.com/PaddlePaddle/PaddleX
- 官方 QQ 用户群: 1045148026
- GitHub Issue 反馈: http://www.github.com/PaddlePaddle/PaddleX/issues

CHAPTER 1

10 分钟快速上手使用

本文档在一个小数据集上展示了如何通过 PaddleX 进行训练。本示例同步在 AIStudio 上，可直接在线体验模型训练。

本示例代码源于 Github tutorials/train/classification/mobilenetv3_small_ssld.py，用户可自行下载至本地运行。

PaddleX 中的所有模型训练跟随以下 3 个步骤，即可快速完成训练代码开发！

> **注意**：不同模型的 transforms、datasets 和训练参数都有较大差异，更多模型训练，可直接根据文档教程获取更多模型的训练代码。模型训练教程

PaddleX 的其它用法

- 使用 VisualDL 查看训练过程中的指标变化
- 加载训练保存的模型进行预测

1. 安装 PaddleX

安装相关过程和问题可以参考 PaddleX 的安装文档。

```
pip install paddlex -i https://mirror.baidu.com/pypi/simple
```

2. 准备蔬菜分类数据集

```
wget https://bj.bcebos.com/paddlex/datasets/vegetables_cls.tar.gz
tar xzvf vegetables_cls.tar.gz
```

3. 定义训练/验证图像处理流程 transforms

由于训练时数据增强操作的加入，因此模型在训练和验证过程中，数据处理流程需要分别进行定义。如下所示，代码在 `train_transforms` 中加入了RandomCrop和RandomHorizontalFlip两种数据增强方式，更多方法可以参考*数据增强文档*。

```
from paddlex.cls import transforms
train_transforms = transforms.Compose([
    transforms.RandomCrop(crop_size=224),
    transforms.RandomHorizontalFlip(),
    transforms.Normalize()
])
eval_transforms = transforms.Compose([
    transforms.ResizeByShort(short_size=256),
    transforms.CenterCrop(crop_size=224),
    transforms.Normalize()
])
```

4. 定义 dataset 加载图像分类数据集

定义数据集，`pdx.datasets.ImageNet` 表示读取 ImageNet 格式的分类数据集

- *paddlex.datasets.ImageNet 接口说明*
- *ImageNet 数据格式说明*

```
train_dataset = pdx.datasets.ImageNet(
    data_dir='vegetables_cls',
    file_list='vegetables_cls/train_list.txt',
    label_list='vegetables_cls/labels.txt',
    transforms=train_transforms,
    shuffle=True)
eval_dataset = pdx.datasets.ImageNet(
    data_dir='vegetables_cls',
    file_list='vegetables_cls/val_list.txt',
    label_list='vegetables_cls/labels.txt',
    transforms=eval_transforms)
```

5. 使用 MobileNetV3_small_ssld 模型开始训练

本文档中使用百度基于蒸馏方法得到的 MobileNetV3 预训练模型，模型结构与 MobileNetV3 一致，但精度更高。PaddleX 内置了 20 多种分类模型，查阅*PaddleX 模型库*了解更多分类模型。

```
num_classes = len(train_dataset.labels)
model = pdx.cls.MobileNetV3_small_ssld(num_classes=num_classes)
```

(continues on next page)

(continued from previous page)

```
model.train(num_epochs=20,
            train_dataset=train_dataset,
            train_batch_size=32,
            eval_dataset=eval_dataset,
            lr_decay_epochs=[4, 6, 8],
            save_dir='output/mobilenetv3_small_ssld',
            use_vdl=True)
```

6. 训练过程使用 VisualDL 查看训练指标变化

模型在训练过程中，训练指标和在验证集上的指标，均会以标准输出流形式，输出到命令终端。在用户设定 use_vdl=True 的前提下，也会使用 VisualDL 格式打点到 save_dir 目录下的 vdl_log 文件夹，用户可都终端通过如下命令启动 visualdl，查看可视化的指标变化趋势。

```
visualdl --logdir output/mobilenetv3_small_ssld --port 8001
```

服务启动后，通过浏览器打开 https://0.0.0.0:8001 或 https://localhost:8001 即可。

> 如果您使用的是 AIStudio 平台进行训练，不能通过此方式启动 visualdl，请参考 AIStudio VisualDL 启动教程使用

7. 加载训练保存的模型预测

模型在训练过程中，会每间隔一定轮数保存一次模型，在验证集上评估效果最好的一轮会保存在 save_dir 目录下的 best_model 文件夹。通过如下方式可加载模型，进行预测。

- load_model 接口说明
- 分类模型 predict 接口说明

```
import paddlex as pdx
model = pdx.load_model('output/mobilenetv3_small_ssld/best_model')
result = model.predict('vegetables_cls/bocai/100.jpg')
print("Predict Result: ", result)
```

预测结果输出如下，

```
Predict Result: Predict Result: [{'score': 0.9999393, 'category': 'bocai', 'category_id
↪': 0}]
```

更多使用教程

- 1.目标检测模型训练
- 2.语义分割模型训练

- 3.实例分割模型训练
- 4.模型太大，想要更小的模型，试试模型裁剪吧!

CHAPTER 2

快速安装

以下安装过程默认用户已安装好 **paddlepaddle-gpu 或 paddlepaddle(版本大于或等于 1.8.1)**，paddlepaddle 安装方式参照飞桨官网

2.1 pip 安装

注意其中 pycocotools 在 Windows 安装较为特殊，可参考下面的 Windows 安装命令

```
pip install paddlex -i https://mirror.baidu.com/pypi/simple
```

2.2 Anaconda 安装

Anaconda 是一个开源的 Python 发行版本，其包含了 conda、Python 等 180 多个科学包及其依赖项。使用 Anaconda 可以通过创建多个独立的 Python 环境，避免用户的 Python 环境安装太多不同版本依赖导致冲突。

- 参考Anaconda 安装 PaddleX 文档

2.3 代码安装

github 代码会跟随开发进度不断更新

```
git clone https://github.com/PaddlePaddle/PaddleX.git
cd PaddleX
git checkout develop
python setup.py install
```

2.4 pycocotools 安装问题

PaddleX 依赖 pycocotools 包，如安装 pycocotools 失败，可参照如下方式安装 pycocotools

Windows 安装时可能会提示缺少 `Microsoft Visual C++ 2015 build tools`，点击下载 VC build tools安装再执行如下 pip 命令

```
pip install cython
pip install git+https://gitee.com/jiangjiajun/philferriere-cocoapi.git
↪#subdirectory=PythonAPI
```

Linux/Mac 系统下，直接使用 pip 安装如下两个依赖即可

```
pip install cython
pip install pycocotools
```

CHAPTER 3

数据准备

3.1 数据标注工具

PaddleX 支持图像分类、目标检测、实例分割和语义分割四大视觉领域常见的任务，对于每类视觉任务，都支持了特定的数据格式。PaddleX 目前支持了图像分类的 ImageNet 格式，目标检测的 PascalVOC 格式，实例分割的 MSCOCO 格式（MSCOCO 也可以用于目标检测）以及语义分割数据格式。

3.1.1 常见标注工具

> 图像分类无需标注工具，用户只需以 txt 文件记录每张图片的类别标签即可。对于目标检测、实例分割和语义分割，PaddleX 已经与主流的标注工具进行了适配，用户可根据自己的需求，选择以下标注工具进行数据标注。

数据标注完成后，参照如下流程，将标注数据转为可用 PaddleX 模型训练的数据组织格式。

> 注：精灵标注的目标检测数据可以在工具内部导出为 PascalVOC 格式，因此 paddlex 未提供精灵标注数据到 PascalVOC 格式的转换

3.1.2 标注数据格式转换

目前所有标注工具，生成的标注文件，均为与原图同名的 json 格式文件，如 `1.jpg` 在标注完成后，则会在标注文件保存的目录生成 `1.json` 文件。转换时参照以下步骤

1. 将所有的原图文件放在同一个目录下，如 `pics` 目录

2. 将所有的标注 json 文件放在同一个目录下，如 annotations 目录

3. 使用如下命令进行转换

   注：精灵标注的目标检测数据可以在工具内部导出为 PascalVOC 格式，因此 paddlex 未提供精灵标注数据到 PascalVOC 格式的转换

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

   --source 表示数据标注来源，支持 labelme、jingling 和 easydata（分别表示数据来源于 LabelMe，精灵标注助手和 EasyData）--to 表示数据需要转换成为的格式，支持 ImageNet（图像分类）、PascalVOC（目标检测），MSCOCO（实例分割，也可用于目标检测）和 SEG(语义分割)--pics 指定原图所在的目录路径--annotations 指定标注文件所在的目录路径

3.2 数据格式说明

3.2.1 图像分类 ImageNet

数据文件夹结构

在 PaddleX 中，图像分类支持 ImageNet 数据集格式。数据集目录 data_dir 下包含多个文件夹，每个文件夹中的图像均属于同一个类别，文件夹的命名即为类别名（注意路径中不要包括中文，空格）。如下为示例结构

```
MyDataset/ # 图像分类数据集根目录
|--dog/ # 当前文件夹所有图片属于 dog 类别
|   |--d1.jpg
|   |--d2.jpg
|   |--...
|   |--...
|
|--...
|
|--snake/ # 当前文件夹所有图片属于 snake 类别
|   |--s1.jpg
|   |--s2.jpg
|   |--...
|   |--...
```

划分训练集验证集

为了用于训练，我们需要在 `MyDataset` 目录下准备 `train_list.txt`, `val_list.txt` 和 `labels.txt` 三个文件，分别用于表示训练集列表，验证集列表和类别标签列表。点击下载图像分类示例数据集

labels.txt

labels.txt 用于列出所有类别，类别对应行号表示模型训练过程中类别的 id(行号从 0 开始计数)，例如 labels.txt 为以下内容

```
dog
cat
snake
```

即表示该分类数据集中共有 3 个类别，分别为 `dog`，`cat` 和 `snake`，在模型训练中 `dog` 对应的类别 id 为 0，`cat` 对应 1，以此类推

train_list.txt

train_list.txt 列出用于训练时的图片集合，与其对应的类别 id，示例如下

```
dog/d1.jpg 0
dog/d2.jpg 0
cat/c1.jpg 1
... ...
snake/s1.jpg 2
```

其中第一列为相对对 `MyDataset` 的相对路径，第二列为图片对应类别的类别 id

val_list.txt

val_list 列出用于验证时的图片集成，与其对应的类别 id，格式与 train_list.txt 一致

PaddleX 数据集加载

示例代码如下，

```
import paddlex as pdx
from paddlex.cls import transforms
train_transforms = transforms.Compose([
    transforms.RandomCrop(crop_size=224), transforms.RandomHorizontalFlip(),
    transforms.Normalize()
])
eval_transforms = transforms.Compose([
    transforms.ResizeByShort(short_size=256),
    transforms.CenterCrop(crop_size=224), transforms.Normalize()
```

(continues on next page)

(continued from previous page)

```
])
train_dataset = pdx.datasets.ImageNet(
                    data_dir='./MyDataset',
                    file_list='./MyDataset/train_list.txt',
                    label_list='./MyDataset/labels.txt',
                    transforms=train_transforms)
eval_dataset = pdx.datasets.ImageNet(
                    data_dir='./MyDataset',
                    file_list='./MyDataset/eval_list.txt',
                    label_list='./MyDataset/labels.txt',
                    transforms=eval_transforms)
```

3.2.2 目标检测 PascalVOC

数据集文件夹结构

在 PaddleX 中，目标检测支持 PascalVOC 数据集格式。建议用户将数据集按照如下方式进行组织，原图均放在同一目录，如 `JPEGImages`，标注的同名 xml 文件均放在同一目录，如 `Annotations`，示例如下

```
MyDataset/ # 目标检测数据集根目录
|--JPEGImages/ # 原图文件所在目录
|   |--1.jpg
|   |--2.jpg
|   |--...
|   |--...
|
|--Annotations/ # 标注文件所在目录
|   |--1.xml
|   |--2.xml
|   |--...
|   |--...
```

划分训练集验证集

为了用于训练，我们需要在 `MyDataset` 目录下准备 `train_list.txt`, `val_list.txt` 和 `labels.txt` 三个文件，分别用于表示训练集列表，验证集列表和类别标签列表。点击下载目标检测示例数据集

labels.txt

labels.txt 用于列出所有类别，类别对应行号表示模型训练过程中类别的 id(行号从 0 开始计数)，例如 labels.txt 为以下内容

```
dog
cat
snake
```

表示该检测数据集中共有 3 个目标类别，分别为 dog，cat 和 snake，在模型训练中 dog 对应的类别 id 为 0, cat 对应 1，以此类推

train_list.txt

train_list.txt 列出用于训练时的图片集合，与其对应的标注文件，示例如下

```
JPEGImages/1.jpg Annotations/1.xml
JPEGImages/2.jpg Annotations/2.xml
... ...
```

其中第一列为原图相对 MyDataset 的相对路径，第二列为标注文件相对 MyDataset 的相对路径

val_list.txt

val_list 列出用于验证时的图片集成，与其对应的标注文件，格式与 val_list.txt 一致

PaddleX 数据集加载

示例代码如下，

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

train_transforms = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.Normalize(),
    transforms.ResizeByShort(short_size=800, max_size=1333),
    transforms.Padding(coarsest_stride=32)
])

eval_transforms = transforms.Compose([
    transforms.Normalize(),
    transforms.ResizeByShort(short_size=800, max_size=1333),
    transforms.Padding(coarsest_stride=32),
])

train_dataset = pdx.datasets.VOCDetection(
                        data_dir='./MyDataset',
                        file_list='./MyDataset/train_list.txt',
```

(continues on next page)

(continued from previous page)

```
                        label_list='./MyDataset/labels.txt',
                        transforms=train_transforms)
eval_dataset = pdx.datasets.VOCDetection(
                        data_dir='./MyDataset',
                        file_list='./MyDataset/val_list.txt',
                        label_list='MyDataset/labels.txt',
                        transforms=eval_transforms)
```

3.2.3 实例分割 MSCOCO

数据集文件夹结构

在 PaddleX 中，实例分割支持 MSCOCO 数据集格式（MSCOCO 格式同样也可以用于目标检测）。建议用户将数据集按照如下方式进行组织，原图均放在同一目录，如 JPEGImages，标注文件（如 annotations.json）放在与 JPEGImages 所在目录同级目录下，示例结构如下

```
MyDataset/ # 实例分割数据集根目录
|--JPEGImages/ # 原图文件所在目录
|   |--1.jpg
|   |--2.jpg
|   |--...
|   |--...
|
|--annotations.json # 标注文件所在目录
```

划分训练集验证集

在 PaddleX 中，为了区分训练集和验证集，在 `MyDataset` 同级目录，使用不同的 json 表示数据的划分，例如 `train.json` 和 `val.json`。点击下载实例分割示例数据集。

MSCOCO 数据的标注文件采用 json 格式，用户可使用 Labelme, 精灵标注助手或 EasyData 等标注工具进行标注，参见数据标注工具

PaddleX 加载数据集

示例代码如下，

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

(continues on next page)

(continued from previous page)

```
train_transforms = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.Normalize(),
    transforms.ResizeByShort(short_size=800, max_size=1333),
    transforms.Padding(coarsest_stride=32)
])

eval_transforms = transforms.Compose([
    transforms.Normalize(),
    transforms.ResizeByShort(short_size=800, max_size=1333),
    transforms.Padding(coarsest_stride=32),
])

train_dataset = pdx.dataset.CocoDetection(
                    data_dir='./MyDataset/JPEGImages',
                    ann_file='./MyDataset/train.json',
                    transforms=train_transforms)
eval_dataset = pdx.dataset.CocoDetection(
                    data_dir='./MyDataset/JPEGImages',
                    ann_file='./MyDataset/val.json',
                    transforms=eval_transforms)
```

3.2.4 语义分割 Seg

数据集文件夹结构

在 PaddleX 中，**标注文件为 png 文件**。建议用户将数据集按照如下方式进行组织，原图均放在同一目录，如 `JPEGImages`，标注的同名 png 文件均放在同一目录，如 `Annotations`，示例如下

```
MyDataset/ # 语义分割数据集根目录
|--JPEGImages/ # 原图文件所在目录
|   |--1.jpg
|   |--2.jpg
|   |--...
|   |--...
|
|--Annotations/ # 标注文件所在目录
|   |--1.png
|   |--2.png
```

(continues on next page)

(continued from previous page)

```
|   |--...
|   |--...
```

语义分割的标注图像，如 1.png，为单通道图像，像素标注类别需要从 0 开始递增（一般 0 表示 background 背景），例如 0，1，2，3 表示 4 种类别，标注类别最多 255 个类别 (其中像素值 255 不参与训练和评估)。

划分训练集验证集

为了用于训练，我们需要在 `MyDataset` 目录下准备 `train_list.txt`，`val_list.txt` 和 `labels.txt` 三个文件，分别用于表示训练集列表，验证集列表和类别标签列表。点击下载语义分割示例数据集

labels.txt

labels.txt 用于列出所有类别，类别对应行号表示模型训练过程中类别的 id(行号从 0 开始计数)，例如 labels.txt 为以下内容

```
backgrond
human
car
```

表示该检测数据集中共有 3 个分割类别，分别为 `background`，`human` 和 `car`，在模型训练中 `background` 对应的类别 id 为 0，`human` 对应 1，以此类推，如不知具体类别标签，可直接在 labels.txt 逐行写 0，1，2··· 序列即可。

train_list.txt

train_list.txt 列出用于训练时的图片集合，与其对应的标注文件，示例如下

```
JPEGImages/1.jpg Annotations/1.png
JPEGImages/2.jpg Annotations/2.png
... ...
```

其中第一列为原图相对 `MyDataset` 的相对路径，第二列为标注文件相对 `MyDataset` 的相对路径

val_list.txt

val_list 列出用于验证时的图片集成，与其对应的标注文件，格式与 val_list.txt 一致

PaddleX 数据集加载

示例代码如下，

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

(continues on next page)

(continued from previous page)

```
train_transforms = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.ResizeRangeScaling(),
    transforms.RandomPaddingCrop(crop_size=512),
    transforms.Normalize()
])

eval_transforms = transforms.Compose([
    transforms.ResizeByLong(long_size=512),
    transforms.Padding(target_size=512),
    transforms.Normalize()
])

train_dataset = pdx.datasets.SegDataset(
                        data_dir='./MyDataset',
                        file_list='./MyDataset/train_list.txt',
                        label_list='./MyDataset/labels.txt',
                        transforms=train_transforms)
eval_dataset = pdx.datasets.SegDataset(
                        data_dir='./MyDataset',
                        file_list='./MyDataset/val_list.txt',
                        label_list='MyDataset/labels.txt',
                        transforms=eval_transforms)
```

CHAPTER 4

模型训练

PaddleX 集成了 PaddleClas、PaddleDetection 和 PaddleSeg 三大 CV 工具套件中，在工业领域应用成熟的模型，并提供了统一，易用的 API 使用接口，帮助用户快速完成视觉领域的图像分类、目标检测、实例分割和语义分割模型的训练。

4.1 图像分类

4.1.1 介绍

PaddleX 共提供了 20+ 的图像分类模型，可满足开发者不同场景的需求下的使用。

- **Top1 精度**: 模型在 ImageNet 数据集上的测试精度
- **预测速度**：单张图片的预测用时（不包括预处理和后处理)
- "- "表示指标暂未更新

4.1.2 开始训练

代码保存到本地后，即可直接训练，**训练代码会自动下载训练数据开始训练**

如保存为 `mobilenetv3_small_ssld.py`，如下命令即可开始训练

```
python mobilenetv3_small_ssld.py
```

4.1.3 相关文档

- **【重要】**针对自己的机器环境和数据，调整训练参数？先了解下 PaddleX 中训练参数作用。——>> 传送门
- **【有用】**没有机器资源？使用 AIStudio 免费的 GPU 资源在线训练模型。——>> 传送门
- **【拓展】**更多图像分类模型，查阅*PaddleX 模型库*和API 使用文档。

4.2 目标检测

4.2.1 介绍

PaddleX 目前提供了 FasterRCNN 和 YOLOv3 两种检测结构, 多种 backbone 模型，可满足开发者不同场景和性能的需求。

- **Box MMAP**: 模型在 COCO 数据集上的测试精度
- **预测速度**：单张图片的预测用时（不包括预处理和后处理）
- “- “表示指标暂未更新

4.2.2 开始训练

代码保存到本地后，即可直接训练，**训练代码会自动下载训练数据开始训练**

如保存为 `yolov3_mobilenetv1.py`，如下命令即可开始训练

```
python yolov3_mobilenetv1.py
```

4.2.3 相关文档

- **【重要】**针对自己的机器环境和数据，调整训练参数？先了解下 PaddleX 中训练参数作用。——>> 传送门
- **【有用】**没有机器资源？使用 AIStudio 免费的 GPU 资源在线训练模型。——>> 传送门
- **【拓展】**更多目标检测模型，查阅*PaddleX 模型库*和API 使用文档。

4.3 实例分割

4.3.1 介绍

PaddleX 目前提供了 MaskRCNN 实例分割模型结构, 多种 backbone 模型，可满足开发者不同场景和性能的需求。

- **Box MMAP/Seg MMAP**: 模型在 COCO 数据集上的测试精度
- **预测速度**：单张图片的预测用时（不包括预处理和后处理)
- "-"表示指标暂未更新

4.3.2 开始训练

代码保存到本地后，即可直接训练，**训练代码会自动下载训练数据开始训练**

如保存为 `mask_r50_fpn.py`，如下命令即可开始训练

```
python mask_r50_fpn.py
```

4.3.3 相关文档

- **【重要】**针对自己的机器环境和数据，调整训练参数？先了解下 PaddleX 中训练参数作用。——>> 传送门
- **【有用】**没有机器资源？使用 AIStudio 免费的 GPU 资源在线训练模型。——>> 传送门
- **【拓展】**更多实例分割模型，查阅*PaddleX 模型库*和API 使用文档。

4.4 语义分割

4.4.1 介绍

PaddleX 目前提供了 DeepLabv3p、UNet、HRNet 和 FastSCNN 四种语义分割结构, 多种 backbone 模型，可满足开发者不同场景和性能的需求。

- **mIOU**: 模型在 CityScape 数据集上的测试精度
- **预测速度**：单张图片的预测用时（不包括预处理和后处理)
- "-"表示指标暂未更新

4.4.2 开始训练

代码保存到本地后，即可直接训练，**训练代码会自动下载训练数据开始训练**

如保存为 `deeplabv3p_mobilenetv2_x0.25.py`，如下命令即可开始训练

```
python deeplabv3p_mobilenetv2_x0.25.py
```

4.4.3 相关文档

- **【重要】**针对自己的机器环境和数据，调整训练参数？先了解下 PaddleX 中训练参数作用。——>> 传送门
- **【有用】**没有机器资源？使用 AIStudio 免费的 GPU 资源在线训练模型。——>> 传送门
- **【拓展】**更多语义分割模型，查阅*PaddleX 模型库*和API 使用文档。

4.5 加载模型预测

PaddleX 可以使用 `paddlex.load_model` 接口加载模型（包括训练过程中保存的模型，导出的部署模型，量化模型以及裁剪的模型）进行预测，同时 PaddleX 中也内置了一系列的可视化工具函数，帮助用户方便地检查模型的效果。

注意：使用 `paddlex.load_model` 接口加载仅用于模型预测，如需要在此模型基础上继续训练，可以将该模型作为预训练模型进行训练，具体做法是在训练代码中，将 train 函数中的 `pretrain_weights` 参数指定为预训练模型路径。

4.5.1 图像分类

点击下载如下示例代码中模型

```
import paddlex as pdx
test_jpg = 'mobilenetv3_small_ssld_imagenet/test.jpg'
model = pdx.load_model('mobilenetv3_small_ssld_imagenet')
result = model.predict(test_jpg)
print("Predict Result: ", result)
```

结果输入如下

```
Predict Result: [{'category_id': 21, 'category': 'killer_whale', 'score': 0.8262267}]
```

测试图片如下

- 分类模型 predict 接口说明文档

4.5.2 目标检测

点击下载如下示例代码中模型

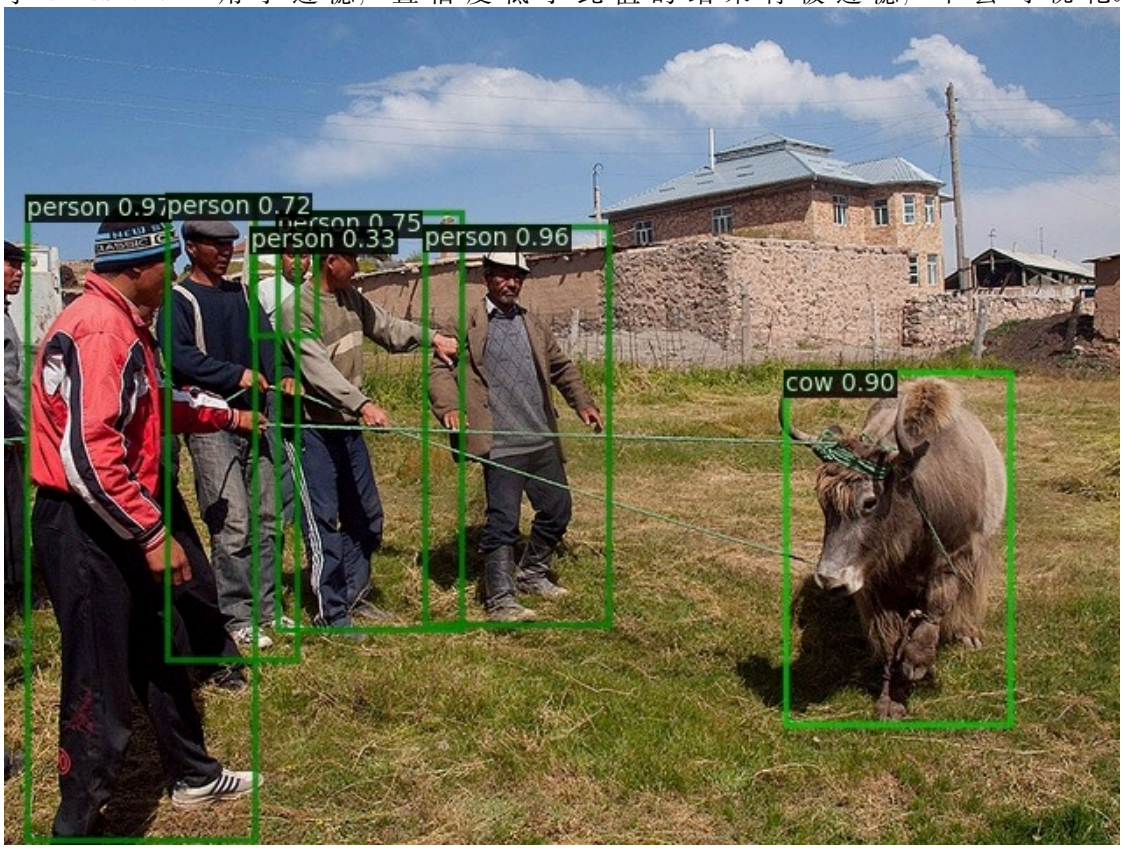
```
import paddlex as pdx
test_jpg = 'yolov3_mobilenetv1_coco/test.jpg'
model = pdx.load_model('yolov3_mobilenetv1_coco')

# predict 接口并未过滤低置信度识别结果，用户根据需求按 score 值进行过滤
result = model.predict(test_jpg)

# 可视化结果存储在./visualized_test.jpg，见下图
pdx.det.visualize(test_jpg, result, threshold=0.3, save_dir='./')
```

- YOLOv3 模型 predict 接口说明文档
- 可视化 pdx.det.visualize 接口说明文档

注意：目标检测和实例分割模型在调用 predict 接口得到的结果需用户自行过滤低置信度结果，在 paddlex.det.visualize 接口中，我们提供了 threshold 用于过滤，置信度低于此值的结果将被过滤，不会可视化。



4.5.3 实例分割

点击下载如下示例代码中模型

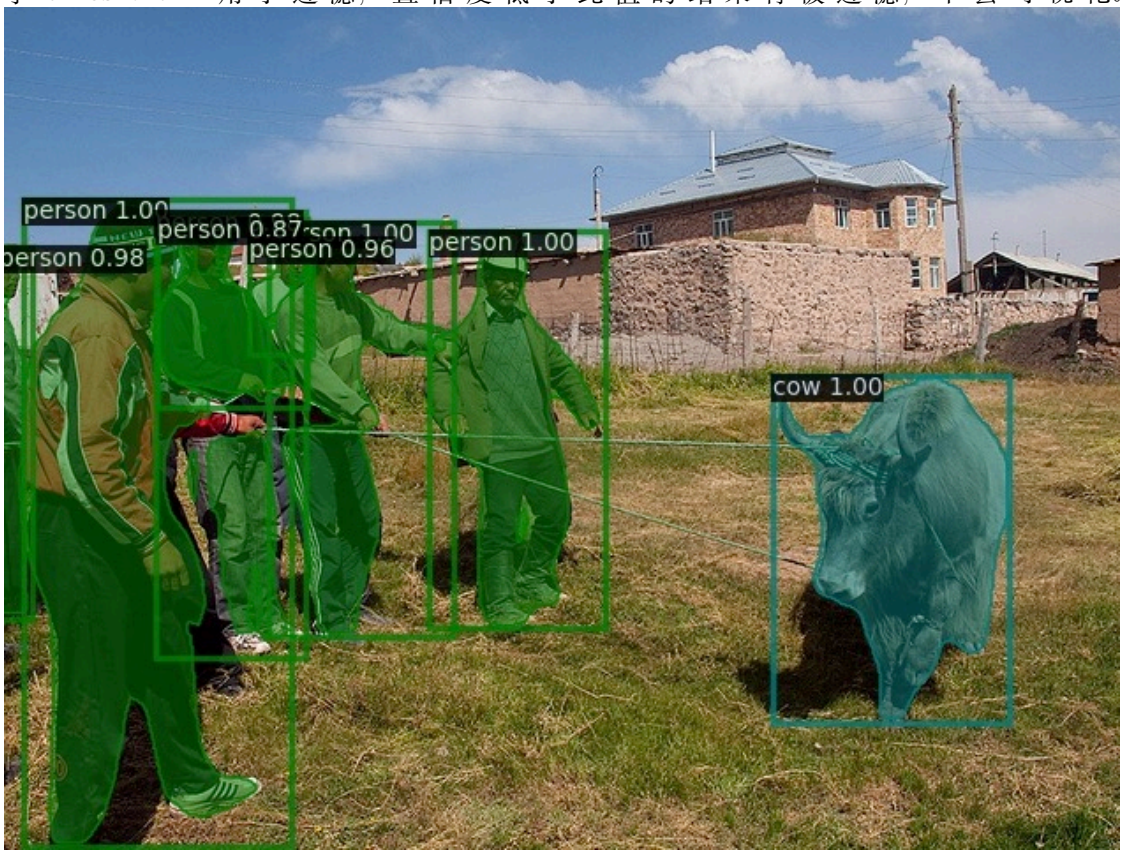
```
import paddlex as pdx
test_jpg = 'mask_r50_fpn_coco/test.jpg'
model = pdx.load_model('mask_r50_fpn_coco')

# predict 接口并未过滤低置信度识别结果，用户根据需求按 score 值进行过滤
result = model.predict(test_jpg)

# 可视化结果存储在./visualized_test.jpg，见下图
pdx.det.visualize(test_jpg, result, threshold=0.5, save_dir='./')
```

- MaskRCNN 模型 predict 接口说明文档
- 可视化 pdx.det.visualize 接口说明文档

注意：目标检测和实例分割模型在调用 predict 接口得到的结果需用户自行过滤低置信度结果，在 paddlex.det.visualize 接口中，我们提供了 threshold 用于过滤，置信度低于此值的结果将被过滤，不会可视化。



4.5.4 语义分割

