

参数解析

```
MODEL:  
    META_ARCHITECTURE: "GeneralizedRCNN"  
    WEIGHT: "catalog://ImageNetPretrained/MSRA/R-50" # imagenet 预训练模型  
    RPN_ONLY: True # ?  
    ATSS_ON: True # 使用 Adaptive Training Sample Selection 策略  
    BACKBONE:  
        CONV_BODY: "R-50-FPN-RETINANET" # backbone + FPN 网络结构  
    RESNETS:  
        BACKBONE_OUT_CHANNELS: 256  
    RETINANET:  
        USE_C5: False # ?  
    ATSS:  
        ANCHOR_SIZES: (64, 128, 256, 512, 1024) # 8S ?  
        ASPECT RATIOS: (1.0,) # ATSS 里面不用配置 anchor 的长宽比  
        SCALES_PER_OCTAVE: 1 # ATSS 里面不用配置 anchor 的大小  
        USE_DCN_IN_TOWER: False # 是否使用 deformable convolution  
        POSITIVE_TYPE: 'ATSS' # how to select positives: ATSS (Ours) , SSC (FCOS),  
        IoU (RetinaNet)  
        TOPK: 9 # 每一层选取的候选 anchor 数量  
        REGRESSION_TYPE: 'BOX' # 回归点还是框, ATSS 只是一种 anchor 自动匹配策略, 可以适应  
        两种, BOX是RetinaNet, POINT是FCOS  
    DATASETS:  
        TRAIN: ("coco_2017_train",)  
        TEST: ("coco_2017_val",)  
    INPUT:  
        MIN_SIZE_TRAIN: (800,)  
        MAX_SIZE_TRAIN: 1333  
        MIN_SIZE_TEST: 800  
        MAX_SIZE_TEST: 1333  
    DATALOADER:  
        SIZE_DIVISIBILITY: 32  
    SOLVER:  
        BASE_LR: 0.01  
        WEIGHT_DECAY: 0.0001  
        STEPS: (60000, 80000)  
        MAX_ITER: 90000  
        IMS_PER_BATCH: 4  
        WARMUP_METHOD: "constant"
```

