PaddleX

May 28, 2020

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。

全流程打通

- **数据准备**: 支持 EasyData 智能数据服务平台数据协议 ，通过平台便捷完成智能标注，低质数据清洗工作；同时兼容主流标注工具协议, 助力开发者更快完成数据准备工作。

- **模型训练**: 基于飞桨核心框架集成 PaddleClas ，PaddleDetection ，PaddleSeg 视觉开发套件，丰富的高质量预训练模型，更快实现工业级模型训练。

- **模型调优**: 内置模型可解释性模块、VisualDL 可视化分析组件，提供丰富的信息更好地理解模型，优化模型。

- **多端安全部署**: 内置 PaddleSlim 模型压缩工具和 AES 模型加密 SDK，结合 Paddle Inference 和 Paddle Lite 便捷完成高性能的多端安全部署。

融合产业实践

- 精选飞桨产业实践的成熟模型结构，开放案例实践教程，加速开发者产业落地。

易用易集成

- 统一易用的全流程 API，5 步完成模型训练，10 行代码实现 Python/C++ 高性能部署。

- 提供以 PaddleX 为核心集成的跨平台可视化开发工具 PaddleX-GUI，更低门槛快速体验飞桨深度学习全流程。

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，您可以阅读 PaddleX 的**使用教程**来了解更多模型任务的训练使用方式。本示例同步在 AIStudio 上，可直接在线体验模型训练

1.1 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
```

1.2 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
```

1.3 3. 训练代码开发

PaddleX 的所有模型训练和预测均只涉及到 5 个 API 接口，分别是

- transforms 图像数据处理
- *datasets* 数据集加载

- *models* 模型类型定义
- train 开始训练
- predict 模型预测

在本示例，通过如下 `train.py` 代码进行训练，训练环境为 1 张 Tesla P40 GPU 卡。

1.3.1 3.1 定义 transforms 数据处理流程

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()
])
```

1.3.2 3.2 定义 dataset 加载数据集

定义数据集，`pdx.datasets.ImageNet` 表示读取 ImageNet 格式的分类数据集，更多数据集细节可以查阅数据集格式说明和*ImageNet 接口文档*

```
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)
```

1.3.3 3.3 定义分类模型

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)
```

1.3.4 3.4 定义训练参数

定义好模型后，即可直接调用 `train` 接口，定义训练时的参数，分类模型内置了 `piecewise_decay` 学习率衰减策略，相关参数见分类 train 接口文档。

```
model.train(num_epochs=10,
            train_dataset=train_dataset,
            train_batch_size=32,
            eval_dataset=eval_dataset,
            lr_decay_epochs=[4, 6, 8],
            learning_rate=0.025,
            save_dir='output/mobilenetv2',
            use_vdl=True)
```

1.4 4. 模型开始训练

`train.py` 与解压后的数据集目录 `vegetables_cls` 放在同一目录下，在此目录下运行 `train.py` 即可开始训练。如果您的电脑上有 GPU，这将会在 10 分钟内训练完成，如果为 CPU 也大概会在 30 分钟内训练完毕。

```
python train.py
```

1.5 5. 训练过程中查看训练指标

模型在训练过程中，所有的迭代信息将以标注输出流的形式，输出到命令执行的终端上，用户也可通过 visualdl 以可视化的方式查看训练指标的变化，通过如下方式启动 visualdl 后，在浏览器打开 https://0.0.0.0:8001 (或 https://localhost:8001) 即可。

```
visualdl --logdir output/mobilenetv2/vdl_log --port 8000
```

1.6 6. 训练完成使用模型进行测试

如下代码使用训练过程中第 8 轮保存的模型进行测试。

```
import paddlex as pdx
model = pdx.load_model('output/mobilenetv2/epoch_8')
result = model.predict('vegetables_cls/bocai/100.jpg', topk=3)
print("Predict Result:", result)
```

预测结果输出如下，预测按 score 进行排序，得到前三分类结果

```
Predict Result: Predict Result: [{'score': 0.9999393, 'category': 'bocai', 'category_id
↪': 0}, {'score': 6.010089e-05, 'category': 'hongxiancai', 'category_id': 2}, {'score':␣
↪5.593914e-07, 'category': 'xilanhua', 'category_id': 5}]
```

1.7 其它推荐

- 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
- 3.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 1.7.1)**，paddlepaddle 安装方式参照飞桨官网

推荐使用 Anaconda Python 环境，Anaconda 下安装 PaddleX 参考文档Anaconda 安装使用

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 Github 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
```

2.3 安装问题

2.3.1 1. 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

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

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

Linux/Mac 安装

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

CHAPTER 3

PaddleX 全流程开发教程

3.1 数据准备

3.1.1 数据标注

3.1.2 主流标注软件支持

3.1.3 EasyData 数据标注支持

3.2 模型训练

3.2.1 训练图像分类模型

本文档训练代码可参考 PaddleX 的代码 tutorial/train/classification/mobilenetv2.py

1. 下载并解压训练所需的数据集

使用 1 张显卡训练并指定使用 0 号卡。

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0'
import paddlex as pdx
```

这里使用蔬菜数据集，训练集、验证集和测试集共包含 6189 个样本，18 个类别。

```
veg_dataset = 'https://bj.bcebos.com/paddlex/datasets/vegetables_cls.tar.gz'
pdx.utils.download_and_decompress(veg_dataset, path='./')
```

2. 定义训练和验证过程中的数据处理和增强操作

transforms 用于指定训练和验证过程中的数据处理和增强操作流程，如下代码在训练过程中使用了 `RandomCrop` 和 `RandomHorizontalFlip` 进行数据增强，transforms 的使用见paddlex.cls.transforms

```
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()
])
```

3. 创建数据集读取器，并绑定相应的数据预处理流程

通过不同的数据集读取器可以加载不同格式的数据集，数据集 API 的介绍见文档paddlex.datasets

```
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)
```

4. 创建模型进行训练

模型训练会默认自动下载和使用 imagenet 图像数据集上的预训练模型，用户也可自行指定 `pretrain_weights` 参数来设置预训练权重。模型训练过程每间隔 `save_interval_epochs` 轮会保存一次模型在 `save_dir` 目录下，同时在保存的过程中也会在验证数据集上计算相关指标。

分类模型的接口可见文档paddlex.cls.models

```
model = pdx.cls.MobileNetV2(num_classes=len(train_dataset.labels))
model.train(
    num_epochs=10,
    train_dataset=train_dataset,
    train_batch_size=32,
    eval_dataset=eval_dataset,
    lr_decay_epochs=[4, 6, 8],
    learning_rate=0.025,
    save_dir='output/mobilenetv2',
    use_vdl=True)
```

将 `use_vdl` 设置为 `True` 时可使用 VisualDL 查看训练指标。按以下方式启动 VisualDL 后，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。

```
visualdl --logdir output/mobilenetv2/vdl_log --port 8001
```

5. 验证或测试

利用训练完的模型可继续在验证集上进行验证。

```
eval_metrics = model.evaluate(eval_dataset, batch_size=8)
print("eval_metrics:", eval_metrics)
```

结果输出：

```
eval_metrics: OrderedDict([('acc1', 0.9895916733386709), ('acc5', 0.9983987189751802)])
```

训练完用模型对图片进行测试。

```
predict_result = model.predict('./vegetables_cls/bocai/IMG_00000839.jpg', topk=5)
print("predict_result:", predict_result)
```

结果输出：

```
predict_result: [{'category_id': 13, 'category': 'bocai', 'score': 0.8607276},
                 {'category_id': 11, 'category': 'kongxincai', 'score': 0.06386806},
                 {'category_id': 2, 'category': 'suanmiao', 'score': 0.03736042},
                 {'category_id': 12, 'category': 'heiqiezi', 'score': 0.007879922},
                 {'category_id': 17, 'category': 'huluobo', 'score': 0.006327283}]
```

3.2.2 训练目标检测模型

更多检测模型在 VOC 数据集或 COCO 数据集上的训练代码可参考代码 tutorials/train/detection/faster_rcnn_r50_fpn.py、代码 tutorials/train/detection/yolov3_darknet53.py。

1. 下载并解压训练所需的数据集

使用 1 张显卡训练并指定使用 0 号卡。

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0'
import paddlex as pdx
```

这里使用昆虫数据集，训练集、验证集和测试集共包含 217 个样本，6 个类别。

```
insect_dataset = 'https://bj.bcebos.com/paddlex/datasets/insect_det.tar.gz'
pdx.utils.download_and_decompress(insect_dataset, path='./')
```

2. 定义训练和验证过程中的数据处理和增强操作

在训练过程中使用 `RandomHorizontalFlip` 进行数据增强，由于接下来选择的模型是带 FPN 结构的 Faster RCNN，所以使用 `Padding` 将输入图像的尺寸补齐到 32 的倍数，以保证 FPN 中两个需做相加操作的特征层的尺寸完全相同。transforms 的使用见*paddlex.det.transforms*

```
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),
])
```

3. 创建数据集读取器，并绑定相应的数据预处理流程

数据集读取器的介绍见文档paddlex.datasets

```
train_dataset = pdx.datasets.VOCDetection(
    data_dir='insect_det',
```

(continues on next page)

(continued from previous page)

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

4. 创建 Faster RCNN 模型，并进行训练

创建带 FPN 结构的 Faster RCNN 模型，`num_classes` 需要设置为包含背景类的类别数，即: 目标类别数量 (6) + 1

```
num_classes = len(train_dataset.labels) + 1
model = pdx.det.FasterRCNN(num_classes=num_classes)
```

模型训练默认下载并使用在 ImageNet 数据集上训练得到的 Backone，用户也可自行指定 `pretrain_weights` 参数来设置预训练权重。训练过程每间隔 `save_interval_epochs` 会在 `save_dir` 保存一次模型，与此同时也会在验证数据集上计算指标。检测模型的接口可见文档paddlex.cv.models

```
model.train(
    num_epochs=12,
    train_dataset=train_dataset,
    train_batch_size=2,
    eval_dataset=eval_dataset,
    learning_rate=0.0025,
    lr_decay_epochs=[8, 11],
    save_dir='output/faster_rcnn_r50_fpn',
    use_vdl=True)
```

将 `use_vdl` 设置为 `True` 时可使用 VisualDL 查看训练指标。按以下方式启动 VisualDL 后，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。

```
visualdl --logdir output/faster_rcnn_r50_fpn/vdl_log --port 8001
```

5. 验证或测试

训练完利用模型可继续在验证集上进行验证。

```
eval_metrics = model.evaluate(eval_dataset, batch_size=2)
print("eval_metrics:", eval_metrics)
```

结果输出：

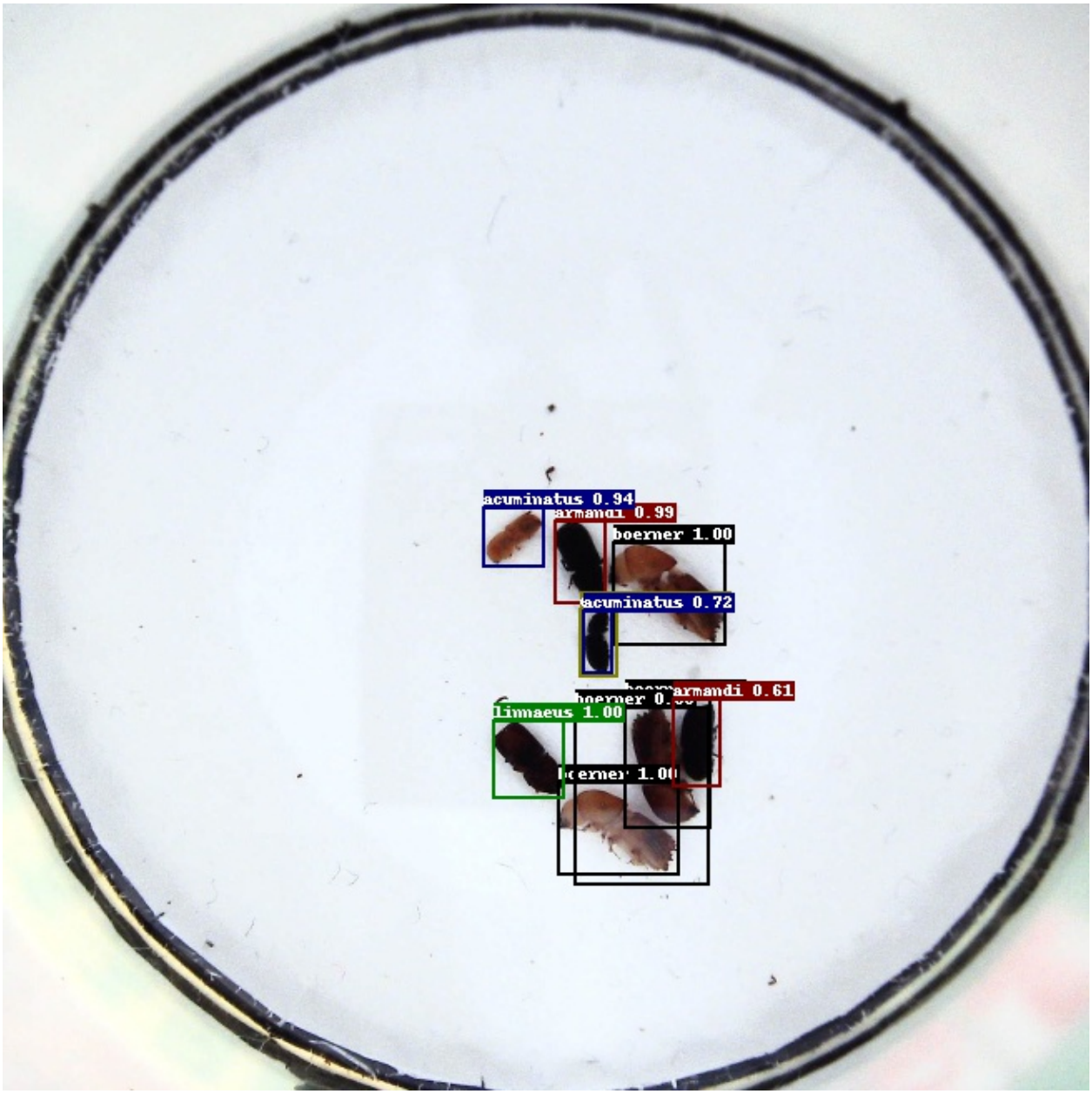
```
eval_metrics: {'bbox_map': 76.085371}
```

训练完用模型对图片进行测试。