```
import paddlex as pdx
test_jpg = './deeplabv3p_mobilenetv2_coco/test.jpg'
model = pdx.load_model('./deeplabv3p_mobilenetv2_coco')
result = model.predict(test_jpg)
pdx.seg.visualize(test_jpg, result, weight=0.0, save_dir='./')
```

在上述示例代码中，通过调用 `paddlex.seg.visualize` 可以对语义分割的预测结果进行可视化，可视化的结果保存在 `save_dir` 下。其中 `weight` 参数用于调整预测结果和原图结果融合展现时的权重，0.0 时只展示预测结果 mask 的可视化，1.0 时只展示原图可视化。

4.5.5 公开数据集训练模型下载

PaddleX 提供了部分公开数据集上训练好的模型，用户可以直接下载后参照本文档加载使用。

PaddleX 的 `load_model` 接口可以满足用户一般的模型调研需求，如若为更高性能的预测部署，可以参考如下文档

- *服务端 Python 部署*
- 服务端 C++ 部署

CHAPTER 5

模型部署

5.1 部署模型导出

在服务端部署的模型需要首先将模型导出为 inference 格式模型，导出的模型将包括 `__model__`、`__params__` 和 `model.yml` 三个文名，分别为模型的网络结构，模型权重和模型的配置文件（包括数据预处理参数等等）。在安装完 PaddleX 后，在命令行终端使用如下命令导出模型到当前目录 `inferece_model` 下。

可直接下载小度熊分拣模型测试本文档的流程xiaoduxiong_epoch_12.tar.gz

```
paddlex --export_inference --model_dir=./xiaoduxiong_epoch_12 --save_dir=./inference_
↪model
```

使用 TensorRT 预测时，需指定模型的图像输入 shape:[w,h]。**注**：

- 分类模型请保持于训练时输入的 shape 一致。
- 指定 [w,h] 时，w 和 h 中间逗号隔开，不允许存在空格等其他字符

```
paddlex --export_inference --model_dir=./xiaoduxiong_epoch_12 --save_dir=./inference_
↪model --fixed_input_shape=[640,960]
```

5.2 服务端部署

5.2.1 Python 部署

PaddleX 已经集成了基于 Python 的高性能预测接口，在安装 PaddleX 后，可参照如下代码示例，进行预测。相关的接口文档可参考paddlex.deploy

导出预测模型

可参考模型导出将模型导出为 inference 格式的模型。

预测部署

点击下载测试图片 xiaoduxiong_test_image.tar.gz

- 单张图片预测

```
import paddlex as pdx
predictor = pdx.deploy.Predictor('./inference_model')
result = predictor.predict(image='xiaoduxiong_test_image/JPEGImages/WeChatIMG110.jpeg')
```

- 批量图片预测

```
import paddlex as pdx
predictor = pdx.deploy.Predictor('./inference_model')
image_list = ['xiaoduxiong_test_image/JPEGImages/WeChatIMG110.jpeg',
    'xiaoduxiong_test_image/JPEGImages/WeChatIMG111.jpeg']
result = predictor.predict(image_list=image_list)
```

关于预测速度的说明：采用 Paddle 的 Predictor 进行预测时，由于涉及到内存显存初始化等原因，在模型加载后刚开始预测速度会较慢，一般在模型运行 20~50 后（即预测 20~30 张图片）预测速度才会稳定。

预测性能对比

测试环境

- CUDA 9.0
- CUDNN 7.5
- PaddlePaddle 1.71
- GPU: Tesla P40

- AnalysisPredictor 指采用 Python 的高性能预测方式
- Executor 指采用 paddlepaddle 普通的 python 预测方式
- Batch Size 均为 1，耗时单位为 ms/image，只计算模型运行时间，不包括数据的预处理和后处理

性能对比

5.2.2 服务端 C++ 部署

Windows 平台部署

说明

Windows 平台下，我们使用 `Visual Studio 2019 Community` 进行了测试。微软从 `Visual Studio 2017` 开始即支持直接管理 `CMake` 跨平台编译项目，但是直到 `2019` 才提供了稳定和完全的支持，所以如果你想使用 CMake 管理项目编译构建，我们推荐你使用 `Visual Studio 2019` 环境下构建。

前置条件

- Visual Studio 2019
- CUDA 9.0 / CUDA 10.0, CUDNN 7+ （仅在使用 GPU 版本的预测库时需要）
- CMake 3.0+

请确保系统已经安装好上述基本软件，我们使用的是 `VS2019` 的社区版。

下面所有示例以工作目录为 `D:\projects` 演示。

Step1: 下载 PaddleX 预测代码

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

说明：其中 `C++` 预测代码在 `PaddleX\deploy\cpp` 目录，该目录不依赖任何 `PaddleX` 下其他目录。

Step2: 下载 PaddlePaddle 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，基于 Paddle 1.8 的 Paddle 预测库下载链接如下所示:

请根据实际情况选择下载，如若以上版本不满足您的需求，请至C++ 预测库下载列表选择符合的版本。

将预测库解压后，其所在目录（例如 D:\projects\fluid_inference\）下主要包含的内容有：

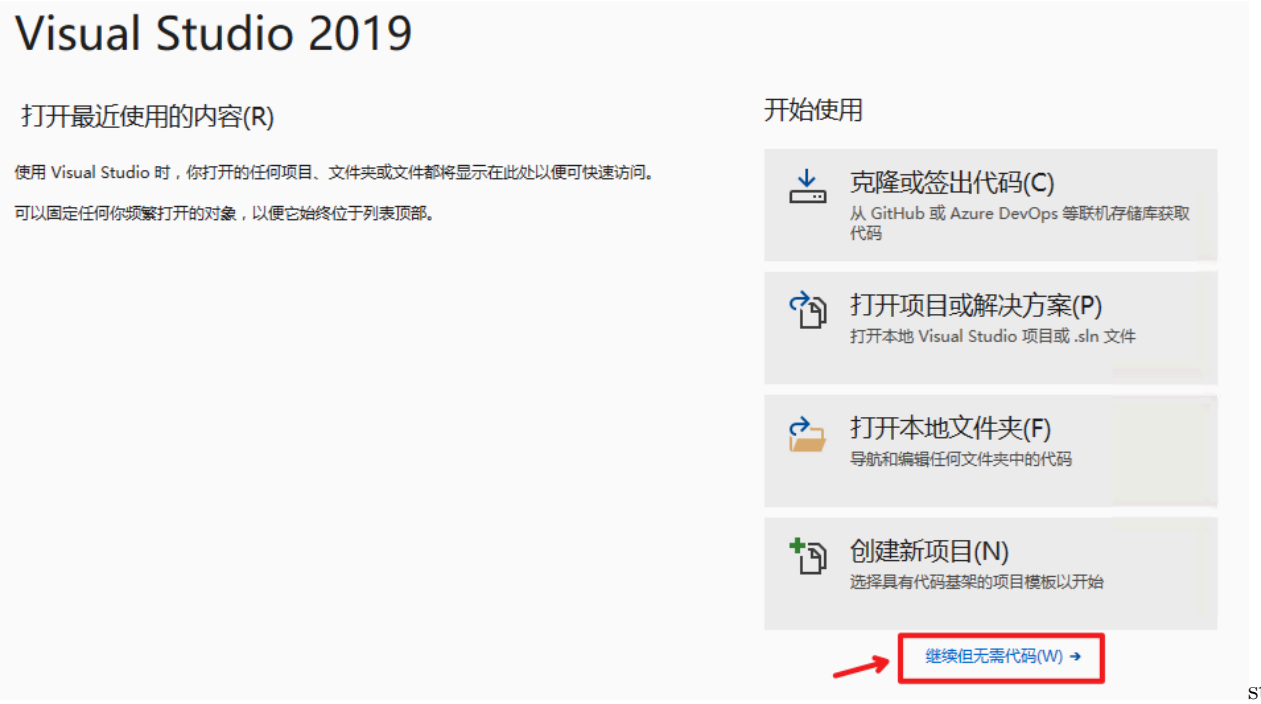
```
  \paddle\ # paddle 核心库和头文件
|
  \third_party\ # 第三方依赖库和头文件
|
  \version.txt # 版本和编译信息
```

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

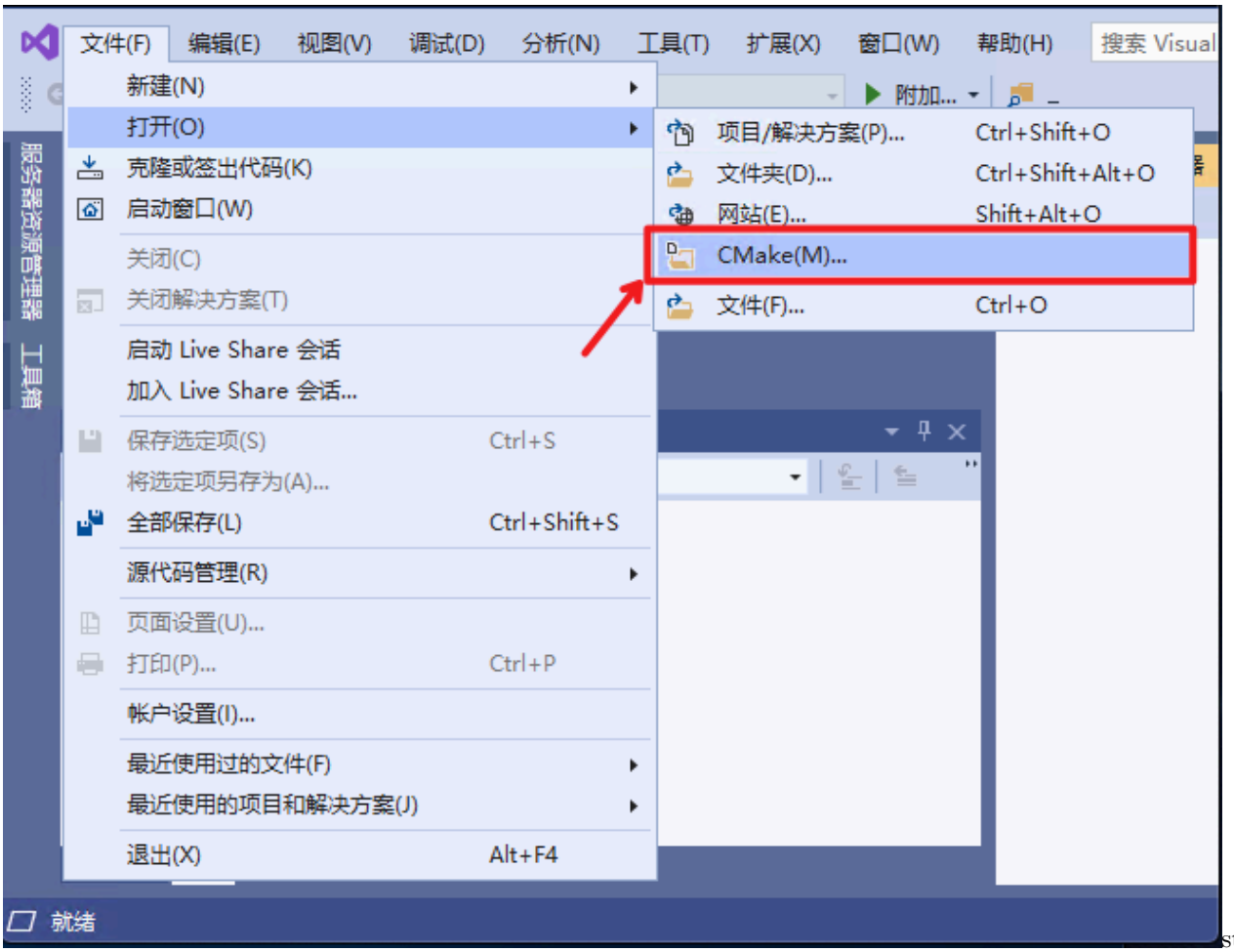
Step4: 使用 Visual Studio 2019 直接编译 CMake

1. 打开 Visual Studio 2019 Community，点击继续但无需代码



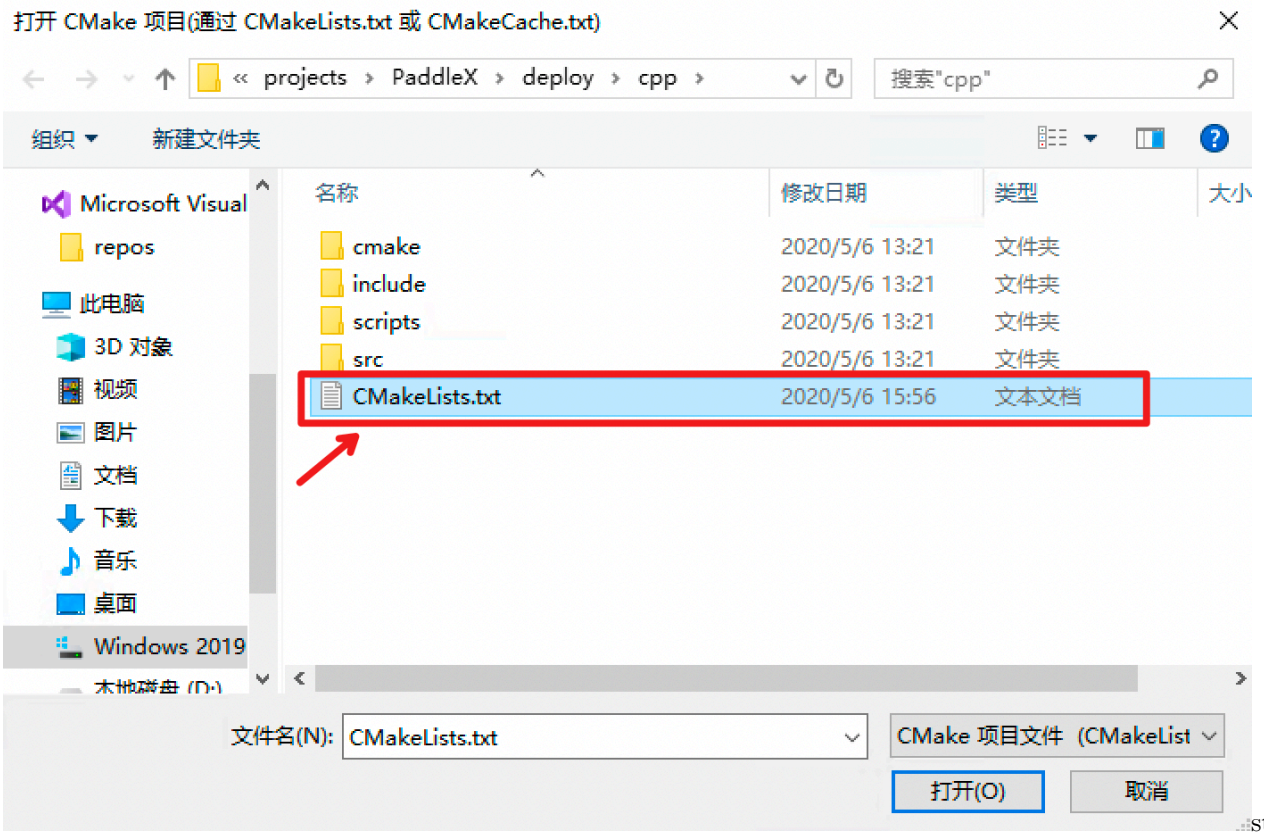
step2

2. 点击：文件-> 打开->CMake



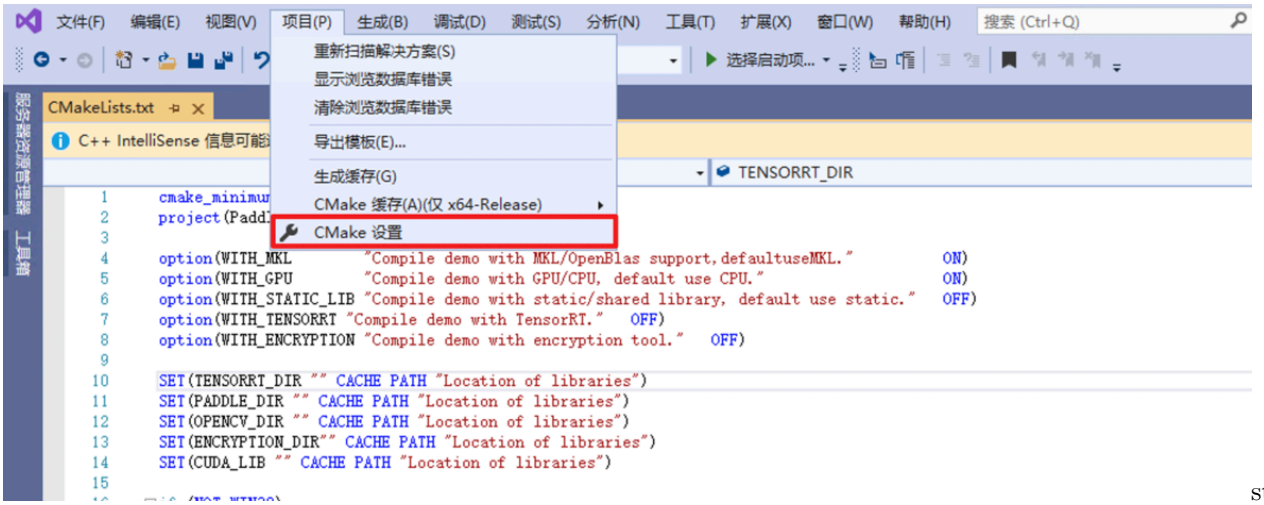
step2.1

选择 C++ 预测代码所在路径（例如 `D:\projects\PaddleX\deploy\cpp`），并打开 `CMakeList.txt`：



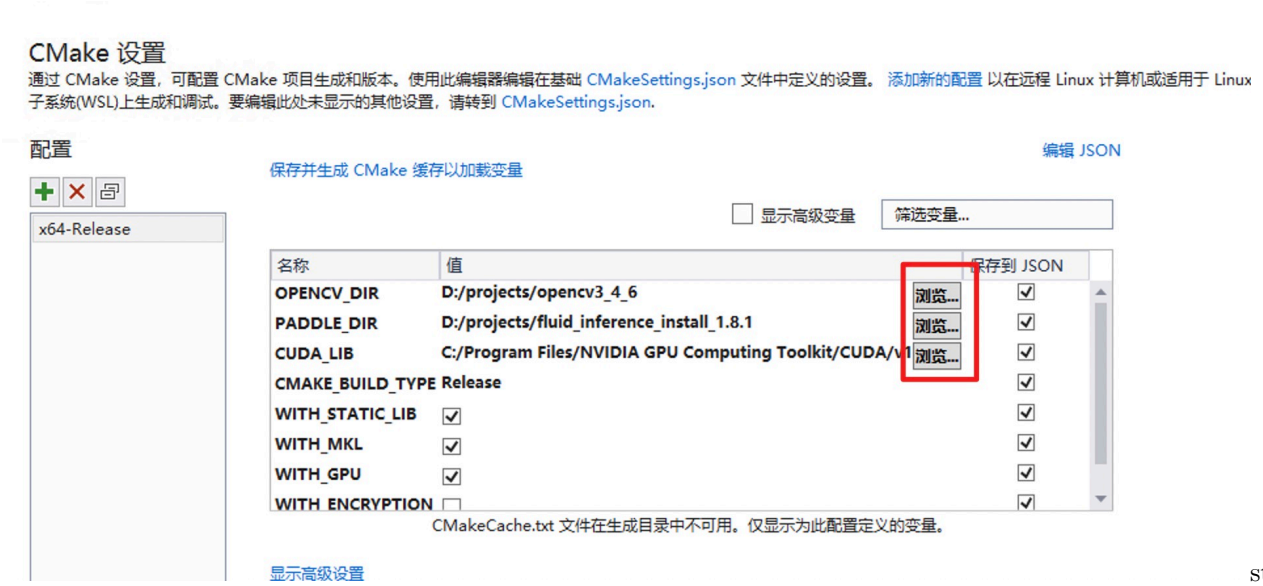
step2.2

3. 点击：项目->CMake 设置



step3

4. 点击浏览，分别设置编译选项指定 CUDA、OpenCV、Paddle 预测库的路径

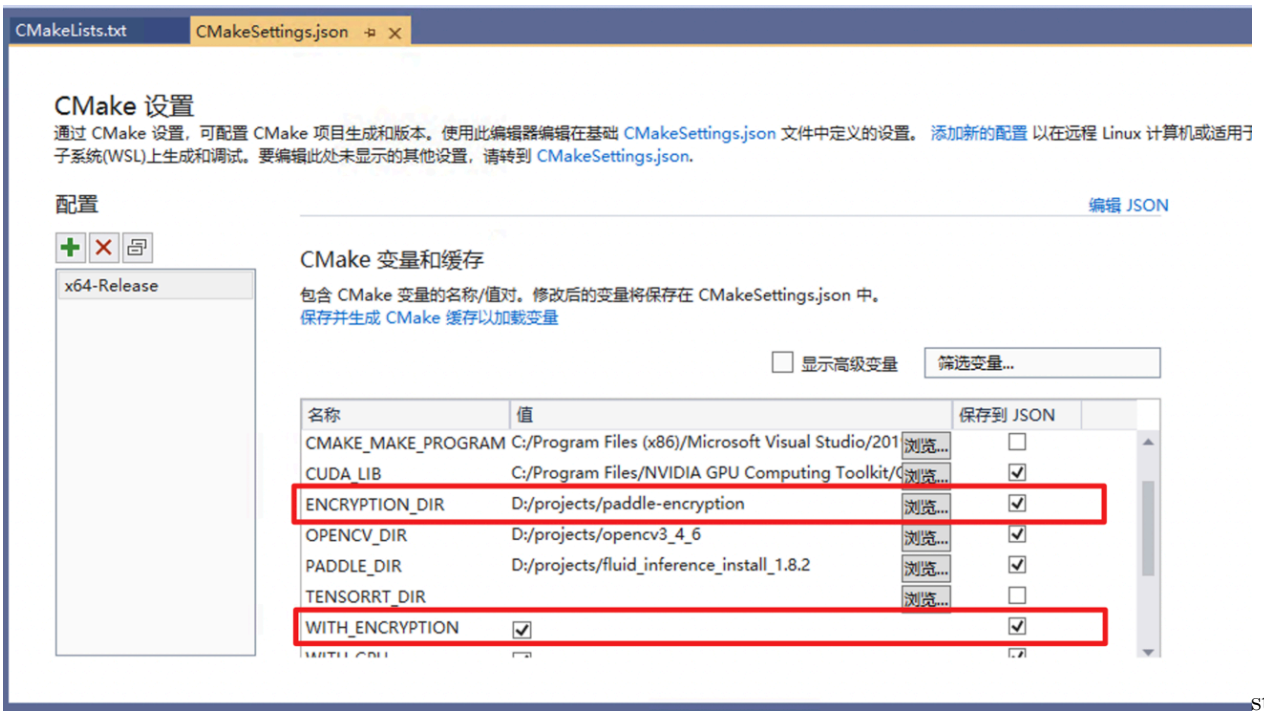


step3

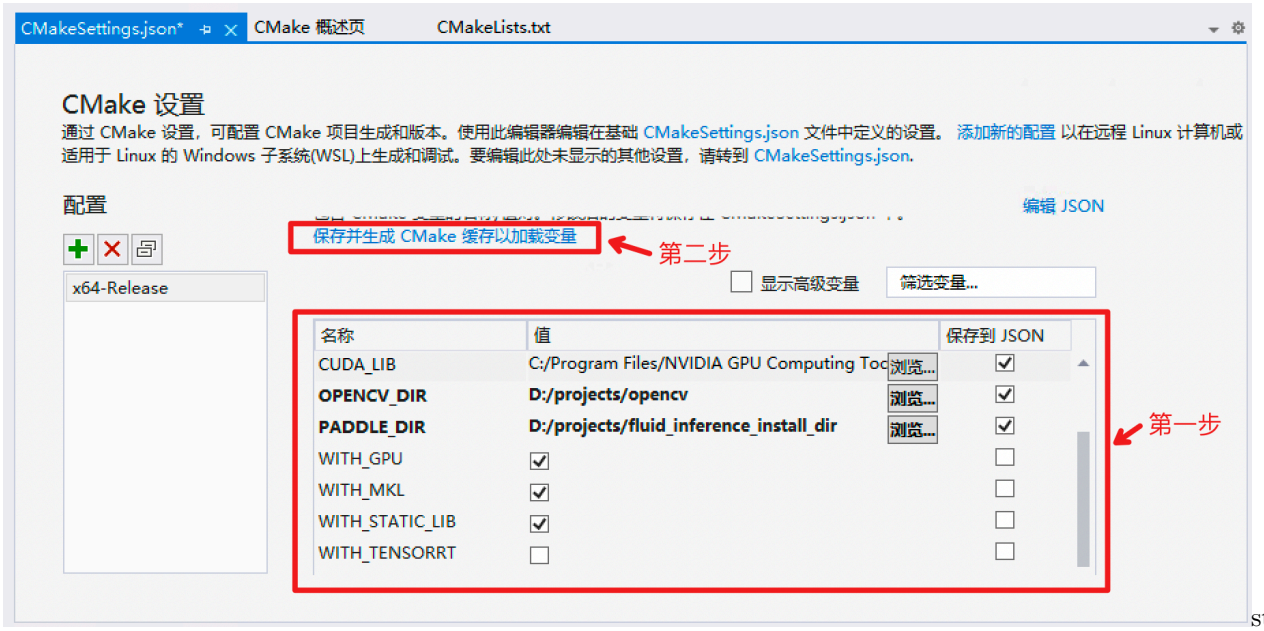
依赖库路径的含义说明如下(带 * 表示仅在使用 **GPU 版本**预测库时指定, 其中 CUDA 库版本尽量与 Paddle 预测库的对齐，例如 Paddle 预测库是**使用 9.0、10.0 版本**编译的，则编译 PaddleX 预测代码时**不使用 9.2、10.1 等版本** CUDA 库):

注意:

1. 如果使用 `CPU` 版预测库，请把 `WITH_GPU` 的值去掉勾
2. 如果使用的是 `openblas` 版本，请把 `WITH_MKL` 的值去掉勾
3. Windows 环境下编译会自动下载 YAML, 如果编译环境无法访问外网, 可手动下载: yaml-cpp.zip yaml-cpp.zip 文件下载后无需解压，在 cmake/yaml.cmake 中将 `URL https://bj.bcebos.com/paddlex/deploy/deps/yaml-cpp.zip` 中的网址，改为下载文件的路径。
4. 如果需要使用模型加密功能，需要手动下载Windows 预测模型加密工具。例如解压到 D:/projects，解压后目录为 D:/projects/paddlex-encryption。编译时需勾选 WITH_EBNCRYPTION 并且在 ENCRTYPTION_DIR 填入 D:/projects/paddlex-encryption。

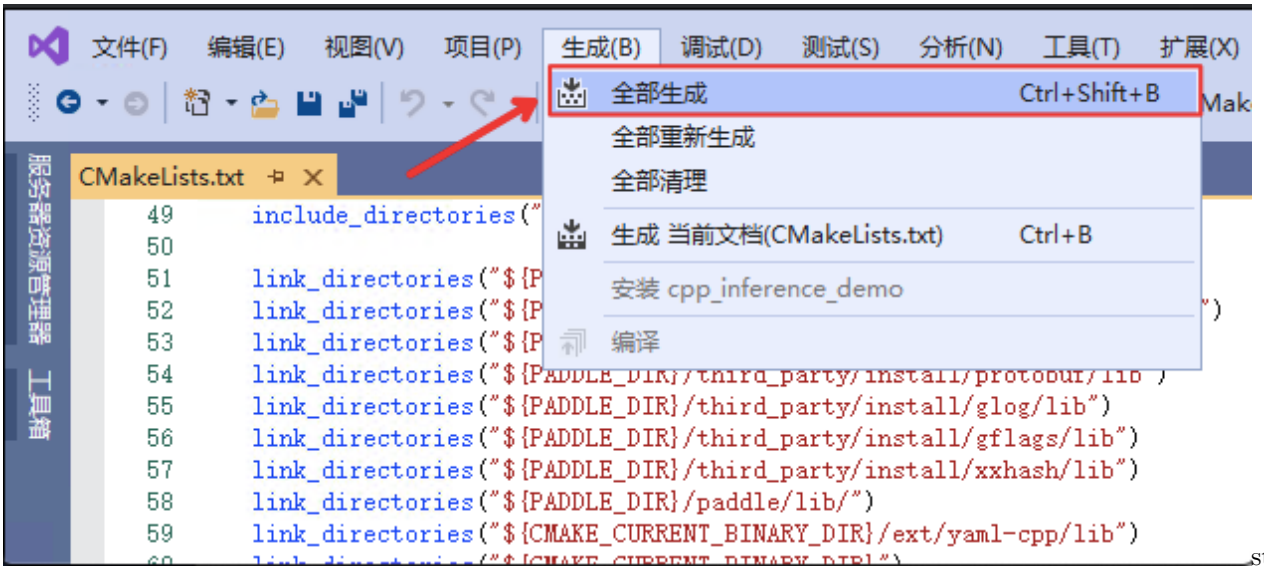


step_encryption



step4

设置完成后，点击上图中保存并生成 CMake 缓存以加载变量。5. 点击生成-> 全部生成



step6

Step5: 预测及可视化

在加载模型前，请检查你的模型目录中文件应该包括 model.yml、__model__ 和 __params__ 三个文件。如若不满足这个条件，请参考模型导出为 Inference 文档将模型导出为部署格式。

上述 Visual Studio 2019 编译产出的可执行文件在 out\build\x64-Release 目录下，打开 cmd，并切换到该目录：

```
D:
cd D:\projects\PaddleX\deploy\cpp\out\build\x64-Release
```

编译成功后，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，其主要命令参数说明如下：

样例

可使用小度熊识别模型中导出的 inference_model 和测试图片进行预测，例如导出到 D:\projects，模型路径为 D:\projects\inference_model。

> 关于预测速度的说明：Paddle 在部署预测时，由于涉及到内存显存初始化等原因，在模型加载后刚开始预测速度会较慢，一般在模型运行 20~50 后（即预测 20~30 张图片）预测速度才会稳定。

样例一：(使用未加密的模型对单张图像做预测)

不使用 GPU 测试图片 D:\images\xiaoduxiong.jpeg

```
.\paddlex_inference\detector.exe --model_dir=D:\projects\inference_model --
↪image=D:\images\xiaoduxiong.jpeg --save_dir=output
```

图片文件可视化预测结果会保存在 save_dir 参数设置的目录下。

样例二：(使用未加密的模型对图像列表做预测)

使用 GPU 预测多个图片 D:\images\image_list.txt，image_list.txt 内容的格式如下：

```
D:\images\xiaoduxiong1.jpeg
D:\images\xiaoduxiong2.jpeg
...
D:\images\xiaoduxiongn.jpeg
```

```
.\paddlex_inference\detector.exe --model_dir=D:\projects\inference_model --image_
↪list=D:\images\image_list.txt --use_gpu=1 --save_dir=output --batch_size=2 --thread_
↪num=2
```

图片文件可视化预测结果会保存在 save_dir 参数设置的目录下。

样例三：(使用加密后的模型对单张图片进行预测)

如果未对模型进行加密，请参考加密 PaddleX 模型对模型进行加密。例如加密后的模型所在目录为 D:\projects\encrypted_inference_model。

