旷视AI智慧交通开源赛道

复杂场景下十字路口的交通标志检测
赛题讲解

旷视科技

Sep. 07, 2021

Outline

- 赛题背景
- 任务说明
- 数据说明
- 评价指标
- 比赛要求
- 代码说明

赛题介绍

† 赛题背景

- 深度学习技术日趋成熟，智慧交通需求日益增长
  - 智能驾驶、交通安防、违章辅助报警
  - 交通标志检测是道路信息提取的重要分支之一
- 难点与挑战
  - 交通标志本身种类众多，大小不定，极易受光照、天气等因素的影响
  - 交通在十字路口场景下，复杂的交通情况会进一步加大难度

赛题介绍

† 任务说明

- 比赛题目: 复杂场景下十字路口的交通标志检测
- 比赛任务: 主办方提供真实场景下涵盖多种复杂道路场景下的十字路口交通标志的图片数据集，参赛团队基于此数据集，使用开源的旷视天元深度学习框架，通过实现或提出更好的交通标志检测算法，尽可能提升在测试集上的检测性能。
- 比赛时间: 2021.9.13 - 2021.9.26

旷视AI智慧交通开源赛道 报名中

交通标志本身种类众多，大小不定，并且在交通复杂的十字路口场景下，由于光照、天气等因素的影响，使其被精确检测变得更加困难。提高上述场景下交通标志检测准确度，将有助于降低十字路口交通事故发生的概率。本次...

复杂场景的交通标志检测 | 39个人已参赛 | 2021/09/13 - 09/26

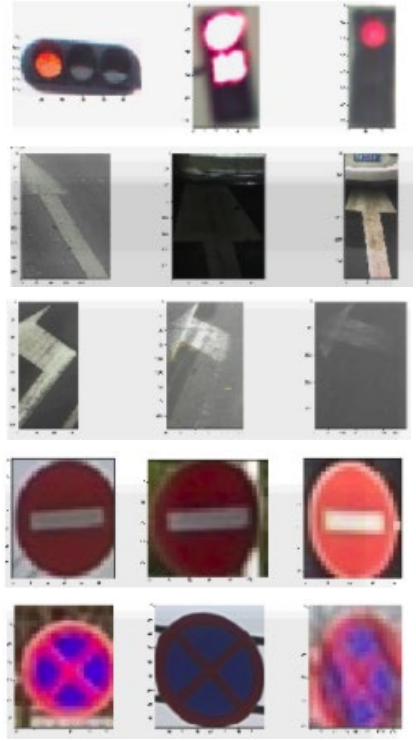
87,000
奖金池

比赛链接

赛题介绍

† 数据说明

- 本次比赛提供真实场景的道路图片，共计对5个交通标志类别进行了标注
  - 红灯、直行标志、向左转弯标志、禁止驶入和禁止临时停车



- 本次比赛数据划分为两个集合
  - 训练集: train (2226 images, 5000 boxes) + val (299 images, 965 boxes)
  - 测试集: test (584 images)
- 参赛者可直接在比赛关联的 MegStudio 项目中找到数据集，在项目内点击【启动环境】，进入环境查看数据集，并在线完成模型训练

赛题介绍

† 评价指标

- 输出格式: 符合COCO评测格式的JSON文件, 在Github Repo已经提供相应接口
- COCO Evaluation Tools: https://cocodataset.org/#detection-eval, submit format
- mAP, AP75, APs
  - Score = 0.5 x mAP + 0.3 x AP75 + 0.2 x APs

```
Average Precision (AP):
  AP                % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50        % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75        % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small          % AP for small objects: area < 32^2
  AP^medium         % AP for medium objects: 32^2 < area < 96^2
  AP^large          % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1          % AR given 1 detection per image
  AR^max=10         % AR given 10 detections per image
  AR^max=100        % AR given 100 detections per image
AR Across Scales:
  AR^small          % AR for small objects: area < 32^2
  AR^medium         % AR for medium objects: 32^2 < area < 96^2
  AR^large          % AR for large objects: area > 96^2
```

赛题介绍

† 比赛要求

- 框架: 本次比赛要求参赛者必须使用开源的旷视天元深度学习框架——MegEngine 作为模型训练的框架，关于 MegEngine 的教程、API 以及更多学习资料可前往 MegEngine 官网 获取。
- 数据: 不允许对任何数据做任何不合规的操作，如非法增加带标注或无标注数据
- 数据: 主办方提供的验证集（val）不允许参与任何训练，仅可用于验证和选择模型
- 模型: 不允许使用除COCO和ImageNet-1K之外的其它预训练模型
- 模型: 不允许多个模型 ensemble，最终结果只能使用单模型进行预测
- 模型: 不允许对模型进行离线训练
- 代码: 成果开源及代码审核，具体参考比赛链接
- 可复现性: 结果必须稳定可复现，最终获奖队伍需提供代码、模型及解决方案（MegStudio）
- 上述行为，必须全部符合才能记为有效成绩，否则 取消名次