```
predict_result = model.predict('./insect_det/JPEGImages/1968.jpg')
```

可视化测试结果：

```
pdx.det.visualize('./insect_det/JPEGImages/1968.jpg', predict_result, threshold=0.5,␣
↪save_dir='./output/faster_rcnn_r50_fpn')
```



3.2.3 训练实例分割模型

本文档训练代码可直接下载代码 tutorials/train/detection/mask_rcnn_r50_fpn.py。

1. 下载并解压训练所需的数据集

使用 1 张显卡训练并指定使用 0 号卡。

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0'
import paddlex as pdx
```

这里使用小度熊分拣数据集，训练集、验证集和测试共包含 21 个样本，1 个类别。

```
xiaoduxiong_dataset = 'https://bj.bcebos.com/paddlex/datasets/xiaoduxiong_ins_det.tar.gz'
pdx.utils.download_and_decompress(xiaoduxiong_dataset, path='./')
```

2. 定义训练和验证过程中的数据处理和增强操作

在训练过程中使用 `RandomHorizontalFlip` 进行数据增强，由于接下来选择的模型是带 FPN 结构的 Mask RCNN，所以使用 `PaddingImage` 将输入图像的尺寸补齐到 32 的倍数，以保证 FPN 中两个需做相加操作的特征层的尺寸完全相同。transforms 的使用见*paddlex.cv.transforms*

```
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)
])
```

3. 创建数据集读取器，并绑定相应的数据预处理流程

数据集读取器的介绍见文档paddlex.datasets

```
train_dataset = pdx.datasets.CocoDetection(
    data_dir='xiaoduxiong_ins_det/JPEGImages',
    ann_file='xiaoduxiong_ins_det/train.json',
    transforms=train_transforms,
    shuffle=True)
eval_dataset = pdx.datasets.CocoDetection(
    data_dir='xiaoduxiong_ins_det/JPEGImages',
    ann_file='xiaoduxiong_ins_det/val.json',
    transforms=eval_transforms)
```

4. 创建 Mask RCNN 模型，并进行训练

创建带 FPN 结构的 Mask RCNN 模型，num_classes 需要设置为包含背景类的类别数，即：目标类别数量 (1) + 1。

```
num_classes = len(train_dataset.labels)
model = pdx.det.MaskRCNN(num_classes=num_classes
```

模型训练默认下载并使用在 ImageNet 数据集上训练得到的 Backone，用户也可自行指定 pretrain_weights 参数来设置预训练权重。训练过程每间隔 save_interval_epochs 会在 save_dir 保存一次模型，与此同时也会在验证数据集上计算指标。检测模型的接口可见文档paddlex.det.models。

```
model.train(
    num_epochs=12,
    train_dataset=train_dataset,
    train_batch_size=1,
    eval_dataset=eval_dataset,
    learning_rate=0.00125,
    warmup_steps=10,
    lr_decay_epochs=[8, 11],
    save_dir='output/mask_rcnn_r50_fpn',
    use_vdl=True)
```

将 use_vdl 设置为 True 时可使用 VisualDL 查看训练指标。按以下方式启动 VisualDL 后，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。

```
visualdl --logdir output/faster_rcnn_r50_fpn/vdl_log --port 8001
```

5. 验证或测试

训练完利用模型可继续在验证集上进行验证。

```
eval_metrics = model.evaluate(eval_dataset, batch_size=1)
print("eval_metrics:", eval_metrics)
```

结果输出：

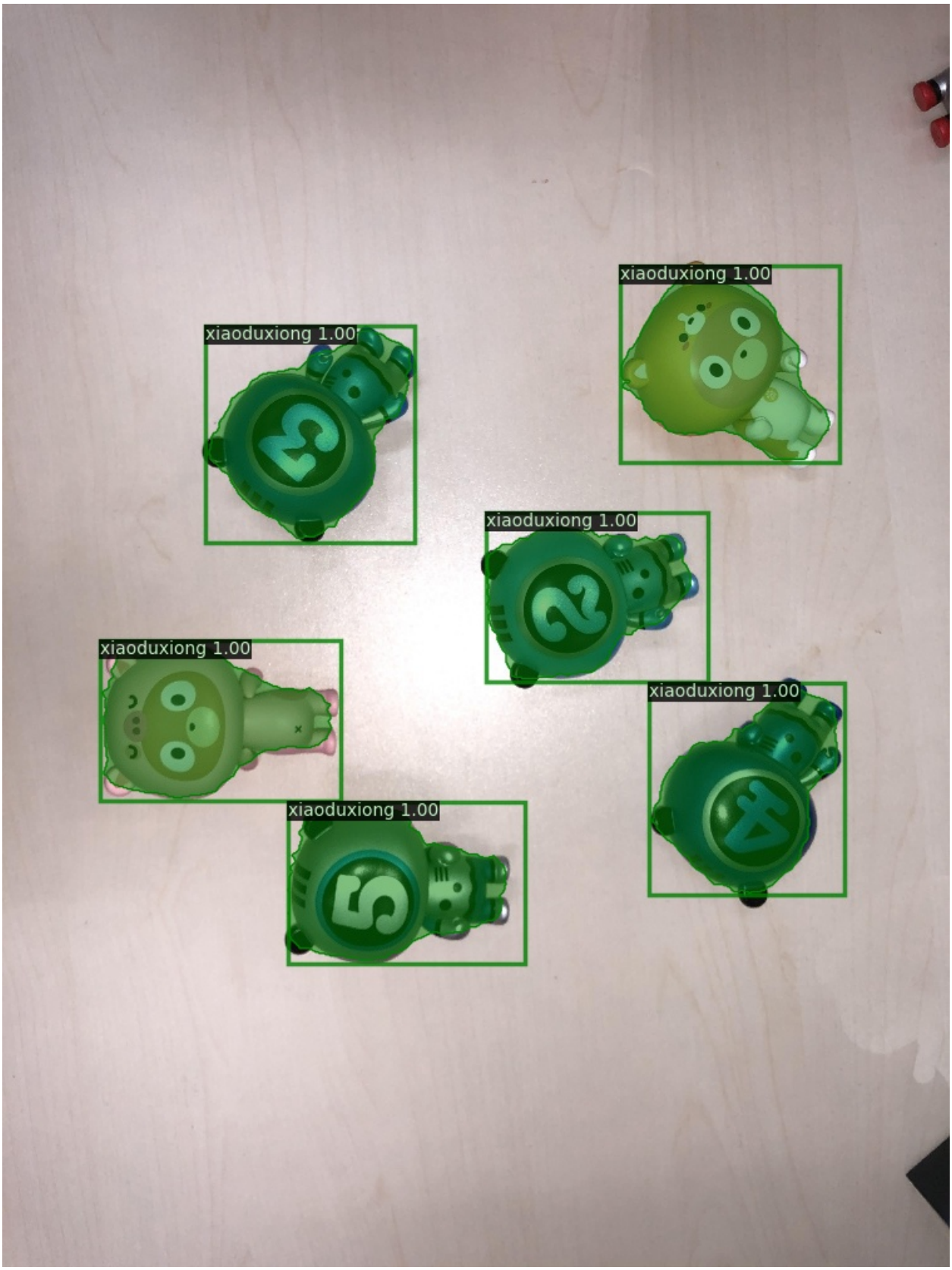
```
eval_metrics: OrderedDict([('bbox_mmap', 0.5038283828382838), ('segm_mmap', 0.
↪7025202520252025)])
```

训练完用模型对图片进行测试。

```
predict_result = model.predict('./xiaoduxiong_ins_det/JPEGImages/WechatIMG114.jpeg')
```

可视化测试结果：

```
pdx.det.visualize('./xiaoduxiong_ins_det/JPEGImages/WechatIMG114.jpeg', predict_result,␣
↪threshold=0.7, save_dir='./output/mask_rcnn_r50_fpn')
```


xiaoduxiong 1.00
xiaoduxiong 1.00
xiaoduxiong 1.00
xiaoduxiong 1.00
xiaoduxiong 1.00
xiaoduxiong 1.00

3.2.4 训练语义分割模型

更多语义分割模型在视盘数据集上的训练代码可参考代码 tutorials/train/segmentation/deeplabv3p.py。

1. 下载并解压训练所需的数据集

使用 1 张显卡训练并指定使用 0 号卡。

```
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0'
import paddlex as pdx
```

这里使用视盘分割数据集，训练集、验证集和测试集共包含 343 个样本，2 个类别。

```
optic_dataset = 'https://bj.bcebos.com/paddlex/datasets/optic_disc_seg.tar.gz'
pdx.utils.download_and_decompress(optic_dataset, path='./')
```

2. 定义训练和验证过程中的数据处理和增强操作

在训练过程中使用 `RandomHorizontalFlip` 和 `RandomPaddingCrop` 进行数据增强，transforms 的使用见*paddlex.seg.transforms*

```
train_transforms = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.Resize(target_size=512),
    transforms.RandomPaddingCrop(crop_size=500),
    transforms.Normalize()
])
eval_transforms = transforms.Compose([
    transforms.Resize(512),
    transforms.Normalize()
])
```

3. 创建数据集读取器，并绑定相应的数据预处理流程

数据集读取器的介绍见文档paddlex.cv.datasets

```
train_dataset = pdx.datasets.SegDataset(
    data_dir='optic_disc_seg',
    file_list='optic_disc_seg/train_list.txt',
    label_list='optic_disc_seg/labels.txt',
    transforms=train_transforms,
    shuffle=True)
eval_dataset = pdx.datasets.SegDataset(
```

(continues on next page)

(continued from previous page)

```
    data_dir='optic_disc_seg',
    file_list='optic_disc_seg/val_list.txt',
    label_list='optic_disc_seg/labels.txt',
    transforms=eval_transforms)
```

4. 创建 DeepLabv3+ 模型，并进行训练

创建 DeepLabv3+ 模型，num_classes 需要设置为不包含背景类的类别数，即：目标类别数量 (1)，详细代码可参见demo。

```
num_classes = num_classes
model = pdx.seg.DeepLabv3p(num_classes=num_classes)
```

模型训练默认下载并使用在 ImageNet 数据集上训练得到的 Backone，用户也可自行指定 pretrain_weights 参数来设置预训练权重。训练过程每间隔 save_interval_epochs 会在 save_dir 保存一次模型，与此同时也会在验证数据集上计算指标。检测模型的接口可见文档paddlex.seg.models。

```
model.train(
    num_epochs=40,
    train_dataset=train_dataset,
    train_batch_size=4,
    eval_dataset=eval_dataset,
    learning_rate=0.01,
    save_dir='output/deeplab',
    use_vdl=True)
```

将 use_vdl 设置为 True 时可使用 VisualDL 查看训练指标。按以下方式启动 VisualDL 后，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。

```
visualdl --logdir output/deeplab/vdl_log --port 8001
```

5. 验证或测试

训练完利用模型可继续在验证集上进行验证。

```
eval_metrics = model.evaluate(eval_dataset, batch_size=2)
print("eval_metrics:", eval_metrics)
```

结果输出：

```
eval_metrics: {'miou': 0.8915175875548873, 'category_iou': [0.9956445981924432, 0.
↪7873905769173314], 'macc': 0.9957137358816046, 'category_acc': [0.9975360650317765, 0.
↪8948120441157331], 'kappa': 0.8788684558629085}
```

(continues on next page)