```
.\paddlex_inference\detector.exe --model_dir=D:\projects\encrypted_inference_model --
↪image=D:\images\xiaoduxiong.jpeg --save_dir=output --key=kLAl1qOs5uRbFt0/
↪RrIDTZW2+tOf5bzvUIaHGF8lJ1c=
```

--key 传入加密工具输出的密钥，例如 kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=，图片文件可视化预测结果会保存在 save_dir 参数设置的目录下。

Linux 平台部署

说明

本文档在 Linux 平台使用 GCC 4.8.5 和 GCC 4.9.4 测试过，如果需要使用更高 G++ 版本编译使用，则需要重新编译 Paddle 预测库，请参考: 从源码编译 Paddle 预测库。

前置条件

- G++ 4.8.2 ~ 4.9.4
- CUDA 9.0 / CUDA 10.0, CUDNN 7+ （仅在使用 GPU 版本的预测库时需要）
- CMake 3.0+

请确保系统已经安装好上述基本软件，**下面所有示例以工作目录 /root/projects/演示**。

Step1: 下载代码

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

说明：其中 C++ 预测代码在/root/projects/PaddleX/deploy/cpp 目录，该目录不依赖任何 PaddleX 下其他目录。

Step2: 下载 PaddlePaddle C++ 预测库 paddle_inference

PaddlePaddle C++ 预测库针对不同的 CPU，CUDA，以及是否支持 TensorRT，提供了不同的预编译版本，目前 PaddleX 依赖于 Paddle1.8 版本，以下提供了多个不同版本的 Paddle 预测库:

更多和更新的版本，请根据实际情况下载: C++ 预测库下载列表

下载并解压后/root/projects/fluid_inference 目录包含内容为:

```
fluid_inference
  paddle # paddle 核心库和头文件
|
  third_party # 第三方依赖库和头文件
|
  version.txt # 版本和编译信息
```

注意: 预编译版本除 nv-jetson-cuda10-cudnn7.5-trt5 以外其它包都是基于 GCC 4.8.5 编译，使用高版本 GCC 可能存在 ABI 兼容性问题，建议降级或自行编译预测库。

Step3: 编译

编译 cmake 的命令在 scripts/build.sh 中，请根据实际情况修改主要参数，其主要内容说明如下:

```
# 是否使用 GPU(即是否使用 CUDA)
WITH_GPU=OFF
# 使用 MKL or openblas
WITH_MKL=ON
```

(continues on next page)

(continued from previous page)

```
# 是否集成 TensorRT(仅 WITH_GPU=ON 有效)
WITH_TENSORRT=OFF
# TensorRT 的路径，如果需要集成 TensorRT，需修改为您实际安装的 TensorRT 路径
TENSORRT_DIR=/root/projects/TensorRT/
# Paddle 预测库路径，请修改为您实际安装的预测库路径
PADDLE_DIR=/root/projects/fluid_inference
# Paddle 的预测库是否使用静态库来编译
# 使用 TensorRT 时，Paddle 的预测库通常为动态库
WITH_STATIC_LIB=OFF
# CUDA 的 lib 路径
CUDA_LIB=/usr/local/cuda/lib64
# CUDNN 的 lib 路径
CUDNN_LIB=/usr/local/cuda/lib64

# 是否加载加密后的模型
WITH_ENCRYPTION=ON
# 加密工具的路径，如果使用自带预编译版本可不修改
sh $(pwd)/scripts/bootstrap.sh # 下载预编译版本的加密工具
ENCRYPTION_DIR=$(pwd)/paddlex-encryption

# OPENCV 路径，如果使用自带预编译版本可不修改
sh $(pwd)/scripts/bootstrap.sh  # 下载预编译版本的 opencv
OPENCV_DIR=$(pwd)/deps/opencv3gcc4.8/

# 以下无需改动
rm -rf build
mkdir -p build
cd build
cmake .. \
    -DWITH_GPU=${WITH_GPU} \
    -DWITH_MKL=${WITH_MKL} \
    -DWITH_TENSORRT=${WITH_TENSORRT} \
    -DWITH_ENCRYPTION=${WITH_ENCRYPTION} \
    -DTENSORRT_DIR=${TENSORRT_DIR} \
    -DPADDLE_DIR=${PADDLE_DIR} \
    -DWITH_STATIC_LIB=${WITH_STATIC_LIB} \
    -DCUDA_LIB=${CUDA_LIB} \
    -DCUDNN_LIB=${CUDNN_LIB} \
    -DENCRYPTION_DIR=${ENCRYPTION_DIR} \
    -DOPENCV_DIR=${OPENCV_DIR}
```

(continues on next page)

(continued from previous page)

```
make
```

注意： linux 环境下编译会自动下载 OPENCV, PaddleX-Encryption 和 YAML，如果编译环境无法访问外网，可手动下载：

- opencv3gcc4.8.tar.bz2
- paddlex-encryption.zip
- yaml-cpp.zip

opencv3gcc4.8.tar.bz2 文件下载后解压，然后在 script/build.sh 中指定 `OPENCE_DIR` 为解压后的路径。

paddlex-encryption.zip 文件下载后解压，然后在 script/build.sh 中指定 `ENCRYPTION_DIR` 为解压后的路径。

yaml-cpp.zip 文件下载后无需解压，在 cmake/yaml.cmake 中将 `URL https://bj.bcebos.com/paddlex/deploy/deps/yaml-cpp.zip` 中的网址，改为下载文件的路径。

修改脚本设置好主要参数后，执行 `build` 脚本：

```
sh ./scripts/build.sh
```

Step4: 预测及可视化

在加载模型前，请检查你的模型目录中文件应该包括 `model.yml`、`__model__` 和 `__params__` 三个文件。如若不满足这个条件，请参考模型导出为 Inference 文档将模型导出为部署格式。

编译成功后，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，其主要命令参数说明如下：

样例

可使用小度熊识别模型中导出的 `inference_model` 和测试图片进行预测，导出到/root/projects，模型路径为/root/projects/inference_model。

> 关于预测速度的说明：Paddle 在部署预测时，由于涉及到内存显存初始化等原因，在模型加载后刚开始预测速度会较慢，一般在模型运行 20~50 后（即预测 20~30 张图片）预测速度才会稳定。

样例一：

不使用 `GPU` 测试图片 `/root/projects/images/xiaoduxiong.jpeg`

```
./build/demo/detector --model_dir=/root/projects/inference_model --image=/root/projects/
↪images/xiaoduxiong.jpeg --save_dir=output
```

图片文件可视化预测结果会保存在 `save_dir` 参数设置的目录下。

样例二：

使用 GPU 预测多个图片/root/projects/image_list.txt，image_list.txt 内容的格式如下：

```
/root/projects/images/xiaoduxiong1.jpeg
/root/projects/images/xiaoduxiong2.jpeg
...
/root/projects/images/xiaoduxiongn.jpeg
```

```
./build/demo/detector --model_dir=/root/projects/inference_model --image_list=/root/
↪projects/images_list.txt --use_gpu=1 --save_dir=output --batch_size=2 --thread_num=2
```

图片文件可视化预测结果会保存在 save_dir 参数设置的目录下。

5.2.3 模型加密部署

PaddleX 提供一个轻量级的模型加密部署方案，通过 PaddleX 内置的模型加密工具对推理模型进行加密，预测部署 SDK 支持直接加载密文模型并完成推理，提升 AI 模型部署的安全性。

目前加密方案已支持 Windows，Linux 系统

1. 方案简介

1.1 简介

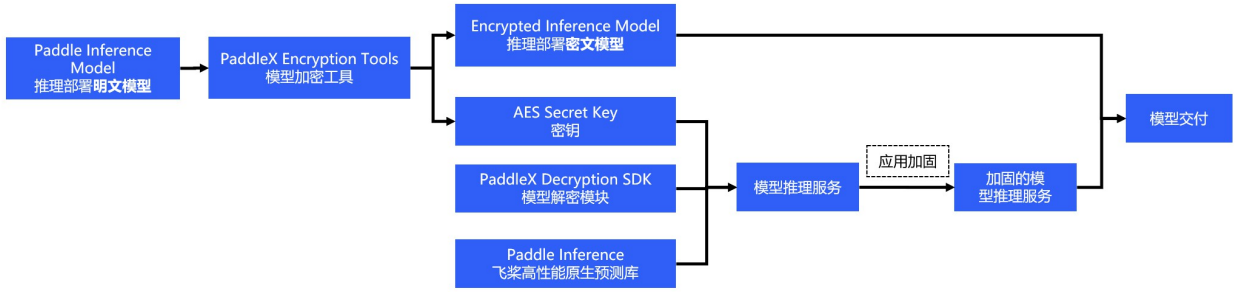
（1）加密算法的选择和支持的库

一般使用 OpenSSL 库来支持数据的加解密，OpenSSL 提供了大量的加解密算法，包括对称加密算法（AES 等）和非对称加密算法（RSA 等）。

两种算法使用的场景不同，非对称加密算法一般应用于数字签名和密钥协商的场景下，而对称加密算法一般应用于纯数据加密场景，性能更优。在对模型的加密过程中使用对称加密算法。

以下对模型加密场景实现的说明中以开发一个 C/C++ 库为基础，采用 AES 对称加密算法，为了加解密前后能够快速判断解密是否成功，使用 AES-GCM 加解密模式，在密钥的安全性上使用长度为 256 位的密钥数据。

（2）实现模型保护的一般步骤：



下面是对提供的 C/C++ 加解密库内部实现的中文描述，参考以下步骤可以实现一套加解密库来适应自己的场景并通过内存数据加载到 Paddle Inference 预测库中

1）考虑到加密的模型文件解密后需要从内存加载数据，使用 conbine 的模式生成模型文件和参数文件。

2）项目集成 OpenSSL，使用静态库的形式。

3）实现 AES 算法接口，借助 OpenSSL 提供的 EVP 接口，在 EVP 接口中指定算法类型，算法使用对称加解密算法中的 AES，加解密模式使用 AES-GCM，密钥长度为 256 位，AES-GCM 的实现可以参考官方提供的例子自己进行封装接口：AES-GCM 实现。

4）利用 OpenSSL 库实现 SHA256 摘要算法，这部分下面有用（可选）。关于 SHA256 的 hash 计算可以参考 OpenSSL 提供的 example：OpenSSL 信息摘要例子。

5）在模型加密环节直接对 model 文件和 params 文件的数据内容进行加密后保存到新的文件，为了新的文件能够被区分和可迭代，除了加密后的数据外还添加了头部信息，比如为了判断该文件类型使用固定的魔数作为文件的开头；为了便于后面需求迭代写入版本号以示区别；为了能够在解密时判断是否采用了相同的密钥将加密时的密钥进行 SHA256 计算后存储；这三部分构成了目前加密后文件的头部信息。加密后的文件包含头部信息 + 密文信息。

6）在模型解密环节根据加密后的文件读取相关的加密数据到内存中，对内存数据使用 AES 算法进行解密，注意解密时需要采用与加密时一致的加密算法和加密的模式，以及密钥的数据和长度，否则会导致解密后数据错误。

7）集成模型预测的 C/C++ 库，在具体使用预测时一般涉及 paddle::AnalysisConfig 和 paddle:Predictor，为了能够从内存数据中直接 load 解密后的模型明文数据（避免模型解密后创建临时文件），这里需要将 AnalysisConfig 的模型加载函数从 SetModel 替换为 SetModelBuffer 来实现从内存中加载模型数据。

需要注意的是，在本方案中，密钥集成在上层预测服务的代码中。故模型的安全强度等同于代码抵御逆向调试的强度。为了保护密钥和模型的安全，开发者还需对自己的应用进行加固保护。常见的应用加固手段有：代码混淆，二进制文件加壳等等，亦或将加密机制更改为 AES 白盒加密技术来保护密钥。这类技术领域内有大量商业和开源产品可供选择，此处不一一赘述。

1.2 加密工具

Linux 版本 PaddleX 模型加密工具，编译脚本会自动下载该版本加密工具，您也可以选择手动下载。

Windows 版本 PaddleX 模型加密工具，该版本加密工具需手动下载，如果您在使用 Visual Studio 2019 编译 C++ 预测代码的过程中已经下载过该工具，此处可不必重复下载。

Linux 加密工具包含内容为：

```
paddlex-encryption
  include # 头文件：paddle_model_decrypt.h（解密）和 paddle_model_encrypt.h（加密）
|
```

(continues on next page)

(continued from previous page)

```
  lib # libpmodel-encrypt.so 和 libpmodel-decrypt.so 动态库
|
  tool # paddlex_encrypt_tool
```

Windows 加密工具包含内容为：

```
paddlex-encryption
  include # 头文件：paddle_model_decrypt.h（解密）和 paddle_model_encrypt.h（加密）
|
  lib # pmodel-encrypt.dll 和 pmodel-decrypt.dll 动态库 pmodel-encrypt.lib 和 pmodel-
↪encrypt.lib 静态库
|
  tool # paddlex_encrypt_tool.exe 模型加密工具
```

1.3 加密 PaddleX 模型

对模型完成加密后，加密工具会产生随机密钥信息（用于 AES 加解密使用），需要在后续加密部署时传入该密钥来用于解密。

> 密钥由 32 字节 key + 16 字节 iv 组成，注意这里产生的 key 是经过 base64 编码后的，这样可以扩充 key 的选取范围

Linux 平台：

```
# 假设模型在/root/projects 下
./paddlex-encryption/tool/paddlex_encrypt_tool -model_dir /root/projects/paddlex_
↪inference_model -save_dir /root/projects/paddlex_encrypted_model
```

Windows 平台：

```
# 假设模型在 D:/projects 下
.\paddlex-encryption\tool\paddlex_encrypt_tool.exe -model_dir D:\projects\paddlex_
↪inference_model -save_dir D:\projects\paddlex_encrypted_model
```

-model_dir 用于指定 inference 模型路径（参考导出 inference 模型将模型导出为 inference 格式模型），可使用导出小度熊识别模型中导出的 inference_model。加密完成后，加密过的模型会保存至指定的-save_dir 下，包含 __model__.encrypted、__params__.encrypted 和 model.yml 三个文件，同时生成密钥信息，命令输出如下图所示，密钥为 kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=

```
Output: Encryption key:
        kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=
Success, Encrypt __model__, __params__ to paddlex_encrypted_model(dir) success!
```

2. PaddleX C++ 加密部署

2.1 Linux 平台使用

参考Linux 平台编译指南编译 C++ 部署代码。编译成功后，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，其主要命令参数说明如下：

样例

可使用导出小度熊识别模型中的测试图片进行预测。

样例一：

不使用 GPU 测试图片 `/root/projects/images/xiaoduxiong.jpeg`

```
./build/demo/detector --model_dir=/root/projects/paddlex_encrypted_model --image=/root/
↪projects/xiaoduxiong.jpeg --save_dir=output --key=kLAl1qOs5uRbFt0/
↪RrIDTZW2+tOf5bzvUIaHGF8lJ1c=
```

`--key` 传入加密工具输出的密钥，例如 `kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=`，图片文件可视化预测结果会保存在 `save_dir` 参数设置的目录下。

样例二:

使用 GPU 预测多个图片`/root/projects/image_list.txt`，image_list.txt 内容的格式如下：

```
/root/projects/images/xiaoduxiong1.jpeg
/root/projects/xiaoduxiong2.jpeg
...
/root/projects/xiaoduxiongn.jpeg
```

```
./build/demo/detector --model_dir=/root/projects/models/paddlex_encrypted_model --image_
↪list=/root/projects/images_list.txt --use_gpu=1 --save_dir=output --
↪key=kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=
```

`--key` 传入加密工具输出的密钥，例如 `kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=`，图片文件可视化预测结果会保存在 `save_dir` 参数设置的目录下。

2.2 Windows 平台使用

参考Windows 平台编译指南。需自行下载 Windows 版 PaddleX 加密工具压缩包，解压，在编译指南的编译流程基础上，在 CMake 设置中勾选 WITH_ENCRYPTION，ENCRYPTION_DIR 填写为加密工具包解压后的目录，再进行编译。参数与 Linux 版本预测部署一致。预测 demo 的入口程序为 paddlex_inference\detector.exe，paddlex_inference\classifier.exe，paddlex_inference\segmenter.exe。

样例

可使用导出小度熊识别模型中的测试图片进行预测。

样例一：

不使用 GPU 测试单张图片，例如图片为 D:\images\xiaoduxiong.jpeg，加密后的模型目录为 D:\projects\paddlex_encrypted_model

```
.\paddlex_inference\detector.exe --model_dir=D:\projects\paddlex_encrypted_model --
↪image=D:\images\xiaoduxiong.jpeg --save_dir=output --key=kLAl1qOs5uRbFt0/
↪RrIDTZW2+tOf5bzvUIaHGF8lJ1c=
```

--key 传入加密工具输出的密钥，例如 kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=，图片文件可视化预测结果会保存在 save_dir 参数设置的目录下。

样例二：

使用 GPU 预测图片列表，例如图片列表为 D:\projects\image_list.txt，image_list.txt 的内容如下：

```
D:\projects\images\xiaoduxiong1.jpeg
D:\projects\images\xiaoduxiong2.jpeg
...
D:\projects\images\xiaoduxiongn.jpeg
```

加密后的模型目录例如为 D:\projects\paddlex_encrypted_model

```
.\paddlex_inference\detector.exe --model_dir=D:\projects\paddlex_encrypted_model --image_
↪list=D:\projects\images_list.txt --use_gpu=1 --save_dir=output --key=kLAl1qOs5uRbFt0/
↪RrIDTZW2+tOf5bzvUIaHGF8lJ1c=
```

--key 传入加密工具输出的密钥，例如 kLAl1qOs5uRbFt0/RrIDTZW2+tOf5bzvUIaHGF8lJ1c=，图片文件可视化预测结果会保存在 save_dir 参数设置的目录下。

5.3 Nvidia-Jetson 开发板

5.3.1 说明

本文档在 Linux 平台使用 GCC 4.8.5 和 GCC 4.9.4 测试过，如果需要使用更高 G++ 版本编译使用，则需要重新编译 Paddle 预测库，请参考: NVIDIA Jetson 嵌入式硬件预测库源码编译。

5.3.2 前置条件

- G++ 4.8.2 ~ 4.9.4
- CUDA 9.0 / CUDA 10.0, CUDNN 7+ （仅在使用 GPU 版本的预测库时需要）
- CMake 3.0+

请确保系统已经安装好上述基本软件，**下面所有示例以工作目录 /root/projects/演示**。

Step1: 下载代码

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

说明：其中 C++ 预测代码在/root/projects/PaddleX/deploy/cpp 目录，该目录不依赖任何 PaddleX 下其他目录。

Step2: 下载 PaddlePaddle C++ 预测库 paddle_inference

目前 PaddlePaddle 为 Nvidia-Jetson 提供了一个基于 1.6.2 版本的 C++ 预测库。

下载并解压后/root/projects/fluid_inference 目录包含内容为:

```
fluid_inference
  paddle # paddle 核心库和头文件
|
  third_party # 第三方依赖库和头文件
|
  version.txt # 版本和编译信息
```

Step3: 编译

编译 cmake 的命令在 scripts/jetson_build.sh 中，请根据实际情况修改主要参数，其主要内容说明如下:

```
# 是否使用 GPU(即是否使用 CUDA)
WITH_GPU=OFF
# 使用 MKL or openblas
WITH_MKL=OFF
# 是否集成 TensorRT(仅 WITH_GPU=ON 有效)
WITH_TENSORRT=OFF
# TensorRT 的路径，如果需要集成 TensorRT，需修改为您实际安装的 TensorRT 路径
TENSORRT_DIR=/root/projects/TensorRT/
# Paddle 预测库路径，请修改为您实际安装的预测库路径
PADDLE_DIR=/root/projects/fluid_inference
# Paddle 的预测库是否使用静态库来编译
# 使用 TensorRT 时，Paddle 的预测库通常为动态库
WITH_STATIC_LIB=OFF
# CUDA 的 lib 路径
CUDA_LIB=/usr/local/cuda/lib64
# CUDNN 的 lib 路径
CUDNN_LIB=/usr/local/cuda/lib64

# 是否加载加密后的模型
WITH_ENCRYPTION=OFF

# OPENCV 路径，如果使用自带预编译版本可不修改
sh $(pwd)/scripts/jetson_bootstrap.sh  # 下载预编译版本的 opencv
OPENCV_DIR=$(pwd)/deps/opencv3/

# 以下无需改动
rm -rf build
mkdir -p build
cd build
cmake .. \
    -DWITH_GPU=${WITH_GPU} \
    -DWITH_MKL=${WITH_MKL} \
    -DWITH_TENSORRT=${WITH_TENSORRT} \
    -DWITH_ENCRYPTION=${WITH_ENCRYPTION} \
    -DTENSORRT_DIR=${TENSORRT_DIR} \
    -DPADDLE_DIR=${PADDLE_DIR} \
    -DWITH_STATIC_LIB=${WITH_STATIC_LIB} \
    -DCUDA_LIB=${CUDA_LIB} \
    -DCUDNN_LIB=${CUDNN_LIB} \
    -DENCRYPTION_DIR=${ENCRYPTION_DIR} \
```

(continues on next page)

(continued from previous page)

```
    -DOPENCV_DIR=${OPENCV_DIR}
make
```

注意： linux 环境下编译会自动下载 OPENCV 和 YAML，如果编译环境无法访问外网，可手动下载：

- opencv3_aarch.tgz
- yaml-cpp.zip

opencv3_aarch.tgz 文件下载后解压，然后在 script/build.sh 中指定 `OPENCE_DIR` 为解压后的路径。

yaml-cpp.zip 文件下载后无需解压，在 cmake/yaml.cmake 中将 `URL https://bj.bcebos.com/paddlex/deploy/deps/yaml-cpp.zip` 中的网址，改为下载文件的路径。

修改脚本设置好主要参数后，执行 `build` 脚本：

```
sh ./scripts/jetson_build.sh
```

Step4: 预测及可视化

在加载模型前，请检查你的模型目录中文件应该包括 `model.yml`、`__model__` 和 `__params__` 三个文件。如若不满足这个条件，请参考模型导出为 Inference 文档将模型导出为部署格式。

编译成功后，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，其主要命令参数说明如下：

5.3.3 样例

可使用小度熊识别模型中导出的 `inference_model` 和测试图片进行预测，导出到/root/projects，模型路径为/root/projects/inference_model。

样例一：

不使用 `GPU` 测试图片 `/root/projects/images/xiaoduxiong.jpeg`

```
./build/demo/detector --model_dir=/root/projects/inference_model --image=/root/projects/
↪images/xiaoduxiong.jpeg --save_dir=output
```

图片文件可视化预测结果会保存在 `save_dir` 参数设置的目录下。

样例二:

使用 `GPU` 预测多个图片`/root/projects/image_list.txt`，image_list.txt 内容的格式如下：

```
/root/projects/images/xiaoduxiong1.jpeg
/root/projects/images/xiaoduxiong2.jpeg
```

(continues on next page)

(continued from previous page)

```
...
/root/projects/images/xiaoduxiongn.jpeg
```

```
./build/demo/detector --model_dir=/root/projects/inference_model --image_list=/root/
↪projects/images_list.txt --use_gpu=1 --save_dir=output --batch_size=2 --thread_num=2
```

图片文件可视化预测结果会保存在 `save_dir` 参数设置的目录下。

5.4 PaddleLite 移动端部署

5.4.1 模型压缩

模型量化

为了更好地满足端侧部署场景下，低内存带宽、低功耗、低计算资源占用以及低模型存储等需求，PaddleX 通过集成 PaddleSlim 实现模型量化，可提升 PaddleLite 端侧部署性能。

原理介绍

定点量化使用更少的比特数（如 8-bit、3-bit、2-bit 等）表示神经网络的权重和激活值，从而加速模型推理速度。PaddleX 提供了训练后量化技术，其原理可参见训练后量化原理，该量化使用 KL 散度确定量化比例因子，将 FP32 模型转成 INT8 模型，且不需要重新训练，可以快速得到量化模型。

使用 PaddleX 量化模型

PaddleX 提供了 `export_quant_model` 接口，让用户以接口的形式完成模型以 post_quantization 方式量化并导出。点击查看量化接口使用文档。

量化性能对比

模型量化后的性能对比指标请查阅PaddleSlim 模型库

模型裁剪

为了更好地满足端侧部署场景下，低内存带宽、低功耗、低计算资源占用以及低模型存储等需求，PaddleX 通过集成 PaddleSlim 实现模型裁剪，可提升 PaddleLite 端侧部署性能。

原理介绍

模型裁剪通过裁剪卷积层中 Kernel 输出通道的大小及其关联层参数大小，来减小模型大小和降低模型计算复杂度，可以加快模型部署后的预测速度，其关联裁剪的原理可参见PaddleSlim 相关文档。**一般而言，在同等模型精度前提下，数据复杂度越低，模型可以被裁剪的比例就越高**。

裁剪方法

PaddleX 提供了两种方式:

1. 用户自行计算裁剪配置（推荐），整体流程包含三个步骤，

第一步：使用数据集训练原始模型**第二步**：利用第一步训练好的模型，在验证数据集上计算模型中各个参数的敏感度，并将敏感度信息存储至本地文件**第三步**：使用数据集训练裁剪模型（与第一步差异在于需要在 `train` 接口中，将第二步计算得到的敏感信息文件传给接口的 `sensitivities_file` 参数）

在如上三个步骤中，**相当于模型共需要训练两遍**，分别对应第一步和第三步，但其中第三步训练的是裁剪后的模型，因此训练速度较第一步会更快。第二步会遍历模型中的部分裁剪参数，分别计算各个参数裁剪后对于模型在验证集上效果的影响，**因此会反复在验证集上评估多次**。

2. 使用 PaddleX 内置的裁剪方案

PaddleX 内置的模型裁剪方案是**基于标准数据集**上计算得到的参数敏感度信息，由于不同数据集特征分布会有较大差异，所以该方案相较于第 1 种方案训练得到的模型**精度一般而言会更低（且用户自定义数据集与标准数据集特征分布差异越大，导致训练的模型精度会越低）**，仅在用户想节省时间的前提下可以参考使用，使用方式只需一步，

一步：使用数据集训练裁剪模型，在训练调用 `train` 接口时，将接口中的 `sensitivities_file` 参数设置为’DEFAULT’字符串

注：各模型内置的裁剪方案分别依据的数据集为：图像分类——ImageNet 数据集、目标检测——PascalVOC 数据集、语义分割——CityScape 数据集

裁剪实验

基于上述两种方案，我们在 PaddleX 上使用样例数据进行了实验，在 Tesla P40 上实验指标如下所示:

图像分类

实验背景：使用 MobileNetV2 模型，数据集为蔬菜分类示例数据，裁剪训练代码见tutorials/compress/classification

目标检测

实验背景：使用 YOLOv3-MobileNetV1 模型，数据集为昆虫检测示例数据，裁剪训练代码见tutorials/compress/detection

语义分割

实验背景：使用 UNet 模型，数据集为视盘分割示例数据，裁剪训练代码见tutorials/compress/segmentation

5.4.2 Android 平台

PaddleX 的安卓端部署由 PaddleLite 实现，部署的流程如下，首先将训练好的模型导出为 inference model，然后对模型进行优化，最后使用 PaddleLite 的预测库进行部署，PaddleLite 的详细介绍和使用可参考：PaddleLite 文档

> PaddleX –> Inference Model –> PaddleLite Opt –> PaddleLite Inference

文章简介：

- 1. 介绍如何将 PaddleX 导出为 inference model
- 2. 使用 PaddleLite 的 OPT 模块对模型进行优化
- 3. 介绍基于 PaddleX Android SDK 的安卓 demo，以及如何快速部署训练好的模型
- 4. 介绍 PaddleX Android SDK 和二次开发

1. 将 PaddleX 模型导出为 inference 模型

参考导出 inference 模型将模型导出为 inference 格式模型。

2. 将 inference 模型优化为 PaddleLite 模型

目前提供了两种方法将 Paddle 模型优化为 PaddleLite 模型:

- 1.python 脚本优化模型，简单上手，目前支持最新的 PaddleLite 2.6.1 版本
- 2.bin 文件优化模型 (linux)，支持 develop 版本 (Commit Id:11cbd50e)，适用于部署 `DeepLab` 模型和 `Unet` 模型的用户。

2.1 使用 python 脚本优化模型

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

其中 `export_lite.py` 脚本请至 github 下载：https://github.com/PaddlePaddle/PaddleX/blob/develop/deploy/lite/ex

2.3 使用 bin 文件优化模型 (linux)

首先下载并解压: 模型优化工具 opt

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

详细的使用方法和参数含义请参考: 使用 opt 转化模型

3. 移动端（Android）Demo

PaddleX 提供了一个基于 Mobilenetv2 模型和 PaddleX Android SDK 的安卓 demo，可供用户体验，该 demo 位于`/PaddleX/deploy/lite/android/demo`，可直接导入 Android Studio 后运行，并支持用户替换其他 PaddleX 导出的检测或分割模型进行预测。

3.1 要求

- Android Studio 3.4
- Android 手机或开发板

3.2 分类 Demo

3.2.1 导入工程并运行

- 打开 Android Studio，在" Welcome to Android Studio" 窗口点击" Open an existing Android Studio project"，在弹出的路径选择窗口中进入`/PaddleX/deploy/lite/android/demo` 目录，然后点击右下角的" Open" 按钮，导入工程；
- 通过 USB 连接 Android 手机或开发板；
- 载入工程后，点击菜单栏的 Run->Run ‘App’ 按钮，在弹出的" Select Deployment Target" 窗口选择已经连接的 Android 设备，然后点击" OK" 按钮；
- 运行成功后，Android 设备将加载一个名为 PaddleX Demo 的 App，默认会加载一个测试图片，同时还支持拍照和从图库选择照片进行预测；

注意：在工程构建的过程中会远程下载 Mobilenetv2 模型、yml 配置文件、测试的图片，以及 PaddleX Android SDK。

3.3 部署自定义模型

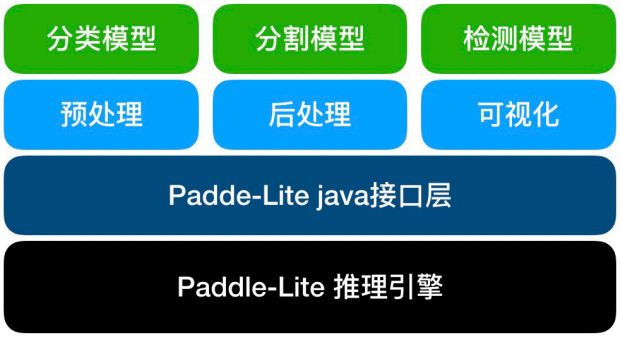
该 demo 还支持用户自定义模型来进行预测，可帮助用户快速验证自己训练好的模型，首先我们已经根据 step1~step2 描述，准备好了 Lite 模型 (.nb 文件) 和 yml 配置文件 (注意：导出 Lite 模型时需指定–place=arm)，然后在 Android Studio 的 project 视图中：

- 将.nb 文件拷贝到/src/main/assets/model/目录下, 根据.nb 文件的名字，修改文件/src/main/res/values/strings.xml 中的 MODEL_PATH_DEFAULT;
- 将.yml 文件拷贝到/src/main/assets/config/目录下，根据.yml 文件的名字，修改文件/src/main/res/values/strings.xml 中的 YAML_PATH_DEFAULT;
- 可根据需要替换测试图片，将图片拷贝到/src/main/assets/images/目录下，根据图片文件的名字，修改文件/src/main/res/values/strings.xml 中的 IMAGE_PATH_DEFAULT;
- 点击菜单栏的 Run->Run ‘App’ 按钮，在弹出的” Select Deployment Target” 窗口选择已经连接的 Android 设备，然后点击” OK” 按钮;

4. PaddleX Android SDK 和二次开发

PaddleX Android SDK 是 PaddleX 基于 Paddle-Lite 开发的安卓端 AI 推理工具，以 PaddleX 导出的 Yaml 配置文件为接口，针对不同的模型实现图片的预处理，后处理，并进行可视化，开发者可集成到业务中。该 SDK 自底向上主要包括：Paddle-Lite 推理引擎层，Paddle-Lite 接口层以及 PaddleX 业务层。

- Paddle-Lite 推理引擎层，是在 Android 上编译好的二进制包，只涉及到 Kernel 的执行，且可以单独部署，以支持极致的轻量级部署。
- Paddle-Lite 接口层，以 Java 接口封装了底层 c++ 推理库。
- PaddleX 业务层，封装了 PaddleX 导出模型的预处理，推理和后处理，以及可视化，支持 PaddleX 导出的检测、分割、分类模型。



架

构

4.1 SDK 安装

首先下载并解压PaddleX Android SDK，得到 paddlex.aar 文件，将拷贝到 android 工程目录 app/libs/下面，然后为 app 的 build.gradle 添加依赖：

