Face Detection in Images

Dengxin Dai

Computer Vision Lab, ETH Zurich

March 25, 2017

1 Tasks

In this challenge, you will be implementing a sliding window face detector. The sliding window model is conceptually simple: independently classify all image patches as being face or non-face. Face detection is one of the most noticeable successes of computer vision. For example, modern cameras and photo organization tools have prominent face detection capabilities.

Proclaim:

- Do not use any existing code or commercial software for the detection.
- Write your own code for the parts where it is explicitly required, do not search over the internet for code of these parts. You code will be checked.
- Do not use any extra training data.

1.1 Data

Training Data for Classification: a collection of training samples with positive samples (face) and negatives (non-face) is provided at https://drive.google.com/file/d/0B8VmRliykLOvUTU5dGdFVDhBbmc/view?usp=sharing.

Testing Data for Classification: A collection of images without labels will be provided for testing. You need to generate labels of face vs. non-faces (1 vs. 0) for these images.

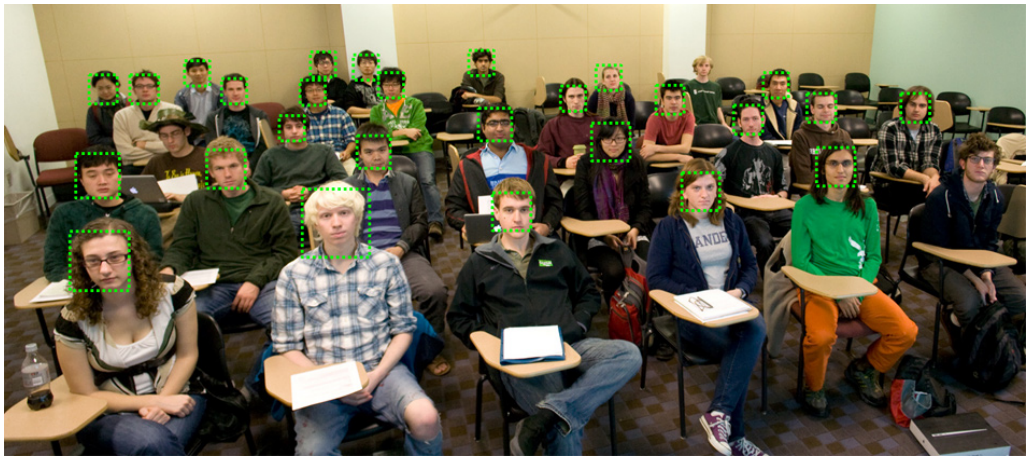

Figure 1: Example of face detection.

Testing Data for Detection: A collection of images will be provided, on which you need to detect the faces. See Fig. 1 for an example. Examples are available at https://drive.google.com/file/d/0B8VmRliykLOvZi1HSm9YdHRvbGc/view?usp=sharing.

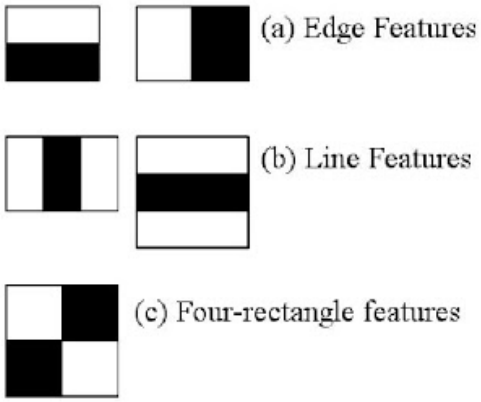

Figure 2: Illustration of the five features.

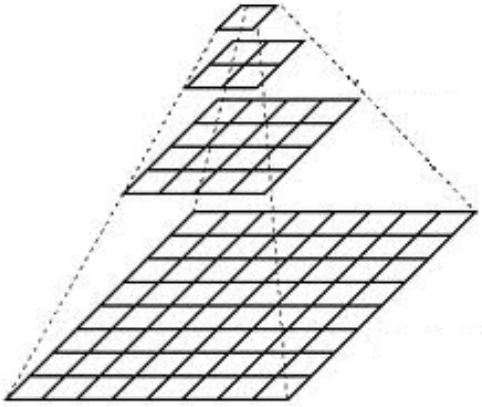

Figure 3: Example of an image pyramid.

1.2 Task 1: Face Classifier

Write your code to train a model to classify images to face vs. non-face:

1. **write your own code** to extract this feature 2 from an image; given an image patch, the value of a feature $= \sum$ (pixel values in white area(s)) $- \sum$ pixel values in black area(s). The white area and black area are equal in size. You may want to extract the features at multiple scales (why?). Using an image pyramid 3 for that purpose, where the five features are extracted from each of the cells of the pyramid. For example, an 1-layer pyramid leads to a 5-dimensional feature, a 2-layer pyramid leads to a 25-dimensional feature $(1+4)*5 = 25$, and a 3-layer leads to an 105-dimensional feature $(1+4+16)*5 = 105$ ...
2. select a good trade-off for your feature dimensions. Explain why this is good. You can do it either empirically or by techniques such as Cross-Validation.
3. train a classifier of your choice for a binary classification of face vs. non-face. you are allowed to use any existing library for the classifier. SVMs, Random Forest, Adaboost, Logistic Regression are among the popular choices.
4. predict the labels for the test data provided for this task; the predicted labels should be written into a '.txt' file, each line contains a filename and its label, separated by a space. The same format as the training data.
5. (Optional) use Hard Negative Mining to further improve your classifier. A hard negative is when you take that falsely detected image, and explicitly create a negative example out of that image, and add that negative to your training set. When you retrain your classifier, it should perform better with this extra knowledge, and not make as many false positives.

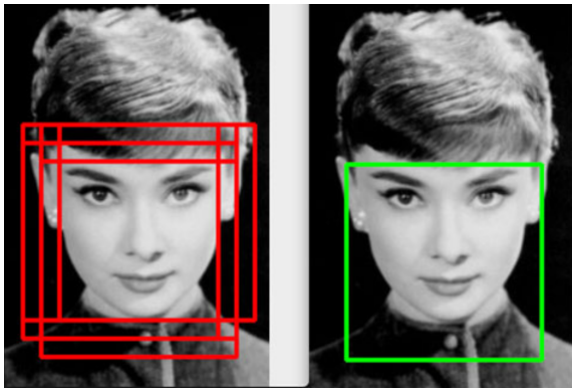

Figure 4: example of non-maximum suppression for face detection.

1.3 Face Detector

The second task is to extend your face classifier to a face detector, which consists of the following tasks. In the end, your algorithm should generate similar results as Fig. 1 shows (maybe not as good).

1. **write your own code** to sample patches at multiple scales, over different positions;
2. Run your face classifier over all patches sampled from an given image, and classify whether they are faces or no-faces;
3. **write your own code** for non-maximum suppression if multiple detections are heavily overlapped; they are actually fired by the same object. See Fig. 4. tip: for overlapping detections, keep the detection with the highest detection score.
4. write your outputs as images with bounding boxes imposed onto the original images.

1.4 What to submit

- A '.txt' file for the classification output over the classification testing images.
- The detection results (bounding boxes imposed over images) of all detection testing images.
- The code, with a brief readme of its functions.