Lab 09: HOG

Adrian S. Volcinschi M.
Universidad de los Andes
https://uniandes.edu.co/en
as.volcinschi@uniandes.edu.co

Abstract

Face detection is a problem that dates back to the 1990s approximately. This problem has been approached in various ways but a very intuitive one was using HOG to extract features of the shape of a face to detect them in an image later on. This was not enough so a multiscale HOG was proposed as a better solution for the problem. The multiscale HOG using a SVM as a supervised classifier is the proposed approach in this paper. The results are good for the test dataset, 0.8561 of average precision, but it fails to work with more natural images.

1. Introduction

The detection of faces is a problem that has been around for a while now. The problem consists of placing a rectangle around the face of each person in a photo. This is a problem of interest because of the potential uses it may have. One very important application of these face detectors was to detect faces when using a digital camera. Other examples can be for security tapes to follow each person along the video.
One of the first attempts of this was done by Turk et al. [3] in 1991 where they calculated something called "eigenfaces" which are some vectors that create a "face space" which represent the characteristics a face may have. This problem began to be studied more and a very important advance came in 2004. Viola et al. [5] proposed a method with some rectangle features that concluded in face detection because the eyes are a dark rectangular region that is divided by a bright perpendicular object (nose). Nowadays the state of art face detectors like the one in Zhang et al. [6] are using deep convolutional networks to fulfill this task in a better way.

2. Materials and Methods

To train the face detector the Caltech 10,000 Web Faces [2] dataset was used. This dataset consists of 10,524 human faces in different settings and different resolutions. For this paper, only 6713 of these images were used. Additionally, each image is of size 36x36 to be consistent. Some of the train images are shown in Figure 1

Figure 1: Pictures of the Caltech 10,000 Web Faces dataset

2.1. HOG Features and linear SVM

First, the HOG features were calculated for each image using the Vlfeat libraries [4] implemented for Matlab. The HOG function would receive as parameters the window size (36) and a cell size (6) to calculate the mentioned features. To train a better model, not only the image was used but the mirror image also to double the HOG features. Having all the HOG features for the train images, a vector with all these features was created. An average of this HOG features of a face can be seen in Figure 2. Second, to train a linear SVM negative features needed to be calculated. Since the dataset had approximately 10,000 images, it was in our best

interest to calculate the same amount of negative features. For this, 254 random images were used and calculated HOG features in a random window of 36x36 inside this images.

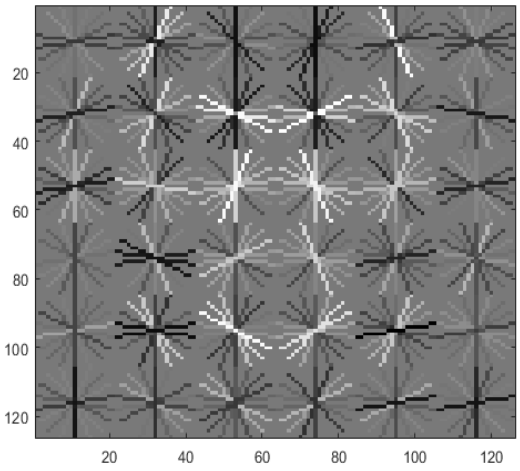

Figure 2: Average of the HOG features for faces in the Caltech dataset

A linear SVM was train to classify between face and not face categories. The positive feature vector ended of size 13426x1116 and the negative feature vector of size 10000x1116. The corresponding labels, 1 and -1, were paired with these feature vectors and the linear SVM was trained. The two parameters of interest, w and b, were calculated to later be able to classify a HOG feature vector as face or not.