赛题介绍

† 代码说明

- **MegEngine baseline:** https://github.com/er-muyue/megengine-trafficsign
  - 本Repo包含了采用MegEngine实现的FRCN、FCOS、ATSS三个主流模型，并提供了在交通标志数据集上的完整训练和测试代码
- Baseline Performance

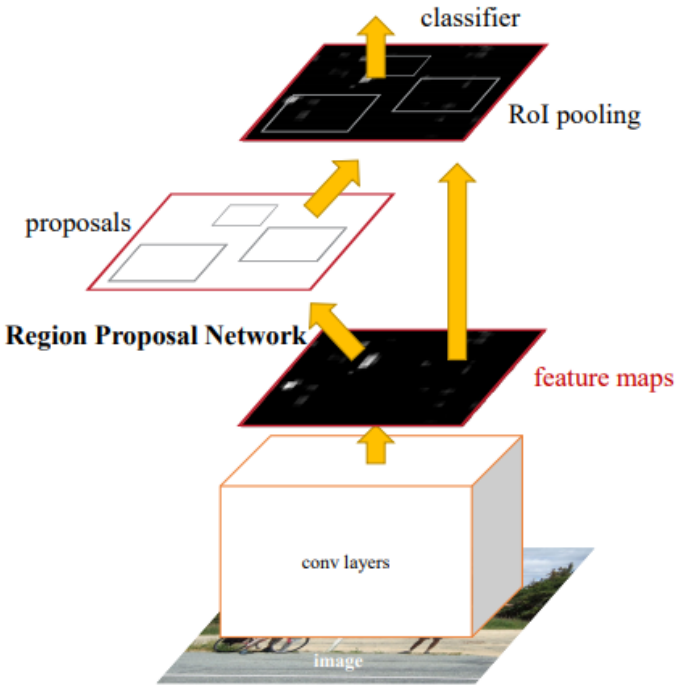
Model	mAP	AP50	AP75	APs	APm	APl	AR@1	AR@10	AR@100	ARs	ARm	ARl	
FRCN	44.5	69.4	49.6	30.3	52.4	67.9	42.3	56.3	56.6	41.6	63.3	76.9	1x
FRCN	**48.0**	71.4	**55.3**	**32.7**	58.0	74.2	44.7	58.6	58.7	42.3	67.8	81.5	2x
FCOS	38.2	60.4	41.2	18.8	48.2	68.3	37.5	51.0	52.3	31.7	63.0	80.3	1x
FCOS	46.6	66.9	51.7	26.3	57.5	75.0	45.0	60.0	60.9	40.8	71.9	84.7	2x
ATSS	38.4	59.6	42.2	20.4	48.4	65.7	37.6	51.8	52.8	33.3	63.0	77.3	1x
ATSS	46.8	67.5	52.6	25.7	58.8	75.1	44.3	60.5	61.2	40.3	73.2	85.7	2x

- 我们还在 **MegStudio** 平台提供了一份完整PPL的手把手教程项目，以便于参赛者能够快速执行一次训练和提交

赛题介绍

† 代码说明

- **Faster-RCNN (two-stage) VS. FCOS (one-stage)**
  - RPN + RCNN: 精度高但速度慢
  - FCN + Output: 精度低但速度快