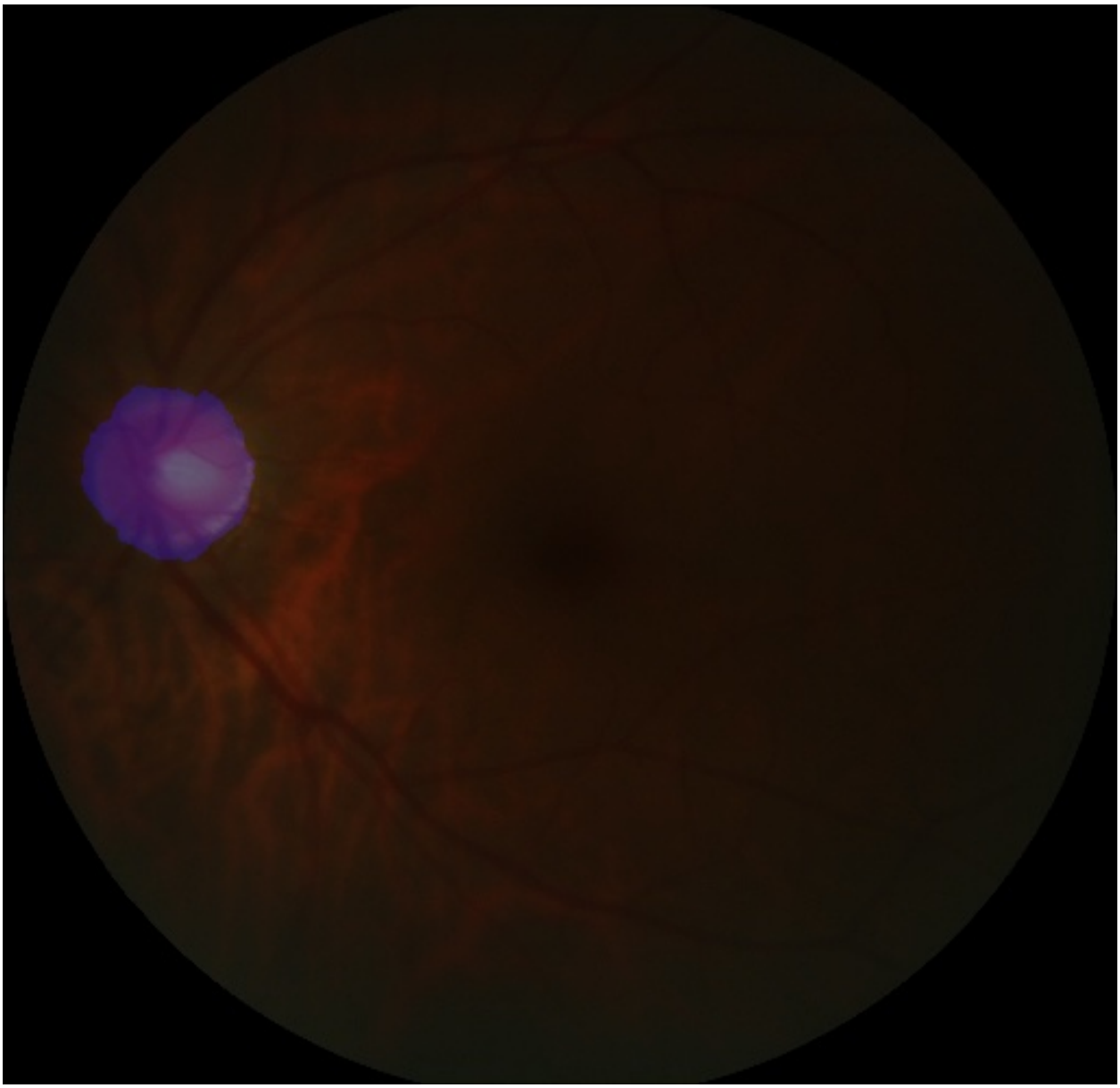
(continued from previous page)

训练完用模型对图片进行测试。

```
image_name = 'optic_disc_seg/JPEGImages/H0005.jpg'
predict_result = model.predict(image_name)
```

可视化测试结果:

```
import paddlex as pdx
pdx.seg.visualize(image_name, predict_result, weight=0.4)
```



3.2.5 VisualDL 可视化训练指标

在使用 PaddleX 训练模型过程中，各个训练指标和评估指标会直接输出到标准输出流，同时也可通过 VisualDL 对训练过程中的指标进行可视化，只需在调用 `train` 函数时，将 `use_vdl` 参数设为 `True` 即可，如下代码所示，

```
model = paddlex.cls.ResNet50(num_classes=1000)
model.train(num_epochs=120, train_dataset=train_dataset,
            train_batch_size=32, eval_dataset=eval_dataset,
            log_interval_steps=10, save_interval_epochs=10,
            save_dir='./output', use_vdl=True)
```

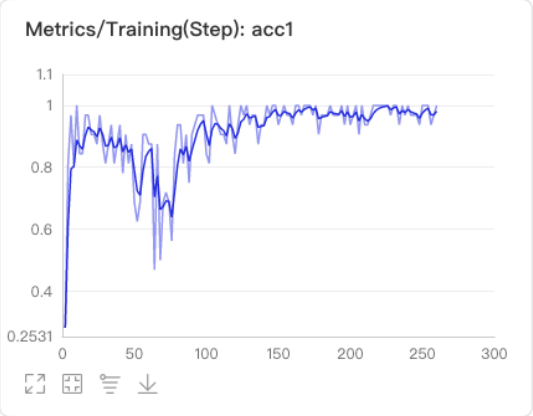
模型在训练过程中，会在 `save_dir` 下生成 `vdl_log` 目录，通过在命令行终端执行以下命令，启动 VisualDL。

```
visualdl --logdir=output/vdl_log --port=8008
```

在浏览器打开 `http://0.0.0.0:8008` 便可直接查看随训练迭代动态变化的各个指标（0.0.0.0 表示启动 VisualDL 所在服务器的 IP，本机使用 0.0.0.0 即可）。

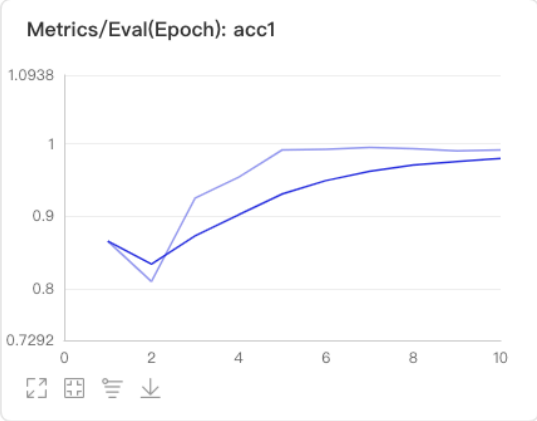
在训练分类模型过程中，使用 VisualDL 进行可视化的示例图如下所示。

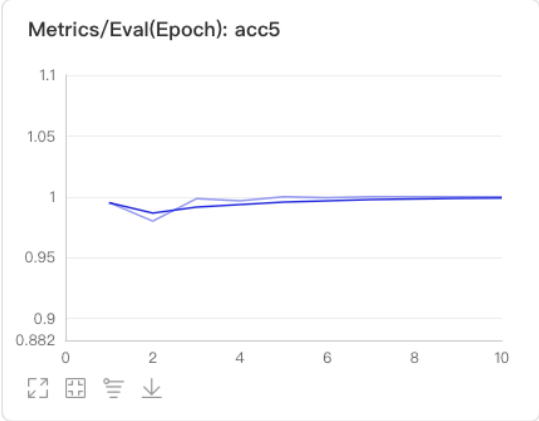
训练过程中每个 Step 的 `Loss` 和相应 `Top1` 准确率变化趋势：



训练过程中每个 Step 的学习率 `lr` 和相应 `Top5` 准确率变化趋势：

训练过程中，每次保存模型时，模型在验证数据集上的 `Top1` 准确率和 `Top5` 准确率：





3.3 模型压缩

3.3.1 分类模型裁剪

本文档训练代码可直接在 PaddleX 的 Repo 中下载，代码 tutorials/compress/classification本文档按如下方式对模型进行了裁剪

> 第一步：在训练数据集上训练 MobileNetV2 第二步：在验证数据集上计算模型中各个参数的敏感度信息第三步：根据第二步计算的敏感度，设定 `eval_metric_loss`，对模型裁剪后重新在训练数据集上训练

步骤一训练 MobileNetV2

> 模型训练使用文档可以直接参考分类模型训练，本文档在该代码基础上添加了部分参数选项，用户可直接下载模型训练代码tutorials/compress/classification/mobilenetv2.py使用如下命令开始模型训练

```
python mobilenetv2.py
```

步骤二计算参数敏感度

参数敏感度的计算可以直接使用 PaddleX 提供的 APIpaddlex.slim.cal_params_sensitivities，使用代码如下，敏感度信息文件会保存至 save_file

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

model_dir = './output/mobilenetv2/best_model'
model = pdx.load_model(model_dir)

# 定义验证所用的数据集
eval_dataset = pdx.datasets.ImageNet(
    data_dir=dataset,
    file_list=os.path.join(dataset, 'val_list.txt'),
    label_list=os.path.join(dataset, 'labels.txt'),
    transforms=model.eval_transforms)

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

本步骤代码已整理至tutorials/compress/classification/cal_sensitivities_file.py，用户可直接下载使用使用如下命令开始计算敏感度

```
python cal_sensitivities_file.py --model_dir output/mobilenetv2/best_model --dataset␣
↪vegetables_cls --save_file sensitivities.data
```

步骤三开始裁剪训练

本步骤代码与步骤一使用同一份代码文件，使用如下命令开始裁剪训练

```
python mobilenetv2.py --model_dir output/mobilenetv2/best_model --sensitivities_file␣
↪sensitivities.data --eval_metric_loss 0.10
```

实验效果

本教程的实验效果可以查阅模型压缩文档

3.3.2 检测模型裁剪

本文档训练代码可直接在 PaddleX 的 Repo 中下载，代码 tutorials/compress/detection本文档按如下方式对模型进行了裁剪

> 第一步：在训练数据集上训练 YOLOv3 第二步：在验证数据集上计算模型中各个参数的敏感度信息第三步：根据第二步计算的敏感度，设定 `eval_metric_loss`，对模型裁剪后重新在训练数据集上训练

步骤一训练 YOLOv3

> 模型训练使用文档可以直接参考检测模型训练，本文档在该代码基础上添加了部分参数选项，用户可直接下载模型训练代码tutorials/compress/detection/yolov3_mobilnet.py使用如下命令开始模型训练

```
python yolov3_mobilenet.py
```

步骤二计算参数敏感度

> 参数敏感度的计算可以直接使用 PaddleX 提供的 APIpaddlex.slim.cal_params_sensitivities，使用代码如下，敏感度信息文件会保存至 `save_file`

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

model = pdx.load_model(model_dir)

# 定义验证所用的数据集
eval_dataset = pdx.datasets.ImageNet(
    data_dir=dataset,
    file_list=os.path.join(dataset, 'val_list.txt'),
    label_list=os.path.join(dataset, 'labels.txt'),
    transforms=model.eval_transforms)
```

(continues on next page)

(continued from previous page)

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

本步骤代码已整理至tutorials/compress/detection/cal_sensitivities_file.py，用户可直接下载使用使用如下命令开始计算敏感度

```
python cal_sensitivities_file.py --model_dir output/yolov3_mobile/best_model --dataset␣
↪insect_det --save_file sensitivities.data
```

步骤三开始裁剪训练

本步骤代码与步骤一使用同一份代码文件，使用如下命令开始裁剪训练

```
python yolov3_mobilenet.py --model_dir output/yolov3_mobile/best_model --sensitivities_
↪file sensitivities.data --eval_metric_loss 0.10
```

实验效果

本教程的实验效果可以查阅模型压缩文档

3.3.3 分割模型裁剪

本文档训练代码可直接在 PaddleX 的 Repo 中下载，代码 tutorials/compress/segmentation本文档按如下方式对模型进行了裁剪

第一步：在训练数据集上训练 UNet 第二步：在验证数据集上计算模型中各个参数的敏感度信息 第三步：根据第二步计算的敏感度，设定 `eval_metric_loss`，对模型裁剪后重新在训练数据集上训练

步骤一训练 UNet

模型训练使用文档可以直接参考检测模型训练，本文档在该代码基础上添加了部分参数选项，用户可直接下载模型训练代码tutorials/compress/segmentation/unet.py使用如下命令开始模型训练

```
python unet.py
```

步骤二计算参数敏感度

参数敏感度的计算可以直接使用 PaddleX 提供的 APIpaddlex.slim.cal_params_sensitivities，使用代码如下，敏感度信息文件会保存至 save_file

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

model = pdx.load_model(model_dir)

# 定义验证所用的数据集
eval_dataset = pdx.datasets.ImageNet(
    data_dir=dataset,
    file_list=os.path.join(dataset, 'val_list.txt'),
    label_list=os.path.join(dataset, 'labels.txt'),
    transforms=model.eval_transforms)

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

本步骤代码已整理至tutorials/compress/detection/cal_sensitivities_file.py，用户可直接下载使用使用如下命令开始计算敏感度

```
python cal_sensitivities_file.py --model_dir output/unet/best_model --dataset optic_disc_
↪seg --save_file sensitivities.data
```

步骤三开始裁剪训练

本步骤代码与步骤一使用同一份代码文件，使用如下命令开始裁剪训练

