



The Simplest Faster - RCNN

by cmd23333

$\geq \triangleright \leq$

github : cmd23333



`opt.parse(kwargs)` 更新并打印 config 信息

`dataset = Dataset(opt)` 导入数据集

`dataloader = torch.utils.data.DataLoader`

`(dataset, batch_size=1, shuffle=True, num_workers=0)`

默认 `n_fg_class = 20`

`faster_rcnn = FasterRCNNVGG16()` 构建模型

→ 提特征
extractor, classifier = decom_vgg16()

`rpn = RegionProposalNetwork(512, 512,` input 通道数 中间层通道数

`ratios=ratios, anchor_scales=~,`

`feat_stride=16)`

vgg 的 conv5 把原图变小 316 倍。

→ rpn 的 `anchor_base = generate_anchor_base`
`(8, 16, 32, 0.5, 1, 2)` (anchor_scales, ratios)

以 $(0, 0)$ 为中心，产生 9 个 anchors。基础尺寸 = 16

$$16 \times 8 \times 16 \times 8 = 2^{14}$$

$$\text{ratio} = \frac{x}{y}$$

$$0.5 \cdot y = 2^{15}$$

$$y = 2^{7.5}$$

$$x = 2^{6.5}$$

$$0.5 \quad w=2^{6.5}, h=2^{7.5}$$

1

$$(16 \times 8)^2$$

2

$$(16 \times 16)^2$$

$$w=256, h=256$$

$$(16 \times 32)^2$$

→ RPN 的 proposal_layer = Proposal Creator (self, ..)

输入: Loc. score 和约 2W 个 anchor 的坐标和原图尺寸 scale
输出: ~2000 个训练样本 rois (~2000, 4) 坐标. 无 gt

9↑ n_anchor = self.anchor_base.shape[0]

self.conv1 = nn.Conv2d(in_channel, mid_channel,
kernel_size 3, 1, 1) pad=1. $\frac{h-3+2 \times 1}{1} + 1 = h.$

self.score = nn.Conv2d(mid_channel, n_anchor * 2,
1, 1, 0)

self.loc = nn.Conv2d(mid_channel, n_anchor * 4,
1, 1, 0)

然后用 normal_init() 初始化上面的层。

→ head = VGG16 RoI Head (n_class = n_fg_class + 1,
roi_size = 7,

spatial_scale = $\frac{1}{self.feat_stride}$,
classifier = classifier)

→ head 的 classifier = classifier

同样，初始化
这一层。

`cls_loc = nn.Linear(4096, n_class * 4)`

`score = nn.Linear(4096, n_class)`

`head_for_roi = ROI Pooling2D(roi_size,`
`roi_size, spatial_scale)`

再初始化父类 FasterRCNN

`self.loc_normalize_mean = loc_normalize_mean (0, 0, 0, 0)`
`self.loc_normalize_std = loc_normalize_std (0.1, 0.1, 0.2, 0.2)`

`self.use_preset('evaluate')`

设置 `nms_thresh = 0.3`

`score_thresh = 0.05`

这样，model construct completed.

`trainer = FasterRCNNTrainer(faster_rcnn).cuda()`



`self.faster_rcnn = faster_rcnn`

`self.rpn_sigma = opt.rpn_sigma 3`

`self.roi_sigma = opt.roi_sigma 1`

正负样本量一半。

默认 `n_sample = 256`. `neg_iou_thres = 0.3` `pos_iou_thres = 0.7`

\uparrow
`pos_ratio = 0.5`

`trainer.anchor_target_creator = AnchorTargetCreator()`

输入：~2w个 anchor 坐标 和 四个 bbox 真实坐标

输出：(~2w, 1) label: 128个1, 128个0, 其余-1

(~2W, 4) 的圆圈目标. (所有 anchor 的坐标都有)

trainer 的 proposal_target_creator = ProposalTargetCreator()
默认 n-sample = 128

从 RPN 产生的 ~2000 个 Roi 中挑 128 个训练

Roi 和 gt 的 $\text{iou} > 0.5$ 选 32 个, < 0.1 选 $128 - 32 = 96$ 个
 $\frac{\text{pos-thres}}{\text{pos-ratio}}$

输入: ~200 个 roi, 一张图中所有 bbox gt (R, 4). 对应类 (R, 1)

输出: 128 个 sample roi (128, 4)

gt_roi_loc (128, 4), gt_roi_label (128, 1)

trainer 的 optimizer = self.faster_rcnn.get_optimizer()

t.optim.SGD(params, momentum=0.9)

trainer 的 meters = {k: AverageValueMeter() for k in

LossTuple._fields}

加载保存的权重如有.