```
dependencies {
    implementation fileTree(include: ['*.jar','*aar'], dir: 'libs')
}
```

4.2 SDK 使用用例

```
import com.baidu.paddlex.Predictor;
import com.baidu.paddlex.config.ConfigParser;
import com.baidu.paddlex.postprocess.DetResult;
import com.baidu.paddlex.postprocess.SegResult;
import com.baidu.paddlex.postprocess.ClsResult;
import com.baidu.paddlex.visual.Visualize;

// Predictor
Predictor predictor = new Predictor();
// model config
ConfigParser configParser = new ConfigParser();
// Visualize
Visualize visualize = new Visualize();
// image to predict
Mat predictMat;

// initialize
configParser.init(context, model_path, yaml_path, cpu_thread_num, cpu_power_mode);
visualize.init(configParser.getNumClasses());
predictor.init(context, configParser)

// run model
if (predictImage != null && predictor.isLoaded()) {
    predictor.setInputMat(predictMat);
    runModel();
}
```

(continues on next page)

(continued from previous page)

```
// get result & visualize
if (configParser.getModelType().equalsIgnoreCase("segmenter")) {
    SegResult segResult = predictor.getSegResult();
    Mat visualizeMat  = visualize.draw(segResult, predictMat, predictor.getImageBlob());
} else if (configParser.getModelType().equalsIgnoreCase("detector")) {
    DetResult detResult = predictor.getDetResult();
    Mat visualizeMat = visualize.draw(detResult, predictMat);
} else if (configParser.getModelType().equalsIgnoreCase("classifier")) {
    ClsResult clsResult = predictor.getClsResult();
}
```

4.3 Result 成员变量

注意：Result 所有的成员变量以 java bean 的方式获取。

```
com.baidu.paddlex.postprocess.ClsResult
```

Fields

- **type** (String|static)：值为” cls”。
- **categoryId** (int)：类别 ID。
- **category** (String)：类别名称。
- **score** (float)：预测置信度。

```
com.baidu.paddlex.postprocess.DetResult
```

Nested classes

- **DetResult.Box** 模型预测的 box 结果。

Fields

- **type** (String|static)：值为” det”。
- **boxes** (List<DetResult.Box>)：模型预测的 box 结果。

```
com.baidu.paddlex.postprocess.DetResult.Box
```

Fields

- **categoryId** (int): 类别 ID。
- **category** (String): 类别名称。
- **score** (float): 预测置信度。
- **coordinate** (float[4]): 预测框值:{xmin, ymin, xmax, ymax}。

```
com.baidu.paddlex.postprocess.SegResult
```

Nested classes

- **SegResult.Mask**: 模型预测的 mask 结果。

Fields

- **type** (String|static): 值为” Seg”。
- **mask** (SegResult.Mask): 模型预测的 mask 结果。

```
com.baidu.paddlex.postprocess.SegResult.Mask
```

Fields

- **scoreData** (float[]): 模型预测在各个类别的置信度，长度为: 1 * numClass * H * W
- **scoreShape** (long[4]): scoreData 的 shape 信息，[1,numClass,H,W]
- **labelData** (long[]): 模型预测置信度最高的 label，长度为: 1 * H * W * 1
- **labelShape** (long[4]): labelData 的 shape 信息，[1,H,W,1]

4.4 SDK 二次开发

- 打开 Android Studio 新建项目 (或加载已有项目)。点击菜单 File->New->Import Module，导入工程/PaddleX/deploy/lite/android/sdk, Project 视图会新增名为 sdk 的 module
- 在 app 的 build.grade 里面添加依赖:

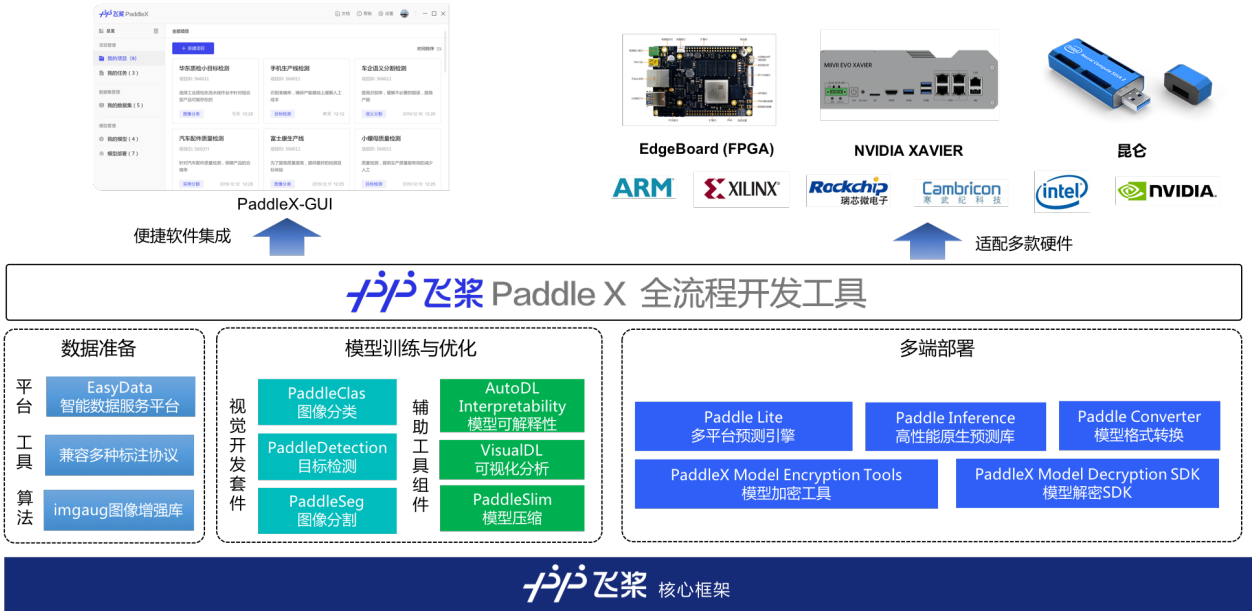
```
dependencies {
    implementation project(':sdk')
}
```

- 源代码位于 sdk/main/java/下，修改源码进行二次开发后，点击菜单栏的 Build->Run ‘sdk’按钮可编译生成 aar，文件位于 sdk/build/outputs/aar/路径下。

CHAPTER 6

产业案例集

PaddleX 精选飞桨视觉开发套件在产业实践中的成熟模型结构，提供统一易用的全流程 API 和模型部署 SDK，打通模型在各种硬件设备上的部署流程，开放从模型训练到多端安全部署的全流程案例实践教程。



6.1 PaddleX 模型介绍

PaddleX 针对图像分类、目标检测、实例分割和语义分割 4 种视觉任务提供了丰富的模型算法，用户根据在实际场景中的需求选择合适的模型。

6.1.1 图像分类

图像分类任务指的是输入一张图片，模型预测图片的类别，如识别为风景、动物、车等。

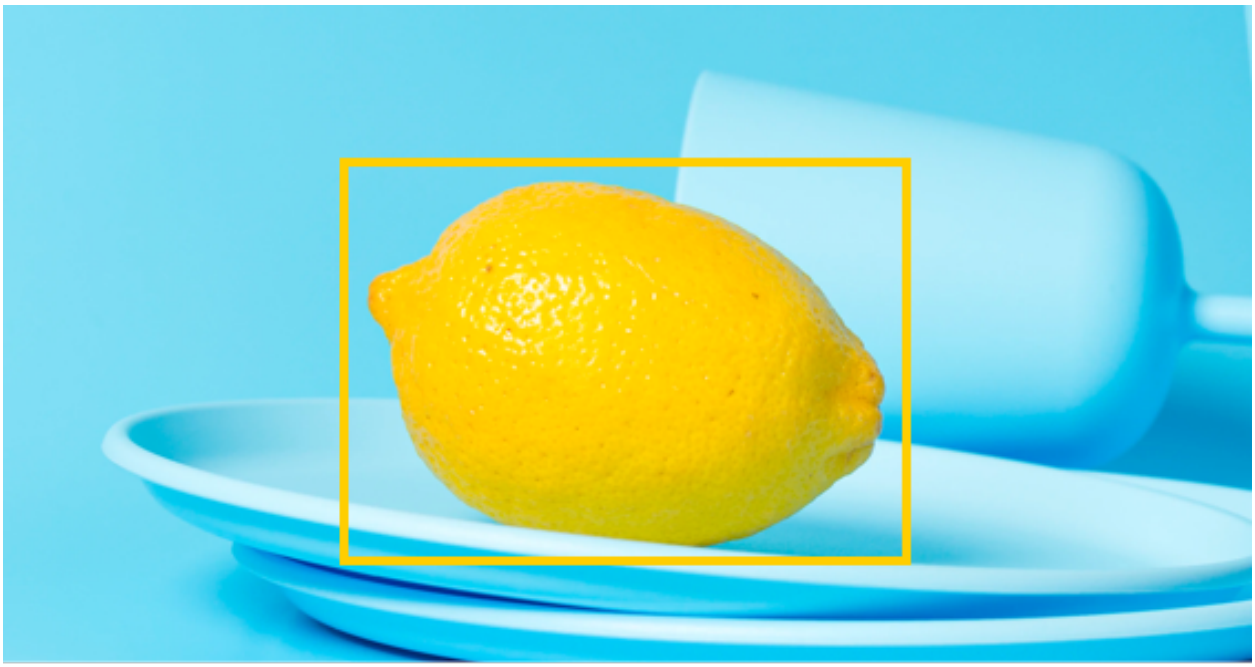
对于图像分类任务，针对不同的应用场景，PaddleX 提供了百度改进的模型，见下表所示:

> 表中 GPU 预测速度是使用 PaddlePaddle Python 预测接口测试得到（测试 GPU 型号为 Nvidia Tesla P40）。表中 CPU 预测速度 (测试 CPU 型号为)。表中骁龙 855 预测速度是使用处理器为骁龙 855 的手机测试得到。测速时模型输入大小为 224 x 224，Top1 准确率为 ImageNet-1000 数据集上评估所得。

包括上述模型，PaddleX 支持近 20 种图像分类模型，其余模型可参考*PaddleX 模型库*

6.1.2 目标检测

目标检测任务指的是输入图像，模型识别出图像中物体的位置（用矩形框框出来，并给出框的位置），和物体的类别，如在手机等零件质检中，用于检测外观上的瑕疵等。



对于目标检测，针对不同的应用场景，PaddleX 提供了主流的 YOLOv3 模型和 Faster-RCNN 模型，见下表所示

表中 GPU 预测速度是使用 PaddlePaddle Python 预测接口测试得到（测试 GPU 型号为 Nvidia Tesla P40）。表中 CPU 预测速度 (测试 CPU 型号为)。表中骁龙 855 预测速度是使用处理器为骁龙 855 的手机测试得到。测速时 YOLOv3 的输入大小为 608 x 608，FasterRCNN 的输入大小为 800 x 1333，Box mmAP 为 COCO2017 数据集上评估所得。

除上述模型外，YOLOv3 和 Faster RCNN 还支持其他 backbone，详情可参考*PaddleX 模型库*

实例分割

在目标检测中，模型识别出图像中物体的位置和物体的类别。而实例分割则是在目标检测的基础上，做了像素级的分类，将框内的属于目标物体的像素识别出来。

PaddleX 目前提供了实例分割 MaskRCNN 模型，支持 5 种不同的 backbone 网络，详情可参考*PaddleX 模型库*

表中 GPU 预测速度是使用 PaddlePaddle Python 预测接口测试得到（测试 GPU 型号为 Nvidia Tesla P40）。表中 CPU 预测速度 (测试 CPU 型号为)。表中骁龙 855 预测速度是使用处理器为骁龙 855 的手机测试得到。测速时 MaskRCNN 的输入大小为 800 x 1333，Box mmAP 和 Seg mmAP 为 COCO2017 数据集上评估所得。

6.1.3 语义分割

语义分割用于对图像做像素级的分类，应用在人像分类、遥感图像识别等场景。

对于语义分割，PaddleX 也针对不同的应用场景，提供了不同的模型选择，如下表所示

> 表中 GPU 预测速度是使用 PaddlePaddle Python 预测接口测试得到（测试 GPU 型号为 Nvidia Tesla P40）。表中 CPU 预测速度 (测试 CPU 型号为)。表中骁龙 855 预测速度是使用处理器为骁龙 855 的手机测试得到。测速时模型的输入大小为 1024 x 2048，mIOU 为 Cityscapes 数据集上评估所得。

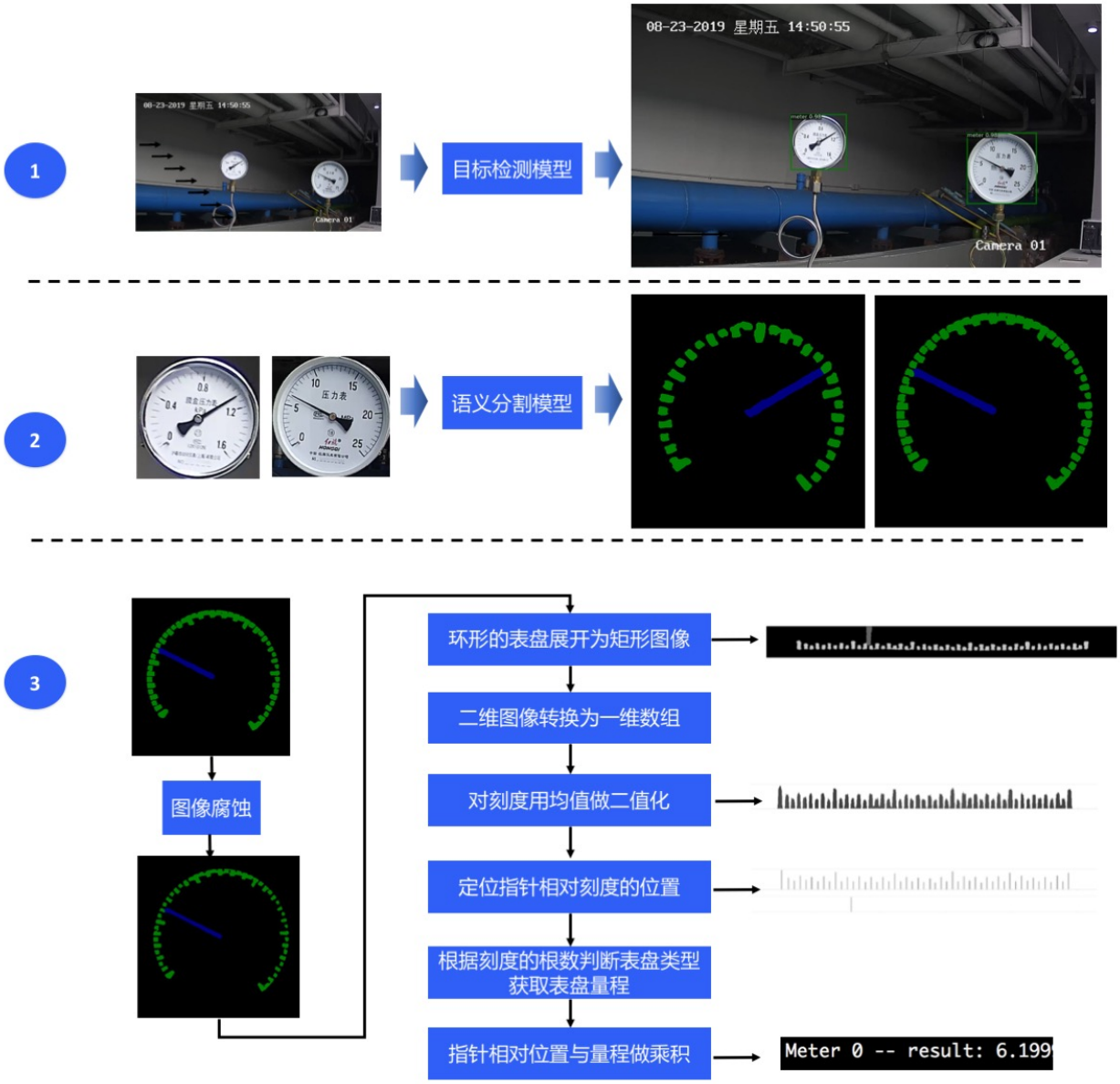
6.2 工业表计读数

本案例基于 PaddleX 实现对传统机械式指针表计的检测与自动读数功能，开放表计数据和预训练模型，并提供在 windows 系统的服务器端以及 linux 系统的 jetson 嵌入式设备上的部署指南。

6.2.1 读数流程

表计读数共分为三个步骤完成：

- 第一步，使用目标检测模型检测出图像中的表计
- 第二步，使用语义分割模型将各表计的指针和刻度分割出来
- 第三步，根据指针的相对位置和预知的量程计算出各表计的读数



MeterReader_A

- **表计检测**：由于本案例中没有面积较小的表计，所以目标检测模型选择性能更优的 **YOLOv3**。考虑到本案例主要在有 GPU 的设备上部署，所以骨干网路选择精度更高的 **DarkNet53**。
- **刻度和指针分割**：考虑到刻度和指针均为细小区域，语义分割模型选择效果更好的 **DeepLapv3**。
- **读数后处理**：首先，对语义分割的预测类别图进行图像腐蚀操作，以达到刻度细分的目的。然后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。接着计算刻度数组的均值，用均值对刻度数组进行二值化操作。最后定位出指针相对刻度的位置，根据刻度的根数判断表盘的类型以此获取表盘的量程，将指针相对位置与量程做乘积得到表盘的读数。

6.2.2 表计数据和预训练模型

本案例开放了表计测试图片，用于体验表计读数的预测推理全流程。还开放了表计检测数据集、指针和刻度分割数据集，用户可以使用这些数据集重新训练模型。

本案例开放了预先训练好的检测模型和语义分割模型，可以使用这些模型快速体验表计读数全流程，也可以直接将这些模型部署在服务器端或 jetson 嵌入式设备上进行推理预测。

6.2.3 快速体验表盘读数

可以使用本案例提供的预训练模型快速体验表计读数的自动预测全流程。如果不需要预训练模型，可以跳转至小节*模型训练*重新训练模型。

前置依赖

- Paddle paddle >= 1.8.0
- Python >= 3.5
- PaddleX >= 1.0.0

安装的相关问题参考*PaddleX 安装*

测试表计读数

1. 下载 PaddleX 源码:

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

1. 预测执行文件位于 `PaddleX/examples/meter_reader/`，进入该目录:

```
cd PaddleX/examples/meter_reader/
```

预测执行文件为 `reader_infer.py`，其主要参数说明如下:

1. 预测

若要使用 GPU，则指定 GPU 卡号（以 0 号卡为例）:

```
export CUDA_VISIBLE_DEVICES=0
```

若不使用 GPU，则将 CUDA_VISIBLE_DEVICES 指定为空:

```
export CUDA_VISIBLE_DEVICES=
```

- 预测单张图片

```
python3 reader_infer.py --detector_dir /path/to/det_inference_model --segmenter_dir /
↪path/to/seg_inference_model --image /path/to/meter_test/20190822_168.jpg --save_dir ./
↪output --use_erode
```

- 预测多张图片

```
python3 reader_infer.py --detector_dir /path/to/det_inference_model --segmenter_dir /
↪path/to/seg_inference_model --image_dir /path/to/meter_test --save_dir ./output --use_
↪erode
```

- 开启摄像头预测

```
python3 reader_infer.py --detector_dir /path/to/det_inference_model --segmenter_dir /
↪path/to/seg_inference_model --save_dir ./output --use_erode --use_camera
```

6.2.4 推理部署

Windows 系统的服务器端安全部署

c++ 部署

1. 下载 PaddleX 源码:

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

1. 将 `PaddleX\examples\meter_reader\deploy\cpp` 下的 `meter_reader` 文件夹和 `CMakeList.txt` 拷贝至 `PaddleX\deploy\cpp` 目录下，拷贝之前可以将 `PaddleX\deploy\cpp` 下原本的 `CMakeList.txt` 做好备份。
2. 按照 *Windows 平台部署*中的 Step2 至 Step4 完成 C++ 预测代码的编译。
3. 编译成功后，可执行文件在 `out\build\x64-Release` 目录下，打开 `cmd`，并切换到该目录:

```
cd PaddleX\deploy\cpp\out\build\x64-Release
```

预测程序为 paddle_inference\meter_reader.exe，其主要命令参数说明如下:

1. 推理预测:

用于部署推理的模型应为 inference 格式，本案例提供的预训练模型均为 inference 格式，如若是重新训练的模型，需参考部署模型导出将模型导出为 inference 格式。

- 使用未加密的模型对单张图片做预测

```
.\paddlex_inference\meter_reader.exe --det_model_dir=\path\to\det_inference_model --seg_
↪model_dir=\path\to\seg_inference_model --image=\path\to\meter_test\20190822_168.jpg --
↪use_gpu=1 --use_erode=1 --save_dir=output
```

- 使用未加密的模型对图像列表做预测

```
.\paddlex_inference\meter_reader.exe --det_model_dir=\path\to\det_inference_model --seg_
↪model_dir=\path\to\seg_inference_model --image_list=\path\to\meter_test\image_list.txt␣
↪--use_gpu=1 --use_erode=1 --save_dir=output
```

- 使用未加密的模型开启摄像头做预测

```
.\paddlex_inference\meter_reader.exe --det_model_dir=\path\to\det_inference_model --seg_
↪model_dir=\path\to\seg_inference_model --use_camera=1 --use_gpu=1 --use_erode=1 --save_
↪dir=output
```

- 使用加密后的模型对单张图片做预测

如果未对模型进行加密，请参考加密 PaddleX 模型对模型进行加密。例如加密后的检测模型所在目录为\path\to\encrypted_det_inference_model，密钥为yEBLDiBOdlj+5EsNNrABhfDuQGkdcreYcHcncqwdbx0=；加密后的分割模型所在目录为\path\to\encrypted_seg_inference_model，密钥为 DbVS64I9pFRo5XmQ8MNV2kSGsfEr4FKA6OH9OUhRrsY=

```
.\paddlex_inference\meter_reader.exe --det_model_dir=\path\to\encrypted_det_inference_
↪model --seg_model_dir=\path\to\encrypted_seg_inference_model --image=\path\to\test.jpg␣
↪--use_gpu=1 --use_erode=1 --save_dir=output --det_key␣
↪yEBLDiBOdlj+5EsNNrABhfDuQGkdcreYcHcncqwdbx0= --seg_key␣
↪DbVS64I9pFRo5XmQ8MNV2kSGsfEr4FKA6OH9OUhRrsY=
```

Linux 系统的 jeton 嵌入式设备安全部署

c++ 部署

1. 下载 PaddleX 源码:

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

1. 将 PaddleX/examples/meter_reader/deploy/cpp 下的 meter_reader 文件夹和 CMakeList.txt 拷贝至 PaddleX/deploy/cpp 目录下，拷贝之前可以将 PaddleX/deploy/cpp 下原本的 CMakeList.txt 做好备份。
2. 按照 Nvidia-Jetson 开发板部署中的 Step2 至 Step3 完成 C++ 预测代码的编译。
3. 编译成功后，可执行程为 build/meter_reader/meter_reader，其主要命令参数说明如下：

1. 推理预测：

用于部署推理的模型应为 inference 格式，本案例提供的预训练模型均为 inference 格式，如若是重新训练的模型，需参考部署模型导出将模型导出为 inference 格式。

- 使用未加密的模型对单张图片做预测

```
./build/meter_reader/meter_reader --det_model_dir=/path/to/det_inference_model --seg_
↪model_dir=/path/to/seg_inference_model --image=/path/to/meter_test/20190822_168.jpg --
↪use_gpu=1 --use_erode=1 --save_dir=output
```

- 使用未加密的模型对图像列表做预测

```
./build/meter_reader/meter_reader --det_model_dir=/path/to/det_inference_model --seg_
↪model_dir=/path/to/seg_inference_model --image_list=/path/to/image_list.txt --use_
↪gpu=1 --use_erode=1 --save_dir=output
```

- 使用未加密的模型开启摄像头做预测

```
./build/meter_reader/meter_reader --det_model_dir=/path/to/det_inference_model --seg_
↪model_dir=/path/to/seg_inference_model --use_camera=1 --use_gpu=1 --use_erode=1 --save_
↪dir=output
```

- 使用加密后的模型对单张图片做预测

如果未对模型进行加密，请参考加密 PaddleX 模型对模型进行加密。例如加密后的检测模型所在目录为/path/to/encrypted_det_inference_model，密钥为 yEBLDiBOdlj+5EsNNrABhfDuQGkdcreYcHcncqwdbx0=；加密后的分割模型所在目录为/path/to/encrypted_seg_inference_model，密钥为 DbVS64I9pFRo5XmQ8MNV2kSGsfEr4FKA6OH9OUhRrsY=

```
./build/meter_reader/meter_reader --det_model_dir=/path/to/encrypted_det_inference_model␣
↪--seg_model_dir=/path/to/encrypted_seg_inference_model --image=/path/to/test.jpg --use_
↪gpu=1 --use_erode=1 --save_dir=output --det_key␣
↪yEBLDiBOdlj+5EsNNrABhfDuQGkdcreYcHcncqwdbx0= --seg_key␣
↪DbVS64I9pFRo5XmQ8MNV2kSGsfEr4FKA6OH9OUhRrsY=
```

6.2.5 模型训练

前置依赖

- Paddle paddle >= 1.8.0
- Python >= 3.5
- PaddleX >= 1.0.0

安装的相关问题参考*PaddleX 安装*

训练

- 表盘检测的训练

```
python3 /path/to/PaddleX/examples/meter_reader/train_detection.py
```

- 指针和刻度分割的训练

```
python3 /path/to/PaddleX/examples/meter_reader/train_segmentation.py
```

运行以上脚本可以训练本案例的检测模型和分割模型。如果不需要本案例的数据和模型参数，可更换数据，选择合适的模型并调整训练参数。

6.3 人像分割模型

本教程基于 PaddleX 核心分割模型实现人像分割，开放预训练模型和测试数据、支持视频流人像分割、提供模型 Fine-tune 到 Paddle-Lite 移动端部署的全流程应用指南。

6.3.1 预训练模型和测试数据

预训练模型

本案例开放了两个在大规模人像数据集上训练好的模型，以满足服务器端场景和移动端场景的需求。使用这些模型可以快速体验视频流人像分割，也可以部署到移动端进行实时人像分割，也可以用于完成模型 Fine-tuning。

- Checkpoint Parameter 为模型权重，用于 Fine-tuning 场景，包含 `__params__` 模型参数和 `model.yaml` 基础的模型配置信息。
- Inference Model 和 Quant Inference Model 为预测部署模型，包含 `__model__` 计算图结构、`__params__` 模型参数和 `model.yaml` 基础的模型配置信息。
- 其中 Inference Model 适用于服务端的 CPU 和 GPU 预测部署，Qunat Inference Model 为量化版本，适用于通过 Paddle Lite 进行移动端等端侧设备部署。

预训练模型的存储大小和推理时长如下所示，其中移动端模型的运行环境为 cpu：骁龙 855，内存：6GB，图片大小：192*192

执行以下脚本下载全部的预训练模型：

- 下载 PaddleX 源码：

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

- 下载预训练模型的代码位于 PaddleX/examples/human_segmentation，进入该目录：

```
cd PaddleX/examples/human_segmentation
```

- 执行下载

```
python pretrain_weights/download_pretrain_weights.py
```

测试数据

supervise.ly发布了人像分割数据集 **Supervisely Persons**，本案例从中随机抽取一小部分数据并转化成 PaddleX 可直接加载的数据格式，运行以下代码可下载该数据、以及手机前置摄像头拍摄的人像测试视频 video_test.mp4.

- 下载测试数据的代码位于 PaddleX/xamples/human_segmentation，进入该目录并执行下载：

```
python data/download_data.py
```

6.3.2 快速体验视频流人像分割

前置依赖

- PaddlePaddle >= 1.8.0
- Python >= 3.5
- PaddleX >= 1.0.0

安装的相关问题参考PaddleX 安装

- 下载 PaddleX 源码：

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

- 视频流人像分割和背景替换的执行文件均位于 PaddleX/examples/human_segmentation，进入该目录：

```
cd PaddleX/examples/human_segmentation
```

光流跟踪辅助的视频流人像分割

本案例将 DIS（Dense Inverse Search-basedmethod）光流跟踪算法的预测结果与 PaddleX 的分割结果进行融合，以此改善视频流人像分割的效果。运行以下代码进行体验，以下代码位于 `PaddleX/xamples/human_segmentation`：

- 通过电脑摄像头进行实时分割处理

```
python video_infer.py --model_dir pretrain_weights/humanseg_mobile_inference
```

- 对离线人像视频进行分割处理

```
python video_infer.py --model_dir pretrain_weights/humanseg_mobile_inference --video_
↪path data/video_test.mp4
```

视频分割结果如下所示：

人像背景替换

本案例还实现了人像背景替换功能，根据所选背景对人像的背景画面进行替换，背景可以是一张图片，也可以是一段视频。人像背景替换的代码位于 `PaddleX/xamples/human_segmentation`，进入该目录并执行：

- 通过电脑摄像头进行实时背景替换处理, 通过’–background_video_path’传入背景视频

```
python bg_replace.py --model_dir pretrain_weights/humanseg_mobile_inference --background_
↪image_path data/background.jpg
```

- 对人像视频进行背景替换处理, 通过’–background_video_path’传入背景视频

```
python bg_replace.py --model_dir pretrain_weights/humanseg_mobile_inference --video_path␣
↪data/video_test.mp4 --background_image_path data/background.jpg
```

- 对单张图像进行背景替换

```
python bg_replace.py --model_dir pretrain_weights/humanseg_mobile_inference --image_path␣
↪data/human_image.jpg --background_image_path data/background.jpg
```

背景替换结果如下：

注意:

- 视频分割处理时间需要几分钟，请耐心等待。
- 提供的模型适用于手机摄像头竖屏拍摄场景，宽屏效果会略差一些。

6.3.3 模型 Fine-tune

前置依赖

- PaddlePaddle >= 1.8.0
- Python >= 3.5
- PaddleX >= 1.0.0

安装的相关问题参考PaddleX 安装

- 下载 PaddleX 源码：

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

- 人像分割训练、评估、预测、模型导出、离线量化的执行文件均位于 PaddleX/examples/human_segmentation，进入该目录：

```
cd PaddleX/examples/human_segmentation
```

模型训练

使用下述命令进行基于预训练模型的模型训练，请确保选用的模型结构 model_type 与模型参数 pretrain_weights 匹配。如果不需要本案例提供的测试数据，可更换数据、选择合适的模型并调整训练参数。

```
# 指定 GPU 卡号（以 0 号卡为例）
export CUDA_VISIBLE_DEVICES=0
# 若不使用 GPU，则将 CUDA_VISIBLE_DEVICES 指定为空
# export CUDA_VISIBLE_DEVICES=
python train.py --model_type HumanSegMobile \
--save_dir output/ \
--data_dir data/mini_supervisely \
--train_list data/mini_supervisely/train.txt \
--val_list data/mini_supervisely/val.txt \
--pretrain_weights pretrain_weights/humanseg_mobile_params \
--batch_size 8 \
--learning_rate 0.001 \
--num_epochs 10 \
--image_shape 192 192
```

其中参数含义如下：

- --model_type: 模型类型，可选项为：HumanSegServer 和 HumanSegMobile

- --save_dir: 模型保存路径
- --data_dir: 数据集路径
- --train_list: 训练集列表路径
- --val_list: 验证集列表路径
- --pretrain_weights: 预训练模型路径
- --batch_size: 批大小
- --learning_rate: 初始学习率
- --num_epochs: 训练轮数
- --image_shape: 网络输入图像大小（w, h）

更多命令行帮助可运行下述命令进行查看：

