assignment3

2020 年 4 月 23 日

0.1 作业要求

1. 数据准备（图片读取，对应的 GT 生成)[数据这个没有讲，所以代码直接提供，不用填空，代码位置：week18/dataset.py]
2. EAST 模型建立 [填空代码 week18/model.py]
3. loss 函数建立 [填空代码 week18/loss.py]
4. 优化函数，超参数选择 [填空代码 week18/train.py]

0.2 实现思路

0.2.1 EAST Output Layer

模型的输出层由单通道的 score map 和多通道的 geometry map 组成。其中，geometry map 论文给出了两种方案：1) rotation box(RBOX)；2) quadrangle (QUAD)。这里只讨论 RBOX。

- Score 的 channel = 1，表示每个像素有文字的概率 (score map)；
- RBOX 的 channel = 5，其中，1 个 channel 表示文字区域的角度 (angle map)，其余 4 个 channel 表示文字区域的每像素分别到文字区域上、右、下、左边框的距离 (AABB map)。
- QUAD 的 channel = 8，分别表示四个角的坐标（坐标包含 x 和 y 两个值）。

输出层的 score map、AABB map、angle map 都是由 `Feature-merging branch` 输出的 32 通道 feature map 经过 Conv1x1 得到，表示为：
score map: Conv1×1(32, 1)；
AABB map: Conv1×1(32, 4)；
angle map: Conv1×1(32, 1)。

0.2.2 损失函数

EAST 模型输出层包含三部分：score map，AABB map 和 angle map，因此，整个模型的损失函数是三者各自的损失函数的求和，公式为：

$$L = L_s + \lambda_g(L_{AABB} + \lambda_\theta L_\theta)$$

论文的实验建议，λ_g 取 1，λ_θ 取 10。

Loss for Score Map score map 是二分类的问题，在目标检测领域存在正负样本不平衡问题，也就是一张图片包背景的像素占比远大于包含目标的像素。直接用交叉熵损失函数会导致模型倾向于训练负类，论文 score map 损失函数使用类平衡交叉熵损失函数来解决样本不平衡问题。但是在具体实践中，一般采用 Dice Loss，它的收敛速度会比类平衡交叉熵快。Dice Loss 公式为：

$$d = 1 - 2\frac{|X \bigcap Y|}{|X| + |Y|}$$

Dice Loss 为什么能解决样本不平衡问题？参见 “Dice Loss 理解” 章节。

Loss for AABB Map AABB map 的损失函数使用 IoU loss，表达式为：

$$L = -log(IoU)$$

Loss for Angle Map angle map 的损失函数直接计算余弦值，表达式为：

$$L_\theta = 1 - \cos(\widehat{\theta} - \theta^*)$$

0.2.3 Dice Loss 理解

Dice 系数，是集合相似度的度量指标。用在 CV 领域，是衡量两个样本的重叠程度。Dice 系数表达式为：

$$2\frac{|X \bigcap Y|}{|X| + |Y|}$$

对于目标检测问题而言，X 表示 GT mask，Y 表示 Pred mask。$|X \bigcap Y|$ 近似为 GT mask 与 Pred mask 点乘后求和，$|X|$ 和 $|Y|$ 分别是将 GT mask 和 Pred mask 的所有元素求和。

对于二分类问题，GT mask 只有 0 和 1 两个值，因此可以有效地将在 Pred mask 中对应 GT mask 无目标的像素清零，对于有目标的像素，主要惩罚低置信度的预测，高置信度会得到大的 Dice 系数。

计算示例：

$$|X \bigcap Y| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 0.01 & 0.04 & 0.01 & 0.02 \\ 0.05 & 0.07 & 0.13 & 0.09 \\ 0.99 & 0.96 & 0.89 & 0.91 \\ 0.95 & 0.97 & 0.96 & 0.98 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.95 & 0.97 & 0.96 & 0.98 \end{bmatrix} \rightarrow 3.86$$

$$|X| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow 4$$

$$|Y| = \begin{bmatrix} 0.01 & 0.04 & 0.01 & 0.02 \\ 0.05 & 0.07 & 0.13 & 0.09 \\ 0.05 & 0.12 & 0.01 & 0.03 \\ 0.95 & 0.97 & 0.96 & 0.98 \end{bmatrix} \rightarrow 4.49$$

