Compact convolutional neural network cascade for face detection

I.A. Kalinovskii, V.G. Spitsyn

Tomsk Polytechnic University

This paper presents a new solution to the frontal face detection problem based on a compact convolutional neural networks cascade. Test results on an FDDB dataset show that it is able to compete with state-of-the-art algorithms. This proposed detector is implemented using three technologies: SSE/AVX/AVX2 instruction sets for Intel CPUs, Nvidia CUDA, and OpenCL. The detection speed of our approach exceeds considerably all the existing CPU-based and GPU-based algorithms. Thanks to its high computational efficiency, our detector can process 4K Ultra HD video stream in real time (up to 27 fps) on mobile platforms while searching objects with a dimension of 60×60 pixels or higher. At the same time, its processing speed is almost independent of the background and the number of objects in a scene. This is achieved by asynchronous computation of stages in the cascade.

Keywords: face detection, cascade classifiers, convolutional neural networks, deep learning.

1. Introduction

The need to identify people in millions of photos uploaded daily to social services has led to significant progress in the solution of the problem of detecting faces. New methods are distinguished by invariance with respect to the pose and face expression. Moreover, they are also capable of operating in conditions of complex illumination and strong occlusion. However, many algorithms that demonstrate outstanding performance on a face detection benchmark have very high computational complexity. This circumstance prevents their use for video analysis.

The object of our interest is megapixel video analytics systems that require fast and accurate face detection algorithms. Such systems run on equipment whose computing power is often greatly limited due to the increasing demands for a compact form factor and a lower cost. Because of this, the increase of frame rate or frame resolution is often carried out at the expense of the performance of detection (such as large size objects search only, frames skipping, etc.). Moreover, the use of cameras capable of shooting video with a 4K Ultra HD resolution increases the amount of generated data by several times. In conditions when it is impossible to reduce the search area (for example, by motion analysis), even optimized detectors based on Viola-Jones method are unable to operate effectively at such a video stream resolution.

Despite the fact that the development of modern face detection methods is progressing towards an increased invariance with respect to the head position and the occlusion, we are considering here only a particular problem of frontal faces detection. Our goal is to achieve a high performance of detection at a low computational complexity of the detector, which is difficult to achieve when dealing with this task in the framework of the most general definition of the problem. At the same time, the frontal position of a person in relation to the camera is natural for many scenarios of using video analytics systems. That is why these detectors are so popular in practical applications.

In this paper, we present a frontal face detector based on a cascade of a convolutional neural network (CNN) [12] with a very small number of parameters. Due to the natural parallelism, a small number of cascade stages and a low-level optimization, it is capable of processing a real-time 4K Ultra HD video stream on mobile GPU when searching for faces of 60×60 pixels or higher, and, at the same time, it is 9 times faster than the detector based on Viola-Jones algorithm in the OpenCV implementation (http://opencv.org). Despite the compact CNN architecture, test results on a Face Detection Data Set and Benchmark (FDDB) dataset [8] show that the performance of our CNN cascade is comparable to that of some state-of-the-art frontal face detectors, and its speed surpasses that of any existing CPU and GPU algorithms.

2. Related work

It is not possible to build a simple and rapid detector with a high precision and response to all the possible face image variations because of the big interclass variance, the variety of ambient light conditions, and the complex structure of the background. The standard approach to solving this problem is to use different models for each pose of the head [19, 32, 35]. It has been shown recently that, thanks to their strong generalization capability, deep CNNs can study the whole variety of two-dimensional projections of a face within the limits of a single model [3, 16]. However, the fact that the proposed CNN architectures contain several millions of parameters makes them unsuitable for use in low-power computing devices. Methods based on deformable part models [19, 33], template comparison based models [15], and 3D face models [1] are also unable to work with HD video streams in real time, even to solve the problem of frontal faces search only. Detectors that use manually designed features to describe objects and cascades of boosting classifiers to detect them remain the best solution in terms of processing speed [26].

Many different descriptors were proposed to describe facial features. The most famous ones are rectangular Haar-like features [30] which have shown to be effective for building frontal face detectors and to have a high extraction rate achieved by means of using the integral image. Textural MCT [4, 27] and LBP [29] features, which code pixel intensity in the local domains, have invariance with respect to monotonic light change. LBP in combination with HOG features [24] demonstrated good generalizing properties, and they are able to process complex non-facial images better than the Haar-like features. B. Jun et al. [10] proposed LGP and BHOG features built on the principles of LBP. LGPs are resistant to local changes of light along the borders of the objects, and BHOGs are resistant to local pose changes.

Multidimensional SURF descriptors [17] in combination with the logistic regression make it possible to prepare cascades containing only a few hundreds of weak classifiers. Because of this, SURF cascades exceed the speed of Haar cascades which typically consist of thousands of weak classifiers. A simple comparison of pixel intensities can also be used for faces detection [2, 18]. The detector proposed in [18] has a high execution speed since it does not require any additional processing, including the construction of the image pyramid.

Usually, boosting cascades are trained using grayscale images for the solution of face detection problem. M. Mathias et al. [19] and B. Yang et al. [35] used combinations of different channels (grayscale, RGB, HSV, HOG descriptors and other) for the training of classifiers. The taking into account of both color and geometric information allowed improving the performance of face detection on a complex background.

Recently, H. Li et al. [16] have built a CNN cascade whose speed is the highest among the multi-view face detectors. Similarly to Viola-Jones algorithm, a simple CNN was used for coarse image scanning, while more complex models estimated carefully each selected region. However, despite the significant reduction of the computational complexity (in comparison with the single CNN model [3]), this detector is still incapable of processing an HD video stream even on a powerful GPU.