```
python train.py --help
```

注意：可以通过更换--model_type 变量与对应的--pretrain_weights 使用不同的模型快速尝试。

评估

使用下述命令对模型在验证集上的精度进行评估：

```
python eval.py --model_dir output/best_model \
--data_dir data/mini_supervisely \
--val_list data/mini_supervisely/val.txt \
--image_shape 192 192
```

其中参数含义如下：

- --model_dir: 模型路径
- --data_dir: 数据集路径
- --val_list: 验证集列表路径
- --image_shape: 网络输入图像大小（w, h）

预测

使用下述命令对测试集进行预测，预测可视化结果默认保存在./output/result/文件夹中。

```
python infer.py --model_dir output/best_model \
--data_dir data/mini_supervisely \
--test_list data/mini_supervisely/test.txt \
```

(continues on next page)

(continued from previous page)

```
--save_dir output/result \
--image_shape 192 192
```

其中参数含义如下：

- `--model_dir`: 模型路径
- `--data_dir`: 数据集路径
- `--test_list`: 测试集列表路径
- `--image_shape`: 网络输入图像大小（w, h）

模型导出

在服务端部署的模型需要首先将模型导出为 inference 格式模型，导出的模型将包括 `__model__`、`__params__` 和 `model.yml` 三个文名，分别为模型的网络结构，模型权重和模型的配置文件（包括数据预处理参数等等）。在安装完 PaddleX 后，在命令行终端使用如下命令完成模型导出：

```
paddlex --export_inference --model_dir output/best_model \
--save_dir output/export
```

其中参数含义如下：

- `--model_dir`: 模型路径
- `--save_dir`: 导出模型保存路径

离线量化

```
python quant_offline.py --model_dir output/best_model \
--data_dir data/mini_supervisely \
--quant_list data/mini_supervisely/val.txt \
--save_dir output/quant_offline \
--image_shape 192 192
```

其中参数含义如下：

- `--model_dir`: 待量化模型路径
- `--data_dir`: 数据集路径
- `--quant_list`: 量化数据集列表路径，一般直接选择训练集或验证集
- `--save_dir`: 量化模型保存路径
- `--image_shape`: 网络输入图像大小（w, h）

6.3.4 Paddle-Lite 移动端部署

本案例将人像分割模型在移动端进行部署，部署流程展示如下，通用的移动端部署流程参见PaddleLite 移动端部署。

1. 将 PaddleX 模型导出为 inference 模型

本案例使用 humanseg_mobile_quant 预训练模型，该模型已经是 inference 模型，不需要再执行模型导出步骤。如果不使用预训练模型，则执行上一章节模型训练中的模型导出将自己训练的模型导出为 inference 格式。

2. 将 inference 模型优化为 PaddleLite 模型

下载并解压 模型优化工具 opt，进入模型优化工具 opt 所在路径后，执行以下命令:

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

更详细的使用方法和参数含义请参考: 使用 opt 转化模型

3. 移动端预测

PaddleX 提供了基于 PaddleX Android SDK 的安卓 demo，可供用户体验图像分类、目标检测、实例分割和语义分割，该 demo 位于 `PaddleX/deploy/lite/android/demo`，用户将模型、配置文件和测试图片拷贝至该 demo 下进行预测。

3.1 前置依赖

- Android Studio 3.4
- Android 手机或开发板

3.2 拷贝模型、配置文件和测试图片

- 将 Lite 模型(.nb 文件)拷贝到 `PaddleX/deploy/lite/android/demo/app/src/main/assets/model/` 目录下，根据.nb 文件的名字，修改文件 `PaddleX/deploy/lite/android/demo/app/src/main/res/values/strings.xml` 中的 `MODEL_PATH_DEFAULT`;

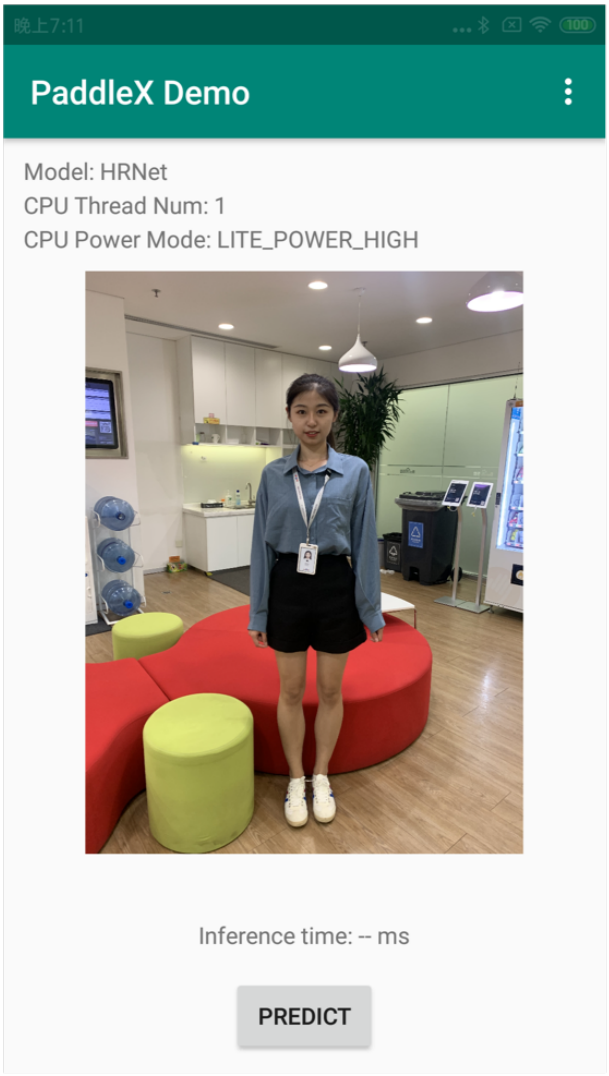
- 将配置文件（.yml 文件）拷贝到 PaddleX/deploy/lite/android/demo/app/src/main/assets/config/目录下，根据.yml 文件的名字，修改文件 PaddleX/deploy/lite/android/demo/app/src/main/res/values/strings.xml 中的 YAML_PATH_DEFAULT；
- 将测试图片拷贝到 PaddleX/deploy/lite/android/demo/app/src/main/assets/images/目录下，根据图片文件的名字，修改文件 PaddleX/deploy/lite/android/demo/app/src/main/res/values/strings.xml 中的 IMAGE_PATH_DEFAULT。

3.3 导入工程并运行

- 打开 Android Studio，在” Welcome to Android Studio” 窗口点击” Open an existing Android Studio project”，在弹出的路径选择窗口中进入 PaddleX/deploy/lite/android/demo 目录，然后点击右下角的” Open” 按钮，导入工程；
- 通过 USB 连接 Android 手机或开发板；
- 工程编译完成后，点击菜单栏的 Run->Run ‘App’ 按钮，在弹出的” Select Deployment Target” 窗口选择已经连接的 Android 设备，然后点击” OK” 按钮；
- 运行成功后，Android 设备将加载一个名为 PaddleX Demo 的 App，默认会加载一个测试图片，同时还支持拍照和从图库选择照片进行预测。

测试图片及其分割结果如下所示：

测试图片


晚上7:11
PaddleX Demo
Model: HRNet
CPU Thread Num: 1
CPU Power Mode: LITE_POWER_HIGH
Inference time: -- ms
PREDICT

分割结果


晚上7:11
PaddleX Demo
Model: HRNet
CPU Thread Num: 1
CPU Power Mode: LITE_POWER_HIGH
Inference time: 152.0 ms
PREDICT

CHAPTER 7

PaddleX GUI

PaddleX GUI 是基于 PaddleX 开发实现的可视化模型训练套件，可以让开发者免去代码开发的步骤，通过点选式地操作就可以快速完成模型的训练开发。PaddleXGUI 具有 **数据集可视化分析**、**模型参数自动推荐**、**跨平台使用**三大特点。

数据集可视化分析

PaddleX 支持导入常见的图像分类、目标检测、实例分割和语义分割数据集，并对数据集的样本分布，标注结果进行可视化展示，数据集的情况一目了然！

模型参数自动推荐

根据用户的电脑配置和数据集情况，自动推荐模型训练参数，免去用户查看文档，被各种参数所烦的忧心事！

跨平台使用

PaddleX GUI 完全跨平台，支持 Linux、Windows 和 Mac 三大主流系统！

- PaddleX GUI 版本: v1.0
- 项目官网: http://www.paddlepaddle.org.cn/paddle/paddlex
- 项目 GitHub: https://github.com/PaddlePaddle/PaddleX/tree/develop
- 官方 QQ 用户群: 1045148026
- GitHub Issue 反馈: http://www.github.com/PaddlePaddle/PaddleX/issues

CHAPTER 8

API 接口说明

8.1 数据处理与增强

transforms 为 PaddleX 的模型训练提供了数据的预处理和数据增强接口。

8.1.1 paddlex.cls.transforms

对图像分类任务的数据进行操作。可以利用*Compose*类将图像预处理/增强操作进行组合。

Compose

```
paddlex.cls.transforms.Compose(transforms)
```

根据数据预处理/增强算子对输入数据进行操作。使用示例

参数

- **transforms** (list): 数据预处理/数据增强列表。

Normalize

```
paddlex.cls.transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

对图像进行标准化。

1. 对图像进行归一化到区间 [0.0, 1.0]。
2. 对图像进行减均值除以标准差操作。

参数

- **mean** (list): 图像数据集的均值。默认为 [0.485, 0.456, 0.406]。
- **std** (list): 图像数据集的标准差。默认为 [0.229, 0.224, 0.225]。

ResizeByShort

```
paddlex.cls.transforms.ResizeByShort(short_size=256, max_size=-1)
```

根据图像的短边调整图像大小（resize）。

1. 获取图像的长边和短边长度。
2. 根据短边与 short_size 的比例，计算长边的目标长度，此时高、宽的 resize 比例为 short_size/原图短边长度。
3. 如果 max_size>0，调整 resize 比例：如果长边的目标长度 >max_size，则高、宽的 resize 比例为 max_size/原图长边长度。
4. 根据调整大小的比例对图像进行 resize。

参数

- **short_size** (int): 调整大小后的图像目标短边长度。默认为 256。
- **max_size** (int): 长边目标长度的最大限制。默认为-1。

CenterCrop

```
paddlex.cls.transforms.CenterCrop(crop_size=224)
```

以图像中心点扩散裁剪长宽为 `crop_size` 的正方形

1. 计算剪裁的起始点。
2. 剪裁图像。

参数

- **crop_size** (int): 裁剪的目标边长。默认为 224。

RandomCrop

```
paddlex.cls.transforms.RandomCrop(crop_size=224, lower_scale=0.08, lower_ratio=3. / 4,␣
↪upper_ratio=4. / 3)
```

对图像进行随机剪裁，模型训练时的数据增强操作。

1. 根据 lower_scale、lower_ratio、upper_ratio 计算随机剪裁的高、宽。
2. 根据随机剪裁的高、宽随机选取剪裁的起始点。
3. 剪裁图像。
4. 调整剪裁后的图像的大小到 crop_size*crop_size。

参数

- **crop_size** (int): 随机裁剪后重新调整的目标边长。默认为 224。
- **lower_scale** (float): 裁剪面积相对原面积比例的最小限制。默认为 0.08。
- **lower_ratio** (float): 宽变换比例的最小限制。默认为 3. / 4。
- **upper_ratio** (float): 宽变换比例的最小限制。默认为 4. / 3。

RandomHorizontalFlip

```
paddlex.cls.transforms.RandomHorizontalFlip(prob=0.5)
```

以一定的概率对图像进行随机水平翻转，模型训练时的数据增强操作。

参数

- **prob** (float): 随机水平翻转的概率。默认为 0.5。

RandomVerticalFlip

```
paddlex.cls.transforms.RandomVerticalFlip(prob=0.5)
```

以一定的概率对图像进行随机垂直翻转，模型训练时的数据增强操作。

参数

- **prob** (float): 随机垂直翻转的概率。默认为 0.5。

RandomRotate

```
paddlex.cls.transforms.RandomRotate(rotate_range=30, prob=0.5)
```

以一定的概率对图像在 [-rotate_range, rotaterange] 角度范围内进行旋转，模型训练时的数据增强操作。

参数

- **rotate_range** (int): 旋转度数的范围。默认为 30。
- **prob** (float): 随机旋转的概率。默认为 0.5。

RandomDistort

```
paddlex.cls.transforms.RandomDistort(brightness_range=0.9, brightness_prob=0.5, contrast_
↪range=0.9, contrast_prob=0.5, saturation_range=0.9, saturation_prob=0.5, hue_range=18,␣
↪hue_prob=0.5)
```

以一定的概率对图像进行随机像素内容变换，模型训练时的数据增强操作。

1. 对变换的操作顺序进行随机化操作。
2. 按照 1 中的顺序以一定的概率对图像在范围 [-range, range] 内进行随机像素内容变换。

【注意】该数据增强必须在数据增强 Normalize 之前使用。

参数

- **brightness_range** (float): 明亮度因子的范围。默认为 0.9。
- **brightness_prob** (float): 随机调整明亮度的概率。默认为 0.5。
- **contrast_range** (float): 对比度因子的范围。默认为 0.9。
- **contrast_prob** (float): 随机调整对比度的概率。默认为 0.5。
- **saturation_range** (float): 饱和度因子的范围。默认为 0.9。
- **saturation_prob** (float): 随机调整饱和度的概率。默认为 0.5。
- **hue_range** (int): 色调因子的范围。默认为 18。
- **hue_prob** (float): 随机调整色调的概率。默认为 0.5。

8.1.2 paddlex.det.transforms

对目标检测/实例分割任务的数据进行操作。可以利用*Compose*类将图像预处理/增强操作进行组合。

Compose

```
paddlex.det.transforms.Compose(transforms)
```

根据数据预处理/增强算子对输入数据进行操作。使用示例

参数

- **transforms** (list): 数据预处理/数据增强列表。

Normalize

```
paddlex.det.transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

对图像进行标准化。

1. 归一化图像到到区间 [0.0, 1.0]。
2. 对图像进行减均值除以标准差操作。

参数

- **mean** (list): 图像数据集的均值。默认为 [0.485, 0.456, 0.406]。
- **std** (list): 图像数据集的标准差。默认为 [0.229, 0.224, 0.225]。

ResizeByShort

```
paddlex.det.transforms.ResizeByShort(short_size=800, max_size=1333)
```

根据图像的短边调整图像大小（resize）。

1. 获取图像的长边和短边长度。
2. 根据短边与 short_size 的比例，计算长边的目标长度，此时高、宽的 resize 比例为 short_size/原图短边长度。
3. 如果 max_size>0，调整 resize 比例：如果长边的目标长度 >max_size，则高、宽的 resize 比例为 max_size/原图长边长度。
4. 根据调整大小的比例对图像进行 resize。

参数

- **short_size** (int): 短边目标长度。默认为 800。
- **max_size** (int): 长边目标长度的最大限制。默认为 1333。

Padding

```
paddlex.det.transforms.Padding(coarsest_stride=1)
```

将图像的长和宽 padding 至 coarsest_stride 的倍数。如输入图像为 [300, 640], `coarest_stride` 为 32，则由于 300 不为 32 的倍数，因此在图像最右和最下使用 0 值进行 padding，最终输出图像为 [320, 640]

1. 如果 coarsest_stride 为 1 则直接返回。
2. 计算宽和高与最邻近的 coarest_stride 倍数差值
3. 根据计算得到的差值，在图像最右和最下进行 padding

参数

- **coarsest_stride** (int): 填充后的图像长、宽为该参数的倍数，默认为 1。

Resize

```
paddlex.det.transforms.Resize(target_size=608, interp='LINEAR')
```

调整图像大小（resize）。

- 当目标大小（target_size）类型为 int 时，根据插值方式，将图像 resize 为 [target_size, target_size]。
- 当目标大小（target_size）类型为 list 或 tuple 时，根据插值方式，将图像 resize 为 target_size。【注意】当插值方式为“RANDOM”时，则随机选取一种插值方式进行 resize，作为模型训练时的数据增强操作。

参数

- **target_size** (int/list/tuple): 短边目标长度。默认为 608。
- **interp** (str): resize 的插值方式，与 opencv 的插值方式对应，取值范围为 [‘NEAREST’，‘LINEAR’，‘CUBIC’，‘AREA’，‘LANCZOS4’，‘RANDOM’]。默认为”LINEAR”。

RandomHorizontalFlip

```
paddlex.det.transforms.RandomHorizontalFlip(prob=0.5)
```

以一定的概率对图像进行随机水平翻转，模型训练时的数据增强操作。

参数

- **prob** (float): 随机水平翻转的概率。默认为 0.5。

RandomDistort

```
paddlex.det.transforms.RandomDistort(brightness_range=0.5, brightness_prob=0.5, contrast_
↪range=0.5, contrast_prob=0.5, saturation_range=0.5, saturation_prob=0.5, hue_range=18,␣
↪hue_prob=0.5)
```

以一定的概率对图像进行随机像素内容变换，模型训练时的数据增强操作。

1. 对变换的操作顺序进行随机化操作。
2. 按照 1 中的顺序以一定的概率对图像在范围 [-range, range] 内进行随机像素内容变换。

【注意】该数据增强必须在数据增强 Normalize 之前使用。

参数

- **brightness_range** (float): 明亮度因子的范围。默认为 0.5。
- **brightness_prob** (float): 随机调整明亮度的概率。默认为 0.5。
- **contrast_range** (float): 对比度因子的范围。默认为 0.5。
- **contrast_prob** (float): 随机调整对比度的概率。默认为 0.5。
- **saturation_range** (float): 饱和度因子的范围。默认为 0.5。
- **saturation_prob** (float): 随机调整饱和度的概率。默认为 0.5。
- **hue_range** (int): 色调因子的范围。默认为 18。
- **hue_prob** (float): 随机调整色调的概率。默认为 0.5。

MixupImage

```
paddlex.det.transforms.MixupImage(alpha=1.5, beta=1.5, mixup_epoch=-1)
```

对图像进行 mixup 操作，模型训练时的数据增强操作，目前仅 YOLOv3 模型支持该 transform。当 label_info 中不存在 mixup 字段时，直接返回，否则进行下述操作：

1. 从随机 beta 分布中抽取出随机因子 factor。
2. 根据不同情况进行处理：
   - 当 factor>=1.0 时，去除 label_info 中的 mixup 字段，直接返回。
   - 当 factor<=0.0 时，直接返回 label_info 中的 mixup 字段，并在 label_info 中去除该字段。
   - 其余情况，执行下述操作：（1）原图像乘以 factor，mixup 图像乘以 (1-factor)，叠加 2 个结果。（2）拼接原图像标注框和 mixup 图像标注框。（3）拼接原图像标注框类别和 mixup 图像标注框类别。（4）原图像标注框混合得分乘以 factor，mixup 图像标注框混合得分乘以 (1-factor)，叠加 2 个结果。
3. 更新 im_info 中的 augment_shape 信息。

参数

- **alpha** (float): 随机 beta 分布的下限。默认为 1.5。
- **beta** (float): 随机 beta 分布的上限。默认为 1.5。
- **mixup_epoch** (int): 在前 mixup_epoch 轮使用 mixup 增强操作；当该参数为-1 时，该策略不会生效。默认为-1。

RandomExpand 类

```
paddlex.det.transforms.RandomExpand(ratio=4., prob=0.5, fill_value=[123.675, 116.28, 103.
↪53])
```

随机扩张图像，模型训练时的数据增强操作。

1. 随机选取扩张比例（扩张比例大于 1 时才进行扩张）。
2. 计算扩张后图像大小。
3. 初始化像素值为输入填充值的图像，并将原图像随机粘贴于该图像上。
4. 根据原图像粘贴位置换算出扩张后真实标注框的位置坐标。
5. 根据原图像粘贴位置换算出扩张后真实分割区域的位置坐标。

参数

- **ratio** (float): 图像扩张的最大比例。默认为 4.0。
- **prob** (float): 随机扩张的概率。默认为 0.5。

- **fill_value** (list): 扩张图像的初始填充值（0-255）。默认为 [123.675, 116.28, 103.53]。

【注意】该数据增强必须在数据增强 Resize、ResizeByShort 之前使用。

RandomCrop

```
paddlex.det.transforms.RandomCrop(aspect_ratio=[.5, 2.], thresholds=[.0, .1, .3, .5, .7,␣
↪.9], scaling=[.3, 1.], num_attempts=50, allow_no_crop=True, cover_all_box=False)
```

随机裁剪图像，模型训练时的数据增强操作。

1. 若 allow_no_crop 为 True，则在 thresholds 加入’ no_crop’。
2. 随机打乱 thresholds。
3. 遍历 thresholds 中各元素：(1) 如果当前 thresh 为’ no_crop’，则返回原始图像和标注信息。(2) 随机取出 aspect_ratio 和 scaling 中的值并由此计算出候选裁剪区域的高、宽、起始点。(3) 计算真实标注框与候选裁剪区域 IoU，若全部真实标注框的 IoU 都小于 thresh，则继续第 3 步。(4) 如果 cover_all_box 为 True 且存在真实标注框的 IoU 小于 thresh，则继续第 3 步。(5) 筛选出位于候选裁剪区域内的真实标注框，若有效框的个数为 0，则继续第 3 步，否则进行第 4 步。
4. 换算有效真值标注框相对候选裁剪区域的位置坐标。
5. 换算有效分割区域相对候选裁剪区域的位置坐标。

【注意】该数据增强必须在数据增强 Resize、ResizeByShort 之前使用。

参数

- **aspect_ratio** (list): 裁剪后短边缩放比例的取值范围，以 [min, max] 形式表示。默认值为 [.5, 2.]。
- **thresholds** (list): 判断裁剪候选区域是否有效所需的 IoU 阈值取值列表。默认值为 [.0, .1, .3, .5, .7, .9]。
- **scaling** (list): 裁剪面积相对原面积的取值范围，以 [min, max] 形式表示。默认值为 [.3, 1.]。
- **num_attempts** (int): 在放弃寻找有效裁剪区域前尝试的次数。默认值为 50。
- **allow_no_crop** (bool): 是否允许未进行裁剪。默认值为 True。
- **cover_all_box** (bool): 是否要求所有的真实标注框都必须在裁剪区域内。默认值为 False。

8.1.3 paddlex.seg.transforms

对用于分割任务的数据进行操作。可以利用*Compose*类将图像预处理/增强操作进行组合。

Compose

```
paddlex.seg.transforms.Compose(transforms)
```

根据数据预处理/数据增强列表对输入数据进行操作。使用示例

参数

- **transforms** (list): 数据预处理/数据增强列表。

RandomHorizontalFlip

```
paddlex.seg.transforms.RandomHorizontalFlip(prob=0.5)
```

以一定的概率对图像进行水平翻转, 模型训练时的数据增强操作。

参数

- **prob** (float): 随机水平翻转的概率。默认值为 0.5。

RandomVerticalFlip

```
paddlex.seg.transforms.RandomVerticalFlip(prob=0.1)
```

以一定的概率对图像进行垂直翻转, 模型训练时的数据增强操作。

参数

- **prob** (float): 随机垂直翻转的概率。默认值为 0.1。

Resize

```
paddlex.seg.transforms.Resize(target_size, interp='LINEAR')
```

调整图像大小（resize）。

- 当目标大小（target_size）类型为 int 时，根据插值方式，将图像 resize 为 [target_size, target_size]。
- 当目标大小（target_size）类型为 list 或 tuple 时，根据插值方式，将图像 resize 为 target_size, target_size 的输入应为 [w, h] 或（w, h）。

参数

- **target_size** (int|list|tuple): 目标大小
- **interp** (str): resize 的插值方式，与 opencv 的插值方式对应，可选的值为 [‘NEAREST’，‘LINEAR’，‘CUBIC’，‘AREA’，‘LANCZOS4’]，默认为”LINEAR”。

ResizeByLong

```
paddlex.seg.transforms.ResizeByLong(long_size)
```

对图像长边 resize 到固定值，短边按比例进行缩放。

参数

- **long_size** (int): resize 后图像的长边大小。

ResizeRangeScaling

```
paddlex.seg.transforms.ResizeRangeScaling(min_value=400, max_value=600)
```

对图像长边随机 resize 到指定范围内，短边按比例进行缩放, 模型训练时的数据增强操作。

参数

- **min_value** (int): 图像长边 resize 后的最小值。默认值 400。
- **max_value** (int): 图像长边 resize 后的最大值。默认值 600。

ResizeStepScaling

```
paddlex.seg.transforms.ResizeStepScaling(min_scale_factor=0.75, max_scale_factor=1.25,␣
↪scale_step_size=0.25)
```

对图像按照某一个比例 resize，这个比例以 scale_step_size 为步长，在 [min_scale_factor, max_scale_factor] 随机变动, 模型训练时的数据增强操作。

参数

- **min_scale_factor** (float), resize 最小尺度。默认值 0.75。
- **max_scale_factor** (float), resize 最大尺度。默认值 1.25。

- **scale__step__size** (float), resize 尺度范围间隔。默认值 0.25。

Normalize

```
paddlex.seg.transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
```

对图像进行标准化。

1. 图像像素归一化到区间 [0.0, 1.0]。2. 对图像进行减均值除以标准差操作。

参数

- **mean** (list): 图像数据集的均值。默认值 [0.5, 0.5, 0.5]。
- **std** (list): 图像数据集的标准差。默认值 [0.5, 0.5, 0.5]。

Padding

```
paddlex.seg.transforms.Padding(target_size, im_padding_value=[127.5, 127.5, 127.5],␣
↪label_padding_value=255)
```

对图像或标注图像进行 padding，padding 方向为右和下。根据提供的值对图像或标注图像进行 padding 操作。

参数

- **target__size** (int|list|tuple): padding 后图像的大小。
- **im__padding__value** (list): 图像 padding 的值。默认为 [127.5, 127.5, 127.5]。
- **label__padding__value** (int): 标注图像 padding 的值。默认值为 255（仅在训练时需要设定该参数）。

RandomPaddingCrop

```
paddlex.seg.transforms.RandomPaddingCrop(crop_size=512, im_padding_value=[127.5, 127.5,␣
↪127.5], label_padding_value=255)
```

对图像和标注图进行随机裁剪，当所需要的裁剪尺寸大于原图时，则进行 padding 操作，模型训练时的数据增强操作。

参数

- **crop_size** (int|list|tuple): 裁剪图像大小。默认为 512。
- **im_padding_value** (list): 图像 padding 的值。默认为 [127.5, 127.5, 127.5]。
- **label_padding_value** (int): 标注图像 padding 的值。默认值为 255。

RandomBlur

```
paddlex.seg.transforms.RandomBlur(prob=0.1)
```

以一定的概率对图像进行高斯模糊，模型训练时的数据增强操作。

参数

- **prob** (float): 图像模糊概率。默认为 0.1。

RandomRotate

```
paddlex.seg.transforms.RandomRotate(rotate_range=15, im_padding_value=[127.5, 127.5, 127.
↪5], label_padding_value=255)
```

对图像进行随机旋转, 模型训练时的数据增强操作。

在旋转区间 [-rotate_range, rotate_range] 内，对图像进行随机旋转，当存在标注图像时，同步进行，并对旋转后的图像和标注图像进行相应的 padding。

参数

- **rotate_range** (float): 最大旋转角度。默认为 15 度。
- **im_padding_value** (list): 图像 padding 的值。默认为 [127.5, 127.5, 127.5]。
- **label_padding_value** (int): 标注图像 padding 的值。默认为 255。

RandomScaleAspect

```
paddlex.seg.transforms.RandomScaleAspect(min_scale=0.5, aspect_ratio=0.33)
```

裁剪并 resize 回原始尺寸的图像和标注图像, 模型训练时的数据增强操作。

按照一定的面积比和宽高比对图像进行裁剪，并 reszie 回原始图像的图像，当存在标注图时，同步进行。

参数

- **min_scale** (float)：裁取图像占原始图像的面积比，取值 [0，1]，为 0 时则返回原图。默认为 0.5。
- **aspect_ratio** (float)：裁取图像的宽高比范围，非负值，为 0 时返回原图。默认为 0.33。

RandomDistort

```
paddlex.seg.transforms.RandomDistort(brightness_range=0.5, brightness_prob=0.5, contrast_
↪range=0.5, contrast_prob=0.5, saturation_range=0.5, saturation_prob=0.5, hue_range=18,␣
↪hue_prob=0.5)
```

以一定的概率对图像进行随机像素内容变换，模型训练时的数据增强操作。

1. 对变换的操作顺序进行随机化操作。2. 按照 1 中的顺序以一定的概率对图像在范围 [-range, range] 内进行随机像素内容变换。

【注意】该数据增强必须在数据增强 Normalize 之前使用。

参数

- **brightness_range** (float)：明亮度因子的范围。默认为 0.5。
- **brightness_prob** (float)：随机调整明亮度的概率。默认为 0.5。
- **contrast_range** (float)：对比度因子的范围。默认为 0.5。
- **contrast_prob** (float)：随机调整对比度的概率。默认为 0.5。
- **saturation_range** (float)：饱和度因子的范围。默认为 0.5。
- **saturation_prob** (float)：随机调整饱和度的概率。默认为 0.5。
- **hue_range** (int)：色调因子的范围。默认为 18。
- **hue_prob** (float)：随机调整色调的概率。默认为 0.5。

8.1.4 数据增强与 imgaug 支持

数据增强操作可用于在模型训练时，增加训练样本的多样性，从而提升模型的泛化能力。

PaddleX 内置增强操作

PaddleX 对于图像分类、目标检测、实例分割和语义分割内置了部分常见的数据增强操作，如下表所示，

imgaug 增强库的支持

PaddleX 目前已适配 imgaug 图像增强库，用户可以直接在 PaddleX 构造 `transforms` 时，调用 imgaug 的方法，如下示例

```
import paddlex as pdx
from paddlex.cls import transforms
import imgaug.augmenters as iaa
train_transforms = transforms.Compose([
    # 随机在 [0.0 3.0] 中选值对图像进行模糊
    iaa.blur.GaussianBlur(sigma=(0.0, 3.0)),
    transforms.RandomCrop(crop_size=224),
    transforms.Normalize()
])
```

除了上述用法，`Compose` 接口中也支持 imgaug 的 `Someof`、`Sometimes`、`Sequential`、`Oneof` 等操作，开发者可以通过这些方法随意组合出增强流程。由于 imgaug 对于标注信息 (目标检测框和实例分割 mask) 与 PaddleX 模型训练逻辑有部分差异，**目前在检测和分割中，只支持 pixel-level 的增强方法，(即在增强时，不对图像的大小和方向做改变) 其它方法仍在适配中**，详情可见下表，

需要注意的是，imgaug 的基础方法中，如 `imgaug.augmenters.blur` 仅为图像处理操作，并无概率设置，而在 CV 模型训练中，增强操作往往是以一定概率应用在样本上，因此我们可以通过 imgaug 的 `Someof`、`Sometimes`、`Sequential`、`Oneof` 等操作来组合实现，如下代码所示，

- `Someof` 执行定义增强方法列表中的部分方法
- `Sometimes` 以一定概率执行定义的增强方法列表
- `Sequential` 按顺序执行定义的增强方法列表

```
image imgaug.augmenters as iaa
from paddlex.cls import transforms
# 以 0.6 的概率对图像样本进行模糊
img_augmenters = iaa.Sometimes(0.6, [
    iaa.blur.GaussianBlur(sigma=(0.0, 3.0))
])
train_transforms = transforms.Compose([
    img_augmenters,
    transforms.RandomCrop(crop_size=224),
    transforms.Normalize()
])
```

8.2 数据集读取

8.2.1 paddlex.datasets.ImageNet

用于图像分类模型

