1. **算法简介**

视频对象检测(VOD)是一项具有挑战性的工作，因为随着时间的推移，物体的外观会发生变化，从而可能导致检测错误。最近的研究集中在聚合相邻帧的特征以补偿帧的劣化外观。此外，还提出了使用远帧来处理多帧的劣化现象。

由于一个目标的位置可能会在一个遥远的帧显著变化，他们只使用目标候选区域的特征，而不依赖于他们的位置。然而，这些方法依赖于目标候选区域的检测性能，对于劣化外观并不实用。在目标候选区域检测之前，VSTAM算法对特征元素进行了明智的增强，提出了带有注意引导记忆(VSTAM)的视频稀疏变换。此外，VSTAM算法稀疏地聚合元素智能特征以减少处理时间和内存成本，引入了一种基于聚合的外部内存更新策略，以有效地保存长期信息。算法在ImageNet VID和UA-DETRAC数据集上实现了8.3%和11.1%的基线精度提高。

VSTAM算法在广泛使用的VOD数据集上显示出优越的性能。

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VSTAM，然后在最后一个epoch评估AP。

./tools/dist\_train.sh configs/vid/vstam/vstam\_config.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VSTAM，并评估bbox mAP。

python tools/test.py configs/vid/vstam/vstam\_config.py \

--checkpoint checkpoints/$CHECKPOINT\_FILE \

--out results.pkl \

--eval bbox

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

./tools/dist\_test.sh configs/vid/vstam/vstam\_config.py 8 \

--checkpoint checkpoints/$CHECKPOINT\_FILE \

--out results.pkl \

--eval bbox