It will be shown further that, with a view to solving the frontal faces detection problem, our CNN cascade may surpass boosting cascades not only in performance but also in speed. The CNN densely extracts high-level features directly from raw data, without requiring any preliminary processing apart from building the image pyramid. Besides, the CNN calculation algorithm can be easily vectorized using SIMD instructions of CPU, and it can be adapted perfectly to the massively parallel architecture of GPU.

3. Compact convolutional neural network cascade

Similarly to [16], we use a CNN to build a cascade detector of frontal faces. This work is based on the following key ideas:

1) A small number of cascade stages. Our CNN cascade has only 3 stages. For example, the shortest boosting cascade consists of 4 stages and uses MCT descriptors for the extraction of facial features [4].

2) The compact design of the CNN architectures. The total number of feature maps in all the CNNs is 355 (in [16] it amounts to 1,949); however, a sample with a smaller variation of face images was used for training the model.

3) Asynchronous execution of the stages. A particular design of the detector makes it possible to execute the second and the third stages of the cascade in parallel with the first one on different processing units. Due to the fact that 99.99% of sliding window positions are rejected already at the first operation stage, in this mode the detector is capable of processing video frames in constant time, regardless of the content of the image.

4) Optimization. During the implementation of the detector, three technologies were used: SIMD extension of CPU, CUDA, and OpenCL. The SIMD (CPU) and CUDA implementations of the CNN were optimized for each of the network architectures used. The giving up on the traditional approach of the CNN calculation through the organization of a stack of layers, combined with optimization at the assembler level, made it possible to achieve the code execution performance which was close to the peak performance of hardware.

3.1 CNN structures

The CNN architectures composing our cascade are shown in Figure 1. Each CNN solves the problem of a background/face binary classification and contains 797 (CNN_1), 1,819 (CNN_2) and 2,923 (CNN_3) parameters. Similarly to the Convolutional Face Finder [5] architecture, the lack of fully-connected layer gives a 50% increase in the speed of the forward propagation procedure. The convolution stride is 1 pixel, and the pooling stride is 2 pixels. Rational approximation of a hyperbolic tangent is used as an activation function:

$$\begin{aligned} f(x) &= 1.7159 \cdot \tanh\left(\frac{2}{3}x\right), \\ \tanh(y) &\approx \operatorname{sgn}(y)\left(1 - \frac{1}{1+|y|+y^2+1.41645 \cdot y^4}\right). \end{aligned} \tag{1}$$

The relative error of the following approximation does not exceed 1.8% on the entire number axis, and only 11 instructions are required to calculate it. Popular ReLU functions turned out to be less efficient in our experiments. It should be noted that CNN_1 contains the smallest number of filters in comparison with previously proposed network architectures for face detection [3, 5, 16, 23].

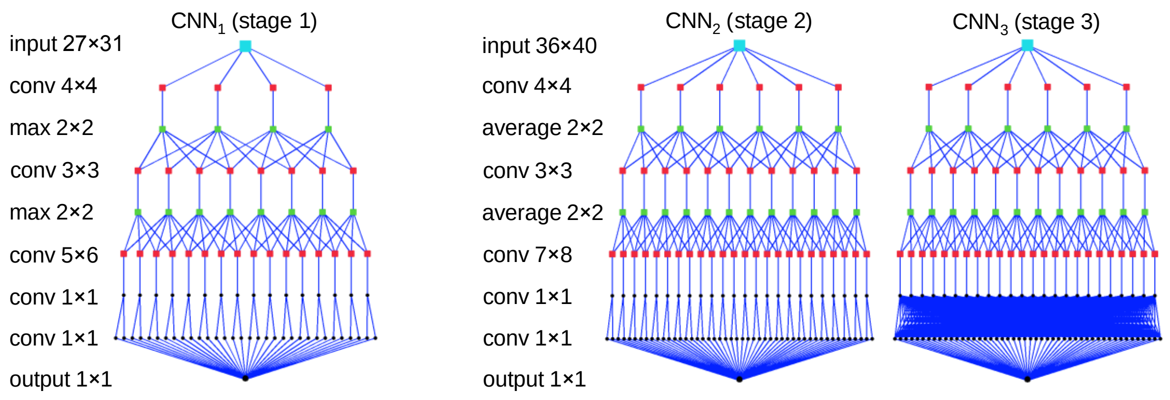

Fig. 1. CNN structures

3.2 Training process

For the CNN training, aligned face images were taken from YouTube Faces Database [31]. This dataset contains face tracks of 1,595 people cut out from 3,425 videos (Fig. 2). Background images were selected from random YouTube videos in several stages during the preparation of models. Face areas (such as eyes, nose, etc.) were also added to the negatives. The total volume of the training set consisted of slightly more than one million grayscale images (433 thousand of positive examples and 585 thousand of negative ones).

The experiments were carried out with simple models which have a small number of parameters. We aimed to find the minimal configuration of a CNN which would be able to classify the test set with an error below 0.5%. For the CNN training, a Levenberg-Marquardt algorithm was used [36].

Fig. 2. Example images from the training set

3.3 Detector design

The detector design is shown in Figure 3. The first CNN densely scans, in series, each image of the pyramid. The responses in the output network layer correspond to the positions of the scanning window with a size of 27×31 pixels during its uniform motion with a 4-pixel step. The coordinates of the windows where the CNN response exceeded the predetermined threshold T_1 are transmitted to the selective unit for a further analysis of these regions of the image. Even when $T_1 = 0$, more than 99.99% of the total number of positions of the window at all pyramid levels are rejected at this stage. For comparison, the first-stage of a Haar cascade of Viola-Jones [30] is able to reject only 50% of negative samples, an MCT cascade [4] – 99%, a SURF cascade [17] – 95%, and a CNN cascade [16] – 92%.