```
paddlex.datasets.ImageNet(data_dir, file_list, label_list, transforms=None, num_workers=
'auto' , buffer_size=100, parallel_method='thread', shuffle=False)
```

读取 ImageNet 格式的分类数据集，并对样本进行相应的处理。ImageNet 数据集格式的介绍可查看文档:数据集格式说明

示例：代码文件

参数

- **data_dir** (str): 数据集所在的目录路径。
- **file_list** (str): 描述数据集图片文件和类别 id 的文件路径（文本内每行路径为相对 data_dir 的相对路径）。
- **label_list** (str): 描述数据集包含的类别信息文件路径。
- **transforms** (paddlex.cls.transforms): 数据集中每个样本的预处理/增强算子，详见paddlex.cls.transforms。
- **num_workers** (int|str): 数据集中样本在预处理过程中的线程或进程数。默认为'auto'。当设为'auto'时，根据系统的实际 CPU 核数设置 num_workers: 如果 CPU 核数的一半大于 8，则 num_workers 为 8，否则为 CPU 核数的一半。
- **buffer_size** (int): 数据集中样本在预处理过程中队列的缓存长度，以样本数为单位。默认为 100。
- **parallel_method** (str): 数据集中样本在预处理过程中并行处理的方式，支持'thread'线程和'process'进程两种方式。默认为'process'（Windows 和 Mac 下会强制使用 thread，该参数无效）。
- **shuffle** (bool): 是否需要对数据集中样本打乱顺序。默认为 False。

8.2.2 paddlex.datasets.VOCDetection

用于目标检测模型

```
paddlex.datasets.VOCDetection(data_dir, file_list, label_list, transforms=None, num_
↪workers= 'auto' , buffer_size=100, parallel_method='thread', shuffle=False)
```

读取 PascalVOC 格式的检测数据集，并对样本进行相应的处理。PascalVOC 数据集格式的介绍可查看文档:数据集格式说明

示例：代码文件

参数

- **data_dir** (str)：数据集所在的目录路径。
- **file_list** (str)：描述数据集图片文件和对应标注文件的文件路径（文本内每行路径为相对 data_dir 的相对路径）。
- **label_list** (str)：描述数据集包含的类别信息文件路径。
- **transforms** (paddlex.det.transforms)：数据集中每个样本的预处理/增强算子，详见paddlex.det.transforms。
- **num_workers** (int|str)：数据集中样本在预处理过程中的线程或进程数。默认为’auto’。当设为’auto’时，根据系统的实际 CPU 核数设置 num_workers：如果 CPU 核数的一半大于 8，则 num_workers 为 8，否则为 CPU 核数的一半。
- **buffer_size** (int)：数据集中样本在预处理过程中队列的缓存长度，以样本数为单位。默认为 100。
- **parallel_method** (str)：数据集中样本在预处理过程中并行处理的方式，支持’thread’线程和’process’进程两种方式。默认为’process’（Windows 和 Mac 下会强制使用 thread，该参数无效）。
- **shuffle** (bool)：是否需要对数据集中样本打乱顺序。默认为 False。

8.2.3 paddlex.datasets.CocoDetection

用于实例分割/目标检测模型

```
paddlex.datasets.CocoDetection(data_dir, ann_file, transforms=None, num_workers='auto',␣
↪buffer_size=100, parallel_method='thread', shuffle=False)
```

读取 MSCOCO 格式的检测数据集，并对样本进行相应的处理，该格式的数据集同样可以应用到实例分割模型的训练中。MSCOCO 数据集格式的介绍可查看文档:数据集格式说明

示例：代码文件

参数

- **data_dir** (str)：数据集所在的目录路径。
- **ann_file** (str)：数据集的标注文件，为一个独立的 json 格式文件。
- **transforms** (paddlex.det.transforms)：数据集中每个样本的预处理/增强算子，详见paddlex.det.transforms。
- **num_workers** (int|str)：数据集中样本在预处理过程中的线程或进程数。默认为’auto’。当设为’auto’时，根据系统的实际 CPU 核数设置 num_workers：如果 CPU 核数的一半大于 8，则 num_workers 为 8，否则为 CPU 核数的一半。

- **buffer_size** (int): 数据集中样本在预处理过程中队列的缓存长度，以样本数为单位。默认为 100。
- **parallel_method** (str): 数据集中样本在预处理过程中并行处理的方式，支持’thread’线程和’process’进程两种方式。默认为’process’（Windows 和 Mac 下会强制使用 thread，该参数无效）。
- **shuffle** (bool): 是否需要对数据集中样本打乱顺序。默认为 False。

8.2.4 paddlex.datasets.SegDataset

用于语义分割模型

```
paddlex.datasets.SegDataset(data_dir, file_list, label_list, transforms=None, num_
↪workers='auto', buffer_size=100, parallel_method='thread', shuffle=False)
```

读取语义分割任务数据集，并对样本进行相应的处理。语义分割任务数据集格式的介绍可查看文档:数据集格式说明

示例：代码文件

参数

- **data_dir** (str): 数据集所在的目录路径。
- **file_list** (str): 描述数据集图片文件和对应标注文件的文件路径（文本内每行路径为相对 `data_dir` 的相对路径）。
- **label_list** (str): 描述数据集包含的类别信息文件路径。
- **transforms** (paddlex.seg.transforms): 数据集中每个样本的预处理/增强算子，详见paddlex.seg.transforms。
- **num_workers** (int|str)：数据集中样本在预处理过程中的线程或进程数。默认为’auto’。当设为’auto’时，根据系统的实际 CPU 核数设置 `num_workers`: 如果 CPU 核数的一半大于 8，则 `num_workers` 为 8，否则为 CPU 核数的一半。
- **buffer_size** (int): 数据集中样本在预处理过程中队列的缓存长度，以样本数为单位。默认为 100。
- **parallel_method** (str): 数据集中样本在预处理过程中并行处理的方式，支持’thread’线程和’process’进程两种方式。默认为’process’（Windows 和 Mac 下会强制使用 thread，该参数无效）。
- **shuffle** (bool): 是否需要对数据集中样本打乱顺序。默认为 False。

8.2.5 paddlex.datasets.EasyDataCls

用于图像分类模型

```
paddlex.datasets.EasyDataCls(data_dir, file_list, label_list, transforms=None, num_
↪workers='auto', buffer_size=100, parallel_method='thread', shuffle=False)
```

读取 EasyData 平台标注图像分类数据集，并对样本进行相应的处理。

参数

- **data_dir** (str): 数据集所在的目录路径。
- **file_list** (str): 描述数据集图片文件和对应标注文件的文件路径（文本内每行路径为相对 data_dir 的相对路径）。
- **label_list** (str): 描述数据集包含的类别信息文件路径。
- **transforms** (paddlex.seg.transforms): 数据集中每个样本的预处理/增强算子，详见paddlex.cls.transforms。
- **num_workers** (int|str): 数据集中样本在预处理过程中的线程或进程数。默认为’auto’。当设为’auto’时，根据系统的实际 CPU 核数设置 num_workers: 如果 CPU 核数的一半大于 8，则 num_workers 为 8，否则为 CPU 核数的一半。
- **buffer_size** (int): 数据集中样本在预处理过程中队列的缓存长度，以样本数为单位。默认为 100。
- **parallel_method** (str): 数据集中样本在预处理过程中并行处理的方式，支持’thread’线程和’process’进程两种方式。默认为’process’（Windows 和 Mac 下会强制使用 thread，该参数无效）。
- **shuffle** (bool): 是否需要对数据集中样本打乱顺序。默认为 False。

8.2.6 paddlex.datasets.EasyDataDet

用于**目标检测/实例分割模型**

```
paddlex.datasets.EasyDataDet(data_dir, file_list, label_list, transforms=None, num_
↪workers= ‘auto’ , buffer_size=100, parallel_method='thread', shuffle=False)
```

读取 EasyData 目标检测/实例分割格式数据集，并对样本进行相应的处理，该格式的数据集同样可以应用到实例分割模型的训练中。

参数

- **data_dir** (str): 数据集所在的目录路径。
- **file_list** (str): 描述数据集图片文件和对应标注文件的文件路径（文本内每行路径为相对 data_dir 的相对路径）。
- **label_list** (str): 描述数据集包含的类别信息文件路径。

- **transforms** (paddlex.det.transforms): 数据集中每个样本的预处理/增强算子，详见paddlex.det.transforms。
- **num_workers** (int|str)：数据集中样本在预处理过程中的线程或进程数。默认为’auto’。当设为’auto’时，根据系统的实际 CPU 核数设置 `num_workers`: 如果 CPU 核数的一半大于 8，则 `num_workers` 为 8，否则为 CPU 核数的一半。
- **buffer_size** (int): 数据集中样本在预处理过程中队列的缓存长度，以样本数为单位。默认为 100。
- **parallel_method** (str): 数据集中样本在预处理过程中并行处理的方式，支持’thread’线程和’process’进程两种方式。默认为’process’（Windows 和 Mac 下会强制使用 thread，该参数无效）。
- **shuffle** (bool): 是否需要对数据集中样本打乱顺序。默认为 False。

8.2.7 paddlex.datasets.EasyDataSeg

用于语义分割模型

```
paddlex.datasets.EasyDataSeg(data_dir, file_list, label_list, transforms=None, num_
↪workers='auto', buffer_size=100, parallel_method='thread', shuffle=False)
```

读取 EasyData 语义分割任务数据集，并对样本进行相应的处理。

参数

- **data_dir** (str): 数据集所在的目录路径。
- **file_list** (str): 描述数据集图片文件和对应标注文件的文件路径（文本内每行路径为相对 `data_dir` 的相对路径）。
- **label_list** (str): 描述数据集包含的类别信息文件路径。
- **transforms** (paddlex.seg.transforms): 数据集中每个样本的预处理/增强算子，详见paddlex.seg.transforms。
- **num_workers** (int|str)：数据集中样本在预处理过程中的线程或进程数。默认为’auto’。当设为’auto’时，根据系统的实际 CPU 核数设置 `num_workers`: 如果 CPU 核数的一半大于 8，则 `num_workers` 为 8，否则为 CPU 核数的一半。
- **buffer_size** (int): 数据集中样本在预处理过程中队列的缓存长度，以样本数为单位。默认为 100。
- **parallel_method** (str): 数据集中样本在预处理过程中并行处理的方式，支持’thread’线程和’process’进程两种方式。默认为’process’（Windows 和 Mac 下会强制使用 thread，该参数无效）。
- **shuffle** (bool): 是否需要对数据集中样本打乱顺序。默认为 False。

8.3 视觉模型集

PaddleX 目前支持 四种视觉任务解决方案 ，包括图像分类、目标检测、实例分割和语义分割。对于每种视觉任务，PaddleX 又提供了 1 种或多种模型，用户可根据需求及应用场景选取。

8.3.1 Image Classification

paddlex.cls.ResNet50

```
paddlex.cls.ResNet50(num_classes=1000)
```

构建 ResNet50 分类器，并实现其训练、评估和预测。

参数

- **num_classes** (int): 类别数。默认为 1000。

train

```
train(self, num_epochs, train_dataset, train_batch_size=64, eval_dataset=None, save_
↪interval_epochs=1, log_interval_steps=2, save_dir='output', pretrain_weights='IMAGENET
↪', optimizer=None, learning_rate=0.025, warmup_steps=0, warmup_start_lr=0.0, lr_decay_
↪epochs=[30, 60, 90], lr_decay_gamma=0.1, use_vdl=False, sensitivities_file=None, eval_
↪metric_loss=0.05, early_stop=False, early_stop_patience=5, resume_checkpoint=None)
```

参数

- **num_epochs** (int): 训练迭代轮数。
- **train_dataset** (paddlex.datasets): 训练数据读取器。
- **train_batch_size** (int): 训练数据 batch 大小。同时作为验证数据 batch 大小。默认值为 64。
- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **save_interval_epochs** (int): 模型保存间隔（单位：迭代轮数）。默认为 1。
- **log_interval_steps** (int): 训练日志输出间隔（单位：迭代步数）。默认为 2。
- **save_dir** (str): 模型保存路径。
- **pretrain_weights** (str): 若指定为路径时，则加载路径下预训练模型；若为字符串’IMAGENET’，则自动下载在 ImageNet 图片数据上预训练的模型权重；若为 None，则不使用预训练模型。默认为’IMAGENET’。

- **optimizer** (paddle.fluid.optimizer): 优化器。当该参数为 None 时，使用默认优化器：fluid.layers.piecewise_decay 衰减策略，fluid.optimizer.Momentum 优化方法。
- **learning_rate** (float): 默认优化器的初始学习率。默认为 0.025。
- **warmup_steps** (int): 默认优化器的 warmup 步数，学习率将在设定的步数内，从 warmup_start_lr 线性增长至设定的 learning_rate，默认为 0。
- **warmup_start_lr**(float): 默认优化器的 warmup 起始学习率，默认为 0.0。
- **lr_decay_epochs** (list): 默认优化器的学习率衰减轮数。默认为 [30, 60, 90]。
- **lr_decay_gamma** (float): 默认优化器的学习率衰减率。默认为 0.1。
- **use_vdl** (bool): 是否使用 VisualDL 进行可视化。默认值为 False。
- **sensitivities_file** (str): 若指定为路径时，则加载路径下敏感度信息进行裁剪；若为字符串’DEFAULT’，则自动下载在 ImageNet 图片数据上获得的敏感度信息进行裁剪；若为 None，则不进行裁剪。默认为 None。
- **eval_metric_loss** (float): 可容忍的精度损失。默认为 0.05。
- **early_stop** (bool): 是否使用提前终止训练策略。默认值为 False。
- **early_stop_patience** (int): 当使用提前终止训练策略时，如果验证集精度在 `early_stop_patience` 个 epoch 内连续下降或持平，则终止训练。默认值为 5。
- **resume_checkpoint** (str): 恢复训练时指定上次训练保存的模型路径。若为 None，则不会恢复训练。默认值为 None。

evaluate

```
evaluate(self, eval_dataset, batch_size=1, epoch_id=None, return_details=False)
```

参数

- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **batch_size** (int): 验证数据批大小。默认为 1。
- **epoch_id** (int): 当前评估模型所在的训练轮数。
- **return_details** (bool): 是否返回详细信息，默认 False。

返回值

- **dict**: 当 return_details 为 False 时，返回 dict, 包含关键字：’acc1’、’acc5’，分别表示最大值的 accuracy、前 5 个最大值的 accuracy。
- **tuple** (metrics, eval_details): 当 `return_details` 为 True 时，增加返回 dict，包含关键字：’true_labels’、’pred_scores’，分别代表真实类别 id、每个类别的预测得分。

predict

```
predict(self, img_file, transforms=None, topk=5)
```

分类模型预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 `ResNet50.test_transforms` 和 `ResNet50.eval_transforms` 中。如未在训练时定义 eval_dataset，那在调用预测 `predict` 接口时，用户需要再重新定义 test_transforms 传入给 `predict` 接口。

参数

- **img_file** (str|np.ndarray): 预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.cls.transforms): 数据预处理操作。
- **topk** (int): 预测时前 k 个最大值。

返回值

- **list**: 其中元素均为字典。字典的关键字为’ category_id’、’ category’、’ score’，分别对应预测类别 id、预测类别标签、预测得分。

batch_predict

```
batch_predict(self, img_file_list, transforms=None, topk=5, thread_num=2)
```

分类模型批量预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 `ResNet50.test_transforms` 和 `ResNet50.eval_transforms` 中。如未在训练时定义 eval_dataset，那在调用预测 `predict` 接口时，用户需要再重新定义 test_transforms 传入给 `predict` 接口。

参数

- **img_file_list** (list|tuple): 对列表（或元组）中的图像同时进行预测，列表中的元素可以是图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.cls.transforms): 数据预处理操作。
- **topk** (int): 预测时前 k 个最大值。
- **thread_num** (int): 并发执行各图像预处理时的线程数。

返回值

- **list**: 每个元素都为列表，表示各图像的预测结果。在各图像的预测列表中，其中元素均为字典。字典的关键字为’ category_id’、’ category’、’ score’，分别对应预测类别 id、预测类别标签、预测得分。

其它分类模型

PaddleX 提供了共计 22 种分类模型，所有分类模型均提供同 `ResNet50` 相同的训练 `train`，评估 `evaluate` 和预测 `predict` 接口，各模型效果可参考模型库。

8.3.2 Object Detection

paddlex.det.YOLOv3

```
paddlex.det.YOLOv3(num_classes=80, backbone='MobileNetV1', anchors=None, anchor_
↪masks=None, ignore_threshold=0.7, nms_score_threshold=0.01, nms_topk=1000, nms_keep_
↪topk=100, nms_iou_threshold=0.45, label_smooth=False, train_random_shapes=[320, 352,␣
↪384, 416, 448, 480, 512, 544, 576, 608])
```

构建 YOLOv3 检测器。**注意在 YOLOv3，num_classes 不需要包含背景类，如目标包括 human、dog 两种，则 num_classes 设为 2 即可，这里与 FasterRCNN/MaskRCNN 有差别**

参数

- **num_classes** (int): 类别数。默认为 80。
- **backbone** (str): YOLOv3 的 backbone 网络，取值范围为 [‘DarkNet53’，‘ResNet34’，‘MobileNetV1’，‘MobileNetV3_large’]。默认为’MobileNetV1’。
- **anchors** (list|tuple): anchor 框的宽度和高度，为 None 时表示使用默认值 [[10, 13], [16, 30], [33, 23], [30, 61], [62, 45], [59, 119], [116, 90], [156, 198], [373, 326]]。
- **anchor_masks** (list|tuple): 在计算 YOLOv3 损失时，使用 anchor 的 mask 索引，为 None 时表示使用默认值 [[6, 7, 8], [3, 4, 5], [0, 1, 2]]。
- **ignore_threshold** (float): 在计算 YOLOv3 损失时，IoU 大于 `ignore_threshold` 的预测框的置信度被忽略。默认为 0.7。
- **nms_score_threshold** (float): 检测框的置信度得分阈值，置信度得分低于阈值的框应该被忽略。默认为 0.01。
- **nms_topk** (int): 进行 NMS 时，根据置信度保留的最大检测框数。默认为 1000。
- **nms_keep_topk** (int): 进行 NMS 后，每个图像要保留的总检测框数。默认为 100。
- **nms_iou_threshold** (float): 进行 NMS 时，用于剔除检测框 IOU 的阈值。默认为 0.45。
- **label_smooth** (bool): 是否使用 label smooth。默认值为 False。
- **train_random_shapes** (list|tuple): 训练时从列表中随机选择图像大小。默认值为 [320, 352, 384, 416, 448, 480, 512, 544, 576, 608]。

train

```
train(self, num_epochs, train_dataset, train_batch_size=8, eval_dataset=None, save_
↪interval_epochs=20, log_interval_steps=2, save_dir='output', pretrain_weights='IMAGENET
↪', optimizer=None, learning_rate=1.0/8000, warmup_steps=1000, warmup_start_lr=0.0, lr_
↪decay_epochs=[213, 240], lr_decay_gamma=0.1, metric=None, use_vdl=False, sensitivities_
↪file=None, eval_metric_loss=0.05, early_stop=False, early_stop_patience=5, resume_
↪checkpoint=None)
```

YOLOv3 模型的训练接口，函数内置了 piecewise 学习率衰减策略和 momentum 优化器。

参数

- **num_epochs** (int): 训练迭代轮数。
- **train_dataset** (paddlex.datasets): 训练数据读取器。
- **train_batch_size** (int): 训练数据 batch 大小。目前检测仅支持单卡评估，训练数据 batch 大小与显卡数量之商为验证数据 batch 大小。默认值为 8。
- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **save_interval_epochs** (int): 模型保存间隔（单位：迭代轮数）。默认为 20。
- **log_interval_steps** (int): 训练日志输出间隔（单位：迭代次数）。默认为 2。
- **save_dir** (str): 模型保存路径。默认值为’output’。
- **pretrain_weights** (str): 若指定为路径时，则加载路径下预训练模型；若为字符串’IMAGENET’，则自动下载在 ImageNet 图片数据上预训练的模型权重；若为字符串’COCO’，则自动下载在 COCO 数据集上预训练的模型权重；若为 None，则不使用预训练模型。默认为 None。
- **optimizer** (paddle.fluid.optimizer): 优化器。当该参数为 None 时，使用默认优化器：fluid.layers.piecewise_decay 衰减策略，fluid.optimizer.Momentum 优化方法。
- **learning_rate** (float): 默认优化器的学习率。默认为 1.0/8000。
- **warmup_steps** (int): 默认优化器进行 warmup 过程的步数。默认为 1000。
- **warmup_start_lr** (int): 默认优化器 warmup 的起始学习率。默认为 0.0。
- **lr_decay_epochs** (list): 默认优化器的学习率衰减轮数。默认为 [213, 240]。
- **lr_decay_gamma** (float): 默认优化器的学习率衰减率。默认为 0.1。
- **metric** (bool): 训练过程中评估的方式，取值范围为 [‘COCO’，‘VOC’]。默认值为 None。
- **use_vdl** (bool): 是否使用 VisualDL 进行可视化。默认值为 False。
- **sensitivities_file** (str): 若指定为路径时，则加载路径下敏感度信息进行裁剪；若为字符串’ DEFAULT’，则自动下载在 PascalVOC 数据上获得的敏感度信息进行裁剪；若为 None，

则不进行裁剪。默认为 None。

- **eval_metric_loss** (float): 可容忍的精度损失。默认为 0.05。
- **early_stop** (bool): 是否使用提前终止训练策略。默认值为 False。
- **early_stop_patience** (int): 当使用提前终止训练策略时，如果验证集精度在 early_stop_patience 个 epoch 内连续下降或持平，则终止训练。默认值为 5。
- **resume_checkpoint** (str): 恢复训练时指定上次训练保存的模型路径。若为 None，则不会恢复训练。默认值为 None。

evaluate

```
evaluate(self, eval_dataset, batch_size=1, epoch_id=None, metric=None, return_
↪details=False)
```

YOLOv3 模型的评估接口，模型评估后会返回在验证集上的指标 box_map(metric 指定为’ VOC’ 时) 或 box_mmap(metric 指定为 COCO 时)。

参数

- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **batch_size** (int): 验证数据批大小。默认为 1。
- **epoch_id** (int): 当前评估模型所在的训练轮数。
- **metric** (bool): 训练过程中评估的方式，取值范围为 [‘COCO’，‘VOC’]。默认为 None，根据用户传入的 Dataset 自动选择，如为 VOCDetection，则 metric 为’ VOC’ ；如为 COCODetection，则 metric 为’ COCO’ 默认为 None，如为 EasyData 类型数据集，同时也会使用’ VOC’。
- **return_details** (bool): 是否返回详细信息。默认值为 False。

返回值

- **tuple** (metrics, eval_details) | **dict** (metrics): 当 return_details 为 True 时，返回 (metrics, eval_details)，当 return_details 为 False 时，返回 metrics。metrics 为 dict，包含关键字：’ bbox_mmap’ 或者’ bbox_map ‘，分别表示平均准确率平均值在各个阈值下的结果取平均值的结果（mmAP）、平均准确率平均值（mAP）。eval_details 为 dict，包含关键字：’ bbox’，对应元素预测结果列表，每个预测结果由图像 id、预测框类别 id、预测框坐标、预测框得分；’ gt ‘：真实标注框相关信息。

predict

```
predict(self, img_file, transforms=None)
```

YOLOv3 模型预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 YOLOv3.test_transforms 和 YOLOv3.eval_transforms 中。如未在训练时定义 eval_dataset，那在调用预测 predict 接口时，用户需要再重新定义 test_transforms 传入给 predict 接口

参数

- **img_file** (str|np.ndarray): 预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.det.transforms): 数据预处理操作。

返回值

- **list**: 预测结果列表，列表中每个元素均为一个 dict，key 包括'bbox'，'category'，'category_id'，'score'，分别表示每个预测目标的框坐标信息、类别、类别 id、置信度，其中框坐标信息为 [xmin, ymin, w, h]，即左上角 x, y 坐标和框的宽和高。

batch_predict

```
batch_predict(self, img_file_list, transforms=None, thread_num=2)
```

YOLOv3 模型批量预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 YOLOv3.test_transforms 和 YOLOv3.eval_transforms 中。如未在训练时定义 eval_dataset，那在调用预测 predict 接口时，用户需要再重新定义 test_transforms 传入给 predict 接口

参数

- **img_file_list** (str|np.ndarray): 对列表（或元组）中的图像同时进行预测，列表中的元素是预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.det.transforms): 数据预处理操作。
- **thread_num** (int): 并发执行各图像预处理时的线程数。

返回值

- **list**: 每个元素都为列表，表示各图像的预测结果。在各图像的预测结果列表中，每个元素均为一个 dict，key 包括'bbox'，'category'，'category_id'，'score'，分别表示每个预测目标的框坐标信息、类别、类别 id、置信度，其中框坐标信息为 [xmin, ymin, w, h]，即左上角 x, y 坐标和框的宽和高。

paddlex.det.FasterRCNN

```
paddlex.det.FasterRCNN(num_classes=81, backbone='ResNet50', with_fpn=True, aspect_
↪ratios=[0.5, 1.0, 2.0], anchor_sizes=[32, 64, 128, 256, 512])
```

构建 FasterRCNN 检测器。**注意在 FasterRCNN 中，num_classes 需要设置为类别数 + 背景类，如目标包括 human、dog 两种，则 num_classes 需设为 3，多的一种为背景 background 类别**

参数

- **num_classes** (int): 包含了背景类的类别数。默认为 81。
- **backbone** (str): FasterRCNN 的 backbone 网络，取值范围为 [‘ResNet18’，‘ResNet50’，‘ResNet50_vd’，‘ResNet101’，‘ResNet101_vd’，‘HRNet_W18’]。默认为’ ResNet50’。
- **with_fpn** (bool): 是否使用 FPN 结构。默认为 True。
- **aspect_ratios** (list): 生成 anchor 高宽比的可选值。默认为 [0.5, 1.0, 2.0]。
- **anchor_sizes** (list): 生成 anchor 大小的可选值。默认为 [32, 64, 128, 256, 512]。

train

```
train(self, num_epochs, train_dataset, train_batch_size=2, eval_dataset=None, save_
↪interval_epochs=1, log_interval_steps=2,save_dir='output', pretrain_weights='IMAGENET',
↪ optimizer=None, learning_rate=0.0025, warmup_steps=500, warmup_start_lr=1.0/1200, lr_
↪decay_epochs=[8, 11], lr_decay_gamma=0.1, metric=None, use_vdl=False, early_stop=False,
↪ early_stop_patience=5, resume_checkpoint=None)
```

FasterRCNN 模型的训练接口，函数内置了 piecewise 学习率衰减策略和 momentum 优化器。

参数

- **num_epochs** (int): 训练迭代轮数。
- **train_dataset** (paddlex.datasets): 训练数据读取器。
- **train_batch_size** (int): 训练数据 batch 大小。目前检测仅支持单卡评估，训练数据 batch 大小与显卡数量之商为验证数据 batch 大小。默认为 2。
- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **save_interval_epochs** (int): 模型保存间隔（单位：迭代轮数）。默认为 1。
- **log_interval_steps** (int): 训练日志输出间隔（单位：迭代次数）。默认为 2。
- **save_dir** (str): 模型保存路径。默认值为’ output’。
- **pretrain_weights** (str): 若指定为路径时，则加载路径下预训练模型；若为字符串’ IMAGENET’，则自动下载在 ImageNet 图片数据上预训练的模型权重；若为字符串’ COCO’，则自动下载在 COCO 数据集上预训练的模型权重（注意：暂未提供 ResNet18 的 COCO 预训练模型）；为 None，则不使用预训练模型。默认为 None。
- **optimizer** (paddle.fluid.optimizer): 优化器。当该参数为 None 时，使用默认优化器：fluid.layers.piecewise_decay 衰减策略，fluid.optimizer.Momentum 优化方法。

- **learning_rate** (float): 默认优化器的初始学习率。默认为 0.0025。
- **warmup_steps** (int): 默认优化器进行 warmup 过程的步数。默认为 500。
- **warmup_start_lr** (int): 默认优化器 warmup 的起始学习率。默认为 1.0/1200。
- **lr_decay_epochs** (list): 默认优化器的学习率衰减轮数。默认为 [8, 11]。
- **lr_decay_gamma** (float): 默认优化器的学习率衰减率。默认为 0.1。
- **metric** (bool): 训练过程中评估的方式，取值范围为 [‘COCO’，‘VOC’]。默认值为 None。
- **use_vdl** (bool): 是否使用 VisualDL 进行可视化。默认值为 False。
- **early_stop** (float): 是否使用提前终止训练策略。默认值为 False。
- **early_stop_patience** (int): 当使用提前终止训练策略时，如果验证集精度在 `early_stop_patience` 个 epoch 内连续下降或持平，则终止训练。默认值为 5。
- **resume_checkpoint** (str): 恢复训练时指定上次训练保存的模型路径。若为 None，则不会恢复训练。默认值为 None。

evaluate

```
evaluate(self, eval_dataset, batch_size=1, epoch_id=None, metric=None, return_
↪details=False)
```

FasterRCNN 模型的评估接口，模型评估后会返回在验证集上的指标 box_map(metric 指定为’VOC’时) 或 box_mmap(metric 指定为 COCO 时)。

参数

- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **batch_size** (int): 验证数据批大小。默认为 1。当前只支持设置为 1。
- **epoch_id** (int): 当前评估模型所在的训练轮数。
- **metric** (bool): 训练过程中评估的方式，取值范围为 [‘COCO’，‘VOC’]。默认为 None，根据用户传入的 Dataset 自动选择，如为 VOCDetection，则 `metric` 为’VOC’；如为 COCODetection，则 `metric` 为’COCO’。
- **return_details** (bool): 是否返回详细信息。默认值为 False。

返回值

- **tuple** (metrics, eval_details) | **dict** (metrics): 当 `return_details` 为 True 时，返回 (metrics, eval_details)，当 `return_details` 为 False 时，返回 metrics。metrics 为 dict，包含关键字：’bbox_mmap’或者’bbox_map‘，分别表示平均准确率平均值在各个 IoU 阈值下的结果取平均值的结果（mmAP）、平均准确率平均值（mAP）。eval_details 为 dict，包含关键字：’bbox’，对应元素预测结果列表，每个预测结果由图像 id、预测框类别 id、预测框坐标、预测框得分；’gt‘：真实标注框相关信息。

predict

```
predict(self, img_file, transforms=None)
```

FasterRCNN 模型预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 `FasterRCNN.test_transforms` 和 `FasterRCNN.eval_transforms` 中。如未在训练时定义 eval_dataset，那在调用预测 `predict` 接口时，用户需要再重新定义 test_transforms 传入给 `predict` 接口。

参数

- **img_file** (str|np.ndarray): 预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.det.transforms): 数据预处理操作。

返回值

- **list**: 预测结果列表，列表中每个元素均为一个 dict，key 包括’bbox’，‘category’，‘category_id’，‘score’，分别表示每个预测目标的框坐标信息、类别、类别 id、置信度，其中框坐标信息为 [xmin, ymin, w, h]，即左上角 x, y 坐标和框的宽和高。

batch_predict

```
batch_predict(self, img_file_list, transforms=None, thread_num=2)
```

FasterRCNN 模型批量预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 `FasterRCNN.test_transforms` 和 `FasterRCNN.eval_transforms` 中。如未在训练时定义 eval_dataset，那在调用预测 `predict` 接口时，用户需要再重新定义 test_transforms 传入给 `predict` 接口。

参数

- **img_file_list** (list|tuple): 对列表（或元组）中的图像同时进行预测，列表中的元素是预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.det.transforms): 数据预处理操作。
- **thread_num** (int): 并发执行各图像预处理时的线程数。

返回值

- **list**: 每个元素都为列表，表示各图像的预测结果。在各图像的预测结果列表中，每个元素均为一个 dict，key 包括’bbox’，‘category’，‘category_id’，‘score’，分别表示每个预测目标的框坐标信息、类别、类别 id、置信度，其中框坐标信息为 [xmin, ymin, w, h]，即左上角 x, y 坐标和框的宽和高。

8.3.3 Instance Segmentation

MaskRCNN