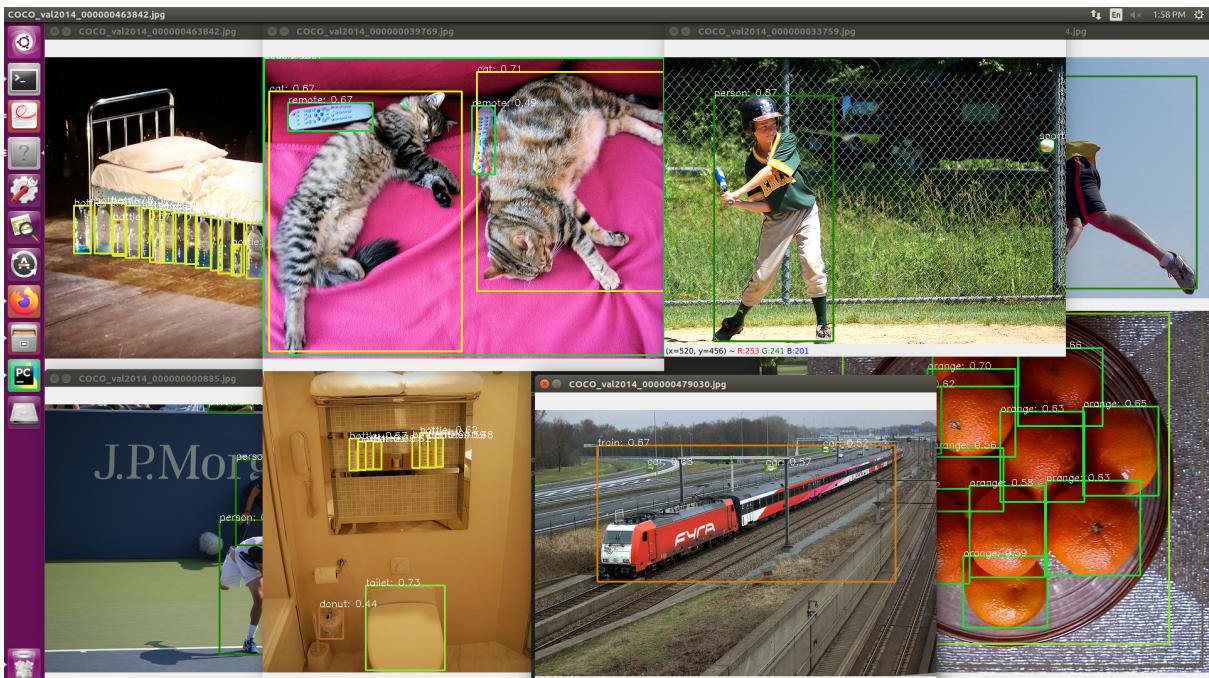
运行demo

1. 安装

```
conda create -n ATSS python=3.7
conda activate ATSS
conda install ipython
pip install ninja yacs cython matplotlib tqdm
conda install -c pytorch torchvision cudatoolkit=10.0
conda install -c conda-forge pycocotools
git clone https://github.com/sfzhang15/ATSS.git
cd ATSS
python setup.py build develop --no-deps
```

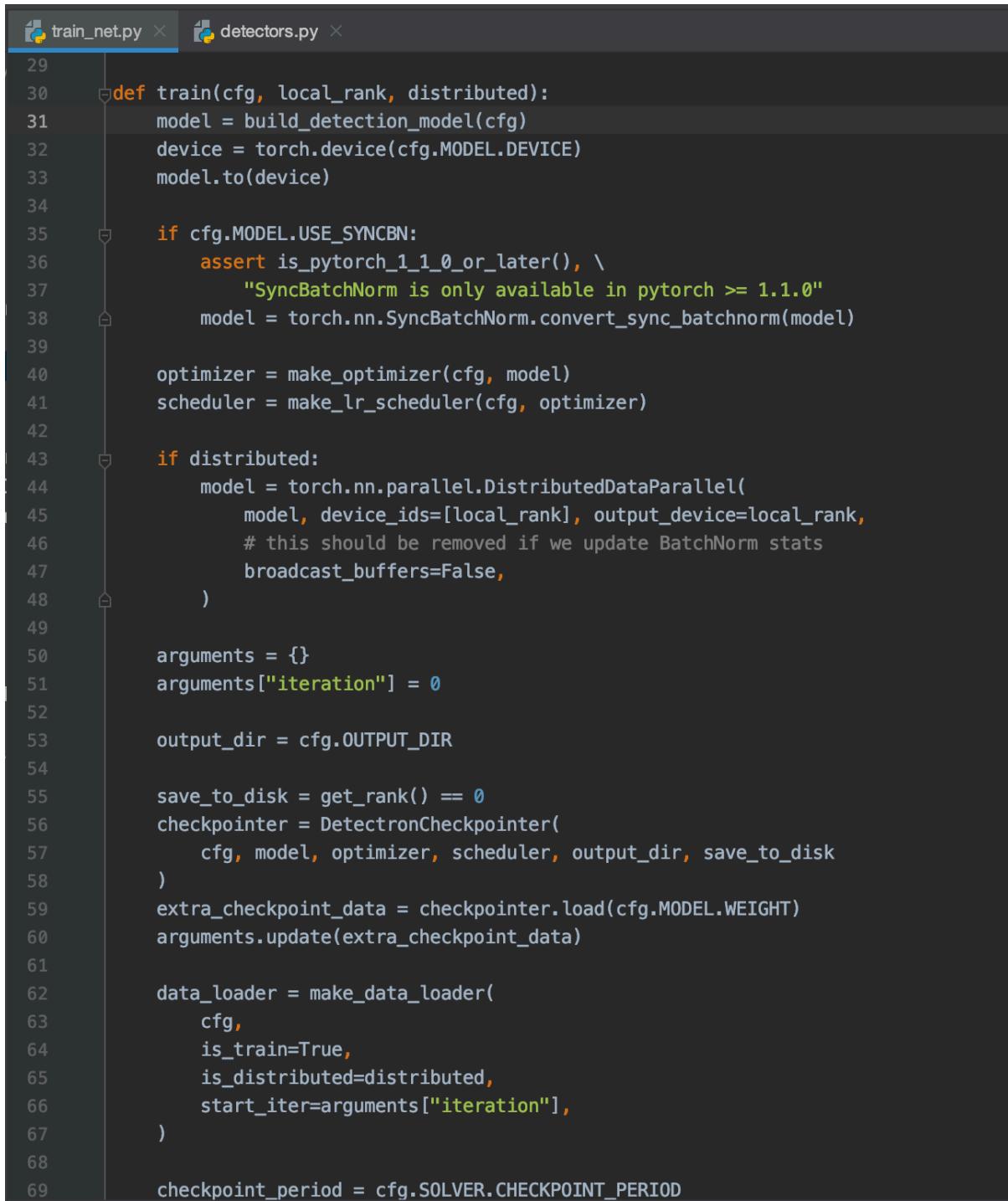
2. 运行 demo

```
python demo/atss_demo.py
```