lr = opt.lr 0.001

下面就开始训练了.

trainer.reset_meters()

就看下一张图的训练过程

img, bbox_, label_, scale = next(iter(dataloader))

img: [1, 3, 600, 900]

全部 {
bbox: [1, 2, 4] (317, 472, 529, 755) (..)
label_: [1, 2] 9.9
.cuda()

scale 转行 float 1.801

trainer.train_step(img, bbox, label, scale)

首先. optimizer.zero_grad()

loss = self.forward(img, bbox, label, scale)

确认 bbox.shape[0] == 1.

img_size = (600, 901)

features = self.faster_rcnn.extractor(img)

(1, 512, 37, 56)

rpn_locs, rpn_scores, rois, roi_indices, anchor =

self.faster_rcnn.rpn(features, img_size, scale)

RPN

$n, h, w = \text{features.shape}$ $37 \times 56 \times 9$

$\text{anchor} = \text{enumerate_shifted_anchor}$ $18648, 4$

$(\text{np.array}(\text{self.anchor_base}), \text{self.feat_stride}, h, w)$ 16

$n_{\text{anchor}} = \frac{\text{anchor.shape}[0]}{h * w} = 9$

$h = F.relu(\text{self.conv1}(\text{features}))$

(1, 512, 37, 56) $512, 512, 3, 1, 1$
 f s p

rpn_locs = self.loc(h)

(1, 36, 37, 56) \downarrow
 Conv2D $1512, 36, 1, 1$
 f s

$rpn_locs = rpn_locs.$ permute(0, 2, 3, 1). contiguous()
(1, 18648, 4) . view(n, -1, 4)

$rpn_scores = self.\text{score}(h)$
(1, 18, 37, 56) \downarrow
Conv2D (512, 18, 1, 1)

$rpn_scores = rpn_scores.\text{permute}(0, 2, 3, 1).\text{contiguous}()$
(1, 37, 56, 18)

$rpn_softmax_scores = F.\text{softmax}($
(1, 37, 56, 9, 2) $rpn_scores.\text{view}(n, 37, 56, 9, 2)$
 $\dim = 4)$

(1, 37, 56, 9) 前景得分
 $rpn_fg_scores = rpn_softmax_scores[:, :, :, :, 1].contiguous()$
 $rpn_fg_scores = rpn_fg_scores.\text{view}(n, -1)$

$rpn_scores = rpn_scores.\text{view}(n, -1, 2)$
(1, 18648, 2)

$rois = list()$, $roi_indices = list()$
for i in range(n): 我们这 batch_size = 1

为检测网络 $roi = self.\text{proposal_layer}$

提供 200 个训练 (rpn_locs[i].cpu().data.numpy(),
(18648, 4))

样本 | $rpn_fg_scores[i].cpu().data.numpy()$,
(18648, 18648)
anchor, img_size, scale = scale
(18648, 4) 600, 901 1.801

$$\left. \begin{array}{l} n_pre_nms = 12000 \\ n_post_nms = 2000 \\ (18648, 4) \end{array} \right\} (18648, 4)$$

rpn_locs 和 $anchor$ 怎么转换.

$$\begin{aligned} \hat{G}_x &= P_w \cdot d_x(P) + P_x & P_x \text{ 为 } anchor \text{ 的} \\ \hat{G}_w &= P_w \exp(d_w(P)) & d_w(P) \Rightarrow rpn_locs \text{ 的} \end{aligned}$$

y 小于高度, > 0 $roi[:, slice(0, 4, 2)] = np.clip$ $H = 600$
 $(roi[:, slice(0, 4, 2)], 0, img_size[0])$

x 小于宽度, > 0 $roi[:, slice(1, 4, 2)] = np.clip$ $W = 900$
 $(roi[:, slice(1, 4, 2)], 0, img_size[1])$

大小的预测框不要了.

28.82 $min_size = self.min_size * scale$ 1.80

$$hs = roi[:, 2] - roi[:, 0]$$

$$ws = roi[:, 3] - roi[:, 1]$$

keep = $np.where((hs \geq min_size) \& (ws \geq min_size)) [0]$ 要求预测框 h, w 都比 min_size 大

可能是用三裁入
的权重. 没有
预测框.