```
python unet.py --model_dir output/unet/best_model --sensitivities_file sensitivities.
↪data --eval_metric_loss 0.10
```

实验效果

本教程的实验效果可以查阅模型压缩文档

3.4 多端安全部署

本文档指引用户如何采用更高性能地方式来部署使用 PaddleX 训练的模型。本文档模型部署采用 Paddle Inference 高性能部署方式，在模型运算过程中，对模型计算图进行优化，同时减少内存操作，具体各模型性能对比见服务端 Python 部署的预测性能对比章节。

同时结合产业实践开发者对模型知识产权的保护需求，提供了轻量级模型加密部署的方案，提升深度学习模型部署的安全性。

3.4.1 服务端部署

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 inference 模型

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]
```

预测部署

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

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

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

C++ 部署

C++ 部署方案位于目录 PaddleX/deploy/cpp/下，且独立于 PaddleX 其他模块。该方案支持在 Windows 和 Linux 完成编译、二次开发集成和部署运行，支持在 Linux 上完成加密部署。

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: 下载代码

下载源代码

```
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 fluid_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 Paddle1.7 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

解压后 `D:\projects\fluid_inference*\`目录下主要包含的内容为:

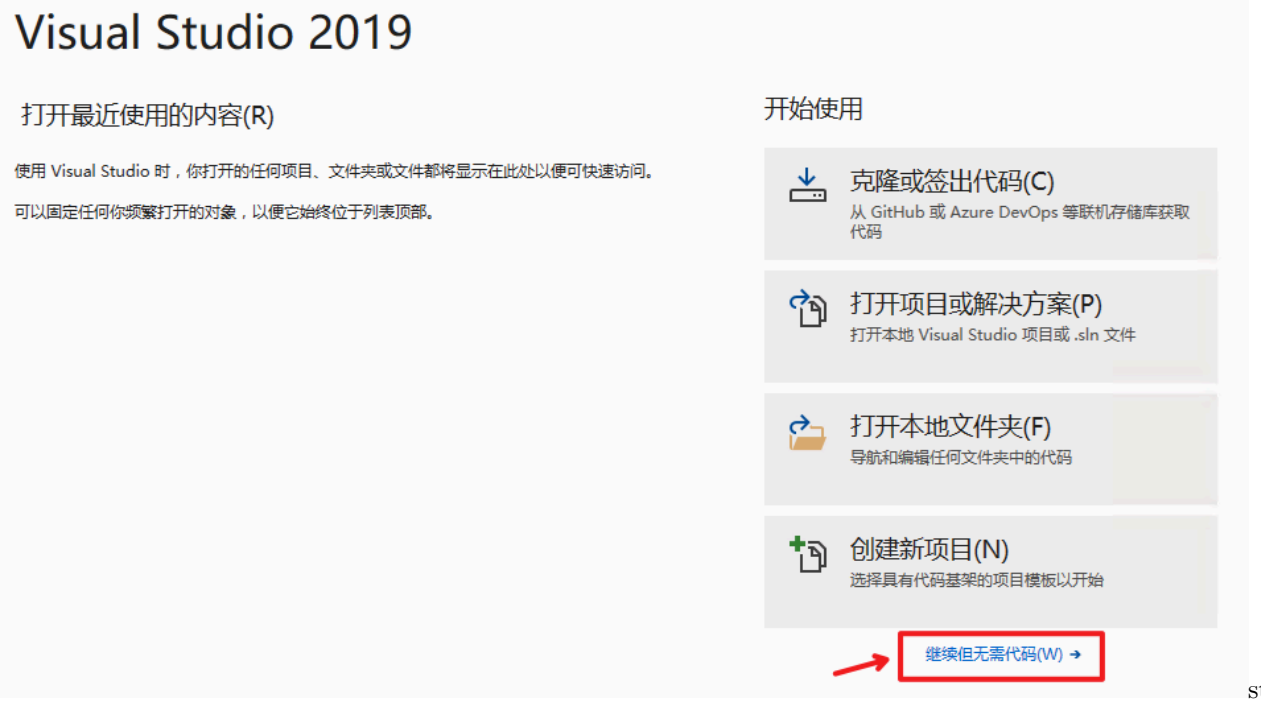
```
  \paddle\ # paddle 核心库和头文件
|
  \third_party\ # 第三方依赖库和头文件
|
  \version.txt # 版本和编译信息
```

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，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`

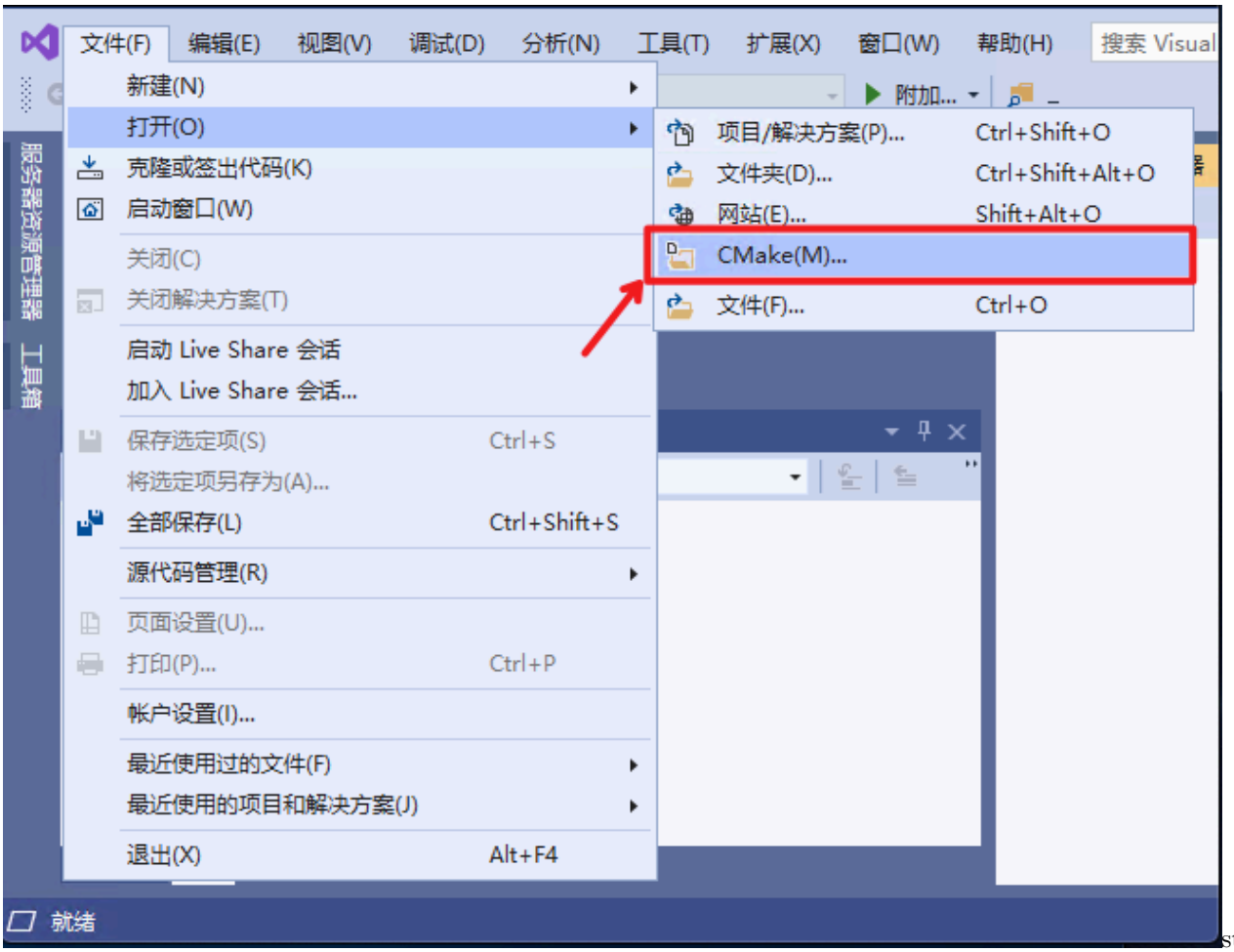
Step4: 使用 Visual Studio 2019 直接编译 CMake

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



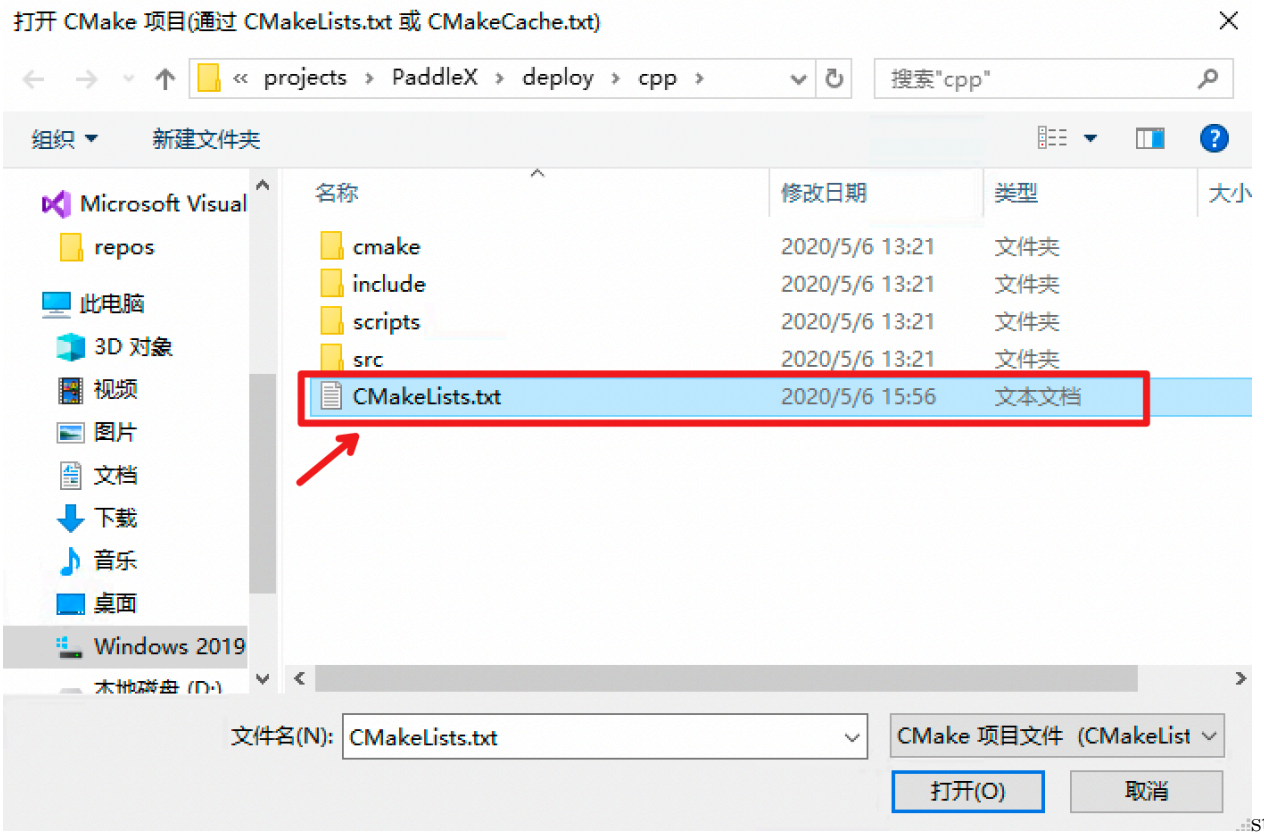
step2

1. 点击：文件-> 打开->CMake



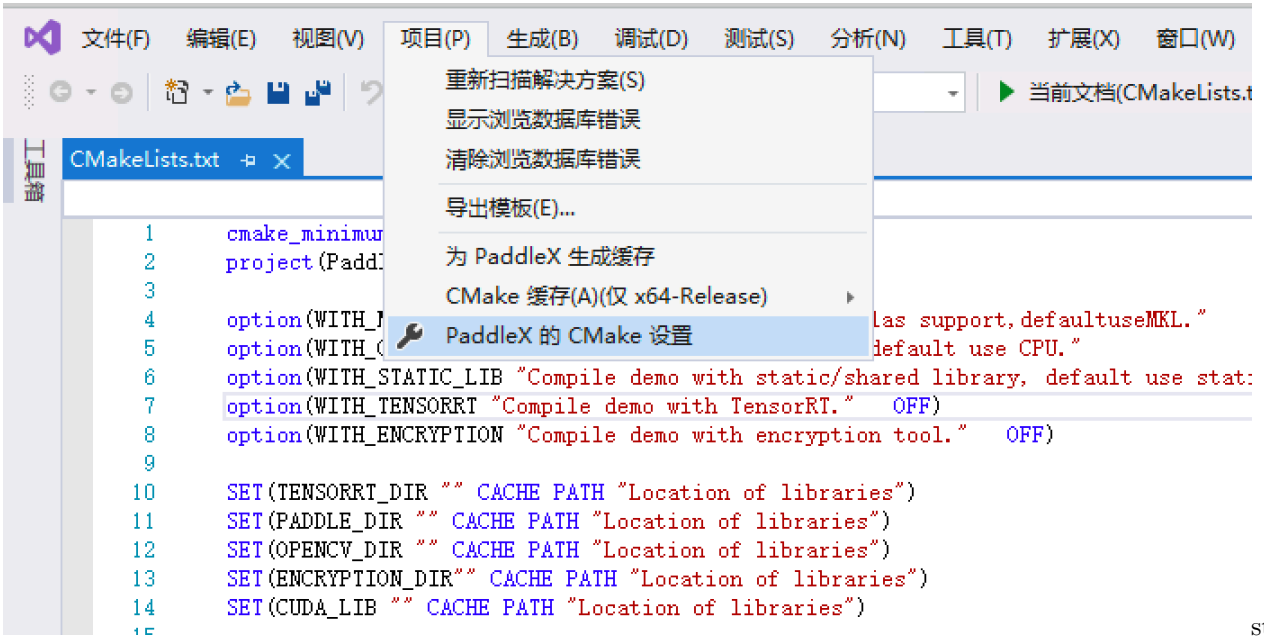
step2.1

选择项目代码所在路径，并打开 `CMakeList.txt`：



step2.2

1. 点击：项目->PADDLEX_INFERENCE 的 CMake 设置

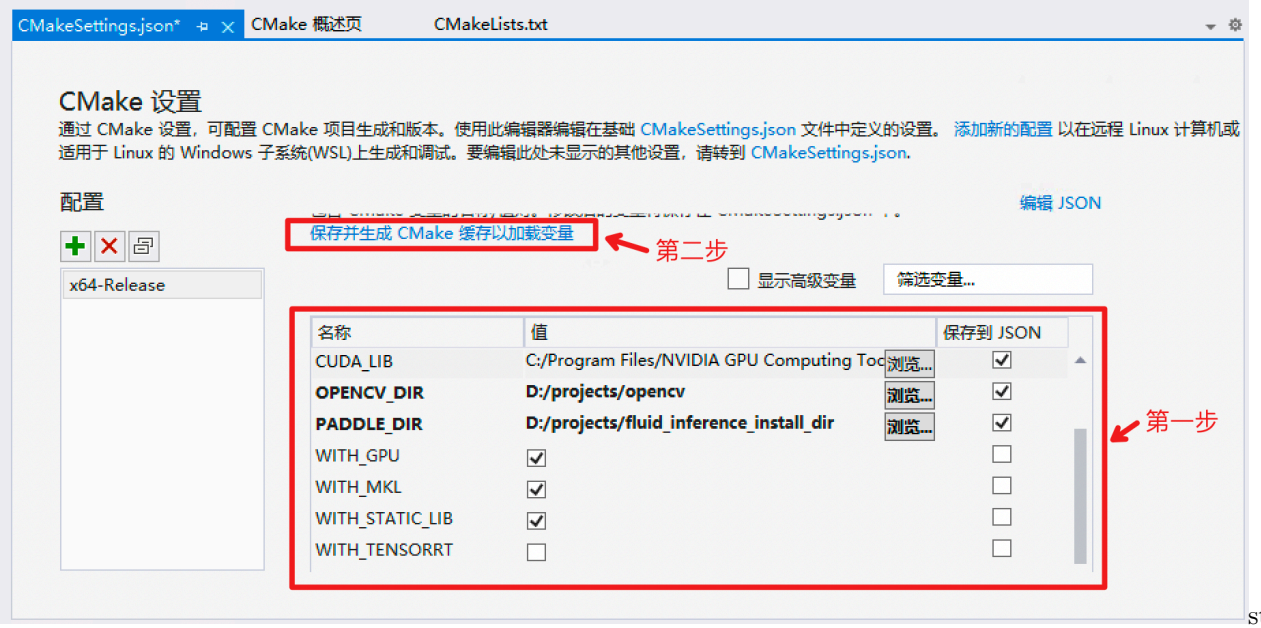


step3

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

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 库版本尽量对齐，**使用 9.0、10.0 版本，不使用 9.2、10.1 等版本 CUDA 库**）：

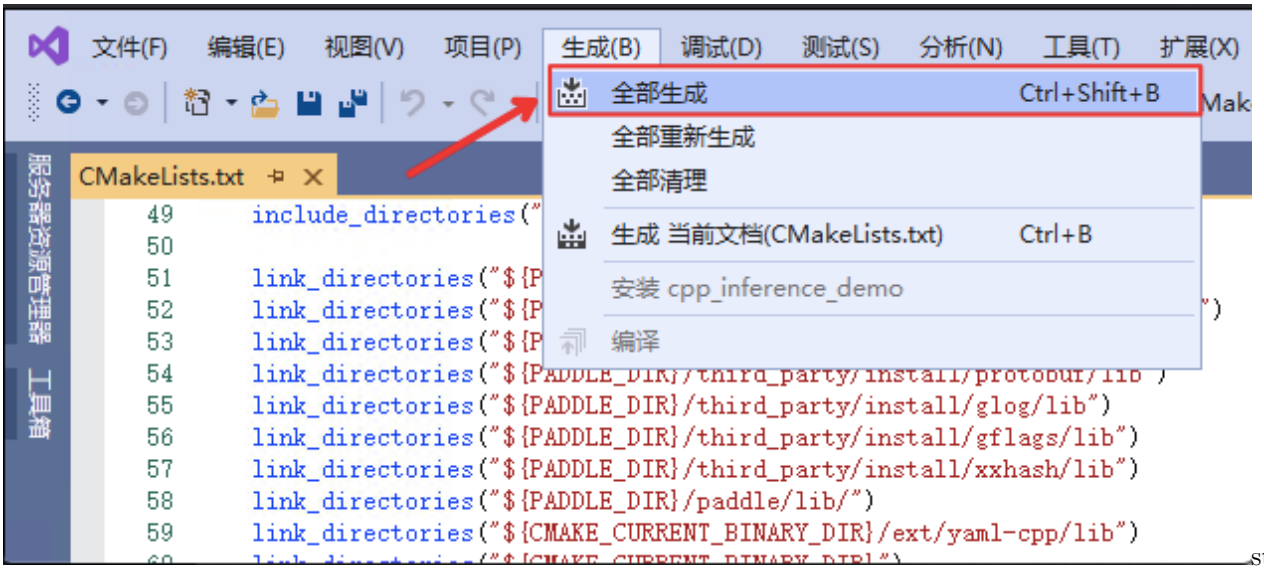
注意： 1. 使用 CPU 版预测库，请把 WITH_GPU 的值去掉勾 2. 如果使用的是 openblas 版本，请把 WITH_MKL 的值去掉勾



step4

设置完成后，点击上图中保存并生成 CMake 缓存以加载变量。

1. 点击生成-> 全部生成



step6

Step5: 预测及可视化

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

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\classifer.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。

样例一：

不使用 GPU 测试图片 \\path\\to\\xiaoduxiong.jpeg