The pre-processing and classification of an image region are carried out in the selective unit, after which the final decision about their belonging to a faces class is made. At the step of pre-processing, the analyzed region is read from the original grayscale image together with a certain neighborhood and is scaled to 51×55 pixels. Then, the equalization of its histogram and mirror reflection with respect to the vertical axis are carried out. Illumination alignment enhances the response of the CNN on shaded faces and effectively suppresses false detections. The use of mirror reflection also reduces the response to complex non-facial images.

During the second step, the second and third stages of the cascade perform region classification. The output of each CNN is a response map of a 5×5 size. Additional classification in the region neighborhood is necessary to prevent the loss of response due to an incorrect positioning of the face in the scanning window. The decision about the type of the region is made on the basis of the number of responses K_{nn} of each classifier that exceeds the predetermined threshold T_2:

$$\delta = \left(K_{nn}^{CNN2} \geq T_{nn} \wedge K_{nn}^{CNN3} > 0\right) \vee \left(K_{nn}^{CNN2} > 0 \wedge K_{nn}^{CNN3} \geq T_{nn}\right),$$
$$\text{decision rule} = \begin{cases} \delta = 1, & face \\ \delta = 0, & no\ \ face \end{cases}. \tag{2}$$

The discrete parameter T_{nn} makes it possible to control the number of detector false alarms more robustly as compared to thresholds T_1 and T_2. If the response of the CNN$_2$ does not agree with the decision rule, further analysis of the region stops.

The last stage of the pipeline detector is the Non-Maximum Suppression (NMS) algorithm which aggregates the detected regions to form the resulting areas of faces localization.

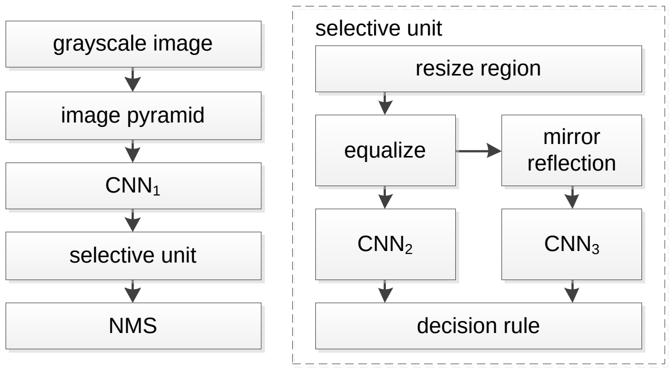

Fig. 3. Detector design

3.4 Implementation

The detector has been implemented using three technologies: SIMD extension of x86 processor family (for each of three instructions sets – SSE, AVX, AVX2 – supported by microarchitectures of Intel Sandy/Ivy Bridge and Haswell/Broadwell processors), Nvidia CUDA, and OpenCL. The calculations are carried out using single precision, and the precision-recall characteristic is identical for all implementations.

Thus, the implementations of all stages of the pipeline detector (except for NMS) are presented in several versions of a manually vectorized code. We used vector intrinsic functions (which are directly translated into the appropriate assembly instructions by the compiler), took into account the limited number of logical registers, and minimized the number of queries to the memory. At the same time, the SSE code can be ported to the ARM platform since all the SSE instructions used have analogs in the NEON set of instructions. The Scalar C++ code and OpenCL allow running the detector on most devices though at a lower execution speed.

Figure 4 shows the comparison of our convolution implementation to its implementation in the Intel IPP, Nvidia NPP and cuDNN, and ArrayFire libraries. Time measurements were made for the first CNN$_1$ convolution layer calculation (Fig. 1) on an image with a 4K Ultra HD resolution and were averaged over 10^3 launches. When implementing the convolution for GPU, we used the method proposed by F.N. Iandola et al. [7]. Due to the fine code optimization for each CNN architecture, the speed of layers calculation is higher than when using more universal functions from the respective libraries.

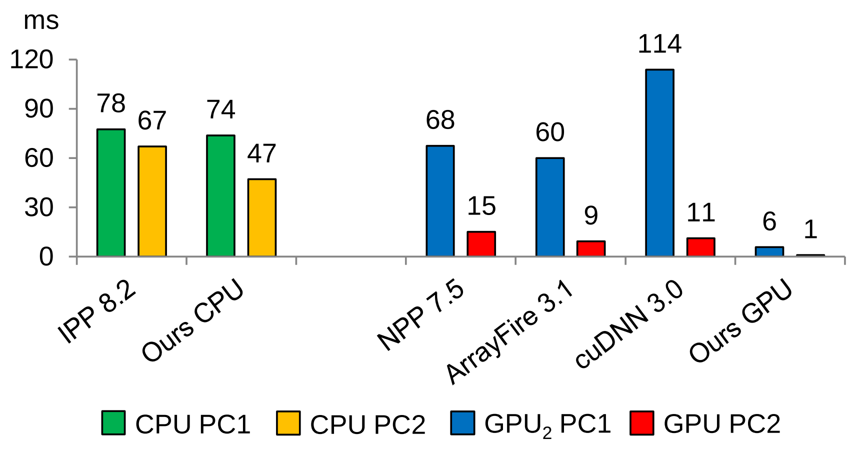

Fig. 4. Comparison of the calculation speed of the first convolution layer for CNN$_1$ for an image with a 4K Ultra HD resolution (equipment specifications are listed in section 4)

In order to provide continuous GPU load, the scanning of several levels of the image pyramid (3 by default) is executed simultaneously in different streams (concurrent kernels). However, a kernel computation start is synchronized between streams as they use common pointers to texture memory.

Also, we implemented several solutions to improve the detection speed.

