University of Ioannina
Polytechnic School
Department of Computer Science and Engineering

Assignment 4
Face detection with a sliding window

Christina
Seventikidou
Semester: 3d

Master Course:
Computer Vision
Tutor: C. Nikou

June, 2021

Contents

1 Introduction

Face detection is usually the first step towards many face-related technologies, such as face recognition or verification. Face detection can have very useful applications,such as in photography. In this case the face detector detects where the faces are and adjusts the focus accordingly.Also, head pose estimation is another application that heavily relies on face detection. In general, face detection is an important part of many biometric, security, and surveillance systems, as well as image and video indexing systems.

1.1 Project description

This project targets face detection in images from a computer vision point of view. To achieve a robust detection method, a HOG plus SVM solution was developed, as proposed by Dalal Triggs at 2005.
The implementation of this method has been done using the MATLAB environment.

The results showed a very good ability of HOG method to recognize faces with an accuracy of about 90 percent. This method has the advantage of recognizing the faces basing on the different facial gestures.Furthermore we used SVM model with Caltech and SUN schene datasets for training and CMU+MIT for testing.The test set is challenging because the images are highly compressed and quantized and in many images there are not human faces but illustrated.

In this project we did not implement HG. We did the rest of the detection pipeline, meaning handling heterogeneous training and testing data, training a linear classifier, and using our classifier to classify millions of sliding windows at multiple scales.

1.2 Object detection background

The concept is that every object(face in this case) has it's own special features that helps in classifying the class. The methods for object detection generally fall into either neural network-based or non-neural approaches.

For non-neural approaches, it becomes necessary to first define features and then using a technique(Machine learning model) to do the classification. On the other hand, neural techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks.

2 Feature extraction

We can use computer vision techniques to perform feature extraction, to encode the information required for face detection as a compact feature vector using techniques and algorithms.Dalal-Triggs introduces the Histogram of Gradients (HG) feature descriptor.

2.1 Hog Method

HOG is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image.

A feature descriptor is the representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information.

Typically, a feature descriptor converts an image of size "width x height x 3" (channels) to a feature vector/array of length n.

The feature vector is not useful for the purpose of viewing one image, but it is very useful for tasks like object detection. The feature vector when fed into an image classification algorithm, like SVM, produces good results.

In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners and so we know that there is a lot information there about the object.

Histogram of oriented Gradients decomposes the image into square cells of a given size, compute a histogram of oriented gradient in each cell, and then re-normalizes the cells by looking into adjacent blocks. We use $vlfeat$ library to achieve this in Matlab. Figure 2 visualize the oriented gradients within each cell of one image.

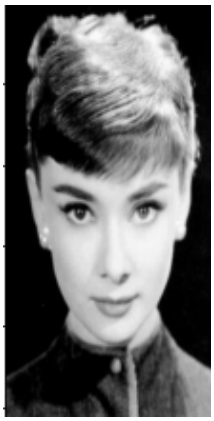

Figure 1: Original image

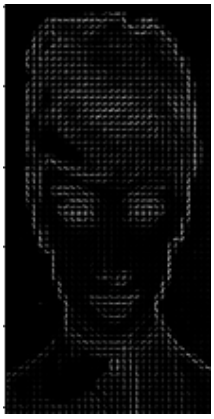

Figure 2: HOG representation

3 SVM

3.1 SVM classifier

SVM or Support Vector Machines are supervised learning models that are able to analyze complex data in order to automatically classify similar data that they have never seen before. We need to provide labeled data for the classes in which we wish to have our data classified. SVMs work by moving data into a relatively high dimensional space and find a high dimensional support vector classifier that can effectively classify the observations.

The way in which SVM deals with this is through kernel trick. Often non linearly separable features become linearly separable when mapped to a high dimensional feature space, without move them there. This allows us to use the same classifying method without an extra effort, the only requirement is to transform in some way our data via a kernel function. In this project we trained a linear SVM for classifying the sliding windows.

3.2 SVM training and testing

Using these HOG features, we can build up a facial detection algorithm with linear support vector machine estimator. The steps are as follows:

1. Obtain a set of image thumbnails of faces("positive" training samples).
2. Obtain a set of image thumbnails of non-faces("negative" training samples).
3. Extract HOG features from these training samples.
4. Train a linear SVM classifier on these samples.
5. Testing: For an "unknown" image, pass a sliding window across the image, using the svm model to evaluate whether that window contains a face or not.

   The steps are described in detail, in the implementation section.

4 Detector

When we have the model in place, we can test a new image and see how the model does. We will use one portion of an image and run a sliding window over it and evaluate each patch.Next, we create a window that iterates over patches of this image, and compute HOG features for each patch. Finally, we can take these HOG-featured patches and use our model to evaluate whether each patch contains a face. This process is actually our detector.

