Human Detection

Konstantinos Raptis

Problem Statement - Tools

- 22 images with fashion models
- Each image, one person
- Detect the person, find the best possible bounding box
- Caffe + iPython (python 2.7)
- R-CNN with ImageNet as pretrained model
- Test on the 22 images data set

Data




- Not only humans but also part of humans such as legs
- Various Scales

Methods

- Cascade Training, HOGs + SVM one of the basic techniques for human detection.

  Problems: few images, not the whole person in every image. It is impossible to train a good detector neither for humans nor for human parts. There are pre-trained models for face, upper body and humans,but they are not applicable to these data.

- cNN can overcome the problems of the cascade training with HOGs, it is suitable for detecting humans as well as human parts. One of the best methods for human detection in images.

  Problem: few images for training, Solution: there are available pretrained models for object detection such as ImageNet.

R-CNN

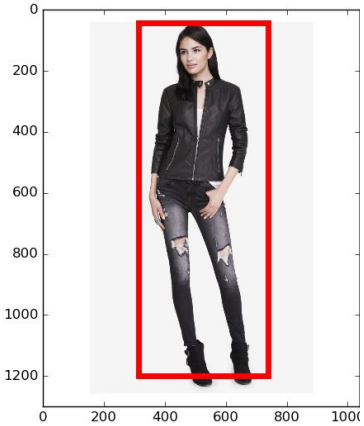
Step 1: Load Imagenet.

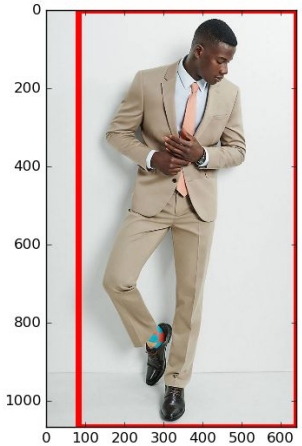
Step 2: Selective Search: Finding possible bounding boxes with objects. “Selective Search for Object Recognition”J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders
2013 IJCV.

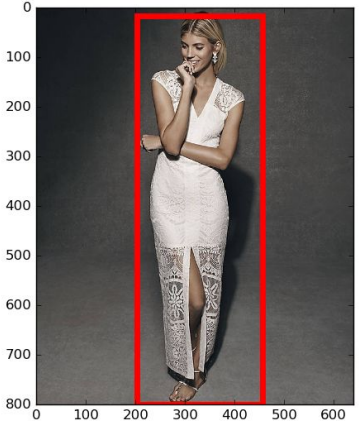
Step 3: Detect objects in the proposed areas (with caffe/python/detect.py). Only the bounding boxes containing a person are kept.

Step 4: Out of all detected objects, we keep the 4 most confident that the detected object is a person. If there is a great difference on the confidence between these 4, the bounding box is created by finding the min and max points of each of the 4 detections, otherwise the min of the bounding box is mean of the mins and the max is the mean of the maxs. If the first proposal is very strong (>1 and there is a great difference between the first and the rest) we only keep that one.

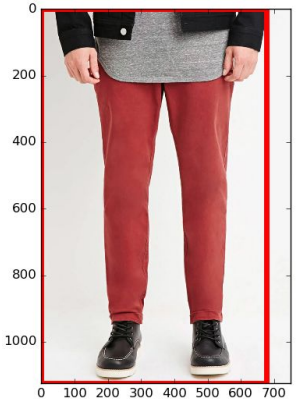
Results (1/4)