3.4.1 Asynchronous mode

Typically, face images occupy only a small area of the image. The main advantage of cascade detectors is the ability to rapidly reject the majority of background regions as early as during the first stages. However, complex non-facial images may be rejected only during the later stages. If there is a large number of stages, the speed of detector becomes strongly dependent on the structure of the background and the number of the objects in the scene. At the same time, the non-uniform distribution of the processing load over the image area negatively affects the runtime efficiency of GPU implementations.

In order to solve this problem, we proposed the asynchronous mode of cascade execution. In this mode, the first stage which is run on GPU scans each level of the image pyramid sequentially. Coordinates of detected regions are transmitted to CPU which operates as a selective unit. CNN_1 proceeds to the scanning of the next level of the pyramid regardless of whether the analysis of the detected regions on CPU has been finished or not.

Due to the low-level optimization, the selective unit carries out a single region analysis in 0.1 ms on a single core processor (CPU PC2, for AVX code). Since only 0.01% (on average) of all window positions passes through CNN_1, by the time the scanning of the last level of the image pyramid is complete, the majority of the detected regions will have already been processed on CPU. A similar situation is possible under the condition of asynchronous execution of the cascade on different CPU cores. Thus, the CNN cascade is able to provide a constant frame processing time dependent only on frame resolution and the productivity of the first stage execution.

3.4.2 Hybrid mode

The effectiveness of GPU in image processing is significantly higher than that of CPU. However, if small images (less than 0.01 megapixels) are considered, the calculation time is limited by delays in the running of kernels. In order to improve the search speed of large size faces, we made it possible to run the first stage of the cascade on CPU and GPU simultaneously. In the hybrid mode, CPU begins scanning the image pyramid from the upper level, while GPU processes a high-resolution image on the lower level. Also, it is possible to use the asynchronous mode. A similar technique was used in [21].

3.4.3 Patchwork mode

In order to reduce the kernel launch delays to a minimum, it is possible to use a patchwork technique [6]. We applied the Floor-ceiling no rotation (FCNR) algorithm [20] for the purposes of dense image pyramid packing into a semi-infinite strip of a predetermined width. Thus, the scanning of all image scales at the same time is carried out in a single run of CNN_1. However, it is not possible to use the asynchronous mode. In this case, the second stage of the cascade is also performed on GPU by means of scanning all the detected regions in a single pass. Usually the asynchronous mode is more effective than the patchwork mode, but the latter improves the low-resolution image processing performance.

4. Experiments

In this section, the results of the proposed detector testing on two public face detection benchmarks – FDDB [8] and AFW [37] – are shown. Since the detector was designed for a frontal face search only, it is obvious that it cannot surpass the multi-view face detectors with these complex datasets. However, it is comparable to the state-of-the-art frontal face detectors on the FDDB benchmark.

In addition, we tested several face detectors whose source code or demo versions are in open access. In order to compare the performance and speed of algorithms for video processing tasks, tests were carried out on the annotated video.

This section also provides execution speed of all detector implementations for different video stream resolutions and asynchronous mode demonstration. The test results show that the CNN cascade provides a very high data processing speed on both GPU and CPU, outperforming all previously proposed algorithms.

Specification of the equipment used:

– PC1 (laptop): Intel Core i7-3610QM CPU (2.3 GHz, Turbo Boost disabled), Intel HD Graphics 4000 GPU_1 (GT2, 16 core, 1,100 MHz) and Nvidia GeForce GT 640M GPU_2 (GK107, 384 core, 709 MHz);

– PC2 (desktop): Intel Core i5-3470 CPU (3.2 GHz, 3.6 GHz with Turbo Boost) and Nvidia GeForce GTX 960 GPU (GM206, 1,024 core, 1,228 MHz).

4.1 Face Detection Data Set and Benchmark

The FDDB [8] benchmark consists of 2,845 pictures (no more than 0.25 megapixels), and it has elliptical shape annotations for 5,171 faces. The authors provide a standardized algorithm for the automatic ROC curves construction based on the detector operation results. The algorithm calculates two types of evaluations: the discrete score and continuous score. ROC curve for the discrete scores reflects the dependence of the detected faces fraction on the number of false alarms by varying the threshold of the decision rule. The detection is considered to be positive if the Intersection-over-Union (IoU) ratio of detection and annotation areas exceeds 0.5. Only one detection can be matched with an annotation. Continuous score reflects the quality of face localization, i.e. the average IoU ratio.

The result of the offered detector evaluation is shown in Figure 5. The following search settings were used: the minimum object size (*minSize*) was 15×15 pixels, the scale factor for the image pyramid construction (*scaleFactor*) was 1.05, $T_1 = 0$, $T_2 = 0$, and $T_{nn} = 1$. Since the detector localizes rectangular areas, in some cases this leads to errors when they are matched with elliptical shape annotations. For a correct evaluation, we manually modified 105 received bounding boxes (Fig. 6) so that their IoU ratio would exceed a predetermined threshold.

Based on the adjusted evaluation, the performance of our detector exceeds the performance of SURF [17], PEP-Adapt [14], and Pico [18] frontal detectors, approaching the multi-view SURF [17] detector performance. Table 1 shows the average number of the sliding window positions selected by each CNN cascade stage on the images from the FDDB collection. Even when $T_1 = 0$, more than 99.99% of window positions were rejected already at the first stage. This is substantially better than the result obtained in [16].