```
paddlex.det.MaskRCNN(num_classes=81, backbone='ResNet50', with_fpn=True, aspect_
↪ratios=[0.5, 1.0, 2.0], anchor_sizes=[32, 64, 128, 256, 512])
```

构建 MaskRCNN 检测器。**注意在 MaskRCNN 中，num_classes 需要设置为类别数 + 背景类，如目标包括 human、dog 两种，则 num_classes 需设为 3，多的一种为背景 background 类别**

参数

- **num_classes** (int): 包含了背景类的类别数。默认为 81。
- **backbone** (str): MaskRCNN 的 backbone 网络，取值范围为 [‘ResNet18’，‘ResNet50’，‘ResNet50_vd’，‘ResNet101’，‘ResNet101_vd’，‘HRNet_W18’]。默认为’ ResNet50’。
- **with_fpn** (bool): 是否使用 FPN 结构。默认为 True。
- **aspect_ratios** (list): 生成 anchor 高宽比的可选值。默认为 [0.5, 1.0, 2.0]。
- **anchor_sizes** (list): 生成 anchor 大小的可选值。默认为 [32, 64, 128, 256, 512]。

train

```
train(self, num_epochs, train_dataset, train_batch_size=1, eval_dataset=None, save_
↪interval_epochs=1, log_interval_steps=20, save_dir='output', pretrain_weights='IMAGENET
↪', optimizer=None, learning_rate=1.0/800, warmup_steps=500, warmup_start_lr=1.0 / 2400,
↪ lr_decay_epochs=[8, 11], lr_decay_gamma=0.1, metric=None, use_vdl=False, early_
↪stop=False, early_stop_patience=5, resume_checkpoint=None)
```

MaskRCNN 模型的训练接口，函数内置了 piecewise 学习率衰减策略和 momentum 优化器。

参数

- **num_epochs** (int): 训练迭代轮数。
- **train_dataset** (paddlex.datasets): 训练数据读取器。
- **train_batch_size** (int): 训练数据 batch 大小。目前检测仅支持单卡评估，训练数据 batch 大小与显卡数量之商为验证数据 batch 大小。默认为 1。
- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **save_interval_epochs** (int): 模型保存间隔（单位：迭代轮数）。默认为 1。
- **log_interval_steps** (int): 训练日志输出间隔（单位：迭代次数）。默认为 2。
- **save_dir** (str): 模型保存路径。默认值为’ output’。

- **pretrain_weights** (str): 若指定为路径时，则加载路径下预训练模型；若为字符串’IMAGENET’, 则自动下载在 ImageNet 图片数据上预训练的模型权重；若为字符串’COCO’, 则自动下载在 COCO 数据集上预训练的模型权重（注意：暂未提供 ResNet18 和 HRNet_W18 的 COCO 预训练模型）；若为 None，则不使用预训练模型。默认为 None。
- **optimizer** (paddle.fluid.optimizer): 优化器。当该参数为 None 时，使用默认优化器：fluid.layers.piecewise_decay 衰减策略，fluid.optimizer.Momentum 优化方法。
- **learning_rate** (float): 默认优化器的初始学习率。默认为 0.00125。
- **warmup_steps** (int): 默认优化器进行 warmup 过程的步数。默认为 500。
- **warmup_start_lr** (int): 默认优化器 warmup 的起始学习率。默认为 1.0/2400。
- **lr_decay_epochs** (list): 默认优化器的学习率衰减轮数。默认为 [8, 11]。
- **lr_decay_gamma** (float): 默认优化器的学习率衰减率。默认为 0.1。
- **metric** (bool): 训练过程中评估的方式，取值范围为 [‘COCO’，‘VOC’]。默认值为 None。
- **use_vdl** (bool): 是否使用 VisualDL 进行可视化。默认值为 False。
- **early_stop** (float): 是否使用提前终止训练策略。默认值为 False。
- **early_stop_patience** (int): 当使用提前终止训练策略时，如果验证集精度在 `early_stop_patience` 个 epoch 内连续下降或持平，则终止训练。默认值为 5。
- **resume_checkpoint** (str): 恢复训练时指定上次训练保存的模型路径。若为 None，则不会恢复训练。默认值为 None。

evaluate

```
evaluate(self, eval_dataset, batch_size=1, epoch_id=None, metric=None, return_
↪details=False)
```

MaskRCNN 模型的评估接口，模型评估后会返回在验证集上的指标 box_mmap(metric 指定为 COCO 时) 和相应的 seg_mmap。

参数

- **eval_dataset** (paddlex.datasets): 验证数据读取器。
- **batch_size** (int): 验证数据批大小。默认为 1。当前只支持设置为 1。
- **epoch_id** (int): 当前评估模型所在的训练轮数。
- **metric** (bool): 训练过程中评估的方式，取值范围为 [‘COCO’，‘VOC’]。默认为 None，根据用户传入的 Dataset 自动选择，如为 VOCDetection，则 `metric` 为’VOC’；如为 COCODetection，则 `metric` 为’COCO’。
- **return_details** (bool): 是否返回详细信息。默认值为 False。

返回值

- **tuple** (metrics, eval_details) | **dict** (metrics): 当 `return_details` 为 True 时，返回 (metrics, eval_details)，当 return_details 为 False 时，返回 metrics。metrics 为 dict，包含关键字：’ bbox_mmap’ 和’ segm_mmap’ 或者’ bbox_map ‘和’ segm_map’，分别表示预测框和分割区域平均准确率平均值在各个 IoU 阈值下的结果取平均值的结果（mmAP）、平均准确率平均值（mAP）。eval_details 为 dict，包含关键字：’ bbox’，对应元素预测框结果列表，每个预测结果由图像 id、预测框类别 id、预测框坐标、预测框得分；’ mask’，对应元素预测区域结果列表，每个预测结果由图像 id、预测区域类别 id、预测区域坐标、预测区域得分；’ gt ‘：真实标注框和标注区域相关信息。

predict

```
predict(self, img_file, transforms=None)
```

MaskRCNN 模型预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 FasterRCNN.test_transforms 和 FasterRCNN.eval_transforms 中。如未在训练时定义 eval_dataset，那在调用预测 predict 接口时，用户需要再重新定义 test_transforms 传入给 predict 接口。

参数

- **img_file** (str|np.ndarray): 预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.det.transforms): 数据预处理操作。

返回值

- **list**: 预测结果列表，列表中每个元素均为一个 dict，key’ bbox’，‘mask’，‘category’，‘category_id’，‘score’，分别表示每个预测目标的框坐标信息、Mask 信息，类别、类别 id、置信度。其中框坐标信息为 [xmin, ymin, w, h]，即左上角 x, y 坐标和框的宽和高。Mask 信息为原图大小的二值图，1 表示像素点属于预测类别，0 表示像素点是背景。

batch_predict

```
batch_predict(self, img_file_list, transforms=None, thread_num=2)
```

MaskRCNN 模型批量预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 FasterRCNN.test_transforms 和 FasterRCNN.eval_transforms 中。如未在训练时定义 eval_dataset，那在调用预测 predict 接口时，用户需要再重新定义 test_transforms 传入给 predict 接口。

参数

- **img_file_list** (list|tuple): 对列表（或元组）中的图像同时进行预测，列表中的元素可以是预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。

- **transforms** (paddlex.det.transforms): 数据预处理操作。
- **thread_num** (int): 并发执行各图像预处理时的线程数。

返回值

- **list**: 每个元素都为列表，表示各图像的预测结果。在各图像的预测结果列表中，每个元素均为一个 dict，key’ bbox’，‘mask’，‘category’，‘category_id’，‘score’，分别表示每个预测目标的框坐标信息、Mask 信息，类别、类别 id、置信度。其中框坐标信息为 [xmin, ymin, w, h]，即左上角 x, y 坐标和框的宽和高。Mask 信息为原图大小的二值图，1 表示像素点属于预测类别，0 表示像素点是背景。

8.3.4 Semantic Segmentation

paddlex.seg.DeepLabv3p

```
paddlex.seg.DeepLabv3p(num_classes=2, backbone='MobileNetV2_x1.0', output_stride=16,␣
↪aspp_with_sep_conv=True, decoder_use_sep_conv=True, encoder_with_aspp=True, enable_
↪decoder=True, use_bce_loss=False, use_dice_loss=False, class_weight=None, ignore_
↪index=255)
```

构建 DeepLabv3p 分割器。

参数

- **num_classes** (int): 类别数。
- **backbone** (str): DeepLabv3+ 的 backbone 网络，实现特征图的计算，取值范围为 [‘Xception65’，‘Xception41’，‘MobileNetV2_x0.25’，‘MobileNetV2_x0.5’，‘MobileNetV2_x1.0’，‘MobileNetV2_x1.5’，‘MobileNetV2_x2.0’]，默认值为’ MobileNetV2_x1.0’。
- **output_stride** (int): backbone 输出特征图相对于输入的下采样倍数，一般取值为 8 或 16。默认 16。
- **aspp_with_sep_conv** (bool): decoder 模块是否采用 separable convolutions。默认 True。
- **decoder_use_sep_conv** (bool): decoder 模块是否采用 separable convolutions。默认 True。
- **encoder_with_aspp** (bool): 是否在 encoder 阶段采用 aspp 模块。默认 True。
- **enable_decoder** (bool): 是否使用 decoder 模块。默认 True。
- **use_bce_loss** (bool): 是否使用 bce loss 作为网络的损失函数，只能用于两类分割。可与 dice loss 同时使用。默认 False。
- **use_dice_loss** (bool): 是否使用 dice loss 作为网络的损失函数，只能用于两类分割，可与 bce loss 同时使用，当 `use_bce_loss` 和 `use_dice_loss` 都为 False 时，使用交叉熵损失

函数。默认 False。

- **class__weight** (list/str): 交叉熵损失函数各类损失的权重。当 `class_weight` 为 list 的时候，长度应为 `num_classes`。当 `class_weight` 为 str 时，weight.lower() 应为’dynamic’，这时会根据每一轮各类像素的比重自行计算相应的权重，每一类的权重为：每类的比例 * num_classes。class_weight 取默认值 None 是，各类的权重 1，即平时使用的交叉熵损失函数。
- **ignore_index** (int): label 上忽略的值，label 为 `ignore_index` 的像素不参与损失函数的计算。默认 255。

train

```
train(self, num_epochs, train_dataset, train_batch_size=2, eval_dataset=None, eval_batch_
↪size=1, save_interval_epochs=1, log_interval_steps=2, save_dir='output', pretrain_
↪weights='IMAGENET', optimizer=None, learning_rate=0.01, lr_decay_power=0.9, use_
↪vdl=False, sensitivities_file=None, eval_metric_loss=0.05, early_stop=False, early_
↪stop_patience=5, resume_checkpoint=None):
```

DeepLabv3p 模型的训练接口，函数内置了 `polynomial` 学习率衰减策略和 `momentum` 优化器。

参数

- **num_epochs** (int): 训练迭代轮数。
- **train_dataset** (paddlex.datasets): 训练数据读取器。
- **train_batch_size** (int): 训练数据 batch 大小。同时作为验证数据 batch 大小。默认 2。
- **eval_dataset** (paddlex.datasets): 评估数据读取器。
- **save_interval_epochs** (int): 模型保存间隔（单位：迭代轮数）。默认为 1。
- **log_interval_steps** (int): 训练日志输出间隔（单位：迭代次数）。默认为 2。
- **save_dir** (str): 模型保存路径。默认’output’
- **pretrain_weights** (str): 若指定为路径时，则加载路径下预训练模型；若为字符串’IMAGENET’，则自动下载在 ImageNet 图片数据上预训练的模型权重；若为字符串’COCO’，则自动下载在 COCO 数据集上预训练的模型权重（注意：暂未提供 Xception41、MobileNetV2_x0.25、MobileNetV2_x0.5、MobileNetV2_x1.5、MobileNetV2_x2.0 的 COCO 预训练模型）；若为字符串’CITYSCAPES’，则自动下载在 CITYSCAPES 数据集上预训练的模型权重（注意：暂未提供 Xception41、MobileNetV2_x0.25、MobileNetV2_x0.5、MobileNetV2_x1.5、MobileNetV2_x2.0 的 CITYSCAPES 预训练模型）；若为 None，则不使用预训练模型。默认’IMAGENET’。
- **optimizer** (paddle.fluid.optimizer): 优化器。当该参数为 None 时，使用默认的优化器：使用 fluid.optimizer.Momentum 优化方法，polynomial 的学习率衰减策略。

- **learning_rate** (float): 默认优化器的初始学习率。默认 0.01。
- **lr_decay_power** (float): 默认优化器学习率衰减指数。默认 0.9。
- **use_vdl** (bool): 是否使用 VisualDL 进行可视化。默认 False。
- **sensitivities_file** (str): 若指定为路径时，则加载路径下敏感度信息进行裁剪；若为字符串’DEFAULT’，则自动下载在 Cityscapes 图片数据上获得的敏感度信息进行裁剪；若为 None，则不进行裁剪。默认为 None。
- **eval_metric_loss** (float): 可容忍的精度损失。默认为 0.05。
- **early_stop** (bool): 是否使用提前终止训练策略。默认值为 False。
- **early_stop_patience** (int): 当使用提前终止训练策略时，如果验证集精度在 `early_stop_patience` 个 epoch 内连续下降或持平，则终止训练。默认值为 5。
- **resume_checkpoint** (str): 恢复训练时指定上次训练保存的模型路径。若为 None，则不会恢复训练。默认值为 None。

evaluate

```
evaluate(self, eval_dataset, batch_size=1, epoch_id=None, return_details=False):
```

DeepLabv3p 模型评估接口。

参数

- **eval_dataset** (paddlex.datasets): 评估数据读取器。
- **batch_size** (int): 评估时的 batch 大小。默认 1。
- **epoch_id** (int): 当前评估模型所在的训练轮数。
- **return_details** (bool): 是否返回详细信息。默认 False。

返回值

- **dict**: 当 `return_details` 为 False 时，返回 dict。包含关键字：’miou’、’category_iou’、’macc’、‘category_acc’ 和’kappa’，分别表示平均 iou、各类别 iou、平均准确率、各类别准确率和 kappa 系数。
- **tuple** (metrics, eval_details)：当 `return_details` 为 True 时，增加返回 dict (eval_details)，包含关键字：’confusion_matrix’，表示评估的混淆矩阵。

predict

```
predict(self, img_file, transforms=None):
```

DeepLabv3p 模型预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 `DeepLabv3p.test_transforms` 和 `DeepLabv3p.eval_transforms` 中。如未在训练时定义 eval_dataset，那在调用预测 `predict` 接口时，用户需要再重新定义 test_transforms 传入给 `predict` 接口。

参数

- **img_file** (str|np.ndarray): 预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.seg.transforms): 数据预处理操作。
- **thread_num** (int): 并发执行各图像预处理时的线程数。**返回值**
- **dict**: 包含关键字’label_map’和’score_map’，‘label_map’存储预测结果灰度图，像素值表示对应的类别，’score_map’存储各类别的概率，shape=(h, w, num_classes)。

batch_predict

```
batch_predict(self, img_file_list, transforms=None):
```

DeepLabv3p 模型批量预测接口。需要注意的是，只有在训练过程中定义了 eval_dataset，模型在保存时才会将预测时的图像处理流程保存在 `DeepLabv3p.test_transforms` 和 `DeepLabv3p.eval_transforms` 中。如未在训练时定义 eval_dataset，那在调用预测 `predict` 接口时，用户需要再重新定义 test_transforms 传入给 `predict` 接口。

参数

- **img_file_list** (list|tuple): 对列表（或元组）中的图像同时进行预测，列表中的元素可以是预测图像路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **transforms** (paddlex.seg.transforms): 数据预处理操作。

返回值

- **dict**: 每个元素都为列表，表示各图像的预测结果。各图像的预测结果用字典表示，包含关键字’label_map’和’score_map’，‘label_map’存储预测结果灰度图，像素值表示对应的类别，’score_map’存储各类别的概率，shape=(h, w, num_classes)。

paddlex.seg.UNet

```
paddlex.seg.UNet(num_classes=2, upsample_mode='bilinear', use_bce_loss=False, use_dice_
↪loss=False, class_weight=None, ignore_index=255)
```

构建 UNet 分割器。

参数

- **num_classes** (int): 类别数。

- **upsample_mode** (str): UNet decode 时采用的上采样方式，取值为’bilinear’时利用双线行差值进行上菜样，当输入其他选项时则利用反卷积进行上菜样，默认为’bilinear’。
- **use_bce_loss** (bool): 是否使用 bce loss 作为网络的损失函数，只能用于两类分割。可与 dice loss 同时使用。默认 False。
- **use_dice_loss** (bool): 是否使用 dice loss 作为网络的损失函数，只能用于两类分割，可与 bce loss 同时使用。当 use_bce_loss 和 use_dice_loss 都为 False 时，使用交叉熵损失函数。默认 False。
- **class_weight** (list/str): 交叉熵损失函数各类损失的权重。当 `class_weight` 为 list 的时候，长度应为 `num_classes`。当 `class_weight` 为 str 时，weight.lower() 应为’dynamic’，这时会根据每一轮各类像素的比重自行计算相应的权重，每一类的权重为：每类的比例 * num_classes。class_weight 取默认值 None 是，各类的权重 1，即平时使用的交叉熵损失函数。
- **ignore_index** (int): label 上忽略的值，label 为 `ignore_index` 的像素不参与损失函数的计算。默认 255。

- train 训练接口说明同*DeepLabv3p 模型 train 接口*
- evaluate 评估接口说明同*DeepLabv3p 模型 evaluate 接口*
- predict 预测接口说明同*DeepLabv3p 模型 predict 接口*
- batch_predict 批量预测接口说明同*DeepLabv3p 模型 predict 接口*

paddlex.seg.HRNet

```
paddlex.seg.HRNet(num_classes=2, width=18, use_bce_loss=False, use_dice_loss=False,␣
↪class_weight=None, ignore_index=255)
```

构建 HRNet 分割器。

参数

- **num_classes** (int): 类别数。
- **width** (int|str): 高分辨率分支中特征层的通道数量。默认值为 18。可选择取值为 [18, 30, 32, 40, 44, 48, 60, 64, ‘18_small_v1’]。’18_small_v1’是 18 的轻量级版本。
- **use_bce_loss** (bool): 是否使用 bce loss 作为网络的损失函数，只能用于两类分割。可与 dice loss 同时使用。默认 False。
- **use_dice_loss** (bool): 是否使用 dice loss 作为网络的损失函数，只能用于两类分割，可与 bce loss 同时使用。当 use_bce_loss 和 use_dice_loss 都为 False 时，使用交叉熵损失函数。默认 False。
- **class_weight** (list|str): 交叉熵损失函数各类损失的权重。当 `class_weight` 为 list 的时候，长度应为 `num_classes`。当 `class_weight` 为 str 时，weight.lower() 应为’dynamic’，

这时会根据每一轮各类像素的比重自行计算相应的权重，每一类的权重为：每类的比例 * num_classes。class_weight 取默认值 None 是，各类的权重 1，即平时使用的交叉熵损失函数。

- **ignore_index** (int): label 上忽略的值，label 为 `ignore_index` 的像素不参与损失函数的计算。默认 255。

- train 训练接口说明同*DeepLabv3p 模型 train 接口*
- evaluate 评估接口说明同*DeepLabv3p 模型 evaluate 接口*
- predict 预测接口说明同*DeepLabv3p 模型 predict 接口*
- batch_predict 批量预测接口说明同*DeepLabv3p 模型 predict 接口*

paddlex.seg.FastSCNN

```
paddlex.seg.FastSCNN(num_classes=2, use_bce_loss=False, use_dice_loss=False, class_
↪weight=None, ignore_index=255, multi_loss_weight=[1.0])
```

构建 FastSCNN 分割器。

参数

- **num_classes** (int): 类别数。
- **use_bce_loss** (bool): 是否使用 bce loss 作为网络的损失函数，只能用于两类分割。可与 dice loss 同时使用。默认 False。
- **use_dice_loss** (bool): 是否使用 dice loss 作为网络的损失函数，只能用于两类分割，可与 bce loss 同时使用。当 use_bce_loss 和 use_dice_loss 都为 False 时，使用交叉熵损失函数。默认 False。
- **class_weight** (list/str): 交叉熵损失函数各类损失的权重。当 `class_weight` 为 list 的时候，长度应为 `num_classes`。当 `class_weight` 为 str 时，weight.lower() 应为’dynamic’，这时会根据每一轮各类像素的比重自行计算相应的权重，每一类的权重为：每类的比例 * num_classes。class_weight 取默认值 None 是，各类的权重 1，即平时使用的交叉熵损失函数。
- **ignore_index** (int): label 上忽略的值，label 为 `ignore_index` 的像素不参与损失函数的计算。默认 255。
- **multi_loss_weight** (list): 多分支上的 loss 权重。默认计算一个分支上的 loss，即默认值为 [1.0]。也支持计算两个分支或三个分支上的 loss，权重按 [fusion_branch_weight, higher_branch_weight, lower_branch_weight] 排列，fusion_branch_weight 为空间细节分支和全局上下文分支融合后的分支上的 loss 权重，higher_branch_weight 为空间细节分支上的 loss 权重，lower_branch_weight 为全局上下文分支上的 loss 权重，若 higher_branch_weight 和 lower_branch_weight 未设置则不会计算这两个分支上的 loss。

- train 训练接口说明同*DeepLabv3p 模型 train 接口*
- evaluate 评估接口说明同*DeepLabv3p 模型 evaluate 接口*
- predict 预测接口说明同*DeepLabv3p 模型 predict 接口*
- batch_predict 批量预测接口说明同*DeepLabv3p 模型 predict 接口*

8.4 模型压缩

8.4.1 paddlex.slim.cal_params_sensitivities

计算参数敏感度

```
paddlex.slim.cal_params_sensitivities(model, save_file, eval_dataset, batch_size=8)
```

计算模型中可裁剪参数在验证集上的敏感度，并将敏感度信息保存至文件 `save_file`

1. 获取模型中可裁剪卷积 Kernel 的名称。
2. 计算每个可裁剪卷积 Kernel 不同裁剪率下的敏感度。【注意】卷积的敏感度是指在不同裁剪率下评估数据集预测精度的损失，通过得到的敏感度，可以决定最终模型需要裁剪的参数列表和各裁剪参数对应的裁剪率。查看使用示例

参数

- **model** (paddlex.cls.models/paddlex.det.models/paddlex.seg.models): paddlex 加载的模型。
- **save_file** (str): 计算的得到的 sensetives 文件存储路径。
- **eval_dataset** (paddlex.datasets): 评估数据集的读取器。
- **batch_size** (int): 评估时的 batch_size 大小。

8.4.2 paddlex.slim.export_quant_model

导出量化模型

```
paddlex.slim.export_quant_model(model, test_dataset, batch_size=2, batch_num=10, save_
↪dir='./quant_model', cache_dir='./temp')
```

导出量化模型，该接口实现了 Post Quantization 量化方式，需要传入测试数据集，并设定 `batch_size` 和 `batch_num`，模型会以 `batch_size` 的大小计算 `batch_num` 批样本数据，并以这些样本数据的计算结果为统计信息进行模型量化。

参数

- **model**(paddlex.cls.models/paddlex.det.models/paddlex.seg.models): paddlex 加载的模型。
- **test_dataset**(paddlex.dataset): 测试数据集
- **batch_size**(int): 进行前向计算时的批数据大小
- **batch_num**(int): 进行向前计算时批数据数量
- **save_dir**(str): 量化后模型的保存目录
- **cache_dir**(str): 量化过程中的统计数据临时存储目录

使用示例

点击下载如下示例中的模型，数据集

```
import paddlex as pdx
model = pdx.load_model('vegetables_mobilenet')
test_dataset = pdx.datasets.ImageNet(
                    data_dir='vegetables_cls',
                    file_list='vegetables_cls/train_list.txt',
                    label_list='vegetables_cls/labels.txt',
                    transforms=model.eval_transforms)
pdx.slim.export_quant_model(model, test_dataset, save_dir='./quant_mobilenet')
```

8.5 预测结果可视化

PaddleX 提供了一系列模型预测和结果分析的可视化函数。

8.5.1 paddlex.det.visualize

目标检测/实例分割预测结果可视化

```
paddlex.det.visualize(image, result, threshold=0.5, save_dir='./')
```

将目标检测/实例分割模型预测得到的 Box 框和 Mask 在原图上进行可视化。

参数

- **image** (str|np.ndarray): 原图文件路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **result** (str): 模型预测结果。

- **threshold**(float): score 阈值，将 Box 置信度低于该阈值的框过滤不进行可视化。默认 0.5
- **save_dir**(str): 可视化结果保存路径。若为 None, 则表示不保存, 该函数将可视化的结果以 np.ndarray 的形式返回；若设为目录路径，则将可视化结果保存至该目录下。默认值为’./’。

使用示例

点击下载如下示例中的模型

```
import paddlex as pdx
model = pdx.load_model('xiaoduxiong_epoch_12')
result = model.predict('./xiaoduxiong_epoch_12/xiaoduxiong.jpeg')
pdx.det.visualize('./xiaoduxiong_epoch_12/xiaoduxiong.jpeg', result, save_dir='./')
# 预测结果保存在./visualize_xiaoduxiong.jpeg
```

8.5.2 paddlex.seg.visualize

语义分割模型预测结果可视化

```
paddlex.seg.visualize(image, result, weight=0.6, save_dir='./')
```

将语义分割模型预测得到的 Mask 在原图上进行可视化。

参数

- **image** (str|np.ndarray): 原图文件路径或 numpy 数组 (HWC 排列，BGR 格式)。
- **result** (str): 模型预测结果。
- **weight**(float): mask 可视化结果与原图权重因子，weight 表示原图的权重。默认 0.6。
- **save_dir**(str): 可视化结果保存路径。若为 None, 则表示不保存, 该函数将可视化的结果以 np.ndarray 的形式返回；若设为目录路径，则将可视化结果保存至该目录下。默认值为’./’。

使用示例

点击下载如下示例中的模型和测试图片

```
import paddlex as pdx
model = pdx.load_model('cityscape_deeplab')
result = model.predict('city.png')
pdx.det.visualize('city.png', result, save_dir='./')
# 预测结果保存在./visualize_city.png
```

8.5.3 paddlex.det.draw_pr_curve

目标检测/实例分割准确率-召回率可视化

```
paddlex.det.draw_pr_curve(eval_details_file=None, gt=None, pred_bbox=None, pred_
↪mask=None, iou_thresh=0.5, save_dir='./')
```

将目标检测/实例分割模型评估结果中各个类别的准确率和召回率的对应关系进行可视化，同时可视化召回率和置信度阈值的对应关系。

注：PaddleX 在训练过程中保存的模型目录中，均包含 `eval_result.json` 文件，可将此文件路径传给 `eval_details_file` 参数，设定 `iou_threshold` 即可得到对应模型在验证集上的 PR 曲线图。

参数

- **eval_details_file** (str): 模型评估结果的保存路径，包含真值信息和预测结果。默认值为 None。
- **gt** (list): 数据集的真值信息。默认值为 None。
- **pred_bbox** (list): 模型在数据集上的预测框。默认值为 None。
- **pred_mask** (list): 模型在数据集上的预测 mask。默认值为 None。
- **iou_thresh** (float): 判断预测框或预测 mask 为真阳时的 IoU 阈值。默认值为 0.5。
- **save_dir** (str): 可视化结果保存路径。默认值为'./'。

注意： `eval_details_file` 的优先级更高，只要 `eval_details_file` 不为 None，就会从 `eval_details_file` 提取真值信息和预测结果做分析。当 `eval_details_file` 为 None 时，则用 `gt`、`pred_mask`、`pred_mask` 做分析。

使用示例

点击下载如下示例中的模型和数据集

方式一：分析训练过程中保存的模型文件夹中的评估结果文件 `eval_details.json`，例如模型中的 `eval_details.json`。

```
import paddlex as pdx
eval_details_file = 'insect_epoch_270/eval_details.json'
pdx.det.draw_pr_curve(eval_details_file, save_dir='./insect')
```

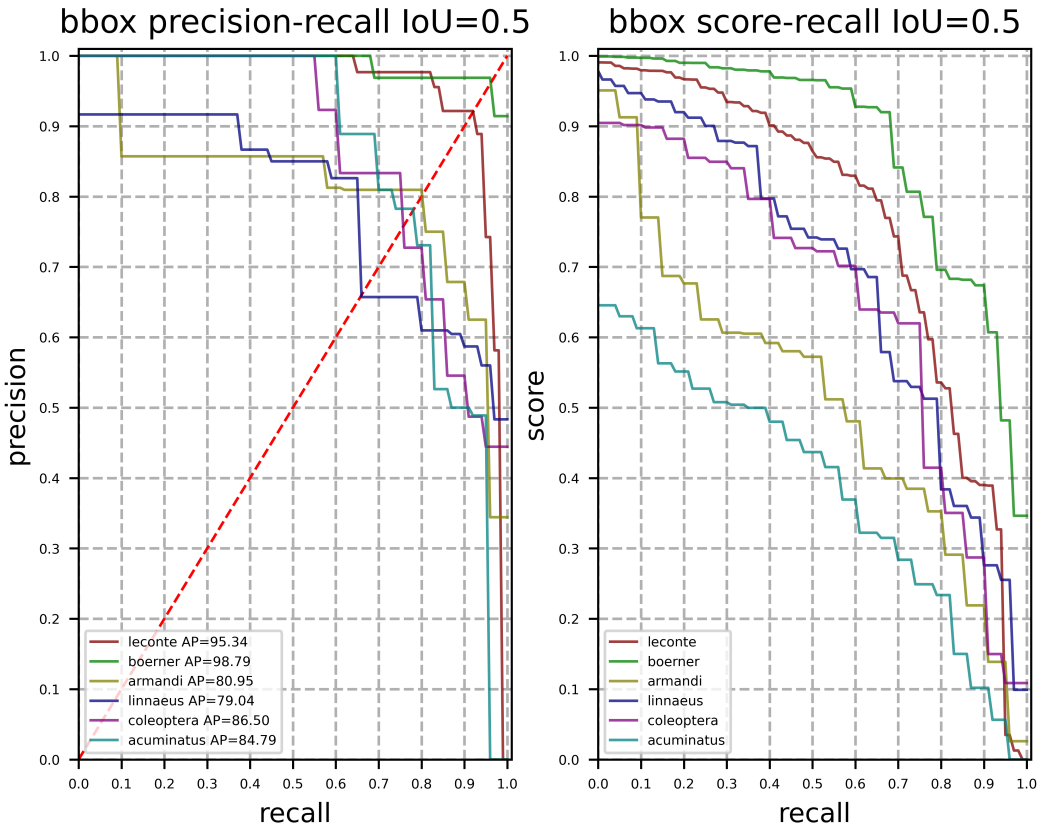
方式二：分析模型评估函数返回的评估结果。

```
import os
# 选择使用 0 号卡
os.environ['CUDA_VISIBLE_DEVICES'] = '0'

from paddlex.det import transforms
import paddlex as pdx

model = pdx.load_model('insect_epoch_270')
eval_dataset = pdx.datasets.VOCDetection(
    data_dir='insect_det',
    file_list='insect_det/val_list.txt',
    label_list='insect_det/labels.txt',
    transforms=model.eval_transforms)
metrics, evaluate_details = model.evaluate(eval_dataset, batch_size=8, return_
↪details=True)
gt = evaluate_details['gt']
bbox = evaluate_details['bbox']
pdx.det.draw_pr_curve(gt=gt, pred_bbox=bbox, save_dir='./insect')
```

预测框的各个类别的准确率和召回率的对应关系、召回率和置信度阈值的对应关系可视化如下:



8.5.4 paddlex.slim.visualzie

模型裁剪比例可视化分析

