ML Reading Group (Fall 2015) **Lecture: 2**

A Convolutional Neural Network Cascade for Face Detection

Lecturer: Yutaro Yamada

1 Today

(– Reading Group Mechanics (MIT/Stanford))

– Example of linear algebra / theory (from last time by Jonathon?)

– Paper discussion

– Sample code in Chainer (http://chainer.org/)

2 Quick Summary

1-Line Summary:

A nice approach to uncontrolled face detection using convolutional neural network as a boosted cascade classifier, with multi-resolution architecture and bounding box calibration to increase robustness.

Weaknesses:

1. Computational cost is (still) high.

2. The developed method is a natural extension of the boosted cascade to the domain of neural network. I would imagine that many researchers have thought of a similar idea for non-frontal face detection.

Strengths:

1. The idea of applying CNN to bounding box calibration is unique.

2. The performance is good.

Comments:

1. The contribution of multi-resolution architecture seems minimal to me.

3 Major Ideas

1. The boosted cascade has been used in face detection for many years as a nicely balanced classifier for both computational expense and performance.

2. Authors claim that, in cascade architecture, only simple classifiers have been used in the past.

3. They apply CNN for this classifier to increase performance.

4. Some tweaks they add include: multi-resolution architecture (12-net, 24-net, and 48-net; Figure 2) and bounding box calibration (Figure 4).

4 Network Architecture

(location: paper-mode/Network/facenet.py and calibnet.py)

They use two shallow nets (12-net, 24net) and one normal net (48-net). The shallow nets correspond to the series of classifier in the cascade step in Viola-Jones. They also apply calibration in-between 12-net and 24-net, and 24-net and 48-net. The calibration is an automated correction of the window so that the detection bounding box is aligned to faces more accurately. The calibration can also be achieved by solving neural net classification problem.

Although the paper says the kernel size for ConvNet is 3, this should be either 2 or 4; otherwise, the pooling step wouldn't work, so I'm assuming it's a typo. We chose 2 in 12-net, 4 in 24-net and 48-net in favor of prediction speed.

12-net: INPUT: 12x12x3, memory:12x12x3=432, weight=0 CONV: i=3, o=16, k=2, s=1, p=0, memory:11x11x16= , weight:(3x3x3)x16= POOL : k=3, s=2, memory: 5x5x16= RELU : size of channel=16 FC1: i=400, o=16, memory: 400x16=6400, weight: 400x16= RELU: FC2: i=16, o=2, memory: 16x2=32, weight: 16x2=32 RELU:

24-net INPUT: 24x24x3, memory:24x24x3=, weight=0 CONV: i=3, o=64, k=4, s=1, p=0, memory:21x21x64= , weight:(3x4x4)x64= POOL : k=3, s=2, memory: 10x10x64=6400 RELU : size of channel=64 FC1: i=6400, o=128, memory: 6400x128=, weight: 6400x128= RELU: INPUT: 12x12x3 -¿ 12-net -¿ output=output of 12-net-FC1 (this will be added to FC1 in 24-net) FC2: i=144, o=2, memory: 144x2=288, weight: 144x2=288 RELU:

48-net INPUT: 48x48x3, memory:48x48x3, weight=0 CONV1: i=3, o=64, k=4, s=1, p=0, memory: 45x45x64=129600, weight:(3x4x4)x64 POOL1: k=3, s=2, memory: 22x22x64 NORM1: window size=9 CONV2: i=64, o=64, k=4, s=1, p=0, memory: 19x19x64, weight: (64x4x4)x64 NORM2: window size=9 POOL2: k=3, s=2, memory: 9x9x64, weight FC1: i=5184, o=256, memory: 5184x256, weight: 5184x256 INPUT: 24x24x3-¿ 24-net -¿ output=output of 24-net-FC1 FC2: i=384, o=2, memory:384x2, weight: 384x2

12-calibration-net INPUT: 12x12x3, memory:12x12x3=432, weight=0 CONV: i=3, o=16, k=2, s=1, p=0, memory:11x11x16= , weight:(3x3x3)x16= POOL : k=3, s=2, memory: 5x5x16= RELU : size of channel=16 FC1: i=400, o=128, memory: 400x128, weight: 400x128 RELU: FC2: i=128, o=45, memory: 128x45, weight: 128x45 RELU:

24-calibration-net INPUT: 24x24x3, memory:24x24x3, weight=0 CONV: i=3, o=32, k=2, s=1, p=0 POOL : k=3, s=2, memory: 5x5x32 RELU : size of channel=32 FC1: i=6400, o=64, memory: 6400x64, weight: 6400x64= RELU: FC2: i=64, o=45, memory: 64x45, weight: 64x45 RELU:

48-calibration-net INPUT: 48x48x3, memory:48x48x3, weight=0 CONV1: i=3, o=64, k=4, s=1, p=0, memory: 45x45x64=129600, weight:(3x4x4)x64 POOL1: k=3, s=2, memory: 22x22x64 NORM1: window size=9 CONV2: i=64, o=64, k=4, s=1, p=0, memory: 19x19x64, weight: (64x4x4)x64 NORM2: window size=9 FC1: i=23104, o=256, memory: 23104x256, weight: 23104x256 FC2: i=256, o=45, memory:256x2, weight: 256x2

5 Training Process

For training the 12-calibration net, 24-calibration net, and 48-calibration net, we used 590082 image samples, and used 9/10 of them as training set and 1/10 as test set. We generated the training set by modifying the bounding box of each annotated face into 45 patterns, that are pre-defined by the paper. The details for the generating process are described in the paper as well as in our code. (We recommend using GPU on Amazon Web Service, once a script is fully developed and finalized on local GPU, since AWS GPU is much faster. For example, it took 2 days to train 12-calibration-net on our machine, but only 4 hours on EC2 in AWS.)

For training 12-net, we followed the procedure that is similar with the previous models. 24-net was trained with the same positive training set as 12-net, but with different negative training set, which are generated by a cascade procedure that consists of 12-net and 12-calibration-net. 48-net was trained similarly, but with a cascade procedure that consists of 12-net, 12-calibration-net, 24-net, and 24-calibration-net.

6 Result of Calibration

Calibration is an automated correction of the window so that the detection bounding box is aligned to faces more accurately. The calibration nets are trained so that they can automatically determine how much a given window of face is off from the real face center in the prediction step. In the cascade detection scheme, the calibration step dramatically reduces the number of faces which the following detection net needs to process, and thus reduces overall computation time. An example of calibration is shown below. The script that generated this example is located at /paper-model/temp4.py.

7 Formalism

I think it's important to understand the theory behind convolutional neural network (CNN) since the proposed method is entirely based on CNN. There are many good articles about CNN on the web (`http://cs231n.github.io/`, `http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/`) so I'd go through them if you are not familiar with CNN.