Deep Learning
Exercise 2

Due Date : 1 December 2017
21 november 2017

Professor: Lior Wolf
TA: Eliya Nachmani

Instruction:

1. Use Pytorch
2. In each group should be 3 students
3. Submit the results to - eliyan@mail.tau.ac.il. The subject of the mail should be **DL course EX2**, in the body of the mail write your names & IDs, and attach the zip file.
4. You should download /afwl and /fddb folders from Link.

1 Introduction

This exercise will partially follow the algorithm suggested in the CVPR 2015 paper: A Convolutional Neural Network Cascade for Face Detection. The method proposed by the writers, is to train a cascade of small convolutional networks. As a result, the method described is both fast and accurate. The reading of the paper is **critical** to the understanding of this exercise.

2 Data

The data for this exercise is based on 2 datasets.

1. AFLW - is a publicly available dataset that contains images with annotated faces. To save you time, the face images were already cropped in appropriate size for your network. You can find the data in this link: /aflw/alfw_12.t7 and /aflw/alfw_24.t7. The first one contains crops of size 12×12 while the other 24×24. You should use torchfile package to open the files.
2. PASCAL VOC 2007 - this dataset contains images labeled with multiple classes and was used as detection benchmark. Images without the class "person" will be used as negative samples for your network training.

3 Code

Face detection is a challenging task due to the fact that faces may appear in various pose, scale, facial expression, occlusion, and lighting settings. A common practice in computer vision - in order to tackle the scale issue - is to apply detection algorithms on various scales of the image. This is done by building the test image into image pyramid. Additionally, to suppress multiple detections that overlap non-maximal suppression(NMS) is applied. NMS removes overlapping detections while keeping the detection with the highest score. Additionally, NMS requires the overlap ratio (intersection-over-union) between the detections to suppress.

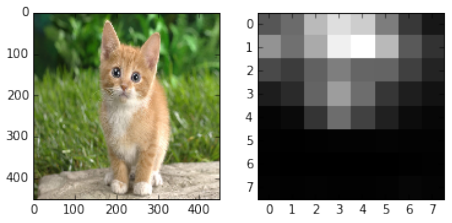
4 Fully convolutional network

One of the simplest ways to perform object detection is sliding window approach. Where each window of fixed sized is scanned for an object by running the network on it. However, this is costly and greatly degenerates a given detector speed. A better solution, would be to transform the network into fully-convolutional network(FCN) by converting fully-connected layers to convolution layers. The reason this method is more efficient is that convolution outputs are shared between multiple windows.

The output of FCN on a given image is a probability map for the existence of the object in each window. The size of the window is the network input size while the stride between each window is decided by the amount of down sampling (i.e. max pooling) the network performs. For example: consider activating a fully convolutional network with input size of 227 and stride of 32 over an input image of size 451×451. Using the equation for output map:

$output = (input - network\_input\_size)/stride + 1 = (451 - 227)/32 + 1 = 8$

Figur 1: Fully convolutional layer input and output



In conclusion, each cell (i, j) in the heat-map contains the probability for an object existing in window with bottom left point $(32i, 32j)$ of spatial size 32×32. In this exercise, each cell will contain the probability of the window containing a face.

5 FDDB

To evaluate the detector you'll write in this exercise we will use the Face Detection Data Set and Benchmark(FDDB). This is done by:

1. Run the detector over images found in /fddb/images.
2. Save the detections in the elliptical format described here. The detections should be written to a file named: fold-01-out.txt.
3. Say "test" is the name of the directory that contains your fold-01-out.txt. Run /fddb/evaluation/runEvaluation.pl test. This will result a text file called testDiscROC.txt which you are required to submit in the following questions.

Note that the detections in FDDB are in elliptical format. Therefore, you may need to adjust the square detections generated by the detector (For example, increase detection height by 20% and shift the detection center upwards).

6 Question 1: 12-net

Train the network depicted as 12-net in the paper. The networks receives as input a patch of size 12×12 from a RGB image - and classifies it as either face or non-face. As positive samples you should use AFLW faces, for negative samples crop random patches from the PASCAL dataset images.

1. Submit the training code.
2. Save the final model on nova.
3. Plot the cost function value for test and train after each epoch, and report final test classification error.

7 Question 2: Simple detector

Implement a simple face detector. The detector will classify windows of size 12×12 with a stride of 2 in the image as face or non-face using the 12-net. This will be done on multiple scales of the image in order to detect faces at various scales. Use FCN version of 12-net in your implementation to make the detector faster. Additionally, apply NMS over **each scale separately** to reduce the number of false-positives.

1. Submit the code that you used to create the FCN.
2. Submit the detector.
3. Report the recall of your model on the FDDB benchmark - submitting the DistROC.txt file.

Hints:

1. When using FCN version of 12-net, you will automatically achieve stride 2 with 12×12 window size.
2. This detector should have a high recall (above 90%) with quite a lot false positives per image (400-500).
3. The exact number of scales to use it up to you(to achieve high recall).

8 Question 3: 24-net

Train the network depicted as 24-net in the paper. You can ignore the concatenation of 12-net. As positive samples you should use AFLW faces. For negative samples you should densely scan each background image with the 12-net - each window classified as positive(i.e. face) will serve as a negative sample (this is known as negative mining/bootstrapping).

1. Submit the training code including the negative mining code.
2. Save the final model on nova.
3. Plot the cost function value for test and train after each epoch, and report final test classification error.

9 Question 4: Better detector

Improve the face detector from question 2 using 24-net. This will be done by taking each window classified as positive by the 12-net, scaling it to 24×24 and using the 24-net to classify it. This time, apply NMS over all scales at once to reduce the number of false positives.

1. Submit the detector.
2. Report the recall of your model on the FDDB benchmark - submitting the DistROC.txt file.

Hints:

1. Recall torch **image**, specifically use image.scale and image.crop method.
2. The recall of your model should be lower, but with fewer false positives.