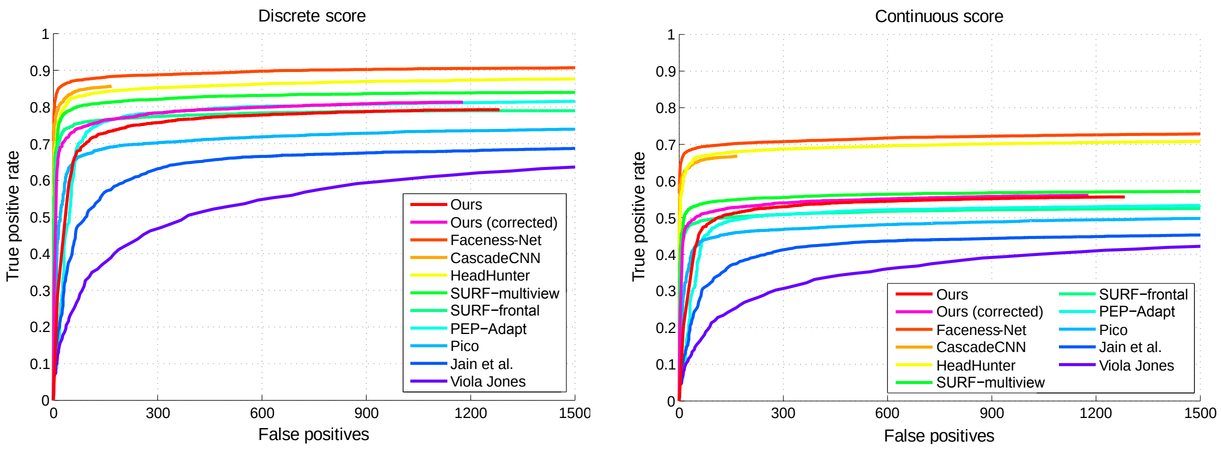

Fig. 5. Discrete score ROC curves and Continuous score ROC curves for different methods on the FDDB dataset, including: our CNN cascade, Faceness-Net [34], CascadeCNN [16], HeadHunter [19], SURF [17], PEP-Adapt [14], Pico [18], Jain et al. [9], and Viola Jones (OpenCV)

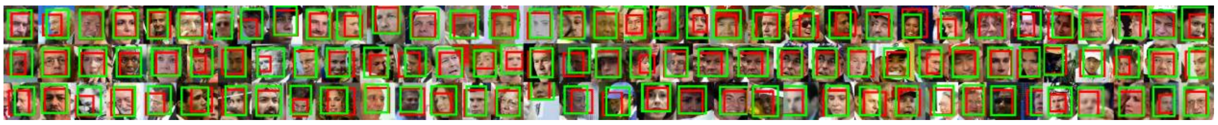

Fig. 6. Manually corrected detections received by the compact CNN cascade on the FDDB benchmark

Table 1. Statistics of the CNN cascade operation on the FDDB. It shows the number of detections averaged over all images made by each cascade stage and the percentage of rejected windows

stages	**number of windows**	**rejected windows, %**
sliding window	2 724 768.2	
stage 1	132.7	99.995
stage 2	57.0	57.019
stage 3	43.3	24.036
NMS	1.9	

4.2 Annotated Faces in the Wild

The Annotated Faces in the Wild (AFW) [37] benchmark consists of 205 large-scale (0.5-5 megapixels) images and contains rectangular annotations for 468 faces.

This benchmark was developed relatively recently and is mainly used for multi-view detectors evaluation. Because of this, we additionally tested 7 frontal face detectors, including: two Haar cascades (OpenCV-default, OpenCV-alt) and an LBP cascade (OpenCV-lbp) from the OpenCV library; a Haar cascade [25] (OpenCV-Pham) and an LBP cascade [11] (OpenCV-Köstinger) trained by the OpenCV object detection framework; an SURF cascade [17] (SURF-frontal, not the same model as for the FDDB); and a cascade of decision trees using the pixels intensity comparison [18] (Pico).

For each detector, precision and recall scores were calculated for different values of *minNeighbors* = {1, 2, 3} (a parameter specifying how many neighbors each candidate rectangle should have to retain it), and the mean value of F_1 measure. *MinNeighbors* parameter is used in the OpenCV and SURF detectors and is equivalent to T_{nn} (2) for our detector. In Pico implementation, the level of false alarms is regulated by the threshold of the decision rule which in this test was assumed to be equal to *minNeighbors* + 2.

Comparison of the detectors was carried out with the following search settings: *minSize* = 80×80 pixels and *scaleFactor* = 1.1. The SURF and OpenCV-Köstinger detectors localize a smaller face area in comparison with other detectors, which is why their *minSize* value was reduced by 25%. The configuration of the detectors was as follows:

– OpenCV: version 3.0.0, *useOptimized* = 1;
– SURF: *modelType* = 0, *fast* = 1, *step* = 1;
– Pico: *strideFactor* = 0.1.

A standard IoU ratio with a threshold of 0.5 was used to evaluate the detections. Moreover, 44 bounding boxes were additionally generated for each annotation by scaling it with a factor from 0.9 to 1.2 (Fig. 7). Such a multiple check of detections made it possible to take into account differences in the size of areas localized by detectors and to eliminate errors appearing during matching.

The test results are presented in Figure 8. The proposed detector (F_1 = 0.75) holds the second position in terms of F_1 measure value. It is second to the OpenCV-Köstinger detector (F_1 = 0.78) due to producing more false alarms. However, on this benchmark, modern multi-view detectors show significantly better results, but they cannot work in real-time mode. For example, the Faceness-Net [34] detector requires 50 ms for image analysis with VGA resolution on Nvidia Titan Black GPU, which is about 200 times longer than the working time of our CNN cascade.