5 Implementation

In this section we give a brief review of the implementation and some details of each detector component.The development of the project has been made using MATLAB. Below are the methods that were made:

- get positive features : find positive training samples that show a variety of faces. We have used Caltech dataset with 6713 images that have dimension 36 by 36. The function gets the path of the images' folder and the feature params which are defined.It returns the hog features of

each image, a matrix with size 6713 by 1116. For each image it creates a 1116-dimension vector that are the hog features of each image.

- get random neg features: Next we need a set of similarly sized thumbnails which do not have a face in them. We used SUN scene database dataset which has 274 images with dimension 212x259x3. It returns a matrix with size 10048 by 1116. So this function return negative training examples (non-faces) from any images in 'non face scn path'. Images are converted to grayscale, because the positive training data is only available in grayscale. One way for better performance is to sample random negative examples at multiple scales. 'num samples' is the number of random negatives to be mined.

- trainClassifier: This function gets the path of training face images and non face images and the feature parameters. We train a linear SVM, and tune the lambda parameter to get W and b which are returned.

- run detector: This function performs the face search over all the images found in the test set folder.The W and b from trainClassifier are used to classify any new image based on a threshold. We can eliminate the threshold parameter by simply retaining only the top or highest values obtained from (W'*X + b). Here, we tune a threshold as the number of faces in the test image is unknown. Patches of test image whose feature descriptor has a value of (W'*X + b) higher than threshold is classified as face. Also, we would prefer not to have many detections of the same face, but to reduce overlapping groups of detections down to a single detection. This was done via a procedural approach: non-maximum suppression - an algorithm common in machine vision - If non maximal suppression in desired all positives matches can be drawn.

6 Performance

The parameters that had to be tuned are lambda and the threshold. The performance is compared based on based on Average Precision and values of confusion matrix (True Positives, False Negatives etc).

Precision, Recall and F-score: In pattern recognition with binary classification two different measures can be done. We would define:

- tp: true positives, the number of correctly labeled as belonging to the positive class.
- tn: true negatives, the number of correctly labeled as belonging to the negative class.
- f tp: false positives, the number of wrong labeled as belonging to the positive class.
- fn: false negatives, the number of wrong labeled as belonging to the negative class. So we have

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

$$Fscore = \frac{precision * recall}{precision + recall}$$

Almost all values of lambda from 0.0001 to 0.1 gave a classifier resulting in 99.3 to 99.8 percent train accuracy. Hence in order to avoid over fitting, we tuned it to get a classifier with slightly lower train accuracy. We keep 0.0001 as final. Below are the results for some trials in threshold parameter.

Table 1: threshold tuning keeping lamda=0.001

threshold	Svm train accuracy	Average precision
0	0.995	0.848
0.2	0.996	0.845
0.5	0.993	0.855
-0.1	0.996	0.866
-0.3	0.996	0.858
-0.5	0.994	0.838

We can see that the best average precision is for threshold value -0.1, and so this value was kept. The detector was tested many times and the value of -0.1 gave the best average precision. Below are the visualizations.