调试

1. 构建模型



```
29
30     def train(cfg, local_rank, distributed):
31         model = build_detection_model(cfg)
32         device = torch.device(cfg.MODEL.DEVICE)
33         model.to(device)
34
35         if cfg.MODEL.USE_SYNCBN:
36             assert is_pytorch_1_1_0_or_later(), \
37                 "SyncBatchNorm is only available in pytorch >= 1.1.0"
38             model = torch.nn.SyncBatchNorm.convert_sync_batchnorm(model)
39
40         optimizer = make_optimizer(cfg, model)
41         scheduler = make_lr_scheduler(cfg, optimizer)
42
43         if distributed:
44             model = torch.nn.parallel.DistributedDataParallel(
45                 model, device_ids=[local_rank], output_device=local_rank,
46                 # this should be removed if we update BatchNorm stats
47                 broadcast_buffers=False,
48             )
49
50         arguments = {}
51         arguments["iteration"] = 0
52
53         output_dir = cfg.OUTPUT_DIR
54
55         save_to_disk = get_rank() == 0
56         checkpointer = DetectronCheckpointer(
57             cfg, model, optimizer, scheduler, output_dir, save_to_disk
58         )
59         extra_checkpoint_data = checkpointer.load(cfg.MODEL.WEIGHT)
60         arguments.update(extra_checkpoint_data)
61
62         data_loader = make_data_loader(
63             cfg,
64             is_train=True,
65             is_distributed=distributed,
66             start_iter=arguments["iteration"],
67         )
68
69         checkpoint_period = cfg.SOLVER.CHECKPOINT_PERIOD
```

2. 构建 backbone 特征提取器，这里没有区分 backbone 和 neck 所以 resnet和fpn是一起构建的

```
48
49     @registry.BACKBONES.register("R-50-FPN-RETINANET")
50     @registry.BACKBONES.register("R-101-FPN-RETINANET")
51     def build_resnet_fpn_p3p7_backbone(cfg):
52         body = resnet.ResNet(cfg)
53         in_channels_stage2 = cfg.MODEL.RESNETS.RES2_OUT_CHANNELS
54         out_channels = cfg.MODEL.RESNETS.BACKBONE_OUT_CHANNELS
55         in_channels_p6p7 = in_channels_stage2 * 8 if cfg.MODEL.RETINANET.USE_C5 \
56             else out_channels
57         fpn = fpn_module.FPN(
58             in_channels_list=[
59                 0,
60                 in_channels_stage2 * 2,
61                 in_channels_stage2 * 4,
62                 in_channels_stage2 * 8,
63             ],
64             out_channels=out_channels,
65             conv_block=conv_with_kaiming_uniform(
66                 cfg.MODEL.FPN.USE_GN, cfg.MODEL.FPN.USE_RELU
67             ),
68             top_blocks=fpn_module.LastLevelP6P7(in_channels_p6p7, out_channels),
69         )
70         model = nn.Sequential(OrderedDict([("body", body), ("fpn", fpn)]))
71         model.out_channels = out_channels
72         return model
```