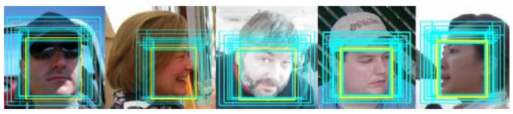

Fig. 7. AFW annotations modification

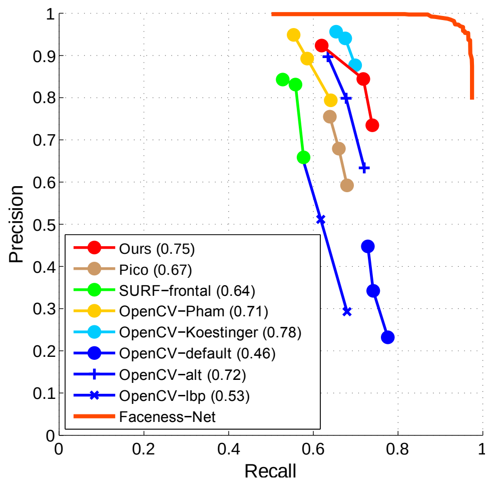

Fig. 8. Test results of the frontal face detectors on the AFW benchmark. The numbers in parentheses are mean values of the F_1-measure for detectors

4.3 Video data

During the last test, the performance of detectors was evaluated on a high-resolution video sequence. We annotated the first 12,000 frames of the 12th episode from the 5th season of «How I Met Your Mother» TV series in HD resolution. This video segment comprises 10 different scenes, and the number of faces in a frame varies from 1 to 88 (39,976 faces in total).

For testing, we used the same detector parameters as for the AFW benchmark, except the *minSize* parameter which was equal to 40×40 pixels. Also, in this test the parameter T_2 for our detector was equal to 1.7. All detectors were running on CPU PC1 in a single-threaded mode.

The proposed detector also ranks second in terms of F_1 measure value (F_1 = 0.61, 10.6 fps), yielding to OpenCV-Köstinger (F_1 = 0.65, 1.7 fps), but it is superior to all the detectors in terms of speed (Fig. 10). Thus, the CNN cascade provides the best performance/speed ratio in comparison with boosting cascades.

A greater level of recall, with a significantly reduced precision, can be achieved by using a weaker decision rule in the selective unit:

$$\delta = K_{nn}^{CNN\,2} \geq T_{nn} \vee K_{nn}^{CNN\,3} \geq T_{nn} \tag{3}$$

A higher level of precision can be achieved by performing an additional validation of detections with a Haar cascade OpenCV-alt. A Haar cascade was used only after the NMS algorithm, and the detections were pre-scaled to the size of 80×80 pixels. This made it possible to keep the high speed of video processing.

The results of the evaluations of detectors with the use of the AFW and video data allow drawing the following conclusions. Frontal detectors whose processing speed (on CPU) is comparable to that of a compact CNN cascade achieve a significantly lower recall and precision of detection. Detectors whose recall is comparable to that of a CNN cascade have a processing speed which is several times lower. Thus, we have shown that CNNs (even classic ones) allow building a frontal faces detector with better characteristics than it can be achieved using the modern modifications of Viola-Jones algorithm.

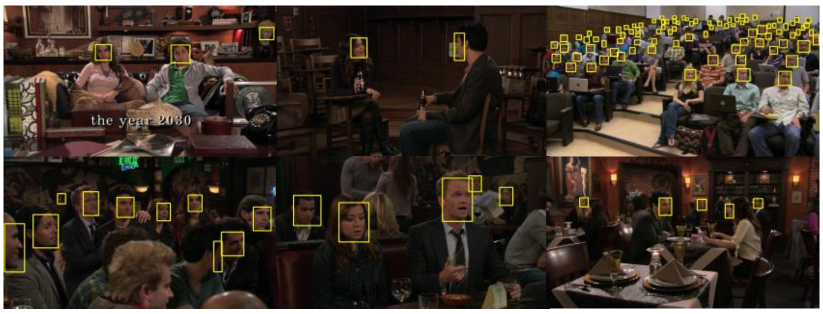

Fig. 9. Examples of video frame annotations

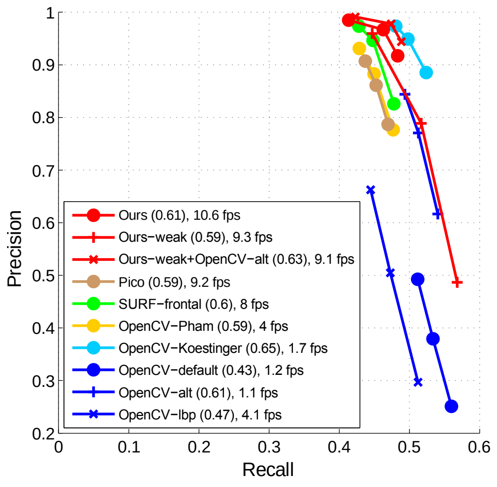

Fig. 10. Test results of the frontal face detectors on HD video. The numbers in parentheses are mean values of the F_1-measure for detectors

4.4 Runtime efficiency

The runtime efficiency was the key priority at all stages of the proposed detector development. Our CNN cascade turned out to be from 2.6 to 10 times faster than Haar and LBP cascades from the speed-optimized OpenCV 3.0 library when processing HD video on CPU PC1 (Fig. 10). Meanwhile, the computing complexity of our CNN cascade is comparable to that of Haar-like cascades. In case with the search parameters from Section 4.3 C++ code (without auto-vectorization), it reaches 1.2 fps, whereas the Haar-like classifier from the OpenCV-alt reaches only 1.1 fps.

Among other open source projects (for example, http://libccv.org and http://dlib.net), we do not know a CPU-based algorithm faster than Pico [18] (integer, it does not require the construction of an image pyramid) and a SURF cascade [17] (5 stages, SIMD optimization is used). However, despite the higher computational complexity, the speed of the CNN cascade execution exceeds the speed of these detectors due to the code vectorization and the efficient use of the processor's cache memory.