2.2. Multi-scale HOG

The previously trained model only works for faces of size 36x36. To solve this problem a multi-scale approach was implemented using the code of user gary30404 [1] in github. The original image, with faces of different sizes, was re scaled to other scales such as 0.85x, 0.6x, and 0.1x. When the image was already re scaled the HOG features were calculated for the entire image. Since the HOG cell size was 6, each spatial dimension, x and y, were going to be divided by this factor. This re scaled HOG feature matrix had therefore the size of the re scaled image width and height divided by 6 and 31 dimensions because of the number of oriented gradients inside the vl_hog function. A sliding window was then used to go over this matrix and extract if it is a face or not using the w and b of the trained SVM. This SVM will return a confidence for the window depending if its a face or not. We then can set a threshold to control how many face detections are considered as true. Finally, all the windows that surpassed the threshold had their corresponding bounding boxes calculated and applied a non-maximum suppression to remove overlapping bounding boxes.

2.3. Evaluation Method

Since this is a detection problem, a Precision-Recall curve was implemented. This is a convenient evaluation method since it will take into account not only how many faces it is actually detecting but also the faces it is not detecting and the false positives where they algorithm says something is a face but it is not. Furthermore, the results reported are based on the average precision (AP) to get a better understanding of how the algorithm is working at its best rather than a various scenarios.

3. Results

3.1. Threshold

The threshold is an important parameter to play with since it is the one that regulates the amount of detections and furthermore the quality of this detections. This parameter was varied between 0.8 and 1 as shown in Table 1. It can be seen that a low threshold would give a worst result because it would let more fake faces to go through possibly increasing the recall but lowering the precision whereas a high threshold would be too selective and decrease the recall of the algorithm but increase the precision.

Table 1: AP variation when the hyperparameter threshold is variated

Threshold	0.8	0.9	1
AP	0.846	**0.851**	0.839

3.2. Average Precision

Since the extraction of negative features was random each time the algorithm is runned, the AP each time would vary. To get a more feasible AP the algorithm was runned 6 times and the average AP is 0.8322 ± 0.0162. The results for each run can be observed in Table 2. Finally, the PR curve of the best run can be seen in Figure 3

Table 2: AP in different runs of the algorithm

Run	1	2	3	4	5	6
AP	0.804	**0.851**	0.832	0.834	0.844	0.828

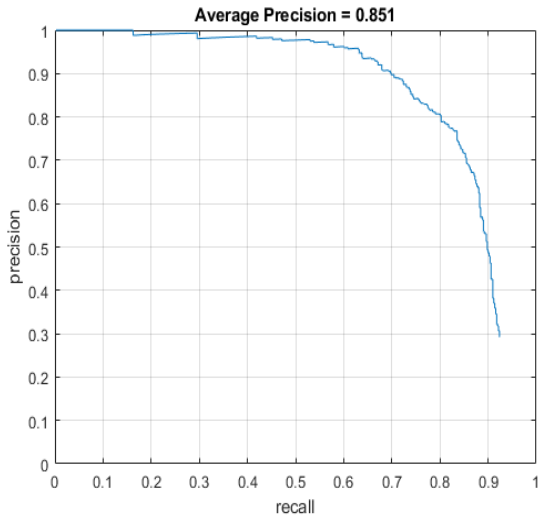

Figure 3: Best PR curve obtained with the algorithm

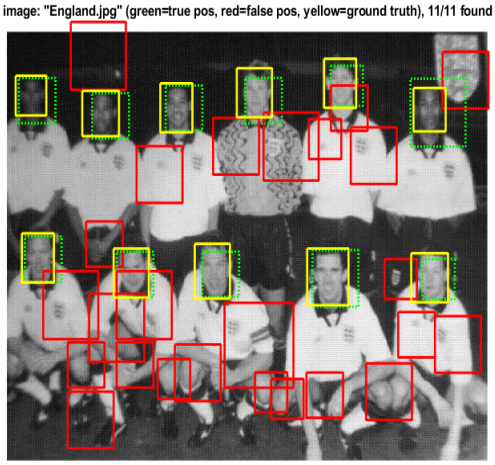

Figure 5: Example of a decent result for the face detections

Some decent results of the algorithm can be seen in Figures 4 and 5. On the other hand, a bad result can be seen in Figure 6. It can be seen in these figures that the algorithm fails to detect faces that are not completely facing forward causing the false negative count to rise. Additionally, when the image is blurry such as the one in Figure 6 it fails completely. From Figure 5 it can be concluded that the algorithm detects a face when there is a rounded object such as the knees of the players in Figures 4 and 5.

