OBJECT CLASS RECOGNITION: FACES - ASSIGNMENT 4

Mirko Morello
920601
m.morello11@campus.unimib.it

Andrea Borghesi
916202
a.borghesi1@campus.unimib.it

January 3, 2025

1 INTRODUCTION

In this task, our objective was to find faces inside images using the Viola-Jones object detection framework.

2 TECHNOLOGIES USED: VIOLA JONES

Viola-Jones object detection framework [3] is a 2001 machine object detection algorithm That uses a combination of integral images, the AdaBoost algorithm, Cascade classifiers, and, in our case, the Local Binary Patterns to compute features.

3 APPROACH

3.1 Preprocessing

3.1.1 Conversion to grey-scale

The Viola-Jones algorithm has been created with one goal in mind, to find objects in grey-scale images. So to start we decided to convert all of our images in grey-scale to shift this computation from the training to the preprocessing phase.

3.1.2 Data Augmentation

The Viola-Jones algorithm is known to be working better when presented with a large amount of negative samples which should resemble as much as possible various backgrounds that could appear in our images. Also, due to the implementation provided by MATLAB, the number of negative samples used at each stage depends on the number of positive samples.
Another advantage for the algorithm is augmenting positive samples. This is because working with a larger number of them lets each predictor have a better generalization. Therefore we decided to augment both classes.

Negative samples Since we had 274 negative images in our dataset, we adopted a couple of data augmentation techniques. First, we flipped the image onto both axes and then we took each flipped image and rotated it 180 times with a step of 2 degrees. This led us to achieve a total of 360 pictures for each original image and a total of almost 100'000 negative samples.

Positive samples Since the positive samples supplied are around 6713, we won't be augmenting them as much as we did for the negative samples. Also, flipping faces upside-down and rotating them too much leads to extreme samples, that rarely occur in reality. Therefore we flipped each face on the y-axes and rotated them twice (along the z-axis) with an angle of ± 30 degrees since it is likely to find faces slightly rotated. With this last augmentation, we ended up with 26852 positive samples.

3.2 Empirical Hyperparameter Tuning

The main function used to adopt the Viola-Jones algorithm in MATLAB is `trainCascadeObjectDetector` [2]. Given a set of positive instances and a set of negative instances, it constructs a cascade of weak classifiers, using AdaBoost to train each stage (Figure 1).

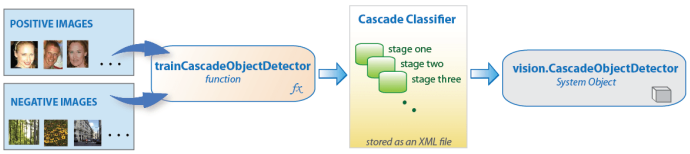

Figure 1: From [1]

The hyperparameters are the following:

- `FalseAlarmRate`: The range is from 0 to 1, this hyperparameter lets us choose the acceptable false

alarm rate at each stage which is the fraction of negative training samples incorrectly classified as positive samples. The lower the values the more the complexity is increased at each stage, leading to a more accurate model (fewer false detection) but higher training times. The default is 0.5.

- `NumCascadeStages`: Any integer value can be assigned. It's the number of training stages the VJ detector will have. The default is 20.
- `FeatureType`: Either HOG, LBP, or Haar. It's the Feature type used. The default is HOG.
- `NegativeSamplesFactor`: Any integer value can be assigned. It's used to calculate the number of negative samples used at each stage since the formula used to calculate them is the following: NegSamplesFactor $\cdot$ NposSamples. The default is 2.

