Technical Manual

Anomaly Detection System

Media System Laboratory

Sungkyunkwan University

December 2018

TABLE OF CONTENTS

1. Installations

This implementation supports python 3+, imutils 0.5.1, torch 0.4.1 and visdom (optional). Install all installation requirements by selecting your environment on the website and running the appropriate command.

This implementation is based on;

https://github.com/amdegroot/ssd.pytorch

and

https://github.com/hli2020/object_detection

2. Datasets

Two datasets are employed for training the deep anomaly detector.

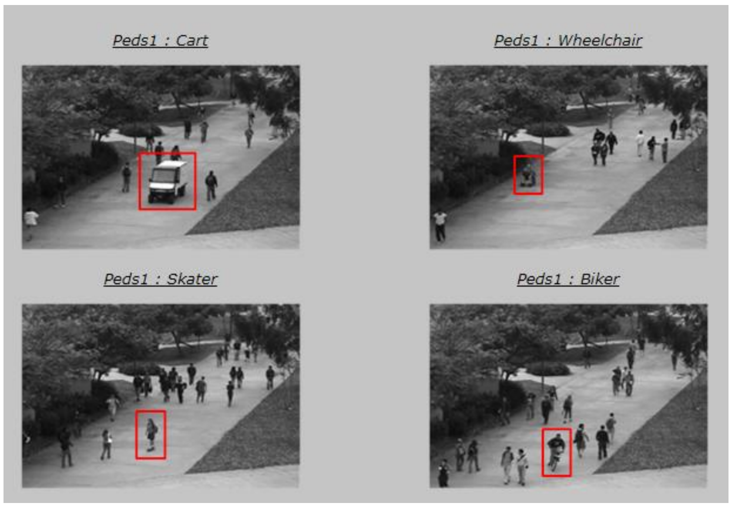
2.1 UCSD anomaly detection dataset

The UCSD Anomaly Detection Dataset was acquired with a stationary camera mounted at an elevation, overlooking pedestrian walkways. The crowd density in the walkways was variable, ranging from sparse to very crowded. In the normal setting, the video contains only pedestrians. Abnormal events are due to either:

- The circulation of non-pedestrian entities in the walkways
- Anomalous pedestrian motion patterns

Commonly occurring anomalies include bikers, skaters, small carts, and people walking across a walkway or in the grass that surrounds it. A few instances of people in wheelchair were also recorded. All abnormalities are naturally occurring, i.e. they were not staged for the purposes of assembling the dataset.

Examples of anomalies:



We have used total 4171 training images of UCSD dataset. The original size of UCSD images is 158 × 238, which is very small, we upsampled the images by factor of 3.

Dataset download

http://www.svcl.ucsd.edu/projects/anomaly/UCSD_Anomaly_Dataset.tar.gz

2.2 Koren CCTV videos dataset

CCTV video dataset is used for training with 4672 examples including both crowded and uncrowded scenes. Other training parameters are same as used for UCSD dataset.

Download original CCTV video from Koren server

```
scp -r $1 /home/datafromServer2
```

Upload processed CCTV video to Koren server

```
scp -r /home/datatoServer2/. $2
```

2.3 Training data preparation

Training dataset annotation is performed using VGG image annotator tool.

Label data using the rectangle option and place in

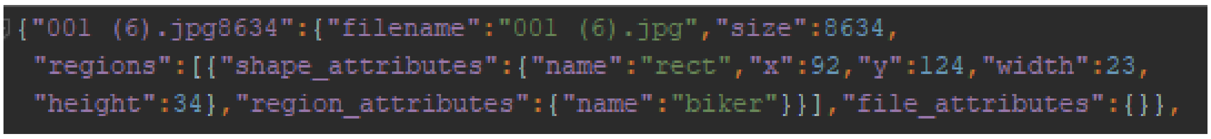
data/image_data/train (images and via .json file)

Processes the VGG Image Annotator json output into a list of dictionaries.

Json annotation file snippet:

{'480835.jpg1406946': {'regions': {}, 'filename': '480835.jpg', 'fileref': '', 'size': 1406946, 'file_attributes': {}, 'base64_img_data': ''}, '480980.jpg2644207': {'regions': {'0': {'region_attributes': {}, 'shape_attributes': {'y': 547, 'x': 2176, 'height': 249, 'name': 'rect', 'width': 178}}}, 'filename': '480980.jpg', 'fileref': '', 'size': 2644207, 'file_attributes': {}, 'base64_img_data': ''},...