```
.\\paddlex_inference\\detector.exe --model_dir=\\path\\to\\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 GPU 预测多个图片\\path\\to\\image_list.txt，image_list.txt 内容的格式如下：

```
\\path\\to\\images\\xiaoduxiong1.jpeg
\\path\\to\\images\\xiaoduxiong2.jpeg
...
\\path\\to\\images\\xiaoduxiongn.jpeg
```

```
.\\paddlex_inference\\detector.exe --model_dir=\\path\\to\\inference_model --image_
↪list=\\path\\to\\images_list.txt --use_gpu=1 --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。

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 fluid_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 Paddle1.7 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。

Step4: 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 lib 路径
TENSORRT_DIR=/path/to/TensorRT/
# Paddle 预测库路径
PADDLE_DIR=/path/to/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=ON
# CUDA 的 lib 路径
CUDA_LIB=/path/to/cuda/lib/
# CUDNN 的 lib 路径
CUDNN_LIB=/path/to/cudnn/lib/

# 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
OPENCV_DIR=$(pwd)/deps/opencv3gcc4.8/
sh $(pwd)/scripts/bootstrap.sh

# 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
```

修改脚本设置好主要参数后，执行 build 脚本：

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

Step5: 预测及可视化

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

编译成功后，预测 demo 的可执行程序分别为 build/demo/detector，build/demo/classifer，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。

样例一：

不使用 GPU 测试图片 /path/to/xiaoduxiong.jpeg

```
./build/demo/detector --model_dir=/path/to/inference_model --image=/path/to/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/path/to/image_list.txt，image_list.txt 内容的格式如下：

```
/path/to/images/xiaoduxiong1.jpeg
/path/to/images/xiaoduxiong2.jpeg
...
/path/to/images/xiaoduxiongn.jpeg
```

```
./build/demo/detector --model_dir=/path/to/models/inference_model --image_list=/root/
↪projects/images_list.txt --use_gpu=1 --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。

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。

注意：目前加密方案仅支持 Linux 系统

1. 方案简介

1.1 简介

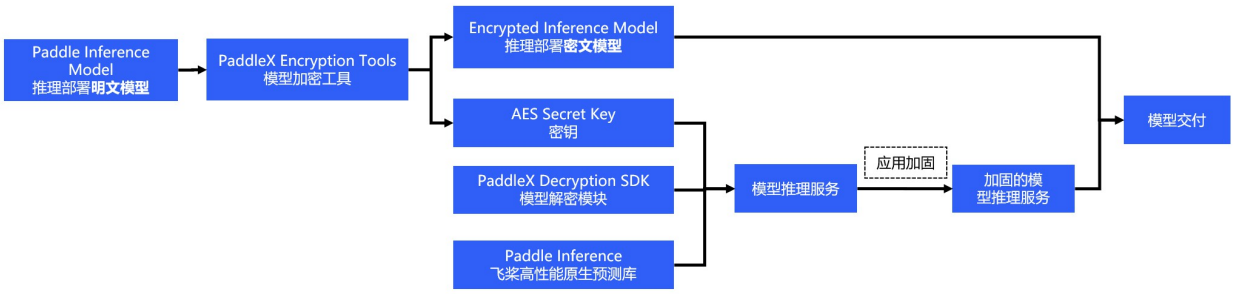
（1）加密算法的选择和支持的库

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 load 到 paddlepaddle 中（c/c++ 预测服务）

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：https://wiki.openssl.org/index.php/EVP_Authenticated_Encryption_and_Decryption。

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：https://wiki.openssl.org/index.php/EVP_Message_Digests

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 paddlepaddle 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

PaddleX 模型加密工具。在编译部署代码时，编译脚本会自动下载加密工具，您也可以选择手动下载。

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

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

```
./paddlex-encryption/tool/paddlex_encrypt_tool -model_dir /path/to/paddlex_inference_
↪model -save_dir /path/to/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

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/classifer，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 /path/to/xiaoduxiong.jpeg

```
./build/demo/detector --model_dir=/path/to/inference_model --image=/path/to/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/path/to/image_list.txt，image_list.txt 内容的格式如下：

```
/path/to/images/xiaoduxiong1.jpeg
/path/to/images/xiaoduxiong2.jpeg
...
/path/to/images/xiaoduxiongn.jpeg
```

```
./build/demo/detector --model_dir=/path/to/models/inference_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。

3.4.2 OpenVINO 部署

方案简介

OpenVINO 部署方案位于目录 `PaddleX/deploy/openvino/`下，且独立于 PaddleX 其他模块，该方案目前支持在 **Linux** 完成编译和部署运行。

PaddleX 到 OpenVINO 的部署流程如下：

> PaddleX –> ONNX –> OpenVINO IR –> OpenVINO Inference Engine

部署流程

说明

本文档在 `Ubuntu` 使用 `GCC 4.8.5` 进行了验证，如果需要使用更多 G++ 版本和平台的 OpenVino 编译，请参考: OpenVINO。

验证环境

- Ubuntu* 16.04 (64-bit) with GCC* 4.8.5
- CMake 3.12
- Python 2.7 or higher

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`**。

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

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

Step1: 软件依赖

- openvino: 编译文档
- gflags: 编译文档
- opencv: 编译文档 说明：/root/projects/PaddleX/deploy/openvino/scripts/bootstrap.sh 提供了预编译版本下载，也可自行编译。
- ngraph: 说明：openvino 编译的过程中会生成 ngraph 的 lib 文件，位于 {openvino 根目录}/bin/intel64/Release/lib/下。

Step2: 编译

编译 cmake 的命令在 scripts/build.sh 中，请根据 Step1 中编译软件的实际情况修改主要参数，其主要内容说明如下：

```
# openvino 预编译库的路径
OPENVINO_DIR=/path/to/inference_engine/
# gflags 预编译库的路径
GFLAGS_DIR=/path/to/gflags
# ngraph lib 的路径，编译 openvino 时通常会生成
NGRAPH_LIB=/path/to/ngraph/lib/
# opencv 预编译库的路径，如果使用自带预编译版本可不修改
OPENCV_DIR=$(pwd)/deps/opencv3gcc4.8/
# 下载自带预编译版本
sh $(pwd)/scripts/bootstrap.sh
rm -rf build
mkdir -p build
cd build
cmake .. \
    -DOPENCV_DIR=${OPENCV_DIR} \
    -DGFLAGS_DIR=${GFLAGS_DIR} \
    -DOPENVINO_DIR=${OPENVINO_DIR} \
    -DNGRAPH_LIB=${NGRAPH_LIB}
make
```

修改脚本设置好主要参数后，执行 build 脚本：

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

Step3: 模型转换

将 PaddleX 模型转换成 ONNX 模型：