3. 构建 ATSS head 在增强后的特征基础上进行分类和回归，同时增加了centerness分支。至此模型构建完成开始模型训练。

```
132
133         kernel_size=3,
134         stride=1,
135         padding=1,
136         bias=True
137     )
138     bbox_tower.append(nn.GroupNorm(32, in_channels))
139     bbox_tower.append(nn.ReLU())
140
141     self.add_module('cls_tower', nn.Sequential(*cls_tower))
142     self.add_module('bbox_tower', nn.Sequential(*bbox_tower))
143     self.cls_logits = nn.Conv2d(
144         in_channels, num_anchors * num_classes, kernel_size=3, stride=1,
145         padding=1
146     )
147     self.bbox_pred = nn.Conv2d(
148         in_channels, num_anchors * 4, kernel_size=3, stride=1,
149         padding=1
150     )
151     self.centerness = nn.Conv2d(
152         in_channels, num_anchors * 1, kernel_size=3, stride=1,
153         padding=1
154     )
155
156     # initialization
157     for modules in [self.cls_tower, self.bbox_tower,
158                     self.cls_logits, self.bbox_pred,
159                     self.centerness]:
160         for l in modules.modules():
161             if isinstance(l, nn.Conv2d):
162                 torch.nn.init.normal_(l.weight, std=0.01)
163                 torch.nn.init.constant_(l.bias, 0)
164
165     # initialize the bias for focal loss
166     prior_prob = cfg.MODEL.ATSS.PRIOR_PROB
167     bias_value = -math.log((1 - prior_prob) / prior_prob)
168     torch.nn.init.constant_(self.cls_logits.bias, bias_value)
169     if self.cfg.MODEL.ATSS.REGRESSION_TYPE == 'POINT':
170         assert num_anchors == 1, "regressing from a point only support num_anchors == 1"
171         torch.nn.init.constant_(self.bbox_pred.bias, 4)
172
173     self.scales = nn.ModuleList([Scale(init_value=1.0) for _ in range(5)])
```

4. generalized_rcnn.py中的forward函数: 这里可以看出大致的运行流程, 先将images转换为ImageList, 经过backbone提取特征, 通过rpn获取proposals和proposal_losses, 这里的rpn就是ATSSModule, 因此核心逻辑都在atss.py里面。后面着重分析atss.py和loss.py两个函数。

The screenshot shows the PyCharm IDE interface with the following details:

- Project View:** Shows the project structure for "solution06 - generalized_rcnn.py". The "atss.py" file is open in the editor.
- Code Editor:** Displays the "forward" method of the "GeneralizedRCNN" class. A red dot marks a breakpoint at line 46. A yellow dot marks a step-over point at line 62. The code includes comments explaining the return types for training and testing.
- Variables View:** Shows the current state of variables during the debug session. It lists "images", "self", "body", "ResNet", "conv1", "bn1", and "targets".
- Breakpoints View:** Shows the locations of all breakpoints set in the code.
- Status Bar:** Shows the PyCharm version (2020.3.2), file encoding (UTF-8), and Python version (3.7).

