fall_detection

项目路径:

`\\192.168.16.105\data_huangzhiyong\fall_detection`

目录结构

文件/目录	作用
fall_detector.py	执行跌倒检测的脚本
jit_models/	存放导出的TorchScript模型
videos/	测试视频的目录
out_videos/	跌倒检测后输出视频的目录
pose/	多人姿态估计项目，包括姿态相关的数据集、算法和公共的预训练模型参数
track/	目标跟踪项目，包含三种目标跟踪算法SORT/ BoT-SORT/ OC-SORT
classifier/	视频分类项目

1. 安装环境

```
pip install - r requirements.txt
cd track/botsort/
python setup.py develop
# Cython-bbox
pip3 install cython_bbox
# faiss cpu / gpu
pip3 install faiss-cpu
pip3 install faiss-gpu
```

2.pose

多人姿态估计算法是基于开源项目KAPAO做改进，得到另外两个算法YOLO-Pose和KAPAO_with_kp_conf。KAPAO的原理也很简单就是在YOLO的回归边界框的同时加上关键点回归，再把关键点当成框预测，最后做一个点框融合来提高关键点的预测精度。

三个算法的区别主要在于输出编码方式和是否有点框匹配，目前主要使用KAPAO_with_kp_conf。

三个算法的目录架构基本相同，以下介绍KAPAO_with_kp_conf的目录结构:

文件/目录	功能
cpp_detect	C++或Python的模型后处理
data	模型结构配置文件和模型训练的超参数文件
demos	测试模型，输出测试视频
models	定义模型结构
res	KAPAO官方效果图
runs	模型训练的输出的日志和模型参数等
utils	包括数据集处理、数据增强、损失函数等
dist_train_run.sh	用于启动训练的脚本
my_train.py	修改了官方的训练代码，便于调试
train.py	官方训练代码
val.sh	用于启动测试的脚本
val.py	测试代码

2.1 Train

```
cd pose/kapao_with_kp_conf
bash dist_train_run.sh
```

2.2 Val

```
cd pose/kapao_with_kp_conf
bash val.sh
```

2.3 Export ONNX or TorchScript

```
# 进入到 pose>*>models>yolo>Detect 文件，修改以下代码
onnx = dict(
    export=True,    # 为真，表示导出ONNX或TorchScript
    with_postprocessing=True    # 为真表示导出的模型带有后处理
    )
```

```
cd pose/kapao_with_kp_conf
python export_model.py    # export TorchScript
python
```

2.4 Debug

了解代码原理，只需要调试一遍就可以了模型的训练代码即可。

基本操作方式就是进入到pose目录，调试各个算法下面的my_train.py即可。

3. track

目录结构

文件/目录	功能
botsort/	BoT-SORT的官方源代码，包含目标检测和ReID模块
bot_sort/	只保留BoT-SORT必须的模块,不包含ReID模块
ocsort/	OC-SORT的官方源代码
oc_sort	只保留OC-SORT必须的模块
mot_benchmark	目标跟踪数据集
sort.py	SORT的官方源代码
object_track.py	整合三种目标跟踪算法的工具类 **ObjectTracker**

```
cd track
python object_track.py --display
```

4. classifier

分类器简单的参考SlowFast模型的思路和X3D的超参数配置，尝试了一下，但效果不好，模型过拟合了。

模型为双分支结构：

- 时间序列分支
  - 使用双层LSTM
  - 输入若干帧的边界框和关键点信息
  - 序列长度为16帧，帧间隔为4
- 空间序列分支
  - 使用多个Bottleneck卷积模块
  - 输入帧序列最后一帧对于的人体图像区域
- 融合模块
  - 将时间序列分支和空间序列分支的特征拼接在一起
  - 融合特征后输出全连接层进行分类

4.1 Train

```
cd classifier
# you can customize the training configures on dist_train_run.sh
bash dist_train_run.sh
```

4.2 Val

```
cd classifier
bash val.sh
```

4.3 Export ONNX or TorchScript

```
cd classifier
python export_model.py
```

5、fall_detection.py

python文件用于执行整套跌倒检测流程，包括多人姿态估计+目标跟踪+视频分类。

输入是一个视频，输出是一个结果可视化视频。

5.1、跌倒检测项目完整流程

1. 训练pose和classifier网络
2. 使用`export_model.py`导出TorchScript模型
3. 设置`fall_detection.py`中的参数
    1. `--pose-pth` 指定pose网络的TorchScript模型路径
    2. `--cls-pth` 指定和classifier网络的TorchScript模型路径
    3. `-t` 指定目标跟踪算法的类型，目前支持SORT\OC-SORT\BoT-SORT
    4. 设置其余检测的超参数，如NMS阈值，置信度阈值等等。
4. 执行`fall_detection.py`

5.2、代码原理

1. pose网络检测出当前帧中所有的人体框和关键点
2. 使用目标跟踪算法（ObjectTracker类）
    1. 给每个人体框匹配一个跟踪ID
    2. 根据跟踪ID将检测信息添加到相应的跟踪序列中，一个跟踪序列包含一个人从过去到现在所有的信息（人体框和关键点）。
3. 遍历所有的跟踪序列，classifier网络根据序列信息和当前帧图像，对每个人进行跌倒检测（分类）。

5.3、代码示例

```
python fall_detector.py --display -p "path/to/your/video"
```