Dice 系数差异函数 (Dice Loss) 定义为：

$$d = 1 - 2\frac{|X \bigcap Y|}{|X| + |Y|}$$

Dice 系数越大（Pred mask 越接近 GT mask），Dice Loss 越小。

0.3 代码实现

完整代码：https://github.com/kysonlok/hct_project3/tree/master/src/east_pytorch

0.3.1 优化算法

为了方便，这里直接使用 Adam 优化器，不需要手动调整 lr。

```
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
```

0.3.2 Loss for Score Map

为了避免无相交导致分母为 0，加上 smooth（一个接近 0 的小数）。

```
def dice_coefficient(gt_score, pred_score):
    smooth = 1e-5

    intersection = torch.sum(gt_score * pred_score)
```

```
    union = torch.sum(gt_score) + torch.sum(pred_score)

    loss = 1. - (2. * (intersection + smooth)) / (union + smooth)

    return loss
```

0.3.3 Loss for Geometry Map

Geometry Map 由 AABB Map（文字边框）和 Angle Map（边框的角度）组成。

```
def get_geo_loss(gt_geo, pred_geo):
    # d1 -> top, d2 -> right, d3 -> bottom, d4 -> left
    d1_gt, d2_gt, d3_gt, d4_gt, theta_gt = torch.split(gt_geo, 1, 1)
    d1_pred, d2_pred, d3_pred, d4_pred, theta_pred = torch.split(pred_geo, 1, 1)

    area_gt = (d1_gt + d3_gt) * (d2_gt + d4_gt)
    area_pred = (d1_pred + d3_pred) * (d2_pred + d4_pred)
    w_union = torch.min(d2_gt, d2_pred) + torch.min(d4_gt, d4_pred)
    h_union = torch.min(d1_gt, d1_pred) + torch.min(d3_gt, d3_pred)
    area_intersect = w_union * h_union
    area_union = area_gt + area_pred - area_intersect

    iou_loss = -torch.log((area_intersect + 1.0) / (area_union + 1.0))
    angle_loss = 1 - torch.cos(theta_pred - theta_gt)

    return iou_loss, angle_loss
```

0.3.4 Loss for EAST

```
class Loss(nn.Module):
    def __init__(self, weight_angle=10):
        super(Loss, self).__init__()
        self.weight_angle = weight_angle

    def forward(self, gt_score, pred_score, gt_geo, pred_geo, ignored_map):
        # 过滤没有文字目标
        if torch.sum(gt_score) < 1:
            return torch.sum(pred_score + pred_geo) * 0
```

```
        # 计算 score loss 使用 dice loss 代替分类平衡交叉熵 loss
        classify_loss = dice_coefficient(gt_score, pred_score*(1-ignored_map))

        # IoU loss + Angle loss
        iou_loss, angle_loss = get_geo_loss(gt_geo, pred_geo)

        iou_loss = torch.sum(iou_loss * gt_score) / torch.sum(gt_score)
        angle_loss = torch.sum(angle_loss * gt_score) / torch.sum(gt_score)

        geo_loss = self.weight_angle * angle_loss + iou_loss

        print('classify loss is {:.8f}, angle loss is {:.8f}, iou loss is {:.8f}'.format(classify_

        return geo_loss + classify_loss
```

0.3.5 Output Layer

```
# 请定义 self.conv1, 用于输出 score map
# 建议代码: self.conv1 = nn.Conv2d(, , )
self.conv1 = nn.Conv2d(32, 1, 1, bias=False)
self.sigmoid1 = nn.Sigmoid()

# 请定义 self.conv2, 用于输出 d1,d2,d3,d4 map
# 建议代码: self.conv2 = nn.Conv2d(,,)
self.conv2 = nn.Conv2d(32, 4, 1, bias=False)
self.sigmoid2 = nn.Sigmoid()

# 请定义 self.conv3, 用于输出 angle_map
# 建议代码: self.conv3 = nn.Conv2d(, , )
self.conv3 = nn.Conv2d(32, 1, 1, bias=False)
self.sigmoid3 = nn.Sigmoid()
```

0.4 TO DO LIST

1. 写 inference 代码，在测试集上测试一下；

2. 调整优化器，对比效果；

3. backbone 用 Resnet 代替 VGG16，对比效果。