$(18648, 4)$

$$roi = roi[keep, :]$$

score = score[keep] $\rightarrow rpn_fg_scores[i]$

$(18648, 1)$

排序 $order = score.ravel().argsort() [::-1]$

前 12000 $roi = roi[order[:n_pre_nms], :]$ 12000

表示保留
的 index $\text{keep} = \text{non_maximum_suppression}$

(917,) (cp. asarray(roi), thresh = self.nms_thresh) 0.7

RPN

$\text{roi} = \text{roi} [\text{keep}[:n_post_nms]]$

返回3 roi (917,4) 原图尺寸的bbox (y_1, x_1, y_2, x_2)坐标

$\text{batch_index} = i * \text{np.ones}(\text{len}(roi),), \text{dtype} = \text{np.int32}$
[917,] 都是0

rois.append(roi)

roi_indices.append(batch_index)

917,4 rois = np.concatenate(rois, axis=0) (roi1, roi2, ...) 不过我们 batch_size=1
917, roi_indices = np.concatenate(roi_indices, axis=0) 这两步是可选的，且无用。

上面的 rpn 网络返回了

rpn_locs, rpn_scores, rois, roi_indices, anchor

F-步，因为 batch_size=1.

$\text{bbox} = \text{bbox}[0] . \text{label} = \text{label}[0]$
[2,4] [1,2,4] [2] [1,2]

18648,2 rpn_score = rpn_scores[0]

18648,4 rpn_loc = rpn_locs[0]

917,4 roi = rois

下面开始 ROI 3.

sample_roi, gt_roi_loc, gt_roi_label

= self.proposal_target_creator(roi, bbox, label,

self.loc_norm_mean, self.loc_norm_std)

从上面生成的 ~200 个 (917 个) roi 中, 选择 128 个 用来训练.

32 ~ $\text{IoU}(\text{roi}, \text{gt}) > 0.5$

给这 32 个属于 ground truth.

128 - 32 ~ $\text{IoU}(\text{roi}, \text{gt}) < 0.1$

roi
→
 $n_bbox, - = bbox.shape$

$\text{roi} = \text{np.concatenate}((\text{roi}, \text{bbox}), \text{axis}=0)$
(919, 4). 不是很懂这操作? 作弊式训练?

pos_roi_per_image = np.round(self.n_sample * self.pos_ratio)

(919, 2) 每个预测和每个 gt 的 iou
 $\text{iou} = bbox_iou(\text{roi}, \text{bbox})$

(919, 1) $\text{gt_assignment} = \text{iou.argmax(axis=1)}$ 找预测和图中哪个 gt 的iou大

(919, 1) $\text{max_iou} = \text{iou.max(axis=1)}$ 找预测和 gt-bbox 的 iou 最大值

[99]

gt_roi_label = label[gt_assignment] + 1

把类别从 0 ~ 19 换到 1 ~ 20 因为 0 用作 bg

(45)

pos_index = np.where(max_iou >= self.pos_iou_thresh)

pos_roi_per_this_image = int(min(pos_roi_per_image,
32
pos_index.size))

45

If pos_index.size > 0 : 从过多 iou > 0.5 的抽 3 个，不放回

pos_index = np.random.choice(..)

neg_index = np.where((max_iou < neg_iou_thresh) &
max_iou > neg_iou_thresh_low)

原文 [1]

hi
0.5
To]

同样从 neg_index 中选取 n_sample - pos_roi_per_this_img

(128, 1) keep_index = np.append(pos_index, neg_index)

gt_roi_label = gt_roi_label[keep_index]

gt_roi_label[pos_roi_per_this_img:] = 0 同样本池

不计算回归损失

(128, 4) sample_roi = roi[keep_index]

(128, 4) 真实的 gt_offset

gt_roi_loc = bbox2loc(sample_roi,

(2, 4) bbox(gt_assignment[keep_index])
128, 4

$$t_x = \frac{G_x - P_x}{P_w} \quad t_w = \log\left(\frac{G_w}{P_w}\right)$$

(128, 4) $gt_roi_loc = \frac{gt_roi_loc - loc_norm_mean}{loc_norm_std}$

上面返回了 sample_roi, gt_roi_loc, gt_roi_label 原因尺寸

128个0，因为 batch_size=1
sample_roi_index = t.zeros(len(sample_roi))
roi_cls_loc, roi_score = self.faster_rcnn.head
(features, sample_roi, sample_roi_index)

→ (128, 7)
roi_indices = at.to tensor(sample_roi_index).float()
(128, 4)
rois = at.totensor(sample_roi).float()
indices_and_rois = t.cat([roi_indices[:, None], rois],
(128, 5)
y x → xy
xy_indices_and_rois = indices_and_rois[:, [0, 2, 1, 4, 3]]
dim=1)

