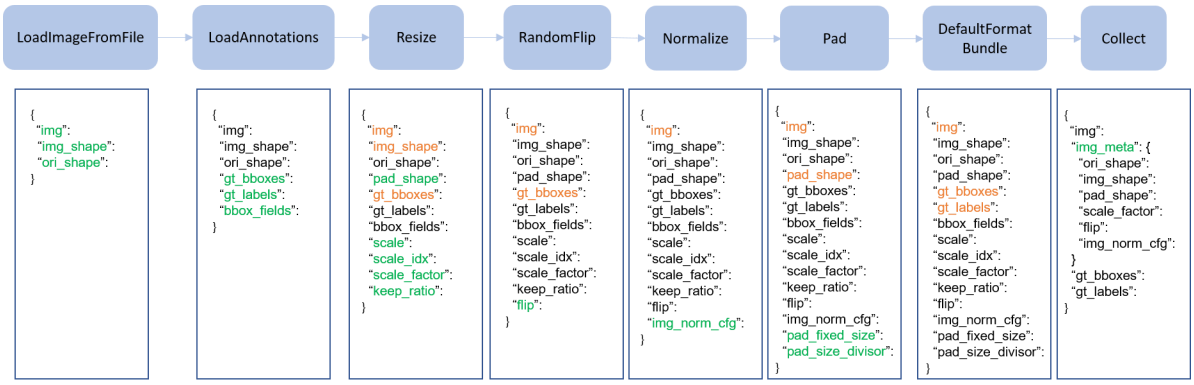
mmdetection2.6自定义数据集pipeline

1. data pipeline的基本使用

mmdetection的数据读取方式分为两部分，第一部分为数据集，第二部分为data pipeline，通常数据集定义如何处理标注信息，而pipeline定义处理数据字典的所有步骤，一个pipeline由一系列的操作组成，每一个操作都采用一个dict作为输入，并且也输出一个dict作为下一个操作的输入。

下图是一个经典的pipeline，蓝色的方块是piplline的基本操作，随着pipeline的移动，每个操作都可以加入新的键（绿色的字）作为结果的dict。并且更新已经存在的键值（橘色的字）。



这些操作被分为数据加载，预处理，格式化，测试时增强。

faster rcnn的pipeline的例子：

```
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True),
    dict(type='Resize', img_scale=(1333, 800), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels']),
]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(1333, 800),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
```

```
            dict(type='RandomFlip'),
            dict(type='Normalize', **img_norm_cfg),
            dict(type='Pad', size_divisor=32),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img']),
        ])
]
```

下边列出了各种operation添加，更新与移除的dict行为。

1.1 数据加载

- `LoadImageFromFile`

添加：img，img_shape，ori_shape

- `LoadAnnotations`

添加：gt_bboxes，gt_bboxes_ignore，gt_labels，gt_masks，gt_semantic_seg，bbox_fields，mask_fields

- `LoadProposals`

添加：proposals

1.2 预处理

- `Resize`

添加：scale，scale_idx，pad_shape，scale_factor，keep_ratio
更新：img，img_shape，* bbox_fields，* mask_fields，* seg_fields

- `RandomFlip`

添加：flip
更新：img，* bbox_fields，* mask_fields，* seg_fields

- `Pad`

添加：pad_fixed_size，pad_size_divisor
更新：img，pad_shape，* mask_fields，* seg_fields

- `RandomCrop`

更新：img，pad_shape，gt_bboxes，gt_labels，gt_masks，* bbox_fields

- `Normalize`

添加：img_norm_cfg
更新：img

- `SegRescale`

更新：gt_semantic_seg

- `PhotoMetricDistortion`

更新：img

- `Expand`

更新：img，gt_bboxes

- `MinIoURandomCrop`

更新：img，gt_bboxes，gt_labels

- `Corrupt`

更新：img

1.3 格式化

- `ToTensor`

更新：specified by `keys`.

- `ImageToTensor`

更新：specified by `keys`.

- `Transpose`

更新：specified by `keys`.

- `ToDataContainer`

更新：specified by `fields`.

- `DefaultFormatBundle`

更新：img，proposals，gt_bboxes，gt_bboxes_ignore，gt_labels，gt_masks，gt_semantic_seg

- `Collect`

添加：img_meta（img_meta的键由指定`meta_keys`）
删除：除由所指定的键外的所有其他键

1.4 测试时增强

- `MultiScaleFlipAug`

2. 扩展并使用自定义的pipeline

1. 新建一个python文件，例如：`my_pipeline.py`，输入为一个dict输出也为一个dict

```
from mmdet.datasets import PIPELINES

@PIPELINES.register_module()
class MyTransform:

    def __call__(self, results):
        results['dummy'] = True
        return results
```

2. Import这个新类

```
from .my_pipeline import MyTransform
```

3. 在config文件中使用这个pipeline

```
img_norm_cfg = dict(
    mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True),
    dict(type='Resize', img_scale=(1333, 800), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='MyTransform'),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels']),
]
```