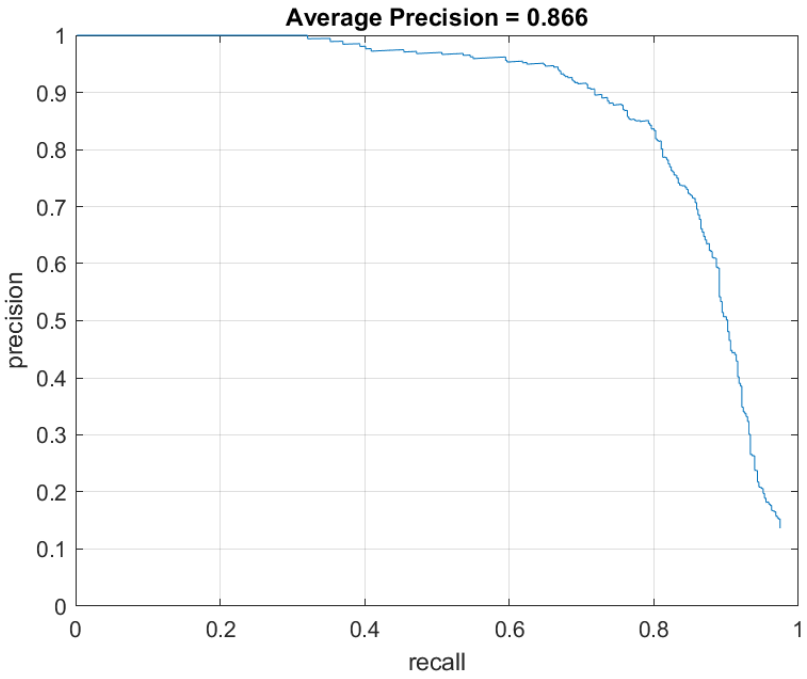

Figure 3: Average precision with threshold -0.1

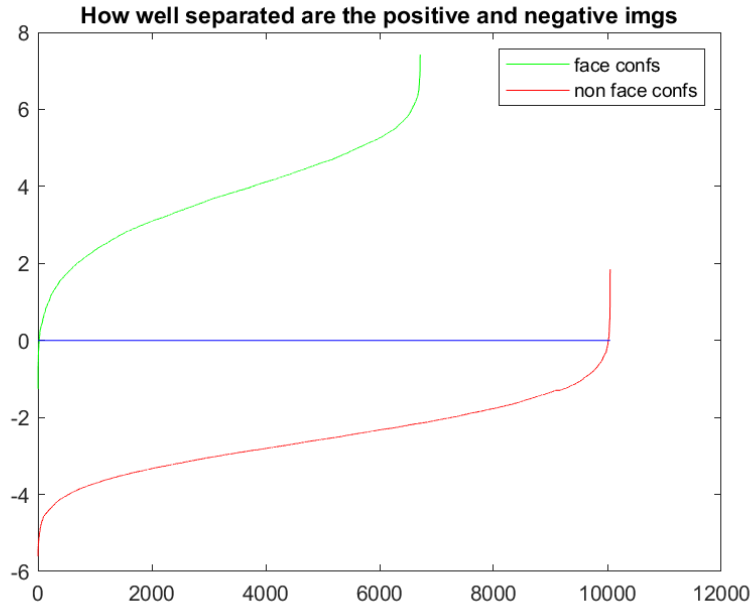

Figure 4: face and non face confidences

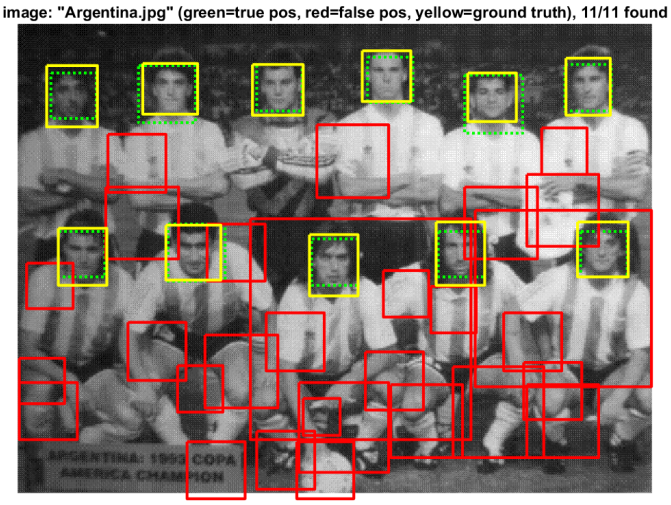

Figure 5: Detector for threshold -0.1

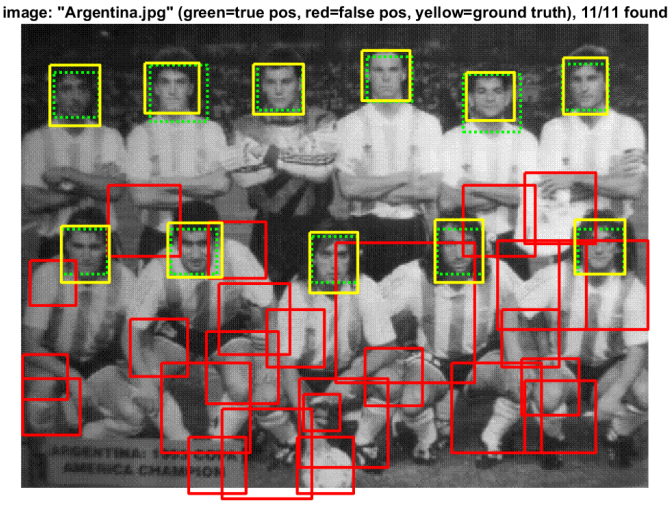

Figure 6: Detector with threshold -0.3

We can see in Table 1 that threshold value -0.3 gives also good results. This happened for all trials for our detector.

We can see that threshold -0.1 reduces the false positive rate, in comparison with -0.3. We chose -0.1 for the final detector.

7 Conclusion

One extension for achieving better precision would be to do negative hard mining. This is a technique to make the classifier more robust by taking taking into consideration the mis classifications we get in the training data itself. One other extension would be to use an other classifier like neural networks.