```
paddlex.slim.visualize(model, sensitivities_file)
```

利用此接口，可以分析在不同的 `eval_metric_loss` 参数下，模型被裁剪的比例情况。可视化结果纵轴为 eval_metric_loss 参数值，横轴为对应的模型被裁剪的比例。

参数

- **model** (paddlex.cv.models): 使用 PaddleX 加载的模型。
- **sensitivities_file** (str): 模型各参数在验证集上计算得到的参数敏感度信息文件。

使用示例

点击下载示例中的模型和sensitivities_file

```
import paddlex as pdx
model = pdx.load_model('vegetables_mobilenet')
pdx.slim.visualize(model, 'mobilenetv2.sensitivities', save_dir='./')
# 可视化结果保存在./sensitivities.png
```

8.5.5 paddlex.transforms.visualize

数据预处理/增强过程可视化

```
paddlex.transforms.visualize(dataset,
                             img_count=3,
                             save_dir='vdl_output')
```

对数据预处理/增强中间结果进行可视化。可使用 VisualDL 查看中间结果：

1. VisualDL 启动方式: visualdl –logdir vdl_output –port 8001
2. 浏览器打开 https://0.0.0.0:8001 即可，其中 0.0.0.0 为本机访问，如为远程服务, 改成相应机器 IP

参数

- **dataset** (paddlex.datasets): 数据集读取器。
- **img_count** (int): 需要进行数据预处理/增强的图像数目。默认为 3。
- **save_dir** (str): 日志保存的路径。默认为’vdl_output’。

8.6 模型可解释性

目前 PaddleX 支持对于图像分类的结果以可视化的方式进行解释，支持 LIME 和 NormLIME 两种可解释性算法。

8.6.1 paddlex.interpret.lime

LIME 可解释性结果可视化

```
paddlex.interpret.lime(img_file,
                       model,
                       num_samples=3000,
                       batch_size=50,
                       save_dir='./')
```

使用 LIME 算法将模型预测结果的可解释性可视化。LIME 表示与模型无关的局部可解释性，可以解释任何模型。LIME 的思想是以输入样本为中心，在其附近的空间中进行随机采样，每个采样通过原模型得到新的输出，这样得到一系列的输入和对应的输出，LIME 用一个简单的、可解释的模型（比如线性回归模型）来拟合这个映射关系，得到每个输入维度的权重，以此来解释模型。

注意：可解释性结果可视化目前只支持分类模型。

参数

- **img_file** (str): 预测图像路径。
- **model** (paddlex.cv.models): paddlex 中的模型。
- **num_samples** (int): LIME 用于学习线性模型的采样数，默认为 3000。
- **batch_size** (int): 预测数据 batch 大小，默认为 50。
- **save_dir** (str): 可解释性可视化结果（保存为 png 格式文件）和中间文件存储路径。

使用示例

对预测可解释性结果可视化的过程可参见代码。

8.6.2 paddlex.interpret.normlime

NormLIME 可解释性结果可视化

```
paddlex.interpret.normlime(img_file,
                           model,
                           dataset=None,
                           num_samples=3000,
                           batch_size=50,
                           save_dir='./',
                           normlime_weights_file=None)
```

使用 NormLIME 算法将模型预测结果的可解释性可视化。NormLIME 是利用一定数量的样本来出一个全局的解释。由于 NormLIME 计算量较大，此处采用一种简化的方式：使用一定数量的测试样本（目前默认使用所有测试样本），对每个样本进行特征提取，映射到同一个特征空间；然后以此特征做为输入，以模型输出做为输出，使用线性回归对其进行拟合，得到一个全局的输入和输出的关系。之后，对一测试样本进行解释时，使用 NormLIME 全局的解释，来对 LIME 的结果进行滤波，使最终的可视化结果更加稳定。

注意：可解释性结果可视化目前只支持分类模型。

参数

- **img_file** (str): 预测图像路径。
- **model** (paddlex.cv.models): paddlex 中的模型。
- **dataset** (paddlex.datasets): 数据集读取器，默认为 None。
- **num_samples** (int): LIME 用于学习线性模型的采样数，默认为 3000。
- **batch_size** (int): 预测数据 batch 大小，默认为 50。
- **save_dir** (str): 可解释性可视化结果（保存为 png 格式文件）和中间文件存储路径。
- **normlime_weights_file** (str): NormLIME 初始化文件名，若不存在，则计算一次，保存于该路径；若存在，则直接载入。

注意： dataset‘读取的是一个数据集，该数据集不宜过大，否则计算时间会较长，但应包含所有类别的数据。NormLIME 可解释性结果可视化目前只支持分类模型。

使用示例

对预测可解释性结果可视化的过程可参见代码。

CHAPTER 9

更新日志

v1.1.0 2020.07.12

- 模型更新
- 新增语义分割模型 HRNet、FastSCNN
- 目标检测 FasterRCNN、实例分割 MaskRCNN 新增 backbone HRNet
- 目标检测/实例分割模型新增 COCO 数据集预训练模型
- 集成 X2Paddle，PaddleX 所有分类模型和语义分割模型支持导出为 ONNX 协议
- 模型部署更新
- 模型加密增加支持 Windows 平台
- 新增 Jetson、PaddleLite 模型部署预测方案
- C++ 部署代码新增 batch 批预测，并采用 OpenMP 对预处理进行并行加速
- 新增 2 个 PaddleX 产业案例
- 人像分割案例
- 工业表计读数案例
- 新增数据格式转换功能，LabelMe、精灵标注助手和 EasyData 平台标注的数据转为 PaddleX 支持加载的数据格式
- PaddleX 文档更新，优化文档结构

v1.0.0 2020.05.20

- 增加模型 C++ 部署和 Python 部署代码
- 增加模型加密部署方案
- 增加分类模型的 OpenVINO 部署方案
- 增加模型可解释性的接口

v0.1.8 2020.05.17

- 修复部分代码 Bug
- 新增 EasyData 平台数据标注格式支持
- 支持 imgaug 数据增强库的 pixel-level 算子

CHAPTER 10

附录

10.1 PaddleX 模型库

10.1.1 图像分类模型

表中模型相关指标均为在 ImageNet 数据集上使用 PaddlePaddle Python 预测接口测试得到（测试 GPU 型号为 Nvidia Tesla P40），预测速度为每张图片预测用时（不包括预处理和后处理），表中符号-表示相关指标暂未测试。

10.1.2 目标检测模型

表中模型相关指标均为在 MSCOCO 数据集上使用 PaddlePaddle Python 预测接口测试得到（测试 GPU 型号为 Nvidia Tesla V100 测试得到），表中符号-表示相关指标暂未测试。

10.1.3 实例分割模型

预测时间是在一张 Nvidia Tesla V100 的 GPU 上通过’evaluate()’接口测试 MSCOCO 验证集得到，包括数据加载、网络前向执行和后处理, batch size 是 1，表中符号-表示相关指标暂未测试。

10.1.4 语义分割模型

以下指标均在 MSCOCO 验证集上测试得到，表中符号-表示相关指标暂未测试。

以下指标均在 Cityscapes 验证集上测试得到，表中符号-表示相关指标暂未测试。

10.2 PaddleX 压缩模型库

10.2.1 图像分类

数据集：ImageNet-1000

量化

分类模型 Lite 时延 (ms)

剪裁

PaddleLite 推理耗时说明：

环境：Qualcomm SnapDragon 845 + armv8

速度指标：Thread1/Thread2/Thread4 耗时

10.2.2 目标检测

量化

数据集：COCO2017

剪裁

数据集：Pasacl VOC & COCO2017

PaddleLite 推理耗时说明：

环境：Qualcomm SnapDragon 845 + armv8

速度指标：Thread1/Thread2/Thread4 耗时

10.2.3 语义分割

数据集：Cityscapes

量化

图像分割模型 Lite 时延 (ms), 输入尺寸 769 x 769

剪裁

PaddleLite 推理耗时说明：

环境：Qualcomm SnapDragon 845 + armv8

速度指标：Thread1/Thread2/Thread4 耗时

10.3 PaddleX 指标及日志

PaddleX 在模型训练、评估过程中，都会有相应的日志和指标反馈，本文档用于说明这些日志和指标的含义。

10.3.1 训练通用统计信息

PaddleX 所有模型在训练过程中，输出的日志信息都包含了 6 个通用的统计信息，用于辅助用户进行模型训练，例如**分割模型**的训练日志，如下图所示。

```
[TRAIN] Epoch=4/20, Step=62/66, loss=0.007226, lr=0.008215, time_each_step=0.41s, eta=0:9:44
[TRAIN] Epoch=4/20, Step=64/66, loss=0.012199, lr=0.008201, time_each_step=0.41s, eta=0:9:43
[TRAIN] Epoch=4/20, Step=66/66, loss=0.003387, lr=0.008187, time_each_step=0.41s, eta=0:9:42
[TRAIN] Epoch 4 finished, loss=0.007828, lr=0.008414 .
```

各字段含义如下:

不同模型的日志中除了上述通用字段外，还有其它字段，这些字段含义可见文档后面对各任务模型的描述。

10.3.2 评估通用统计信息

PaddleX 所有模型在训练过程中会根据用户设定的 `save_interval_epochs` 参数，每间隔一定轮数进行评估和保存。例如**分类模型**的评估日志，如下图所示。

```
Start to evaluating(total_samples=240, total_steps=8)...
##################################################################################################
[EVAL] Finished, Epoch=2, acc1=0.258333, acc5=0.7875 .
Model saved in output/mobilenetv2/best_model.
Model saved in output/mobilenetv2/epoch_2.
Current evaluated best model in eval_dataset is epoch_2, acc1=0.25833333333333336
```

上图中第 1 行表明验证数据集中样本数为 240，需要迭代 8 步才能评估完所有验证数据；第 5 行用于表明第 2 轮的模型已经完成保存操作；第 6 行则表明当前保存的模型中，第 2 轮的模型在验证集上指标最优（分类任务看 `acc1`，此时 `acc1` 值为 0.258333），最优模型会保存在 `best_model` 目录中。

10.3.3 分类特有统计信息

训练日志字段

分类任务的训练日志除了通用统计信息外，还包括 `acc1` 和 `acc5` 两个特有字段。

> 注：acck 准确率是针对一张图片进行计算的：把模型在各个类别上的预测得分按从高往低进行排序，取出前 k 个预测类别，若这 k 个预测类别包含了真值类，则认为该图片分类正确。

```
[TRAIN] Epoch=2/10, Step=22/26, loss=0.370639, acc1=0.8125, acc5=1.0, lr=0.025, time_each_step=0.11s, eta=0:1:14
[TRAIN] Epoch=2/10, Step=24/26, loss=1.782637, acc1=0.71875, acc5=1.0, lr=0.025, time_each_step=0.11s, eta=0:1:14
[TRAIN] Epoch=2/10, Step=26/26, loss=0.462437, acc1=0.90625, acc5=1.0, lr=0.025, time_each_step=0.11s, eta=0:1:14
[TRAIN] Epoch 2 finished, loss=1.240256, acc1=0.759615, acc5=0.998798, lr=0.025 .
```

上图中第 1 行中的 `acc1` 表示参与当前迭代步数的训练样本的平均 top1 准确率，值越高代表模型越优；`acc5` 表示参与当前迭代步数的训练样本的平均 top5（若类别数 n 少于 5，则为 topn）准确率，值越高代表模型越优。第 4 行中的 `loss` 表示整个训练集的平均损失函数值，`acc1` 表示整个训练集的平均 top1 准确率，`acc5` 表示整个训练集的平均 top5 准确率。

评估日志字段

```
 Start to evaluating(total_samples=240, total_steps=8)...
################################################################################
 [EVAL] Finished, Epoch=2, acc1=0.258333, acc5=0.7875 .
 Model saved in output/mobilenetv2/best_model.
 Model saved in output/mobilenetv2/epoch_2.
 Current evaluated best model in eval_dataset is epoch_2, acc1=0.25833333333333336
```

上图中第 3 行中的 `acc1` 表示整个验证集的平均 top1 准确率，`acc5` 表示整个验证集的平均 top5 准确率。

10.3.4 检测特有统计信息

训练日志字段

YOLOv3

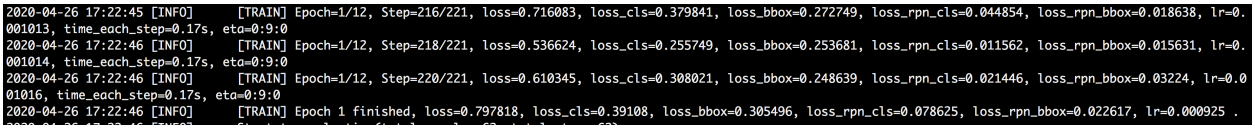
YOLOv3 的训练日志只包括训练通用统计信息（见上文训练通用统计信息）。

```
[TRAIN] Epoch=1/270, Step=204/211, loss=65.632935, lr=2.5e-05, time_each_step=0.41s, eta=7:24:50
[TRAIN] Epoch=1/270, Step=206/211, loss=49.226215, lr=2.6e-05, time_each_step=0.41s, eta=7:22:34
[TRAIN] Epoch=1/270, Step=208/211, loss=66.177971, lr=2.6e-05, time_each_step=0.41s, eta=7:30:9
[TRAIN] Epoch=1/270, Step=210/211, loss=61.262436, lr=2.6e-05, time_each_step=0.42s, eta=7:35:28
[TRAIN] Epoch 1 finished, loss=597.357239, lr=1.3e-05 .
```

上图中第 5 行 `loss` 表示整个训练集的平均损失函数 loss 值。

FasterRCNN

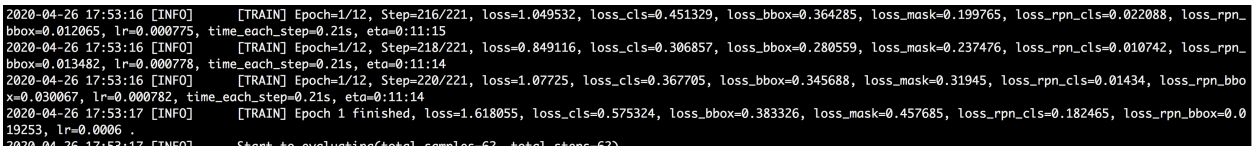
FasterRCNN 的训练日志除了通用统计信息外，还包括 `loss_cls`、`loss_bbox`、`loss_rpn_cls` 和 `loss_rpn_bbox`，这些字段的含义如下:



上图中第 1 行 `loss`, `loss_cls`、`loss_bbox`、`loss_rpn_clss`、`loss_rpn_bbox` 都是参与当前迭代步数的训练样本的损失值，而第 7 行是针整个训练集的损失函数值。

MaskRCNN

MaskRCNN 的训练日志除了通用统计信息外，还包括 `loss_cls`、`loss_bbox`、`loss_mask`、`loss_rpn_cls` 和 `loss_rpn_bbox`，这些字段的含义如下:

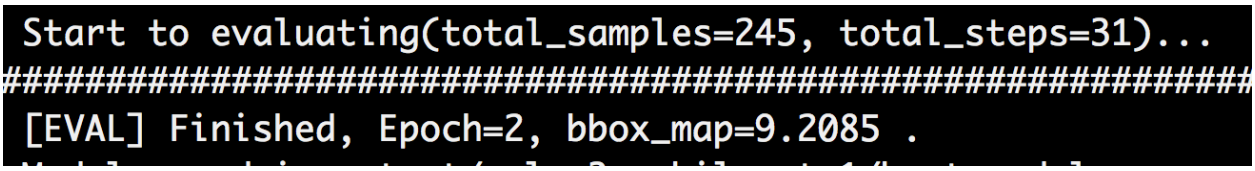


上图中第 1 行 `loss`, `loss_cls`、`loss_bbox`、`loss_mask`、`loss_rpn_clss`、`loss_rpn_bbox` 都是参与当前迭代步数的训练样本的损失值，而第 7 行是针整个训练集的损失函数值。

评估日志字段

检测可以使用两种评估标准：VOC 评估标准和 COCO 评估标准。

VOC 评估标准



> 注: `map` 为平均准确率的平均值，即 IoU(Intersection Over Union) 取 0.5 时各个类别的准确率-召回率曲线下面积的平均值。

上图中第 3 行 `bbox_map` 表示检测任务中整个验证集的平均准确率平均值。

COCO 评估标准

> 注: COCO 评估指标可参见COCO 官网解释。PaddleX 主要反馈 `mmAP`，即 AP at IoU=.50:.05:.95 这项指标，为在各个 IoU 阈值下平均准确率平均值（mAP）的平均值。

COCO 格式的数据集不仅可以用于训练目标检测模型，也可以用于训练实例分割模型。在目标检测中，PaddleX 主要反馈针对检测框的 `bbox_mmAP` 指标；在实例分割中，还包括针对 Mask 的 `seg_mmAP` 指标。如下所示，第一张日志截图为目标检测的评估结果，第二张日志截图为实例分割的评估结果。

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.404
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.862
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.292
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.402
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.406
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.274
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.521
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.532
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.400
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.532
2020-04-26 17:22:55 [INFO]      [EVAL] Finished, Epoch=1, bbox_mmap=0.404178 .
```

上图中红框标注的 bbox_mmap 表示整个验证集的检测框平均准确率平均值。

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.347
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.737
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.255
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.041
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.352
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.241
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.457
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.468
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.440
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.470
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *segm*
DONE (t=0.68s).
Accumulating evaluation results...
DONE (t=0.08s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.466
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.730
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.533
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.005
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.475
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.299
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.579
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.587
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.200
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.593
2020-04-26 17:15:51 [INFO]      [EVAL] Finished, Epoch=1, bbox_mmap=0.347233, segm_mmap=0.466408 .
```

上图中红框标注的 bbox_mmap 和 seg_mmap 分别表示整个验证集的检测框平均准确率平均值、Mask 平均准确率平均值。

10.3.5 分割特有统计信息

训练日志字段

语义分割的训练日志只包括训练通用统计信息（见上文训练通用统计信息）。

```
[TRAIN] Epoch=4/20, Step=62/66, loss=0.007226, lr=0.008215, time_each_step=0.41s, eta=0:9:44
[TRAIN] Epoch=4/20, Step=64/66, loss=0.012199, lr=0.008201, time_each_step=0.41s, eta=0:9:43
[TRAIN] Epoch=4/20, Step=66/66, loss=0.003387, lr=0.008187, time_each_step=0.41s, eta=0:9:42
[TRAIN] Epoch 4 finished, loss=0.007828, lr=0.008414 .
```

评估日志字段

语义分割的评估日志包括了 `miou`、`category_iou`、`macc`、`category_acc`、`kappa`，这些字段的含义如下：

```
2020-04-26 17:58:50 [INFO]      Start to evaluating(total_samples=76, total_steps=19)...
100%|################################################################################################################################################| 19/19 [00:05<00:00,  3.73it/s]
2020-04-26 17:58:55 [INFO]      [EVAL] Finished, Epoch=4, miou=0.901114, categore_iou=[0.9961154 0.8061131], macc=0.996177, category_acc=[0.99748952 0.92140862], kappa=0.89
0705 .
```

10.4 PaddleX 可解释性

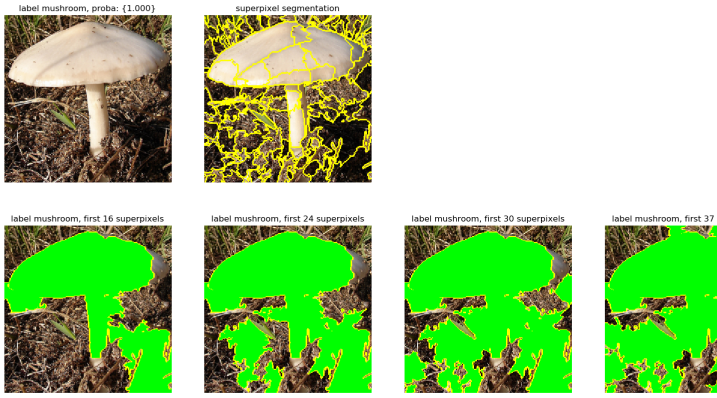
目前深度学习模型普遍存在一个问题，因为使用模型预测还是一个黑盒，几乎无法去感知它的内部工作状态，预测结果的可信度一直遭到质疑。为此，PadlleX 提供了 2 种对图像分类预测结果进行可解释性研究的算法：LIME 和 NormLIME。

10.4.1 LIME

LIME 全称 Local interpretable model-agnostic explanations，表示一种与模型无关的局部可解释性。其实现步骤主要如下：

1. 获取图像的超像素。
2. 以输入样本为中心，在其附近的空间中进行随机采样，每个采样即对对象中的超像素进行随机遮掩（每个采样的权重和该采样与原样本的距离成反比）。
3. 每个采样通过预测模型得到新的输出，这样得到一系列的输入 `X` 和对应的输出 `Y`。
4. 将 `X` 转换为超像素特征 `F`，用一个简单的、可解释的模型 `Model`（这里使用岭回归）来拟合 `F` 和 `Y` 的映射关系。
5. `Model` 将得到 `F` 每个输入维度的权重（每个维度代表一个超像素），以此来解释模型。

LIME 的使用方式可参见代码示例和api 介绍。在使用时，参数中的 `num_samples` 设置尤为重要，其表示上述步骤 2 中的随机采样的个数，若设置过小会影响可解释性结果的稳定性，若设置过大则将在上述步骤 3 耗费较长时间；参数 `batch_size` 则表示在计算上述步骤 3 时，预测的 batch size，若设置过小将在上述步骤 3 耗费较长时间，而上限则根据机器配置决定。



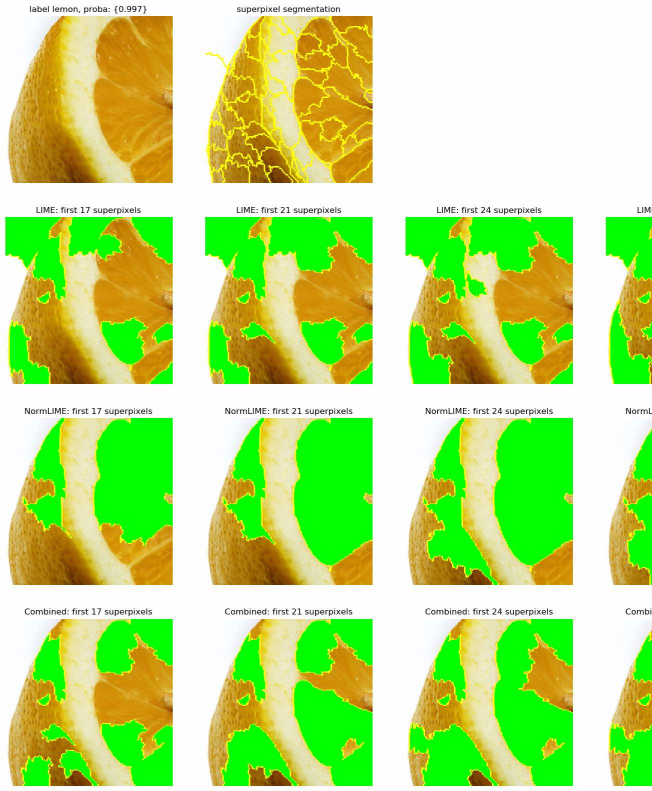
最终 LIME 可解释性算法的可视化结果如下所示：
中绿色区域代表起正向作用的超像素，红色区域代表起反向作用的超像素，”First n superpixels” 代表前 n 个权重比较大的超像素（由上述步骤 5 计算所得结果）。

10.4.2 NormLIME

NormLIME 是在 LIME 上的改进，LIME 的解释是局部性的，是针对当前样本给的特定解释，而 NormLIME 是利用一定数量的样本对当前样本的一个全局性的解释，有一定的降噪效果。其实现步骤如下所示：

1. 下载 Kmeans 模型参数和 ResNet50_vc 网络前三层参数。（ResNet50_vc 的参数是在 ImageNet 上训练所得网络的参数；使用 ImageNet 图像作为数据集，每张图像从 ResNet50_vc 的第三层输出提取对应超象素位置上的平均特征和质心上的特征，训练将得到此处的 Kmeans 模型）
2. 使用测试集中的数据计算 normlime 的权重信息（如无测试集，可用验证集代替）：对每张图像的处理：(1) 获取图像的超像素。(2) 使用 ResNet50_vc 获取第三层特征，针对每个超像素位置，组合质心特征和均值特征 `F`。(3) 把 `F` 作为 Kmeans 模型的输入，计算每个超像素位置的聚类中心。(4) 使用训练好的分类模型，预测该张图像的 `label`。对所有图像的处理：(1) 以每张图像的聚类中心信息组成的向量（若某聚类中心出现在盖章途中设置为 1，反之为 0）为输入，预测的 `label` 为输出，构建逻辑回归函数 `regression_func`。(2) 由 `regression_func` 可获得每个聚类中心不同类别下的权重，并对权重进行归一化。
3. 使用 Kmeans 模型获取需要可视化图像的每个超像素的聚类中心。
4. 对需要可视化的图像的超像素进行随机遮掩构成新的图像。
5. 对每张构造的图像使用预测模型预测 label。
6. 根据 normlime 的权重信息，每个超像素可获不同的权重，选取最高的权重为最终的权重，以此来解释模型。

NormLIME 的使用方式可参见代码示例和api 介绍。在使用时，参数中的 `num_samples` 设置尤为重要，其表示上述步骤 2 中的随机采样的个数，若设置过小会影响可解释性结果的稳定性，若设置过大则将在上述步骤 3 耗费较长时间；参数 `batch_size` 则表示在计算上述步骤 3 时，预测的 batch size，若设置过小将在上述步骤 3 耗费较长时间，而上限则根据机器配置决定；而 `dataset` 则是由测试集或验证集构造的数据。



最终 NormLIME 可解释性算法的可视化结果如下所示：
中绿色区域代表起正向作用的超像素，红色区域代表起反向作用的超像素，" First n superpixels" 代表前 n 个权重比较大的超像素（由上述步骤 5 计算所得结果）。图中最后一行代表把 LIME 和 NormLIME 对应超像素权重相乘的结果。

10.5 训练参数调整

PaddleX 所有训练接口中，内置的参数均为根据单 GPU 卡相应 batch_size 下的较优参数，用户在自己的数据上训练模型，涉及到参数调整时，如无太多参数调优经验，则可参考如下方式

10.5.1 1.Epoch 数的调整

Epoch 数是模型训练过程，迭代的轮数，用户可以设置较大的数值，根据模型迭代过程在验证集上的指标表现，来判断模型是否收敛，进而提前终止训练。此外也可以使用 `train` 接口中的 `early_stop` 策略，模型在训练过程会自动判断模型是否收敛自动中止。

10.5.2 2.batch_size 和 learning_rate

- Batch Size 指模型在训练过程中，一次性处理的样本数量
- 如若使用多卡训练，batch_size 会均分到各张卡上（因此需要让 batch size 整除卡数）
- Batch Size 跟机器的显存/内存高度相关，`batch_size` 越高，所消耗的显存/内存就越高
- PaddleX 在各个 `train` 接口中均配置了默认的 batch size(默认针对单 GPU 卡)，如若训练时提示 GPU 显存不足，则相应调低 BatchSize，如若 GPU 显存高或使用多张 GPU 卡时，可相应调高 BatchSize。
- **如若用户调整 batch size，则也注意需要对应调整其它参数，特别是 train 接口中默认的 learning_rate 值**。如在 YOLOv3 模型中，默认 `train_batch_size` 为 8，`learning_rate` 为 0.000125，当用户将模型在 2 卡机器上训练时，可以将 `train_batch_size` 调整为 16，那么同时 `learning_rate` 也可以对应调整为 0.000125 * 2 = 0.00025

10.5.3 3.warmup_steps 和 warmup_start_lr

在训练模型时，一般都会使用预训练模型，例如检测模型在训练时使用 backbone 在 ImageNet 数据集上的预训练权重。但由于在自行训练时，自己的数据与 ImageNet 数据集存在较大的差异，可能会一开始由于梯度过大使得训练出现问题，因此可以在刚开始训练时，让学习率以一个较小的值，慢慢增长到设定的学习率。因此 `warmup_steps` 和 `warmup_start_lr` 就是这个作用，模型开始训练时，学习率会从 `warmup_start_lr` 开始，在 `warmup_steps` 个 batch 数据迭代后线性增长到设定的学习率。

> 例如 YOLOv3 的 train 接口，默认 `train_batch_size` 为 8，`learning_rate` 为 0.000125，`warmup_steps` 为 1000，`warmup_start_lr` 为 0.0；在此参数配置下表示，模型在启动训练后，在前 1000 个 step(每个 step 表示一个 batch 的数据，也就是 8 个样本) 内，学习率会从 0.0 开始线性增长到设定的 0.000125。

10.5.4 4.lr_decay_epochs 和 lr_decay_gamma

`lr_decay_epochs` 用于让学习率在模型训练后期逐步衰减，它一般是一个 list，如 [6, 8, 10]，表示学习率在第 6 个 epoch 时衰减一次，第 8 个 epoch 时再衰减一次，第 10 个 epoch 时再衰减一次。每次学习率衰减为之前的学习率 *lr_decay_gamma。

> 例如 YOLOv3 的 train 接口，默认 `num_epochs` 为 270,`learning_rate` 为 0.000125，`lr_decay_epochs` 为 [213, 240]，`lr_decay_gamma` 为 0.1；在此参数配置下表示，模型在启动训练后，在前 213 个 epoch 中，训练时使用的学习率为 0.000125，在第 213 至 240 个 epoch 之间，训练使用的学习率为 0.000125x0.1=0.0000125，在 240 个 epoch 之后，使用的学习率为 0.000125x0.1x0.1=0.00000125

10.5.5 5. 参数设定时的约束

根据上述几个参数，可以了解到学习率的变化分为 WarmUp 热身阶段和 Decay 衰减阶段，

- Wamup 热身阶段：随着训练迭代，学习率从较低的值逐渐线性增长至设定的值，以 step 为单位
- Decay 衰减阶段：随着训练迭代，学习率逐步衰减，如每次衰减为之前的 0.1，以 epoch 为单位 step 与 epoch 的关系：1 个 epoch 由多个 step 组成，例如训练样本有 800 张图像，`train_batch_size` 为 8，那么每个 epoch 都要完整用这 800 张图片训一次模型，而每个 epoch 总共包含 800//8 即 100 个 step

在 PaddleX 中，约束 warmup 必须在 Decay 之前结束，因此各参数设置需要满足下面条件

```
warmup_steps <= lr_decay_epochs[0] * num_steps_each_epoch
```

其中 `num_steps_each_epoch` 计算方式如下，

```
num_steps_each_eposh = num_samples_in_train_dataset // train_batch_size
```

因此，如若你在启动训练时，被提示 `warmup_steps should be less than...` 时，即表示需要根据上述公式调整你的参数啦，可以调整 `lr_decay_epochs` 或者是 `warmup_steps`。

10.5.6 6. 如何使用多 GPU 卡进行训练

在 `import paddlex` 前配置环境变量，代码如下

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0' # 使用第 1 张 GPU 卡进行训练
# 注意 paddle 或 paddlex 都需要在设置环境变量后再 import
import paddlex as pdx
```

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '' # 不使用 GPU, 使用 CPU 进行训练
import paddlex as pdx
```

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0,1,3' # 使用第 1、2、4 张 GPU 卡进行训练
import paddlex as pdx
```

10.5.7 相关模型接口

- 图像分类模型 train 接口
- FasterRCNN train 接口
- YOLOv3 train 接口
- MaskRCNN train 接口
- DeepLabv3p train 接口