Choosing a parameter for the FalseAlarmRate was a difficult task since we had to balance out the performance we wanted to achieve and the training times for our model. We wanted to test multiple configurations so a compromise had to be taken. We ended up choosing `FalseAlarmRate=0.1, NumCascadeStages=6, FeatureType=LBP, NegativeSampleFactor=3` since it led to significantly good results without sacrificing too much in terms of complexity.
The number of stages to train is strictly related to the false alarm rate. Our objective is always to have few false positives, and this can be achieved either by setting a low false alarm rate or by using a higher number of stages with a larger false alarm rate, and we decided to go for the former. Note that training many stages requires a large dataset (in the order of thousands of samples). The feature type choice was strongly led by the computation times, since on several occasions we were not able to get past stage 1 with HAAR features, our only options were HOG and LBP. As mentioned in [1] and through various experiments, we found out that LBP is the best choice for detecting faces and their small details.

4 Results

The combination of our choice of hyperparameters for the aforementioned functions leads to an accuracy of 75.96% as shown in Figure 2. The predictor shows good precision, as in Figure 3, 5, although it encounters some difficulties when looking for dark faces as in Figure 4. On the other hand, it has a significant false negative rate as shown in Figure 6. This might be explained by the low quality of the data, especially the negative ones, as they do not present challenging scenarios.

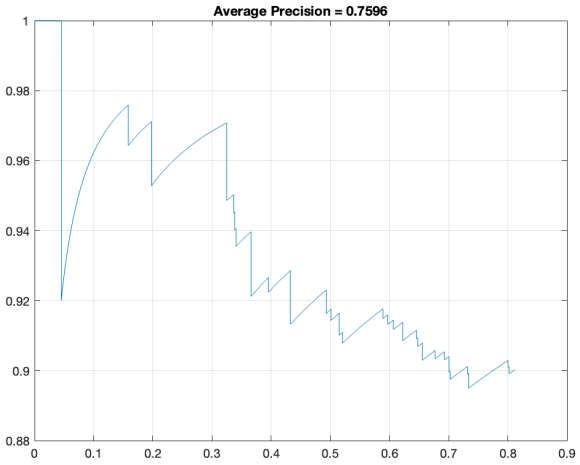

Figure 2: Precision-Recall curve of our predictor

Figure 3: Detection of a large number of faces

Figure 4: Missed dark faces

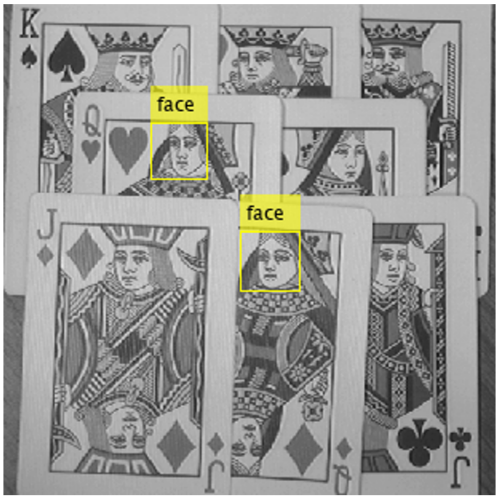

Figure 6: Difficulties in detecting fictional faces

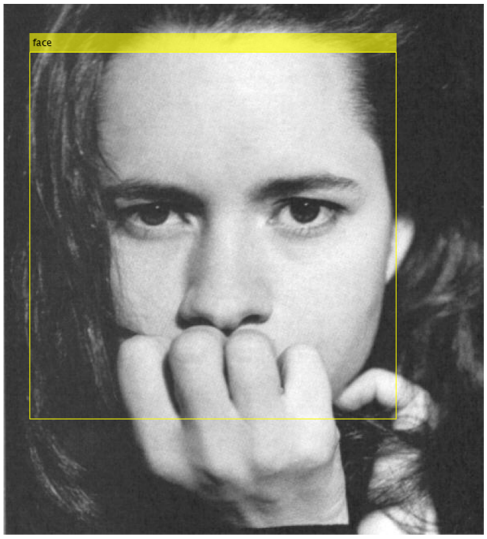

Figure 5: Partially occluded face

REFERENCES

[1] Mathworks. Get started with cascade object detector, 2024.

[2] Mathworks. traincascadeobjectdetector matlab documentation, 2024.

[3] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.