5. `atss.py` 中 `BoxCoder` 是对训练的目标进行编码，当 `cfg.MODEL.ATSS.REGRESSION_TYPE` 为 `POINT` 的时候，回归的目标是 anchor 中心点到真实框四条边的距离。同样预测的时候按照相反的规则进行解码

```
12
13
14     class BoxCoder(object):
15
16         def __init__(self, cfg):
17             self.cfg = cfg
18
19         def encode(self, gt_boxes, anchors):
20             if self.cfg.MODEL.ATSS.REGRESSION_TYPE == 'POINT':
21                 TO_REMOVE = 1 # TODO remove
22                 anchors_w = anchors[:, 2] - anchors[:, 0] + TO_REMOVE
23                 anchors_h = anchors[:, 3] - anchors[:, 1] + TO_REMOVE
24                 anchors_cx = (anchors[:, 2] + anchors[:, 0]) / 2
25                 anchors_cy = (anchors[:, 3] + anchors[:, 1]) / 2
26
27                 w = self.cfg.MODEL.ATSS.ANCHOR_SIZES[0] / self.cfg.MODEL.ATSS.ANCHOR_STRIDES[0]
28                 l = w * (anchors_cx - gt_boxes[:, 0]) / anchors_w
29                 t = w * (anchors_cy - gt_boxes[:, 1]) / anchors_h
30                 r = w * (gt_boxes[:, 2] - anchors_cx) / anchors_w
31                 b = w * (gt_boxes[:, 3] - anchors_cy) / anchors_h
32                 targets = torch.stack([l, t, r, b], dim=1)
33             elif self.cfg.MODEL.ATSS.REGRESSION_TYPE == 'BOX':
34                 TO_REMOVE = 1 # TODO remove
35                 ex_widths = anchors[:, 2] - anchors[:, 0] + TO_REMOVE
36                 ex_heights = anchors[:, 3] - anchors[:, 1] + TO_REMOVE
37                 ex_ctr_x = (anchors[:, 2] + anchors[:, 0]) / 2
38                 ex_ctr_y = (anchors[:, 3] + anchors[:, 1]) / 2
39
40                 gt_widths = gt_boxes[:, 2] - gt_boxes[:, 0] + TO_REMOVE
41                 gt_heights = gt_boxes[:, 3] - gt_boxes[:, 1] + TO_REMOVE
42                 gt_ctr_x = (gt_boxes[:, 2] + gt_boxes[:, 0]) / 2
43                 gt_ctr_y = (gt_boxes[:, 3] + gt_boxes[:, 1]) / 2
44
45                 wx, wy, ww, wh = (10., 10., 5., 5.)
46                 targets_dx = wx * (gt_ctr_x - ex_ctr_x) / ex_widths
47                 targets_dy = wy * (gt_ctr_y - ex_ctr_y) / ex_heights
48                 targets_dw = ww * torch.log(gt_widths / ex_widths)
49                 targets_dh = wh * torch.log(gt_heights / ex_heights)
50                 targets = torch.stack((targets_dx, targets_dy, targets_dw, targets_dh), dim=1)
51
52         return targets
53
```

ATSS

接下来主要看看 loss.py 关于 anchor 和 ground truth 的匹配，以及 threshold 计算等核心逻辑都在这里。

1. atss.py 中 ATSSHead 是对 feature pyramid 中的 cell 进行 classification, box_regression 和 centerness prediction

```
self.centerness = nn.Conv2d(
        in_channels, num_anchors * 1, kernel_size=3, stride=1,
        padding=1
    )

    # initialization
    for modules in [self.cls_tower, self.bbox_tower,
                    self.cls_logits, self.bbox_pred,
                    self.centerness]:
        for l in modules.modules():
            if isinstance(l, nn.Conv2d):
                torch.nn.init.normal_(l.weight, std=0.01)
                torch.nn.init.constant_(l.bias, 0)

    # initialize the bias for focal loss
    prior_prob = cfg.MODEL.ATSS.PRIOR_PROB
    bias_value = -math.log((1 - prior_prob) / prior_prob)
    torch.nn.init.constant_(self.cls_logits.bias, bias_value)
    if self.cfg.MODEL.ATSS.REGRESSION_TYPE == 'POINT':
        assert num_anchors == 1, "regressing from a point only support num_anchors == 1"
        torch.nn.init.constant_(self.bbox_pred.bias, 4)

    self.scales = nn.ModuleList([Scale(init_value=1.0) for _ in range(5)])

def forward(self, x):
    logits = []
    bbox_reg = []
    centerness = []
    for l, feature in enumerate(x):
        cls_tower = self.cls_tower(feature)
        box_tower = self.bbox_tower(feature)

        logits.append(self.cls_logits(cls_tower))

        bbox_pred = self.scales[l](self.bbox_pred(box_tower))
        if self.cfg.MODEL.ATSS.REGRESSION_TYPE == 'POINT':
            bbox_pred = F.relu(bbox_pred)
        bbox_reg.append(bbox_pred)

        centerness.append(self.centerness(box_tower))
    return logits, bbox_reg, centerness
```