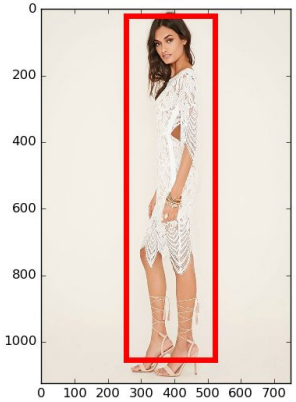

0
200
400
600
800
1000
1200
0
200
400
600
800
1000

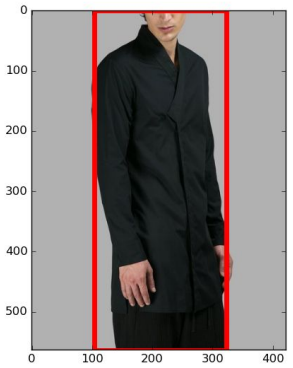

0
200
400
600
800
1000
0
100
200
300
400
500
600


0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

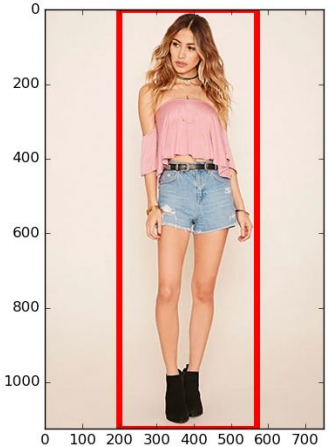
Results (2/4)

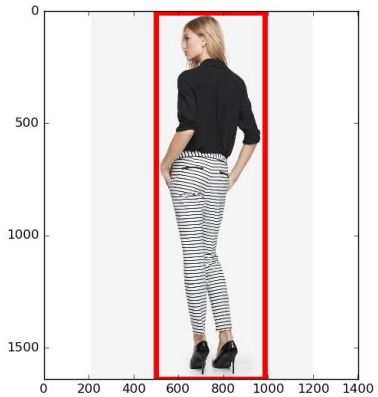

0
200
400
600
800
1000
0 100 200 300 400 500 600 700


0
200
400
600
800
1000
0 100 200 300 400 500 600 700


0
100
200
300
400
500
0 100 200 300 400

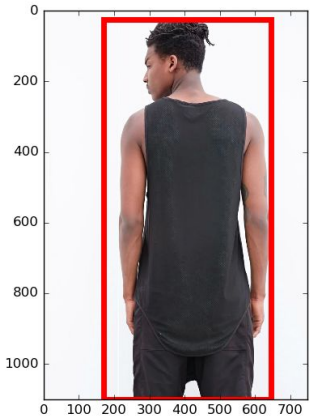
Results (3/4)

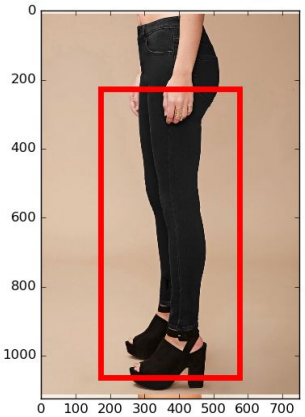

0
200
400
600
800
1000
0 100 200 300 400 500 600 700

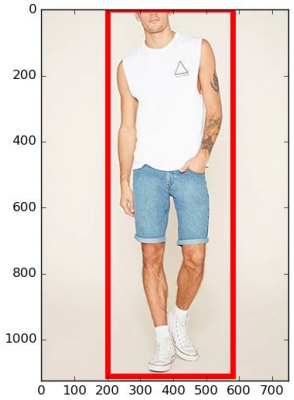

0
500
1000
1500
0 200 400 600 800 1000 1200 1400


0
100
200
300
400
500
600
700
800
0 100 200 300 400 500 600

Results (4/4)


0
200
400
600
800
1000
0 100 200 300 400 500 600 700


0
200
400
600
800
1000
0 100 200 300 400 500 600 700


0
200
400
600
800
1000
0 100 200 300 400 500 600 700

Discussion

Platform: Ubuntu 14.04 with i3 2.5 and 4 GB RAM. Caffe is running only on CPU and not on GPU.

I couldn’t run the example_22_1 because there are too many proposed areas from the selective search and I had an out of memory error. I tried to remove some overlapped boxes but you loose a lot of information. Also, for better results you have to run all the testing data together at once. Unfortunately, because of the lack of memory, I could only test the model for one image at a time. The average time for running an image with the above configuration is about 7 minutes.

Instructions

I uploaded part of the code on my github account (_temp, selective search, detector.py, ipython file, results).

Inside the caffe/_temp folder there is a det_input.txt file, in that file you write the path of the image that you want to test. If you want to test the whole data set in once just add all the paths here.

The ipython file name is caffe_fashion_human_detection.ipynb and it is inside the caffe folder.

For downloading the ImageNet: ./scripts/download_model_binary.py models/bvlc_reference_rcnn_ilsvrc13

It also needs the data/ilsvrc12.

I had to do small changes on the detector.py (it is included on caffe library) and on the selective_search_ijcv_with_python code.