# mmdetection 解析

Sisyphes

2019 年 8 月 1 日

## 目录

# 第一节　结构设计

- BackBone: 特征提取骨架网络,ResNet,ResneXt 等.
- Neck: 连接骨架和头部. 多层级特征融合,FPN,BFP 等.
- DenseHead: 处理特征图上的密集框部分，主要分 AnchorHead，AnchorFreeHead 两大类，分别有 RPNHead，SSDHead,RetinaHead 和 FCOSHead 等.
- RoIExtractor: 汇集不同层级的特征框, 将其大小统一, 为二步定位, 类别优化服务.
- RoIHead (BBoxHead/MaskHead): 类别分类或位置回归等.
- OneStage: BackBone + Neck + DenseHead
- TwoStage: BackBone + Neck + (DenseHead) + RoIExtractor + RoIHead

代码结构:
configs 网络组件结构等配置信息
tools: 训练和测试的最终包装
mmdet:

apis: 分布式环境设定, 推断和训练基类代码
core: anchor,bbox,mask 等在训练前和训练中的各种变换函数
datasets:coco 和 voc 格式的数据类以及一些增强代码
models: 模型组件，采用注册和组合构建的形式完成模型搭建
ops: 优化加速代码, 包括 nms,roialign,dcn,gcb,mask，focal_loss 等

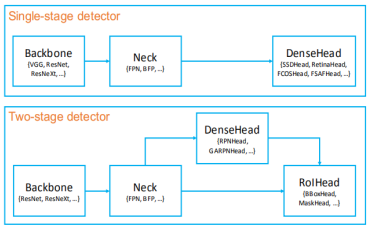


图 1: Framework

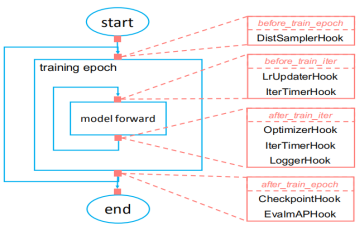


图 2: Trainning pipeline

## 1.1 总体逻辑

在最外层的 train.py 中能看到:

1. mmcv.Config.fromfile 从配置文件解析配置信息, 并做适当更新, 包括预加载模型文件, 分布式相关等
2. mmdet.models.builder 中的 build_detector 根据配置信息构造模型

   2.5 build 函数依次遍历配置中的模型信息, 调用 _build_module 函数, 按 type 关键字从注册表中获取相应的模型对象, 并根据配置参数实例化对象 (配置文件的模型参数只占了各模型构造参数的一小部分, 模型结构并非可以随意更改).

   2.6 registr.py 实现了模型的注册装饰器, 其主要功能就是将各模型组件类对象保存到 registry.module_dict 中, 从而可以实现 2.5 所示功能.

   2.7 目前包含 BACKBONES,NECKS,ROI_EXTRACTORS,SHARED_HEADS,HEADS,LOSSES,DETECTORS 七个 (容器). 注册器可按 @NAME.register_module 方式装饰, 新增. 所有被注册的对象都是一个完整的 pytorch 构图
3. 最后就是数据迭代器和训练 pipeline

## 1.2 Configs

配置方式支持 python/json/yaml, 从 mmcv 的 Config 解析, 其功能同 maskrcnn-benchmark 的 yacs 类似, 将字典的取值方式属性化.

配置文件模型部分包含模型组件及其可改动模型结构的参数, 比如 backbone 的层数, 冻结的 stage;bbox_head 的 in_channel, 类别, 损失函数等; 训练部分主要包括 anchor 采样相关系数; 测试包括非极大抑制等相关参数; 剩下数据, 优化器, 模型管理, 日志等相关信息, 一看即明.

## 1.3 Backbone

特征提取, 略.

## 1.4 Necks

展开描述: 目标尺度和金字塔层级的关系, 这些层级对应的尺度与损失函数, 样本采样等的关系.

### 1.5 Heads

anchor_heads, bbox_heads, mask_heads 都将有 loss 的计算, 是核心点, 需详细描述.

### 1.6 Losses

选描述

### 1.7 Detectors

重点 one_stage, two_stage 详细阐述

## 第二节　数据处理

coco 数据等格式，多格式的转换

## 第三节　模型结构

详细解释一两个模型的细节, 比如正负样本采样，损失函数改写,anchor 相关尺度变换的细节

```
import numpy  as np
class Convolution():

  def __init__(self, nc_in, nc_out, kernel_size, stride=2,padding=1):
    self.kernel_size = kernel_size
    self.weights = np.random.randn(nc_in * kernel_size[0] * kernel_size[1] ,
     nc_out) * np.sqrt(2/nc_in)
    self.biases = np.zeros(nc_out)
    self.stride = stride
    self.padding = padding

  def forward(self,x):
    mb, ch, n, p = x.shape
    y = np.matmul(arr2vec(x,self.kernel_size,self.stride,self.padding), self
     .weights) + self.biases
    y = np.transpose(y,(0,2,1))
    n1 = (n-self.kernel_size[0]+ 2 * self.padding) //self.stride + 1
    p1 = (p-self.kernel_size[1]+2 * self.padding )//self.stride + 1
    return y.reshape(mb,self.biases.shape[0],n1,p1)
```

```
18
19      def backward(self,grad):
20          """
21          写点字可以?
22          """
23      mb, ch_out, n1, p1 = grad.shape
24      grad = np.transpose(grad.reshape(mb,ch_out,n1*p1),(0,2,1))
25      self.grad_b = grad.sum(axis=1).mean(axis=0)
26      self.grad_w = (np.matmul(self.old_x[:,:,:,None],grad[:,:,None,:])).sum(
        axis=1).mean(axis=0)
27      new_grad = np.matmul(grad,self.weights.transpose())
28      return vec2arr(new_grad, self.kernel_size, self.old_size, self.stride,
        self.padding)
```

OneStage，TwoStage, 其他

## 第四节　训练 pipeline

Hook 相关

## 第五节　更改模型

案例

## 第六节　新增模型

RetinaFace, BlazeFace, FaceBoxes 等