1. **算法简介**

近年来，与传统的两级检测器相比，一级检测器在图像数据处理上取得了具有竞争力的精度和更快的速度。然而，在视频目标检测领域，大多数现有的视频目标检测方法仍然是基于两级检测器。此外，直接将现有的VOD方法应用于单级检测器会带来难以承受的计算成本。在此基础上，EOVOD算法使用一个简单而有效的框架来解决计算瓶颈，并利用视频帧的时间一致性实现高效的单阶段视频点播。算法使用一个位置先验网络来过滤背景区域，一个大小先验网络来跳过对特定帧的低级特征映射的不必要计算。

1. **数据准备**
2. **下载数据集**

请从(http://image-net.org/challenges/LSVRC/2015/downloads)下载ILSVRC2015 DET和ILSVRC2015 VID数据集。之后，我们建议将数据集的路径符号链接到' datasets/ '。路径结构如下:

./data/ILSVRC/

./data/ILSVRC/Annotations/DET

./data/ILSVRC/Annotations/VID

./data/ILSVRC/Data/DET

./data/ILSVRC/Data/VID

./data/ILSVRC/ImageSets

注：“ImageSets”文件夹下的txt文件列表可从(<https://github.com/msracver/Flow-Guided-Feature-Aggregation/tree/master/data/ILSVRC2015/ImageSets>)下载。

1. **标签转化**

我们使用CocoVID来维护这个代码库中的所有数据集。在这种情况下，您需要将官方注释转换为这种样式。我们提供脚本，用法如下:

# ImageNet DET

python ./tools/convert\_datasets/ilsvrc/imagenet2coco\_det.py -i ./data/ILSVRC -o ./data/ILSVRC/annotations

# ImageNet VID

python ./tools/convert\_datasets/ilsvrc/imagenet2coco\_vid.py -i ./data/ILSVRC -o ./data/ILSVRC/annotations

1. **算法使用**
   1. **训练**

* **在单卡GPU上训练**

python tools/train.py ${CONFIG\_FILE} [optional arguments]

* **在多卡GPU上训练**

我们提供了' tools/dist\_train.sh '来在多个gpu上启动训练。基本用法如下

bash ./tools/dist\_train.sh \

${CONFIG\_FILE} \

${GPU\_NUM} \

[optional arguments]

可选参数与上面所述保持相同。

如果您想在一台机器上启动多个作业，例如，在一台具有8个gpu的机器上进行4个gpu训练的2个作业，您需要为每个作业指定不同的端口(默认为29500)，以避免通信冲突。

如果使用' dist\_train.sh '启动训练作业，则可以在命令中设置端口。

GetPlayer () .components.builder: GiveAllRecipes ()

CUDA\_VISIBLE\_DEVICES=0,1,2,3 PORT=29500 ./tools/dist\_train.sh ${CONFIG\_FILE} 4

CUDA\_VISIBLE\_DEVICES=4,5,6,7 PORT=29501 ./tools/dist\_train.sh ${CONFIG\_FILE} 4

* **一些可选参数**
* optimizer: 定义训练时使用的优化算法，这里使用的是随机梯度下降（SGD）算法。学习率为0.01/num\_gpus，动量为0.9，权重衰减为0.0001。
* checkpoint\_config: 配置模型保存的间隔，这里设置每隔1个epoch保存一次模型。
* log\_config: 配置训练过程中的日志记录方式，这里设置每隔50个batch记录一次日志。
* dist\_params: 配置分布式训练的参数，这里使用的是NCCL后端。
* log\_level: 配置日志的详细程度，这里设置为INFO级别。
* load\_from: 模型参数的加载路径，用于继续训练时加载预训练的模型。
* resume\_from: 恢复训练的路径，用于在中断的训练过程中恢复训练。
* workflow: 定义训练过程中的不同阶段和对应的迭代次数。
* lr\_config: 配置学习率的调整方式，预热比例为总迭代次数的1/3，然后在第2和第5个迭代时调整学习率。
* total\_epochs: 总的训练迭代次数。
* evaluation: 定义评估的指标和间隔，这里评估指标为bbox（边界框），每隔7个epoch进行一次评估。
* gpu\_ids: 使用的GPU设备ID范围，用于指定训练时使用的GPU设备。这里设置为使用第0号GPU设备。
* **示例**

训练EOVOD(FCOS)，然后在最后一个epoch评估AP。

./tools/dist\_train.sh configs/vid/fcos\_att/fcos\_att\_r101\_fpn\_9x\_vid\_caffe\_random\_level2\_imagenet.py 8

* 1. **测试推理**

本节将展示如何在支持的数据集上测试现有模型。

支持以下测试环境:

* 单GPU
* 单节点多GPU
* 多节点

在测试过程中，不同的任务共享相同的API，我们只支持“samples\_per\_gpu = 1”。

可以使用以下命令进行测试:

# single-gpu testing

python tools/test.py ${CONFIG\_FILE} [--checkpoint ${CHECKPOINT\_FILE}] [--out ${RESULT\_FILE}] [--eval ${EVAL\_METRICS}]

# multi-gpu testing

./tools/dist\_test.sh ${CONFIG\_FILE} ${GPU\_NUM} [--checkpoint ${CHECKPOINT\_FILE}] [--out ${RESULT\_FILE}] [--eval ${EVAL\_METRICS}]

* **测试时的一些可选参数**
* CHECKPOINT\_FILE:检查点的文件名。在应用某些MOT方法时不需要定义它，而是在配置中指定检查点。
* RESULT\_FILE: pickle格式的输出结果文件名。如果没有指定，结果将不会保存到文件中。
* EVAL\_METRICS:要根据结果评估的项目。允许的值取决于数据集，例如，' bbox '可用于ImageNet VID， ' track '可用于LaSOT， ' bbox '和' track '都适用于MOT17。
* --cfg-options:如果指定了，键值对可选的cfg将被合并到配置文件中
* --eval-options:如果指定，键值对可选eval cfg将为datasset .evaluate()函数的kwargs，它仅用于评估
* --format-only :如果指定，结果将被格式化为官方格式。
* **示例**

假设您已经将检查点下载到“检查点/”目录。

1. 在ImageNet VID上测试EOVOD，并评估bbox mAP。

python tools/test.py configs/vid/fcos\_att/fcos\_att\_r101\_fpn\_9x\_vid\_caffe\_random\_level2\_imagenet.py \

--checkpoint checkpoints/$CHECKPOINT\_FILE \

--out results.pkl \

--eval bbox

1. 在ImageNet VID上使用8个gpu测试EOVOD，并评估bbox mAP。

./tools/dist\_test.sh configs/vid/fcos\_att/fcos\_att\_r101\_fpn\_9x\_vid\_caffe\_random\_level2\_imagenet.py 8 \

--checkpoint checkpoints/$CHECKPOINT\_FILE \

--out results.pkl \

--eval bbox