Figure 11 shows the dependence of the speed of various detectors on the content of the scene for the first 4,000 frames of the annotated video. When using the asynchronous mode, the CNN cascade provides nearly constant processing time on both CPU and GPU even when there is a significant increase in the number of faces in the scene (88 faces in frames from 1,771 to 1,834). This property is important for video analytics systems as it makes it possible to predict more precisely the speed of the system in different use cases.

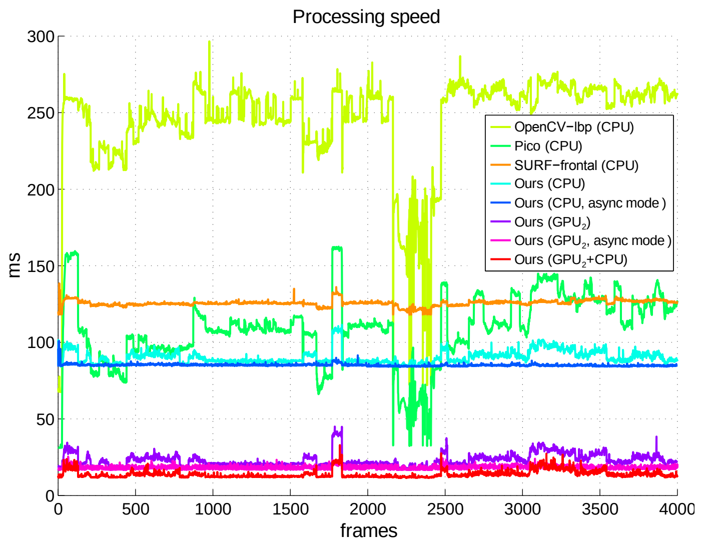

Fig. 11. Dependence of detector operation speed on the scene content during HD video processing (search settings are the same as in section 4.3, single-threaded execution on PC1)

For algorithms that have been tested on the FDDB benchmark, the authors reported the following data on the detection speed of frontal faces for images with VGA resolution: NPD[1] [13] – 178 fps (40×40, 1.2, i5-2400 CPU, 4 threads); ACF [35] – 95 fps (i7 CPU, 4 threads); SURF [17] – 91 fps (40×40, 1.2, i5-2400 CPU, 4 threads); Joint Cascade [2] – 35 fps (80×80, 2.93 GHz CPU, 1 thread); Pico [18] – 417 fps (100×100, i7-2600 CPU, 1 thread); Fast DMP [33] – 42 fps (Intel X5650 CPU, 6 threads). The speed of other detectors does not exceed 10 fps. Only methods based on the CNN [3, 16, 34] and HeadHunter [19] support GPU. For typical search settings (*minSize* = 40×40 pixels, *scaleFactor* = 1.2), a CNN cascade guarantees 85 fps for a single-threaded and 148 fps for a multi-threaded execution on CPU PC2, 171 fps on GPU$_2$ PC1, and 313 fps on GPU PC2 (used hybrid mode).

The CNN cascade proposed by H. Li et al. [16] processes a VGA image in 110 ms on CPU (80×80, 1.414, Intel Xeon E5-2620, 1 thread) and in 10 ms on GPU (Nvidia GeForce GTX TITAN Black, 2,880 CUDA core). With similar search settings, our CNN cascade finishes its operation in 2.5 ms on a single core CPU PC1 and 2 ms on GPU$_2$ PC1.

[1] just 39 fps when processing our annotated video (scaled to the VGA resolution) for multi-threaded execution on CPU PC1.

Many investigations are focused on the optimization of Viola-Jones algorithm for GPU with the purpose of increasing the processing speed of a Full HD video stream: D. Oro et al. [22] – 35 fps for Haar cascade (24×24, 1.1, Nvidia GTX470 GPU); S.C. Tek and M. Gokmen [28] – 35 fps for MCT cascade (24×24, 1.15, Nvidia GTX580 GPU); C. Oh et al. [21] – 29 fps for LBP cascade (30×30, 1.2, Nvidia Tegra K1 GPU + Cortex-A15 CPU). Using the same search parameters, the CNN cascade speed reaches up to 75, 98, and 108 fps respectively when it runs in the hybrid mode on PC2. Thus, the proposed detector provides a higher runtime efficiency on GPU in comparison with Viola-Jones cascade detectors.

Figure 12 shows the diagram of an average video frame rate in 4 standard resolutions which can be reached with different detector implementations optimized for CPU and GPU. Testing was conducted on first 4,000 frames of the annotated video that was scaled to the size of the appropriate resolution. The results indicate that the CNN cascade copes even with the extreme task of real-time processing of the video stream with a 4K Ultra HD resolution. For example, the speed of an LBP cascade face detector from the OpenCV library reaches 3 fps only on GPU$_2$ PC1 with the same search settings.

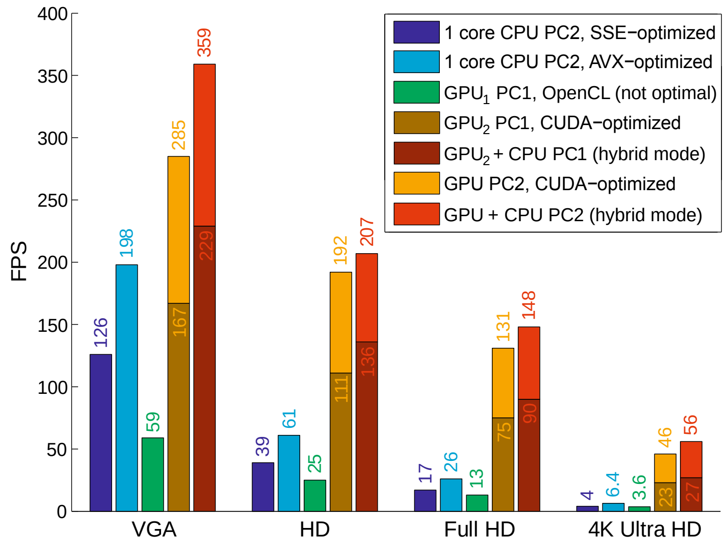

Fig. 12. Processing speed of different frontal face detector implementations based on the compact CNN cascade (*minSize* = 60×60 pixels, *scaleFactor* = 1.2, *minNeighbors* = 2)

5. Conclusion

In this paper, we proposed a cascade of compact CNNs for a rapid detection of frontal faces in an HD video stream. The first stage of the cascade is capable of rejecting 99.99% of windows containing background. In combination with the asynchronous execution mode, this factor substantially reduces the dependence of the detector speed on image content.