115123756
pool = self.roi(x, xy_indices_and_rois)
features → poi
B, C, H, W = x.size
N = roi.size(0)
(128, 25088) output = t.zeros(N, C, 7, 7).cuda()

pool = pool.view(pool.size(0), -1)
(128, 512x7x7)

fc7 = self.classifier(pool)
(128, 4096)

in- out
25088, 4096 Linear
ReLU
FC (4096, 4096)
ReLU

$$\begin{aligned}
 & \text{roi_cls_locs} = \text{self}. \text{cls_loc}(\text{fc7}) \quad \text{FC}(4096, 84) \\
 & \text{roi_scores} = \text{self}. \text{score}(\text{fc7}) \quad \text{FC}(4096, 21)
 \end{aligned}$$

再往下，计算 loss.

trainer 为所有任务分配 gt

$$\begin{aligned}
 & \text{gt_rpn_loc}, \text{gt_rpn_label} = \text{self}. \overset{\uparrow}{\text{anchor_target_creator}} \\
 & (\text{bbox}, \text{anchor}, \text{img_size}) \\
 & [2, 4] \quad [18648, 4] \quad (600, 901)
 \end{aligned}$$

输出 ($\sim 2w, 1$) [label, 128个1, 128个0, 其余-1]

($\sim 2w, 4$) 回归目标。所有 anchors 都有

选择方式：

对每个 gt-bbox, 和它iou最大的正样本

剩下的, iou > 0.7 的正样本, 补足 128 个

iou < 0.3 的负样本, (256 - 正样本个数) 个。

$$\begin{matrix}
 600 \\
 \text{img-H, img-W} = \text{img-size}
 \end{matrix}
 \quad
 \begin{matrix}
 901 \\
 \text{iou}
 \end{matrix}$$

$\text{inside_index} = \text{_get_inside_index}(\text{anchor}, \text{img-H}, \text{img-W})$

在图片内部的 anchor

$$\begin{aligned}
 \text{anchor} &= \text{anchor}[\text{inside_index}] \\
 (6928, 4)
 \end{aligned}$$

$\text{argmax_ious, Label} = \text{self}. \text{_create_Label}$

$$\begin{matrix}
 (6928,) \\
 \uparrow
 \end{matrix}
 \quad
 \begin{matrix}
 (6928, 1) \\
 \text{inside_index, anchor, bbox}
 \end{matrix}$$

每个 anchor 对应 iou 最大的 bbox 是第 \uparrow 个。

(6928, 4) 转换成 offset

$$\text{loc} = \text{bbox2loc}(\text{anchor}, \text{bbox}[\text{argmax_ious}])$$

把不在图里的 label = $\text{unmap}(\text{label}, n_anchor, \text{inside_index}, \text{fill}=-1)$

anchor 恢复过来, loc = $\text{unmap}(\text{label}, n_anchor, \text{inside_index}, \text{fill}=0)$

label 为 -1, loc 为 0.

gt_rpn_label (18648, 1)

gt_rpn_loc (18648, 4)

计算 rpn_loc_loss

rpn_loc_loss = $\text{fast_rcnn_loc_loss}($

label 为 -1 的不参与计算 rpn_loc, gt_rpn_loc,

smooth_L1_loss

gt_rpn_label.data,
self.rpn_sigma)

rpn_cls_loss = $F.\text{cross_entropy}(\text{rpn_score}, \text{gt_rpn_label},$
 $\text{ignore_index}=-1)$

计算 roi_loss

n_sample = $\text{roi_cls_loc.shape}[0]$

roi_cls_loc = $\text{roi_cls_loc.view}(n_sample, -1, 4)$

roi_loc = $\text{roi_cls_loc}[\text{t.arange}(0, n_sample),$
 $128, 1, 4$ gt_roi_label]

roi_loc_loss = $-\text{fast_rcnn_loc_loss}$

(roi_loc, gt_roi_loc, gt_roi_label, data,
128/4 128/4
self.roi_sigma)

1.8238

roi_cls_loss = nn.CrossEntropy()(roi_score, gt_roi_label)

losses = [rpn_loc_loss, rpn_cls_loss, roi_loc_loss,
roi_cls_loss]

losses = losses + [sum losses]

return LossTuple(*losses)

至此，train_step 的 forward 结束。

losses.total_loss.backward()

self.optimizer.step()

trainer

self.update_meters(losses)

至此，train_step 结束