2. atss.py 中 ATSSModule 主要是对 anchor 进行分类和回归同时计算 loss

```
192
193
194     class ATSSModule(torch.nn.Module):
195
196         def __init__(self, cfg, in_channels):
197             super(ATSSModule, self).__init__()
198             self.cfg = cfg
199             self.head = ATSSHead(cfg, in_channels)
200             box_coder = BoxCoder(cfg)
201             self.loss_evaluator = make_atss_loss_evaluator(cfg, box_coder)
202             self.box_selector_test = make_atss_postprocessor(cfg, box_coder)
203             self.anchor_generator = make_anchor_generator_atss(cfg)
204
205         def forward(self, images, features, targets=None):
206             box_cls, box_regression, centerness = self.head(features)
207             anchors = self.anchor_generator(images, features)
208
209             if self.training:
210                 return self._forward_train(box_cls, box_regression, centerness, targets, anchors)
211             else:
212                 return self._forward_test(box_cls, box_regression, centerness, anchors)
213
214         def _forward_train(self, box_cls, box_regression, centerness, targets, anchors):
215             loss_box_cls, loss_box_reg, loss_centerness = self.loss_evaluator(
216                 box_cls, box_regression, centerness, targets, anchors
217             )
218             losses = {
219                 "loss_cls": loss_box_cls,
220                 "loss_reg": loss_box_reg,
221                 "loss_centerness": loss_centerness
222             }
223             return None, losses
224
225         def _forward_test(self, box_cls, box_regression, centerness, anchors):
226             boxes = self.box_selector_test(box_cls, box_regression, centerness, anchors)
227             return boxes, {}
```

3. GloU loss GloU 是对 IoU 的改进，可以直接用 $1 - GIoU$ 作为 loss 函数来训练模型

$$GIoU = IoU - \frac{|Ac - U|}{|Ac|}$$

先计算两个框之间的最小外接矩形面积 Ac , 计算两个框的所占的面积 U , $Ac - U$ 是最小外接矩形不属于两个框的面积。和 IoU 相比 $GIoU$ 是更好的距离度量指标，不仅仅关注重叠区域也关注非重叠区域。

The screenshot shows the PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help.
- Toolbars:** Standard toolbar with icons for Run, Stop, Run, Build, Debug, Terminal, Python Console, and Event Log.
- Project Explorer:** Shows the project structure for "solution06".
- Code Editor:** The main window displays the "loss.py" file under the "atss" directory. The code implements the ATSS loss function using the GiouLoss metric. It includes imports for various modules like "atss", "atss.core", "csrc", "data", "engine", "layers", "modeling", "rpn", and "util". The code uses PyTorch operations such as "torch.max", "torch.min", and "torch.zeros". It also includes C++ code via a shared library "C:/python37-x86_64-linux-gnu.so".
- Status Bar:** Shows the current file is "loss.py", the Python version is 3.7.3, and the current line is 73.