```
paddlex --export_onnx --model_dir=/path/to/xiaoduxiong_epoch_12 --save_dir=/path/to/onnx_
↪model
```

将生成的 onnx 模型转换为 OpenVINO 支持的格式，请参考：Model Optimizer 文档

Step4: 预测

编译成功后，分类任务的预测可执行程序为 classifier，其主要命令参数说明如下：

样例

样例一：

测试图片 `/path/to/xiaoduxiong.jpeg`

```
./build/classifier --model_dir=/path/to/openvino_model --image=/path/to/xiaoduxiong.jpeg
```

样例二：

预测多个图片`/path/to/image_list.txt`，image_list.txt 内容的格式如下：

```
/path/to/images/xiaoduxiong1.jpeg
/path/to/images/xiaoduxiong2.jpeg
...
/path/to/images/xiaoduxiongn.jpeg
```

```
./build/classifier --model_dir=/path/to/models/openvino_model --image_list=/root/
↪projects/images_list.txt
```

3.4.3 移动端部署

step 1: 安装 PaddleLite

```
pip install paddlelite
```

step 2: 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。

step 3: 将 inference 模型转换成 PaddleLite 模型

```
python /path/to/PaddleX/deploy/lite/export_lite.py --model_path /path/to/inference_model␣
↪--save_dir /path/to/onnx_model
```

`--model_path` 用于指定 inference 模型的路径，`--save_dir` 用于指定 Lite 模型的保存路径。

step 4: 预测

Lite 模型预测正在集成中，即将开源…

CHAPTER 4

PaddleX 视觉方案介绍

PaddleX 目前提供了 4 种视觉任务解决方案，分别为图像分类、目标检测、实例分割和语义分割。用户可以根据自己的任务类型按需选取。

4.1 图像分类

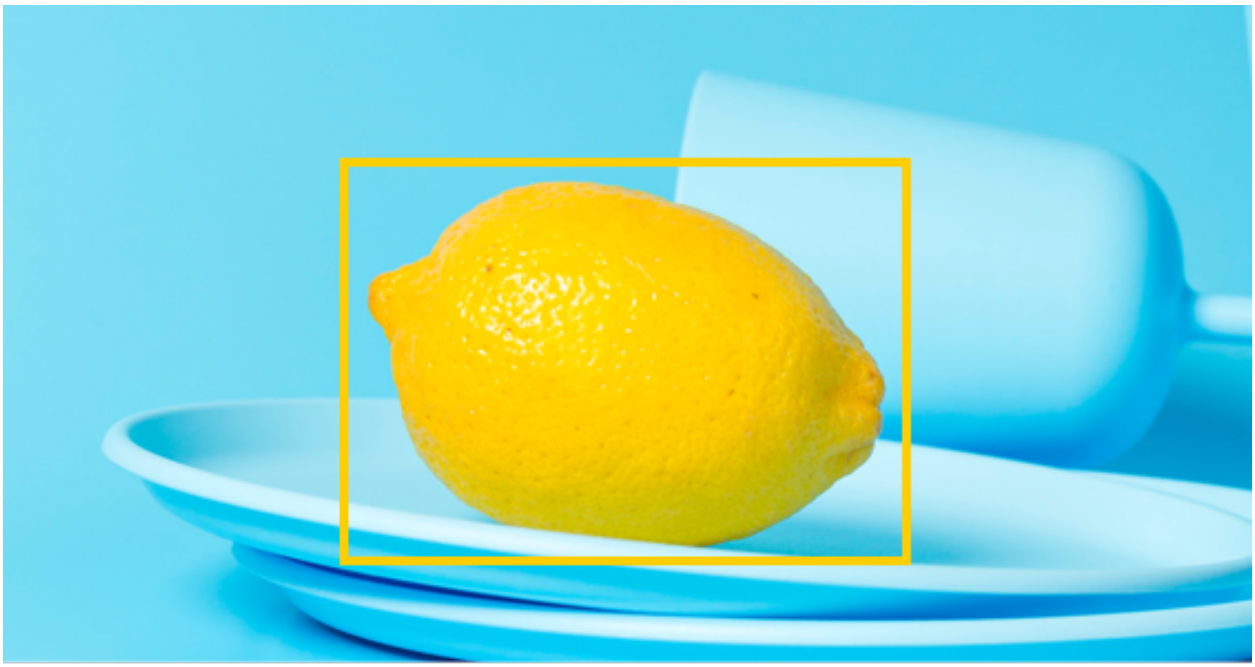
图像分类任务指的是输入一张图片，模型预测图片的类别，如识别为风景、动物、车等。

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

除上述模型外，PaddleX 还支持近 20 种图像分类模型，模型列表可参考*PaddleX 模型库*

4.2 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

除 YOLOv3 模型外，PaddleX 同时也支持 FasterRCNN 模型，支持 FPN 结构和 5 种 backbone 网络，详情可参考*PaddleX 模型库*

4.3 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*

4.4 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

CHAPTER 5

PaddleX API 说明文档

5.1 数据处理-transforms

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

5.1.1 图像分类-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 类

```
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。

RandomCrop 类

```
paddlex.cls.transforms.RandomCrop(crop_size=224, lower_scale=0.88, 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 0.88。
- **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 类

```
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 类

```
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。

Normalize 类

```
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 类

```
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 类

```
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。

RandomRotate 类

```
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 类

```
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。

5.1.2 检测和实例分割-det.transforms

对目标检测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 类

```
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。

ResizeByShort 类

```
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 类

```
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 类

```
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 类

```
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。

Normalize 类

```
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]。

RandomDistort 类

```
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 类

```
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 类

```
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。

5.1.3 语义分割-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 类

```
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 类

```
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 类

```
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 类

```
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 类

```
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 类

```
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 类

```
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 类

```
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 类

```
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 类

```
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 类

```
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。

RandomRotation 类

```
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 类

```
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 类

```
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。

5.1.4 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()
])
```

5.2 数据集-datasets

PaddleX 目前支持主流的 CV 数据集格式和 EasyData 数据标注平台的标注数据格式，此外 PaddleX 也提升了数据格式转换工具 API，支持包括 LabelMe，精灵标注助手和 EasyData 平台数据格式的转换，可以参考 PaddleX 的 tools API 文档。

下表为各数据集格式与相应任务的对应关系，

数据集格式	图像分类	目标检测	实例分割	语义分割
ImageNet	√	•	•	•
VOCDetection	•	√	•	•
CocoDetection	•	√	√	•
SegDataset	•	•	•	√
EasyDataCls	√	•	•	•
EasyDataDet	•	√	√	•
EasyDataSeg	•	•	•	√

5.2.1 图像分类数据集

ImageNet 类

```
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。

EasyDataCls 类

```
paddlex.datasets.EasyDatasetCls(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。EasyData 图像分类任务数据集格式的介绍可查看文档:数据集格式说明。

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。

5.2.2 检测和实例分割数据集

VOCDetection 类

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

仅用于**目标检测**。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。

CocoDetection 类

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

用于**目标检测或实例分割**。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。

EasyDataDet 类

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

用于**目标检测或实例分割**。读取 EasyData 目标检测格式数据集，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。EasyData 目标检测或实例分割任务数据集格式的介绍可查看文档:数据集格式说明

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。

5.2.3 语义分割数据集

SegDataset 类

```
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。

EasyDataSeg 类

```
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。EasyData 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

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。

5.2.4 数据集转换

labelme2voc

```
pdx.tools.labelme2voc(image_dir, json_dir, dataset_save_dir)
```

将 LabelMe 标注的数据集转换为 VOC 数据集。

参数

- **image_dir** (str): 图像文件存放的路径。
- **json_dir** (str): 与每张图像对应的 json 文件的存放路径。
- **dataset_save_dir** (str): 转换后数据集存放路径。

其它数据集转换

easydata2imagenet

```
pdx.tools.easydata2imagenet(image_dir, json_dir, dataset_save_dir)
```

easydata2voc

```
pdx.tools.easydata2voc(image_dir, json_dir, dataset_save_dir)
```

easydata2coco

```
pdx.tools.easydata2coco(image_dir, json_dir, dataset_save_dir)
```

easydata2seg

```
pdx.tools.easydata2seg(image_dir, json_dir, dataset_save_dir)
```

labelme2coco

```
pdx.tools.labelme2coco(image_dir, json_dir, dataset_save_dir)
```

labelme2seg

```
pdx.tools.labelme2seg(image_dir, json_dir, dataset_save_dir)
```

jingling2seg

```
pdx.tools.jingling2seg(image_dir, json_dir, dataset_save_dir)
```

5.3 模型集-models

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。

5.3.1 图像分类

ResNet50 类

```
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 训练接口

```
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, 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。
- **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** (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 评估接口

```
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 预测接口

```
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): 预测图像路径。

- **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。

其它分类器类

PaddleX 提供了共计 22 种分类器，所有分类器均提供同 `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。

ResNet18

```
paddlex.cls.ResNet18(num_classes=1000)
```

ResNet34

```
paddlex.cls.ResNet34(num_classes=1000)
```

ResNet50

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

ResNet50_vd

```
paddlex.cls.ResNet50_vd(num_classes=1000)
```

ResNet50_vd_ssld

```
paddlex.cls.ResNet50_vd_ssld(num_classes=1000)
```

ResNet101

```
paddlex.cls.ResNet101(num_classes=1000)
```

ResNet101_vd

```
paddlex.cls.ResNet101_vdnum_classes=1000)
```

ResNet101_vd_ssld

```
paddlex.cls.ResNet101_vd_ssld(num_classes=1000)
```

DarkNet53

```
paddlex.cls.DarkNet53(num_classes=1000)
```

MobileNetV1

```
paddlex.cls.MobileNetV1(num_classes=1000)
```

MobileNetV2

```
paddlex.cls.MobileNetV2(num_classes=1000)
```

MobileNetV3_small

```
paddlex.cls.MobileNetV3_small(num_classes=1000)
```

MobileNetV3_small_ssld

```
paddlex.cls.MobileNetV3_small_ssld(num_classes=1000)
```

MobileNetV3_large

```
paddlex.cls.MobileNetV3_large(num_classes=1000)
```

MobileNetV3_large_ssld

```
paddlex.cls.MobileNetV3_large_ssld(num_classes=1000)
```

Xception65

```
paddlex.cls.Xception65(num_classes=1000)
```

Xception71

```
paddlex.cls.Xception71(num_classes=1000)
```

ShuffleNetV2

```
paddlex.cls.ShuffleNetV2(num_classes=1000)
```

DenseNet121

```
paddlex.cls.DenseNet121(num_classes=1000)
```

DenseNet161

```
paddlex.cls.DenseNet161(num_classes=1000)
```

DenseNet201

```
paddlex.cls.DenseNet201(num_classes=1000)
```

5.3.2 目标检测

YOLOv3 类

```
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 训练接口

```
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)
```

(continues on next page)

(continued from previous page)

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 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** (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 评估接口

```
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 预测接口

```
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): 预测图像路径。

- **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。

FasterRCNN 类

```
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’]。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 训练接口

```
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 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 接口

```
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 预测接口

```
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): 预测图像路径。
- **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。

5.3.3 实例分割

MaskRCNN 类

```
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’]。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 训练接口

```
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 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 评估接口

```
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 预测接口

```
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): 预测图像路径。
- **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。

5.3.4 语义分割

DeepLabv3p 类

```
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'],'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 训练接口

```
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 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 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** (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 评估接口

```
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 预测接口

```
predict(self, im_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): 预测图像路径。
- **transforms** (paddlex.seg.transforms): 数据预处理操作。

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)。

UNet 类

```
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

```
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='COCO', 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):
```

UNet 模型训练接口。

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 COCO 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’COCO’。
- **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 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** (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 评估接口

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

UNet 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 预测接口

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

UNet 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 `UNet.test_transforms` 和 `UNet.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): 预测图像路径。
- **transforms** (paddlex.seg.transforms): 数据预处理操作。

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)。

5.4 模型压缩-slim

5.4.1 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 查看裁剪教程

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。

5.4.2 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')
```

5.5 模型加载-load_model

PaddleX 提供了统一的模型加载接口，支持加载 PaddleX 保存的模型，并在验证集上进行评估或对测试图片进行预测

5.5.1 函数接口

```
paddlex.load_model(model_dir)
```

参数

- **model_dir**: 训练过程中保存的模型路径

返回值

- **paddlex.cv.models**, 模型类。

示例

1. 点击下载PaddleX 在小度熊分拣数据上训练的 MaskRCNN 模型
2. 点击下载小度熊分拣数据集

```
import paddlex as pdx

model_dir = './xiaoduxiong_epoch_12'
```

(continues on next page)

(continued from previous page)