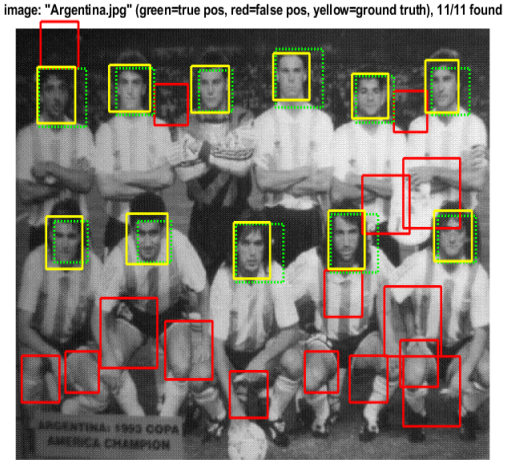

Figure 4: Example of a decent result for the face detections

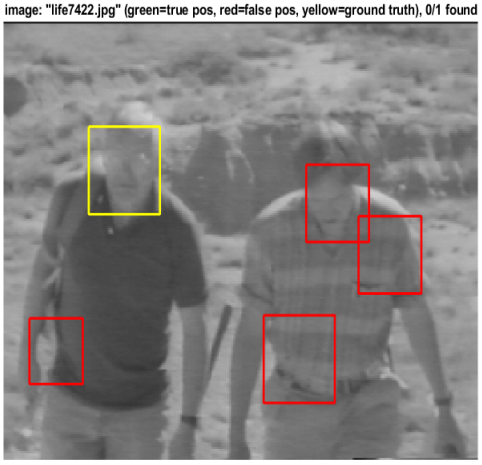

Figure 6: Example of a bad result for the face detections

3.3. Additional Test Images

Some additional test images were taken from the gallery of a person's cellphone. These pictures can be seen in Figure 7. The purpose of this is to see if the algorithm is robust enough to detect faces in more natural images like the ones found in a cellphone. The results can be observed in Figure 8.

Figure 7: Additional natural pictures

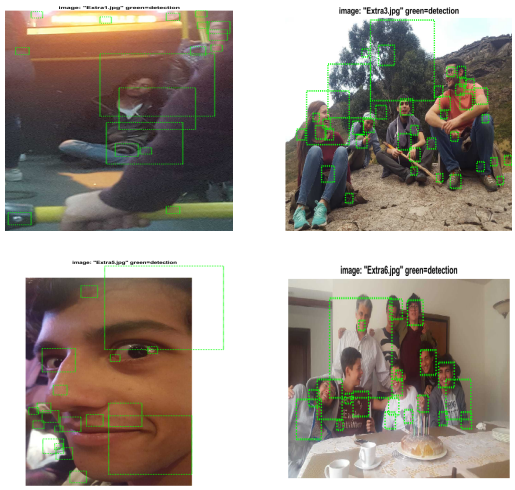

Figure 8: Face detections for the additional photos

Looking at Figure 8 it is clear that the algorithm is not very good for more natural images. For the two images on the left side it didn't even detect the face because of the blurriness and the scale and angle. For the two images of the right it detects almost every face but with a lot of false positives.

4. Conclusions

From what it was seen, the algorithm is very good for faces facing straight forward. When these faces begin to deviate from the forward position the algorithm begins to fail as it doesn't even detect them. Additionally, the algorithm returns a big amount of false positives due to round objects or patterns in the images as it confuses them with faces. It is clear that the recall is overall pretty good but the precision needs to be worked on. For future work it is ideal to look for a way to discriminate harder between round objects and faces to reduce the amount of false positives, which seems to be the principal one.

References

[1] gary30404. Face detection with a sliding window, may 2018.

[2] C. I. of Technology. Caltech 10,000 web faces. http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/, feb 2007.

[3] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591. IEEE, 1991.

[4] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[5] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[6] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

5. Bonus work

Waldo is in picture 13_Interview_Interview_On_Location_13_558 as shown in Figure 9.

Figure 9: Hidden Waldo