4. 先针对每一层计算每个 anchor 到 ground truth 中心点距离，然后根据距离排序选出 topk 候选 anchor

```

elif self.cfg.MODEL.ATSS.POSITIVE_TYPE == 'ATSS':
    num_anchors_per_loc = len(self.cfg.MODEL.ATSS.ASPECT RATIOS) * self.cfg.MODEL.ATSS.SCALES_PER_OCTAVE

    num_anchors_per_level = [len(anchors_per_level.bbox) for anchors_per_level in anchors[im_i]]
    ious = boxlist_iou(anchors_per_im, targets_per_im)

    gt_cx = (bboxes_per_im[:, 2] + bboxes_per_im[:, 0]) / 2.0
    gt_cy = (bboxes_per_im[:, 3] + bboxes_per_im[:, 1]) / 2.0
    gt_points = torch.stack((gt_cx, gt_cy), dim=1)

    anchors_cx_per_im = (anchors_per_im.bbox[:, 2] + anchors_per_im.bbox[:, 0]) / 2.0
    anchors_cy_per_im = (anchors_per_im.bbox[:, 3] + anchors_per_im.bbox[:, 1]) / 2.0
    anchor_points = torch.stack((anchors_cx_per_im, anchors_cy_per_im), dim=1)

    distances = (anchor_points[:, None, :] - gt_points[None, :, :]).pow(2).sum(-1).sqrt()

# Selecting candidates based on the center distance between anchor box and object
candidate_idxs = []
star_idx = 0
for level, anchors_per_level in enumerate(anchors[im_i]):
    end_idx = star_idx + num_anchors_per_level[level]
    distances_per_level = distances[star_idx:end_idx, :]
    topk = min(self.cfg.MODEL.ATSS.TOPK * num_anchors_per_loc, num_anchors_per_level[level])
    _, topk_ids_per_level = distances_per_level.topk(topk, dim=0, largest=False)
    candidate_idxs.append(topk_ids_per_level + star_idx)
    star_idx = end_idx
candidate_idxs = torch.cat(candidate_idxs, dim=0)

```

5. 计算 ground truth 和候选 anchor. 之间的 IoU, 计算这些候选 anchor IOU 的均值和标准差, 筛选该 ground truth 的候选 anchor 的 IoU 阈值时 mean + std

```
# Using the sum of mean and standard deviation as the IoU threshold to select final positive samples
candidate_ious = ious[candidate_idxs, torch.arange(num_gt)]
iou_mean_per_gt = candidate_ious.mean(0)
iou_std_per_gt = candidate_ious.std(0)
iou_thresh_per_gt = iou_mean_per_gt + iou_std_per_gt
is_pos = candidate_ious >= iou_thresh_per_gt[None, :]
```

6. 使用该 IoU 阈值筛选 anchor 同时保证 anchor 中心在 ground truth 里面

```

# Limiting the final positive samples' center to object
anchor_num = anchors_cx_per_im.shape[0]
for ng in range(num_gt):
    candidate_idxs[:, ng] += ng * anchor_num
e_anchors_cx = anchors_cx_per_im.view(1, -1).expand(num_gt, anchor_num).contiguous().view(-1)
e_anchors_cy = anchors_cy_per_im.view(1, -1).expand(num_gt, anchor_num).contiguous().view(-1)
candidate_idxs = candidate_idxs.view(-1)
l = e_anchors_cx[candidate_idxs].view(-1, num_gt) - bboxes_per_im[:, 0]
t = e_anchors_cy[candidate_idxs].view(-1, num_gt) - bboxes_per_im[:, 1]
r = bboxes_per_im[:, 2] - e_anchors_cx[candidate_idxs].view(-1, num_gt)
b = bboxes_per_im[:, 3] - e_anchors_cy[candidate_idxs].view(-1, num_gt)
is_in_gts = torch.stack([l, t, r, b], dim=1).min(dim=1)[0] > 0.01
is_pos = is_pos & is_in_gts

# if an anchor box is assigned to multiple gts, the one with the highest IoU will be selected.
ious_inf = torch.full_like(ious, -INF).t().contiguous().view(-1)
index = candidate_idxs.view(-1)[is_pos.view(-1)]
ious_inf[index] = ious.t().contiguous().view(-1)[index]
ious_inf = ious_inf.view(num_gt, -1).t()

```

问题

1. PyCharm debug torch.distributed.launch debug 的时候报错，可以正常运行

`python -m torch.distributed.launch` 实际运行的是另外一个torch库脚本，路径不在当前工程里面。修改 script path 可以正常调试。