Example of annotated image in json format:



```
{"001 (6).jpg8634":{"filename":"001 (6).jpg","size":8634,
 "regions":[{"shape_attributes":{"name":"rect","x":92,"y":124,"width":23,
 "height":34},"region_attributes":{"name":"biker"}}],"file_attributes":{}},
```

3. Training

Download the fc-reduced VGG-16 PyTorch base network weights at:

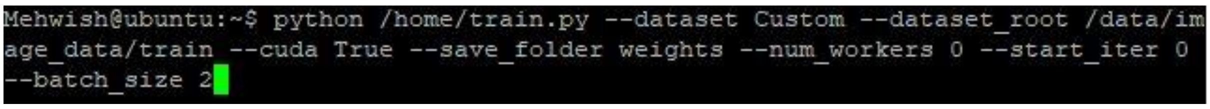
```
https://s3.amazonaws.com/amdegroot-models/vgg16_reducedfc.pth
```

By default, download the file in the ssd.pytorch/weights dir:

```
wget https://s3.amazonaws.com/amdegroot-models/vgg16_reducedfc.pth
```

To train SSD using the train script specify the parameters listed in train.py as a flag or manually change them.

```
python /home/train.py
```



```
Mehwish@ubuntu:~$ python /home/train.py --dataset Custom --dataset_root /data/im
age_data/train --cuda True --save_folder weights --num_workers 0 --start_iter 0
--batch_size 2
```

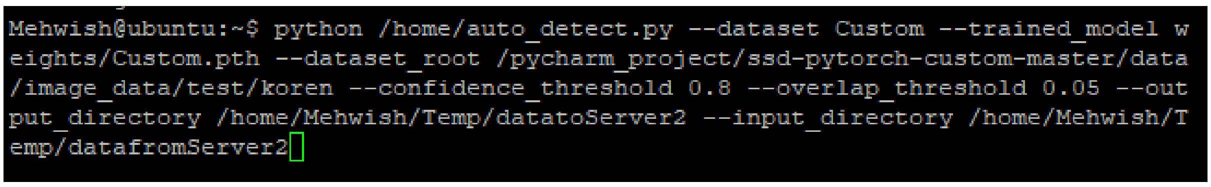
4. Inference

Use following command to test detections.

```
python /home/auto_detect.py
```

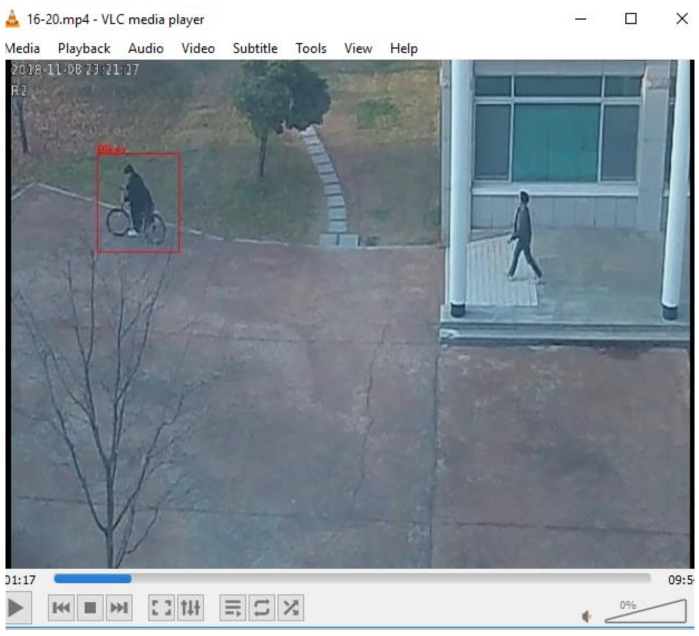
Save model pytorch using .pkl or .pth extensions.

```
--trained model weights/Custom.pth
```



```
Mehwish@ubuntu:~$ python /home/auto_detect.py --dataset Custom --trained_model w
eights/Custom.pth --dataset_root /pycharm_project/ssd-pytorch-custom-master/data
/image_data/test/koren --confidence_threshold 0.8 --overlap_threshold 0.05 --out
put_directory /home/Mehwish/Temp/datatoServer2 --input_directory /home/Mehwish/T
emp/datafromServer2
```

Example of detection during inference:


16-20.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
2018-11-08 23:21:17
Biker
01:17
09:5
0%

5. Evaluation

Evaluate a trained network with respect to mean average precision (MAP): MAP is the metric to measure the accuracy of object detectors. It is the average of the maximum precisions at different recall values.

Specify parameters listed in eval.py file.

```
python /home/eval.py
```

```
Mehwish@ubuntu:~$ python /home/eval.py --dataset Custom --trained_model weights/
Custom.pth --save_folder eval --dataset_root /data/image_data/test --confidence_
threshold 0.6 --overlap_threshold 0.05
```

6. Results

Detection outputs are of the following forms.

3 Detected bounding boxes coordinates, e.g. [223.7 350.9 258.2 422.7]
4 Prediction score for detected class, e.g. [0.624]
5 Number of total anomalies detected category wise

Example of total counts for each anomaly class detected in a video:

```
Number of detected Biker objects : 640
Number of detected Cart objects : 0
Number of detected Skater objects : 0
Number of detected Walking on Grass objects : 0
```