```
data_dir = './xiaoduxiong_ins_det/JPEGImages'
ann_file = './xiaoduxiong_ins_det/val.json'

# 加载垃圾分拣模型
model = pdx.load_model(model_dir)

# 预测
pred_result = model.predict('./xiaoduxiong_ins_det/JPEGImages/WechatIMG114.jpeg')

# 在验证集上进行评估
eval_reader = pdx.cv.datasets.CocoDetection(data_dir=data_dir,
                                            ann_file=ann_file,
                                            transforms=model.eval_transforms)
eval_result = model.evaluate(eval_reader, batch_size=1)
```

5.6 可视化-visualize

PaddleX 提供了一系列模型预测和结果分析的可视化函数。

5.6.1 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): 原图文件路径。
- **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和测试图片

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

5.6.2 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。

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')
```

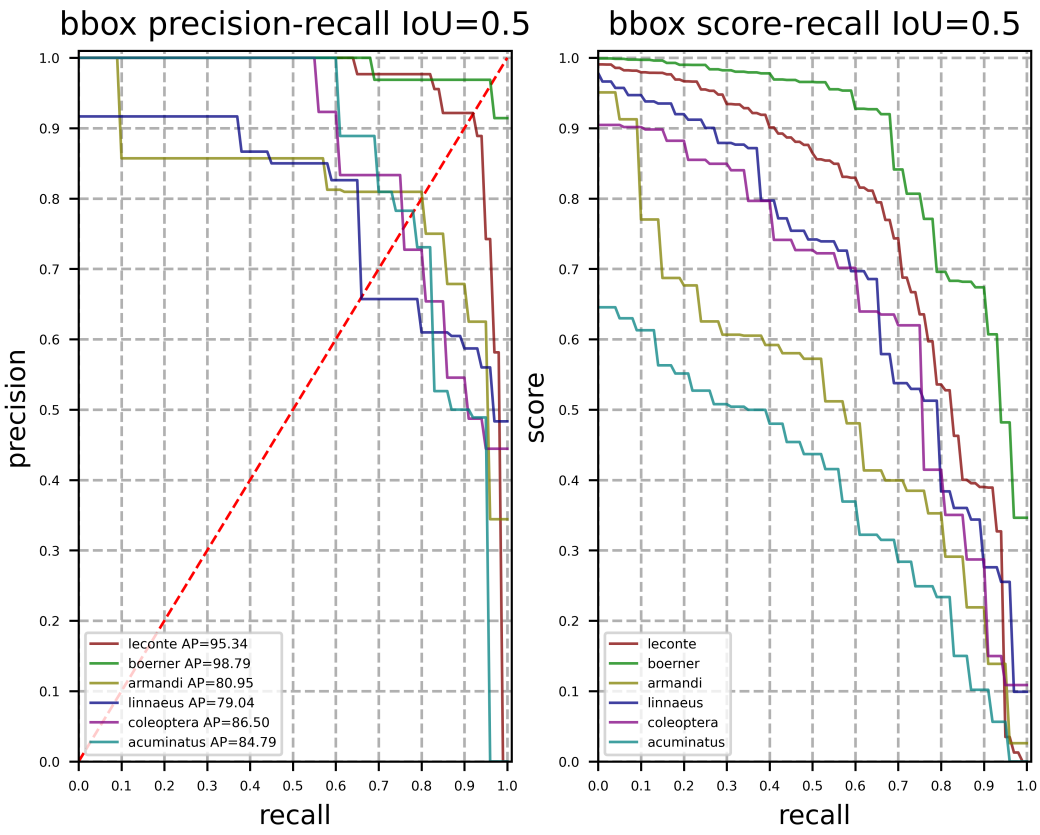
方式二：分析模型评估函数返回的评估结果。

```
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:



5.6.3 语义分割预测结果可视化

```
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): 原图文件路径。
- **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
```

5.6.4 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
```

5.6.5 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。

5.6.6 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='./')
```

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。NormLIME 会提前计算一定数量的测试样本的 LIME 结果，然后对相同的特征进行权重的归一化，这样来得到一个全局的输入和输出的关系。

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。

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。

使用示例

对预测可解释性结果可视化的过程可参见代码。

5.7 Predictor 部署-paddlex.deploy

使用 AnalysisPredictor 进行预测部署。

5.7.1 Predictor 类

```
paddlex.deploy.Predictor(model_dir, use_gpu=False, gpu_id=0, use_mkl=False, use_
↪trt=False, use_glog=False, memory_optimize=True)
```

参数

- **model_dir**: 训练过程中保存的模型路径, 注意需要使用导出的 inference 模型
- **use_gpu**: 是否使用 GPU 进行预测
- **gpu_id**: 使用的 GPU 序列号
- **use_mkl**: 是否使用 mkldnn 加速库
- **use_trt**: 是否使用 TensorRT 预测引擎
- **use_glog**: 是否打印中间日志
- **memory_optimize**: 是否优化内存使用

示例

```
import paddlex
model = paddlex.deploy.Predictor(model_dir, use_gpu=True)
result = model.predict(image_file)
```

predict 接口

```
predict(image, topk=1)
```

** 参数

- **image(str|np.ndarray)**: 待预测的图片路径或 np.ndarray，若为后者需注意为 BGR 格式
- **topk(int)**: 图像分类时使用的参数，表示预测前 topk 个可能的分类

CHAPTER 6

PaddleX GUI 使用文档

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！

6.1 PaddleX GUI 下载安装

6.2 PaddleX GUI 如何训练模型

6.3 其它

- PaddleX 版本: v0.1.7

- 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 7

更新日志

- 2020.05.20
- 发布正式版 v1.0
- 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
- 2020.05.17
- 发布 v0.1.8 pip 更新
- 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 8

常见问题

8.1 1. 训练参数如何调整

参考参数调整文档

8.2 2. 训练过程因显存不够出错

通过使用在终端 `nvidia-smi` 命令，查看 GPU 是否被其它任务占用，尝试清除其它任务；调低训练时的 `batch_size` 参数，从而降低显存的要求，注意需等比例调低 `learning_rate` 等参数；选用更小的模型或 backbone。

8.3 3. 是否有更小的模型，适用于更低配置的设备上运行

可以使用模型裁剪，参考文档模型裁剪使用教程，通过调整裁剪参数，可以控制模型裁剪后的大小，在实际实验中，如 VOC 检测数据，使用 yolov3-mobilenet，原模型大小为 XXM，裁剪后为 XX M，精度基本保持不变

8.4 4. 如何配置训练时 GPU 的卡数

通过在终端 export 环境变量，或在 Python 代码中设置，可参考文档CPU/多卡 GPU 训练

8.5 5. 想将之前训练的模型参数上继续训练

在训练调用 `train` 接口时，将 `pretrain_weights` 设为之前的模型保存路径即可

8.6 6. PaddleX 保存的模型分为正常训练过程中产生、裁剪训练产生、导出为部署模型和量化保存这么多种，有什么差别，怎么区分

不同模型的功能差异

1. 正常模型训练保存

   模型在正常训练过程，每间隔 n 个 epoch 保存的模型目录，模型可作为预训练模型参数，可使用 PaddleX 加载预测、或导出部署模型

2. 裁剪训练保存

   模型在裁剪训练过程，每间隔 n 个 epoch 保存的模型目录，模型不可作为预训练模型参数，可使用 PaddleX 加载预测、或导出部署模型

3. 导出部署模型

   为了模型在服务端部署，导出的模型目录，不可作为预训练模型参数，可使用 PaddleX 加载预测

4. 量化保存模型

   为了提升模型预测速度，将模型参数进行量化保存的模型目录，模型不可作为预训练模型参数，可使用 PaddleX 加载预测

区分方法

通过模型目录下 model.yml 文件中 `status` 字段来区别不同的模型类型，'Normal'、'Prune'、'Infer'、'Quant' 分别表示正常模型训练保存、裁剪训练保存、导出的部署模型、量化保存模型

8.7 7. 模型训练需要太久时间，或者训练速度太慢，怎么提速

1. 模型训练速度与用户选定的模型大小，和设定的 `batch_size` 相关，模型大小可直接参考模型库中的指标，一般而言，模型越大，训练速度就越慢；

2. 在模型速度之外，模型训练完成所需的时间又与用户设定的 `num_epochs` 迭代轮数相关，用户可以通过观察模型在验证集上的指标来决定是否提示结束掉训练进程（训练时设定 `save_interval_epochs` 参数，训练过程会每间隔 `save_interval_epochs` 轮数在验证集上计算指标，并保存模型）；

8.8 8. 如何设定迭代的轮数

1. 用户自行训练时，如不确定迭代的轮数，可以将轮数设高一些，同时注意设置 `save_interval_epochs`，这样模型迭代每间隔相应轮数就会在验证集上进行评估和保存，可以根据不同轮数模型在验证集上的评估指标，判断模型是否已经收敛，若模型已收敛，可以自行结束训练进程

8.9 9. 只有 CPU，没有 GPU，如何提升训练速度

当没有 GPU 时，可以根据自己的 CPU 配置，选择是否使用多 CPU 进行训练，具体配置方式可以参考文档多卡 CPU/GPU 训练

8.10 10. 电脑不能联网，训练时因为下载预训练模型失败，如何解决

可以预先通过其它方式准备好预训练模型，然后训练时自定义 `pretrain_weights` 即可，可参考文档无联网模型训练

8.11 11. 每次训练新的模型，都需要重新下载预训练模型，怎样可以下载一次就搞定

1. 可以按照 9 的方式来解决这个问题 2. 每次训练前都设定 `paddlex.pretrain_dir` 路径，如设定 `paddlex.pretrain_dir='/usrname/paddlex'`，如此下载完的预训练模型会存放至`/usrname/paddlex` 目录下，而已经下载在该目录的模型也不会再次重复下载

CHAPTER 9

附录

9.1 PaddleX 模型库

9.1.1 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。

9.1.2 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。

9.1.3 实例分割模型

表中模型相关指标均为在 MSCOCO 数据集上测试得到。

9.2 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。

9.2.1 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。

9.2.2 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。

9.2.3 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。

9.2.4 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:

```
2020-04-26 17:22:45 [INFO]      [TRAIN] Epoch=1/12, Step=216/221, loss=0.716083, loss_cls=0.379841, loss_bbox=0.272749, loss_rpn_cls=0.044854, loss_rpn_bbox=0.018638, lr=0.
001013, time_each_step=0.17s, eta=0:9:0
2020-04-26 17:22:46 [INFO]      [TRAIN] Epoch=1/12, Step=218/221, loss=0.536624, loss_cls=0.255749, loss_bbox=0.253681, loss_rpn_cls=0.011562, loss_rpn_bbox=0.015631, lr=0.
001014, time_each_step=0.17s, eta=0:9:0
2020-04-26 17:22:46 [INFO]      [TRAIN] Epoch=1/12, Step=220/221, loss=0.610345, loss_cls=0.308021, loss_bbox=0.248639, loss_rpn_cls=0.021446, loss_rpn_bbox=0.03224, lr=0.0
01016, time_each_step=0.17s, eta=0:9:0
2020-04-26 17:22:46 [INFO]      [TRAIN] Epoch 1 finished, loss=0.797818, loss_cls=0.39108, loss_bbox=0.305496, loss_rpn_cls=0.078625, loss_rpn_bbox=0.022617, lr=0.000925 .
```

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:

```
2020-04-26 17:53:16 [INFO]      [TRAIN] Epoch=1/12, Step=216/221, loss=1.049532, loss_cls=0.451329, loss_bbox=0.364285, loss_mask=0.199765, loss_rpn_cls=0.022088, loss_rpn_
bbox=0.012065, lr=0.000775, time_each_step=0.21s, eta=0:11:15
2020-04-26 17:53:16 [INFO]      [TRAIN] Epoch=1/12, Step=218/221, loss=0.849116, loss_cls=0.306857, loss_bbox=0.280559, loss_mask=0.237476, loss_rpn_cls=0.010742, loss_rpn_
bbox=0.013482, lr=0.000778, time_each_step=0.21s, eta=0:11:14
2020-04-26 17:53:16 [INFO]      [TRAIN] Epoch=1/12, Step=220/221, loss=1.07725, loss_cls=0.367705, loss_bbox=0.345688, loss_mask=0.31945, loss_rpn_cls=0.01434, loss_rpn_bbo
x=0.030067, lr=0.000782, time_each_step=0.21s, eta=0:11:14
2020-04-26 17:53:17 [INFO]      [TRAIN] Epoch 1 finished, loss=1.618055, loss_cls=0.575324, loss_bbox=0.383326, loss_mask=0.457685, loss_rpn_cls=0.182465, loss_rpn_bbox=0.0
19253, lr=0.0006 .
```

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

```
Start to evaluating(total_samples=245, total_steps=31)...
##################################################################
[EVAL] Finished, Epoch=2, bbox_map=9.2085 .
```

> 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。

9.2.5 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 .
```

