mmdetection2.6 自定义模型

mmdetection将目标检测模型的基本组件分为五类：

- backbone： 利用全卷积网络来提取特征，例如resnet或者mobilenet
- neck： 在beckbone与head之间的部分，例如FPN与PAFPN
- head：针对特定任务的组成部分，例如：bbox预测，mask预测
- roi extractor： 从feature map中提取roi 特征，例如： ROI Align
- loss： head中计算loss的部分。例如与focal loss 与 GHM loss

每个部分的添加基本上要经历以下几个过程：

定义新的模块——>导入模块——> 在config文件中使用这个模块

一、 添加新的backbone

这里利用mobilenet展示如何开发新的组件。

1.1 定义新的backbone（例如mobilenet）

创建一个新文件 `mmdet/models/backbones/mobilenet.py`.

```
import torch.nn as nn

from ..builder import BACKBONES


@BACKBONES.register_module()
class MobileNet(nn.Module):

    def __init__(self, arg1, arg2):
        pass

    def forward(self, x):  # should return a tuple
        pass

    def init_weights(self, pretrained=None):
        pass
```

1.2 导入模块

两种方法:

1. 加入下边一行代码到`mmdet/models/backbones/__init__.py`中。

```
from .mobilenet import MobileNet
```

2. 不修改原始框架的代码，在config文件中加入下边的代码:

```
custom_imports = dict(
    imports=['mmdet.models.backbones.mobilenet'],
    allow_failed_imports=False)
```

1.3 在config文件中使用backbone

```
model = dict(
    ...
    backbone=dict(
        type='MobileNet',
        arg1=xxx,
        arg2=xxx),
    ...
```

二、 添加新的necks

2.1 定义一个neck

创建一个新的文件`mmdet/models/necks/pafpn.py`.

```
from ..builder import NECKS

@NECKS.register
class PAFPN(nn.Module):

    def __init__(self,
                 in_channels,
                 out_channels,
                 num_outs,
                 start_level=0,
                 end_level=-1,
                 add_extra_convs=False):
        pass

    def forward(self, inputs):
        # implementation is ignored
        pass
```

2.2 导入这个模块

两种方法：

1. 加入下边一行代码到`mmdet/models/necks/__init__.py`中。

```
from .pafpn import PAFPN
```

2. 不修改原始框架的代码，在config文件中加入下边的代码：

```
custom_imports = dict(
    imports=['mmdet.models.necks.pafpn'],
    allow_failed_imports=False)
```

2.3 在config文件中使用新的neck

```
neck=dict(
    type='PAFPN',
    in_channels=[256, 512, 1024, 2048],
    out_channels=256,
    num_outs=5)
```

三、 添加新的heads

以Double Head R-CNN作为例子来开发一个新的head。

2.1 定义一个head

首先新建一个文件`mmdet/models/roi_heads/bbox_heads/double_bbox_head.py`， 并在其中加入一个新的bbox head。Double Head R-CNN采用了一个新型的bbox head来进行 目标检测，为了应用bbox head，需要实现下边所示的三个模块。

```
from mmdet.models.builder import HEADS
from .bbox_head import BBoxHead

@HEADS.register_module()
class DoubleConvFCBBoxHead(BBoxHead):
    r"""Bbox head used in Double-Head R-CNN

                                    /-> cls
                 /-> shared convs ->
                                    \-> reg
    roi features
                                    /-> cls
                 \-> shared fc    ->
                                    \-> reg
    """  # noqa: W605
```

```
    def __init__(self,
                 num_convs=0,
                 num_fcs=0,
                 conv_out_channels=1024,
                 fc_out_channels=1024,
                 conv_cfg=None,
                 norm_cfg=dict(type='BN'),
                 **kwargs):
        kwargs.setdefault('with_avg_pool', True)
        super(DoubleConvFCBBoxHead, self).__init__(**kwargs)

    def init_weights(self):
        # conv layers are already initialized by ConvModule

    def forward(self, x_cls, x_reg):
```

接下来， 如果必要的话，还需要实现ROI head，从`StandardRoIHead`继承`DoubleHeadRoIHead`。`standardRoIHead`已经实现了如下的函数：

```
import torch

from mmdet.core import bbox2result, bbox2roi, build_assigner, build_sampler
from ..builder import HEADS, build_head, build_roi_extractor
from .base_roi_head import BaseRoIHead
from .test_mixins import BBoxTestMixin, MaskTestMixin


@HEADS.register_module()
class StandardRoIHead(BaseRoIHead, BBoxTestMixin, MaskTestMixin):
    """Simplest base roi head including one bbox head and one mask head.
    """

    def init_assigner_sampler(self):

    def init_bbox_head(self, bbox_roi_extractor, bbox_head):

    def init_mask_head(self, mask_roi_extractor, mask_head):

    def init_weights(self, pretrained):

    def forward_dummy(self, x, proposals):


    def forward_train(self,
                      x,
                      img_metas,
                      proposal_list,
                      gt_bboxes,
                      gt_labels,
                      gt_bboxes_ignore=None,
                      gt_masks=None):

    def _bbox_forward(self, x, rois):
```

```
    def _bbox_forward_train(self, x, sampling_results, gt_bboxes, gt_labels,
                            img_metas):

    def _mask_forward_train(self, x, sampling_results, bbox_feats, gt_masks,
                            img_metas):

    def _mask_forward(self, x, rois=None, pos_inds=None, bbox_feats=None):


    def simple_test(self,
                    x,
                    proposal_list,
                    img_metas,
                    proposals=None,
                    rescale=False):
        """Test without augmentation."""
```