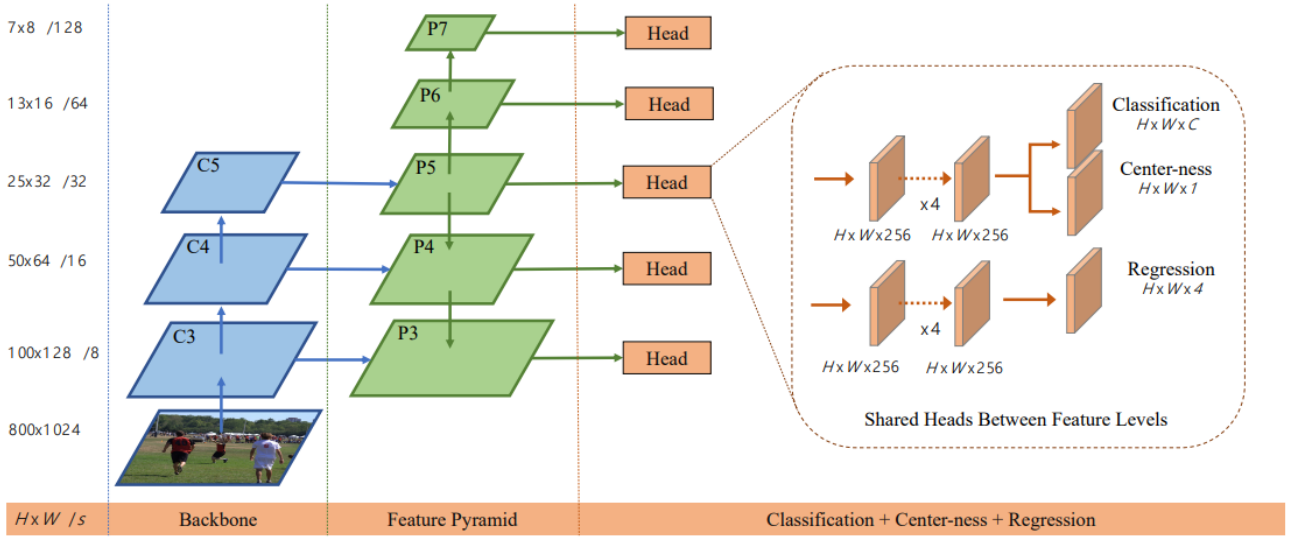



```python
class FasterRCNN(M.Module):
    """
    Implement Faster R-CNN (https://arxiv.org/abs/1506.01497).
    """

    def __init__(self, cfg):
        super().__init__()
        self.cfg = cfg
        # ----------------------- build backbone ------------------------ #
        bottom_up = getattr(resnet, cfg.backbone)(
            norm=layers.get_norm(cfg.backbone_norm), pretrained=cfg.backbone_pretrained
        )
        del bottom_up.fc

        # ----------------------- build FPN ----------------------------- #
        self.backbone = layers.FPN(
            bottom_up=bottom_up,
            in_features=cfg.fpn_in_features,
            out_channels=cfg.fpn_out_channels,
            norm=cfg.fpn_norm,
            top_block=layers.FPNP6(),
            strides=cfg.fpn_in_strides,
            channels=cfg.fpn_in_channels,
        )

        # ----------------------- build RPN ----------------------------- #
        self.rpn = layers.RPN(cfg)

        # ----------------------- build RCNN head ----------------------- #
        self.rcnn = layers.RCNN(cfg)
```

赛题介绍

† 代码说明

- **Faster-RCNN (two-stage) VS. FCOS (one-stage)**
  - RPN + RCNN: 精度高但速度慢
  - FCN + Output: 精度低但速度快




```python
class FCOS(M.Module):
    """
    Implement FCOS (https://arxiv.org/abs/1904.01355).
    """

    def __init__(self, cfg):
        super().__init__()
        self.cfg = cfg

        self.anchor_generator = layers.AnchorPointGenerator(
            cfg.num_anchors,
            strides=self.cfg.stride,
            offset=self.cfg.anchor_offset,
        )
        self.point_coder = layers.PointCoder()

        self.in_features = cfg.in_features

        # ---------------------- build backbone ---------------------- #
        bottom_up = getattr(resnet, cfg.backbone)(
            norm=layers.get_norm(cfg.backbone_norm), pretrained=cfg.backbone_pretrained
        )
        del bottom_up.fc

        # ---------------------- build FPN ---------------------------- #
        self.backbone = layers.FPN(
            bottom_up=bottom_up,
            in_features=cfg.fpn_in_features,
            out_channels=cfg.fpn_out_channels,
            norm=cfg.fpn_norm,
            top_block=layers.LastLevelP6P7(
                cfg.fpn_top_in_channel, cfg.fpn_out_channels, cfg.fpn_top_in_feature
            ),
            strides=cfg.fpn_in_strides,
            channels=cfg.fpn_in_channels,
        )

        backbone_shape = self.backbone.output_shape()
        feature_shapes = [backbone_shape[f] for f in self.in_features]

        # ---------------------- build FCOS Head ---------------------- #
        self.head = layers.PointHead(cfg, feature_shapes)
```

赛题介绍

† 赛题目标

- 我们希望你能:
  - 找到限制交通标志检测算法性能的本质难题并提出对应解决方案，如小目标、模糊目标、极端天气、目标破损等，从而基于baseline算法实现性能提升
  - 另起炉灶，基于Megengine实现SOTA的目标检测算法(EffDet、YOLOX)
  - 通过学习实现SOTA架构并应用到相应问题上
  - ……
- 而不希望仅仅是:
  - 使用了更多数据 or 更多模型 or 更多预训练 (不允许，参考比赛要求)

MEGVII 旷视

持续创新拓展认知边界 非凡科技成就产品价值