9.3 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

9.3.1 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。

9.3.2 2.Batch Size 的调整

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）。这个参数跟机器的显存/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，如若用户调整 batch size，则也注意需要对应调整其它参数，如下表所示展示 YOLOv3 在训练时的参数配置

更多训练接口可以参考

- 分类模型-train
- 目标检测检测 FasterRCNN-train
- 目标检测 YOLOv3-train
- 实例分割 MaskRCNN-train
- 语义分割 DeepLabv3p-train
- 语义分割 UNet

9.4 数据集转换

当前 PaddleX GUI 支持 ImageNet 格式的图像分类数据集、VOC 格式的目标检测数据集、COCO 格式的实例分割数据集、Seg 格式的语义分割的数据集，当使用 LabelMe、EasyData、标注精灵这 3 个工具标注数据时，PaddleX 提供了相应接口可将数据转换成与 PaddleX GUI 想适配的数据集，使用方式如下所示：

```
import paddlex as pdx

# 该接口实现 LabelMe 数据集到 VOC 数据集的转换。
# image_dir 为图像文件存放的路径。
# json_dir 为与每张图像对应的 json 文件的存放路径。
# dataset_save_dir 为转换后数据集存放路径。
```

(continues on next page)

(continued from previous page)

```
pdx.tools.labelme2voc(image_dir='labelme_imgs',
                      json_dir='labelme_jsons',
                      dataset_save_dir='voc_dataset')
```

可替换 labelme2voc 实现不同数据集间的转换，目前提供的转换接口如下：

9.5 数据集格式说明

9.5.1 图像分类 ImageNet

图像分类 ImageNet 数据集包含对应多个标签的图像文件夹、标签文件及图像列表文件。参考数据文件结构如下：

```
./dataset/  # 数据集根目录
|--labelA  # 标签为 labelA 的图像目录
|  |--a1.jpg
|  |--...
|   --...
|
|--...
|
|--labelZ  # 标签为 labelZ 的图像目录
|  |--z1.jpg
|  |--...
|   --...
|
|--train_list.txt  # 训练文件列表文件
|
|--val_list.txt  # 验证文件列表文件
|
 --labels.txt  # 标签列表文件
```

其中，相应的文件名可根据需要自行定义。

`train_list.txt` 和 `val_list.txt` 文本以空格为分割符分为两列，第一列为图像文件相对于 dataset 的相对路径，第二列为图像文件对应的标签 id(从 0 开始)。如下所示：

```
labelA/a1.jpg 0
labelZ/z1.jpg 25
...
```

`labels.txt`: 每一行为一个单独的类别，相应的行号即为类别对应的 id（行号从 0 开始），如下所示：

```
labelA
labelB
...
```

点击这里，下载蔬菜分类分类数据集。在 PaddleX 中，使用 `paddlex.cv.datasets.ImageNet`(API 说明) 加载分类数据集。

9.5.2 目标检测 VOC

目标检测 VOC 数据集包含图像文件夹、标注信息文件夹、标签文件及图像列表文件。参考数据文件结构如下：

```
./dataset/  # 数据集根目录
|--JPEGImages  # 图像目录
|  |--xxx1.jpg
|  |--...
|   --...
|
|--Annotations  # 标注信息目录
|  |--xxx1.xml
|  |--...
|   --...
|
|--train_list.txt  # 训练文件列表文件
|
|--val_list.txt  # 验证文件列表文件
|
 --labels.txt  # 标签列表文件
```

其中，相应的文件名可根据需要自行定义。

`train_list.txt` 和 `val_list.txt` 文本以空格为分割符分为两列，第一列为图像文件相对于 dataset 的相对路径，第二列为标注文件相对于 dataset 的相对路径。如下所示：

```
JPEGImages/xxx1.jpg Annotations/xxx1.xml
JPEGImages/xxx2.jpg Annotations/xxx2.xml
```

(continues on next page)

(continued from previous page)

```
...
```

labels.txt: 每一行为一个单独的类别，相应的行号即为类别对应的 id（行号从 0 开始），如下所示：

```
labelA
labelB
...
```

点击这里，下载昆虫检测数据集。在 PaddleX 中，使用 paddlex.cv.datasets.VOCDetection(API 说明) 加载目标检测 VOC 数据集。

9.5.3 目标检测和实例分割 COCO

目标检测和实例分割 COCO 数据集包含图像文件夹及图像标注信息文件。参考数据文件结构如下：

```
./dataset/  # 数据集根目录
|--JPEGImages  # 图像目录
|  |--xxx1.jpg
|  |--...
|   --...
|
|--train.json  # 训练相关信息文件
|
 --val.json  # 验证相关信息文件
```

其中，相应的文件名可根据需要自行定义。

train.json 和 val.json 存储与标注信息、图像文件相关的信息。如下所示：

```
{
  "annotations": [
    {
      "iscrowd": 0,
      "category_id": 1,
      "id": 1,
      "area": 33672.0,
      "image_id": 1,
      "bbox": [232, 32, 138, 244],
      "segmentation": [[32, 168, 365, 117, ...]]
    },
    ...
```

(continues on next page)

(continued from previous page)

```
  ],
  "images": [
    {
      "file_name": "xxx1.jpg",
      "height": 512,
      "id": 267,
      "width": 612
    },
    ...
  ]
  "categories": [
    {
      "name": "labelA",
      "id": 1,
      "supercategory": "component"
    }
  ]
}
```

其中，每个字段的含义如下所示：

点击这里，下载垃圾实例分割数据集。在 PaddleX 中，使用 `paddlex.cv.datasets.COCODetection`(API 说明) 加载 COCO 格式数据集。

9.5.4 语义分割数据

语义分割数据集包含原图、标注图及相应的文件列表文件。参考数据文件结构如下：

```
./dataset/  # 数据集根目录
|--images  # 原图目录
|  |--xxx1.png
|  |--...
|  --...
|
|--annotations  # 标注图目录
|  |--xxx1.png
|  |--...
|  --...
|
|--train_list.txt  # 训练文件列表文件
|
```

(continues on next page)

(continued from previous page)

```
|--val_list.txt  # 验证文件列表文件
|
 --labels.txt  # 标签列表
```

其中，相应的文件名可根据需要自行定义。

`train_list.txt` 和 `val_list.txt` 文本以空格为分割符分为两列，第一列为图像文件相对于 dataset 的相对路径，第二列为标注图像文件相对于 dataset 的相对路径。如下所示：

```
images/xxx1.png annotations/xxx1.png
images/xxx2.png annotations/xxx2.png
...
```

`labels.txt`: 每一行为一个单独的类别，相应的行号即为类别对应的 id（行号从 0 开始），如下所示：

```
background
labelA
labelB
...
```

标注图像为单通道图像，像素值即为对应的类别，像素标注类别需要从 0 开始递增（一般第一个类别为 `background`），例如 0，1，2，3 表示有 4 种类别，标注类别最多为 256 类。其中可以指定特定的像素值用于表示该值的像素不参与训练和评估（默认为 255）。

点击这里，下载视盘语义分割数据集。在 PaddleX 中，使用 `paddlex.cv.datasets.SegReader`(API 说明) 加载语义分割数据集。

9.5.5 图像分类 EasyDataCls

图像分类 EasyDataCls 数据集包含存放图像和 json 文件的文件夹、标签文件及图像列表文件。参考数据文件结构如下：

```
./dataset/  # 数据集根目录
|--easydata  # 存放图像和 json 文件的文件夹
|  |--0001.jpg
|  |--0001.json
|  |--0002.jpg
|  |--0002.json
|   --...
|
|--train_list.txt  # 训练文件列表文件
|
```

(continues on next page)

(continued from previous page)

```
|--val_list.txt  # 验证文件列表文件
|
 --labels.txt  # 标签列表文件
```

其中，图像文件名应与 json 文件名一一对应。

每个 json 文件存储于 `labels` 相关的信息。如下所示：

```
{"labels": [{"name": "labelA"}]}
```

其中，`name` 字段代表对应图像的类别。

`train_list.txt` 和 `val_list.txt` 文本以空格为分割符分为两列，第一列为图像文件相对于 dataset 的相对路径，第二列为 json 文件相对于 dataset 的相对路径。如下所示：

```
easydata/0001.jpg easydata/0001.json
easydata/0002.jpg easydata/0002.json
...
```

`labels.txt`: 每一行为一个单独的类别，相应的行号即为类别对应的 id（行号从 0 开始），如下所示：

```
labelA
labelB
...
```

点击这里，可以标注图像分类 EasyDataCls 数据集。在 PaddleX 中，使用 `paddlex.cv.datasets.EasyDataCls`(API 说明) 加载分类数据集。

9.5.6 目标检测和实例分割 EasyDataDet

目标检测和实例分割 EasyDataDet 数据集包含存放图像和 json 文件的文件夹、标签文件及图像列表文件。参考数据文件结构如下：

```
./dataset/  # 数据集根目录 ß
|--easydata  # 存放图像和 json 文件的文件夹
|  |--0001.jpg
|  |--0001.json
|  |--0002.jpg
|  |--0002.json
|   --...
|
|--train_list.txt  # 训练文件列表文件
```

(continues on next page)

(continued from previous page)

```
|
|--val_list.txt  # 验证文件列表文件
|
 --labels.txt  # 标签列表文件
```

其中，图像文件名应与 json 文件名一一对应。

每个 json 文件存储于 `labels` 相关的信息。如下所示：

```
"labels": [{"y1": 18, "x2": 883, "x1": 371, "y2": 404, "name": "labelA",
            "mask": "kVfc0`0Zg0
↪<F7J7I5L5K4L4L4L3N3L3N3L3N2N3M2N2N2N2N2N2N1O2N2O1N2N1O2O1N1O1N1O2O1N1O1N10001N1O1N10001N10001O0O10001
↪"},
           {"y1": 314, "x2": 666, "x1": 227, "y2": 676, "name": "labelB",
            "mask":
↪"mdQ8g0Tg0:G8I6K5J5L4L4L4L4M2M4M2M4M2N2N2N3L3N2N2N2N2O1N1O2N2N2O1N1O2N2OOO2O1N1O2OOO2OOO2OOO1N10OO2OO
↪BUcd<"},
           ...]}
```

其中，list 中的每个元素代表一个标注信息，标注信息中字段的含义如下所示：

`train_list.txt` 和 `val_list.txt` 文本以空格为分割符分为两列，第一列为图像文件相对于 dataset 的相对路径，第二列为 json 文件相对于 dataset 的相对路径。如下所示：

```
easydata/0001.jpg easydata/0001.json
easydata/0002.jpg easydata/0002.json
...
```

`labels.txt`: 每一行为一个单独的类别，相应的行号即为类别对应的 id（行号从 0 开始），如下所示：

```
labelA
labelB
...
```

点击这里，可以标注图像分类 EasyDataDet 数据集。在 PaddleX 中，使用 `paddlex.cv.datasets.EasyDataDet`(API 说明) 加载分类数据集。

9.5.7 语义分割 EasyDataSeg

语义分割 EasyDataSeg 数据集包含存放图像和 json 文件的文件夹、标签文件及图像列表文件。参考数据文件结构如下：

```
./dataset/  # 数据集根目录 ß
|--easydata  # 存放图像和 json 文件的文件夹
|  |--0001.jpg
|  |--0001.json
|  |--0002.jpg
|  |--0002.json
|   --...
|
|--train_list.txt  # 训练文件列表文件
|
|--val_list.txt  # 验证文件列表文件
|
 --labels.txt  # 标签列表文件
```

其中，图像文件名应与 json 文件名一一对应。

每个 json 文件存储于 `labels` 相关的信息。如下所示：

```
"labels": [{"y1": 18, "x2": 883, "x1": 371, "y2": 404, "name": "labelA",
            "mask": "kVfc0`0Zg0
↪<F7J7I5L5K4L4L4L3N3L3N3L3N2N3M2N2N2N2N2N2N1O2N2O1N2N1O2O1N101N1O2O1N101N10001N101N10001N10001O0O10001
↪"},
           {"y1": 314, "x2": 666, "x1": 227, "y2": 676, "name": "labelB",
            "mask":
↪"mdQ8g0Tg0:G8I6K5J5L4L4L4L4M2M4M2M4M2N2N2N3L3N2N2N2N2O1N1O2N2N2O1N1O2N2OOO2O1N1O2OOO2OOO2O001N100O2OO
↪BUcd<"},
           ...]}
```

其中，list 中的每个元素代表一个标注信息，标注信息中字段的含义如下所示：

`train_list.txt` 和 `val_list.txt` 文本以空格为分割符分为两列，第一列为图像文件相对于 dataset 的相对路径，第二列为 json 文件相对于 dataset 的相对路径。如下所示：

```
easydata/0001.jpg easydata/0001.json
easydata/0002.jpg easydata/0002.json
...
```

`labels.txt`: 每一行为一个单独的类别，相应的行号即为类别对应的 id（行号从 0 开始），如下所示：

```
labelA
labelB
...
```

点击这里，可以标注图像分类 EasyDataSeg 数据集。在 PaddleX 中，使用 `paddlex.cv.datasets.EasyDataSeg`(API 说明) 加载分类数据集。

- PaddleX 版本: v1.0.0
- 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