Double Head主要修改bbox_forward的逻辑部分，并且直接从`StandardRoIHead`继承其他的函数。

在`mmdet/models/roi_heads/double_roi_head.py`中实现新的ROI Head

```
from ..builder import HEADS
from .standard_roi_head import StandardRoIHead


@HEADS.register_module()
class DoubleHeadRoIHead(StandardRoIHead):
    """RoI head for Double Head RCNN

    https://arxiv.org/abs/1904.06493
    """

    def __init__(self, reg_roi_scale_factor, **kwargs):
        super(DoubleHeadRoIHead, self).__init__(**kwargs)
        self.reg_roi_scale_factor = reg_roi_scale_factor

    def _bbox_forward(self, x, rois):
        bbox_cls_feats = self.bbox_roi_extractor(
            x[:self.bbox_roi_extractor.num_inputs], rois)
        bbox_reg_feats = self.bbox_roi_extractor(
            x[:self.bbox_roi_extractor.num_inputs],
            rois,
            roi_scale_factor=self.reg_roi_scale_factor)
        if self.with_shared_head:
            bbox_cls_feats = self.shared_head(bbox_cls_feats)
            bbox_reg_feats = self.shared_head(bbox_reg_feats)
        cls_score, bbox_pred = self.bbox_head(bbox_cls_feats, bbox_reg_feats)

        bbox_results = dict(
            cls_score=cls_score,
            bbox_pred=bbox_pred,
            bbox_feats=bbox_cls_feats)
        return bbox_results
```

2.2 导入这个模块

两种方法:

1. 将实现的模块加入到`mmdet/models/bbox_heads/__init__.py`与`mmdet/models/roi_heads/__init__.py`中

```
from .double_bbox_head import DoubleConvFCBBoxHead
from .double_roi_head import DoubleHeadRoIHead
```

2. 不修改原始框架的代码，在config文件中加入下边的代码:

```
custom_imports=dict(
    imports=['mmdet.models.roi_heads.double_roi_head',
'mmdet.models.bbox_heads.double_bbox_head'])
```

2.3 在config文件中使用新的head

```
#  Double Head R-CNN的config文件
_base_ = '../faster_rcnn/faster_rcnn_r50_fpn_1x_coco.py'
model = dict(
    roi_head=dict(
        type='DoubleHeadRoIHead',
        reg_roi_scale_factor=1.3,
        bbox_head=dict(
            _delete_=True,
            type='DoubleConvFCBBoxHead',
            num_convs=4,
            num_fcs=2,
            in_channels=256,
            conv_out_channels=1024,
            fc_out_channels=1024,
            roi_feat_size=7,
            num_classes=80,
            bbox_coder=dict(
                type='DeltaXYWHBBoxCoder',
                target_means=[0., 0., 0., 0.],
                target_stds=[0.1, 0.1, 0.2, 0.2]),
            reg_class_agnostic=False,
            loss_cls=dict(
                type='CrossEntropyLoss', use_sigmoid=False, loss_weight=2.0),
            loss_bbox=dict(type='SmoothL1Loss', beta=1.0, loss_weight=2.0))))
```

四、 添加新的losses

2.1 定义一个losss

假设定义一个bbox回归的loss 函数`Myloss`, 新建一个文件`mmdet/models/losses/my_loss.py`

```
import torch
import torch.nn as nn

from ..builder import LOSSES
from .utils import weighted_loss

@weighted_loss
def my_loss(pred, target):
    assert pred.size() == target.size() and target.numel() > 0
    loss = torch.abs(pred - target)
    return loss

@LOSSES.register_module()
class MyLoss(nn.Module):

    def __init__(self, reduction='mean', loss_weight=1.0):
        super(MyLoss, self).__init__()
        self.reduction = reduction
        self.loss_weight = loss_weight

    def forward(self,
                pred,
                target,
                weight=None,
                avg_factor=None,
                reduction_override=None):
        assert reduction_override in (None, 'none', 'mean', 'sum')
        reduction = (
            reduction_override if reduction_override else self.reduction)
        loss_bbox = self.loss_weight * my_loss(
            pred, target, weight, reduction=reduction, avg_factor=avg_factor)
        return loss_bbox
```

2.2 导入这个模块

两种方法:

1. 加入下边一行代码到`mmdet/models/losses/__init__.py`中。

   ```
   from .my_loss import MyLoss, my_loss
   ```

2. 不修改原始框架的代码，在config文件中加入下边的代码:

   ```
   custom_imports=dict(
       imports=['mmdet.models.losses.my_loss'])
   ```

2.3 在config文件中使用新的loss

使用时，针对定义的loss函数的作用，直接修改相应的`loss_xxx`字段，这里的myloss是bbox回归，因此修改下边的字段：

```
loss_bbox=dict(type='MyLoss', loss_weight=1.0))
```

专栏所有文章请点击下列文章列表查看：

知乎专栏：小哲AI专栏文章分类索引跳转查看

AI研习社专栏：小哲AI专栏文章分类索引