The CNN cascade performance is comparable with that of the best frontal face detectors on the FDDB benchmark, but it surpasses them in speed on both CPU and GPU. Thus, the proposed algorithm establishes a new level of performance/speed ratio for the frontal face detection problem and makes it possible to reach acceptable processing speed even on low-power computing devices.

References

1. Barbu A., Lay N., Gramajo G. Face Detection with a 3D Model. arXiv preprint. 2014. URL: http://arxiv.org/pdf/1404.3596.pdf (accessed: 14.02.2016).
2. Chen D., Ren S., Wei Y., et al. Joint cascade face detection and alignment // In European Conference on Computer Vision. 2014. P. 109-122.
3. Farfade S.S., Saberian M., Li L.-J. Multi-view face detection using deep convolutional neural networks. arXiv preprint. 2015. URL: http://arxiv.org/pdf/1502.02766.pdf (accessed: 14.02.2016).

4. Froba B., Ernst A. Face detection with the modified census transform // In IEEE International Conference on Automatic Face and Gesture Recognition. 2004. P. 91-96.
5. Garcia C., Delakis M. Convolutional face finder: A neural architecture for fast and robust face detection // In IEEE Transactions on Pattern Analysis and Machine Intelligence. 2004. P. 1408-1423.
6. Iandola F.N., Moskewicz M.W., Karayev S., et al. Densenet: Implementing efficient convnet descriptor pyramids. arXiv preprint. 2014. URL: http://arxiv.org/pdf/1404.1869.pdf (accessed: 14.02.2016).
7. Iandola F.N., Sheffield D., Anderson M.J., et al. Communication-minimizing 2D convolution in GPU registers // In IEEE International Conference on Image Processing. 2013. P. 2116-2120.
8. Jain V., Learned-Miller E. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UMCS-2010-009. University of Massachusetts. 2010. URL: http://vis-www.cs.umass.edu/fddb/fddb.pdf (accessed: 14.02.2016).
9. Jain V., Learned-Miller E. Online domain adaptation of a pre-trained cascade of classifiers // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2011. P. 577-584.
10. Jun B., Choi I., Kim D. Local transform features and hybridization for accurate face and human detection // In Pattern Analysis and Machine Intelligence. 2013. P. 1423-1436.
11. Köstinger M. Efficient metric learning for real-world face recognition // Graz University of Technology, PhD thesis. 2013.
12. LeCun Y., Bengio Y. Convolutional networks for images, speech, and time series // The handbook of brain theory and neural networks. 1995. P. 255-258.
13. Liao S., Jain A.K., Li S.Z. A fast and accurate unconstrained face detector // In IEEE Transactions on Pattern Analysis and Machine Intelligence. 2015. P. 211-223.
14. Li H., Hua G., Lin Z., et al. Probabilistic elastic part model for unsupervised face detector adaptation // In Proceedings IEEE International Conference on Computer Vision Workshops. 2013. P. 793-800.
15. Li H., Lin Z., Brandt J., et al. Efficient boosted exemplar-based face detection // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014. P. 1843-1850.
16. Li H., Lin Z., Shen X., et al. A convolutional neural network cascade for face detection // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. P. 5325-5334.
17. Li J., Zhang Y. Learning SURF cascade for fast and accurate object detection // In Computer Vision and Pattern Recognition. 2013. P. 3468-3475.
18. Markuš N., Frljak M., Pandžić I.S., et al. Object detection with pixel intensity comparisons organized in decision trees. arXiv preprint. 2013. URL: http://arxiv.org/pdf/1305.4537.pdf (accessed: 14.02.2016).
19. Mathias M., Benenson R., Pedersoli M., et al. Face detection without bells and whistles // In Proceedings of ECCV. 2014. P. 720-735.
20. Ntene N., Van Vuuren J.H. A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem // Discrete Optimization. 2009. P. 174-188.
21. Oh C., Yi S., Yi Y. Real-time face detection in Full HD images exploiting both embedded CPU and GPU // In IEEE International Conference on Multimedia and Expo. 2015. P. 1-6.
22. Oro D., Fernández C., Saeta J.R., et al. Real-time GPU-based Face Detection in HD Video Sequences // In IEEE International Conference on Computer Vision Workshops. 2011. P. 530-537.

23. Osadchy M., LeCun Y., Miller M. Synergistic face detection and pose estimation with energy-based models // Journal of Machine Learning Research. 2007. P. 196-206.
24. Paisitkriangkrai S., Shen C., Zhang J. Face detection with effective feature extraction. arXiv preprint. 2010. URL: http://arxiv.org/pdf/1009.5758.pdf (accessed: 14.02.2016).
25. Pham M.T., Cham T.J. Fast training and selection and Haar features using statistics in boosting-based face detection // In IEEE International Conference on Computer Vision. 2007. P. 1-7.
26. Saberian M.J., Vasconcelos N. Boosting classifier cascades // In Proceedings of the 24th Annual Conference on Neural Information Processing Systems. 2010. P. 7-9.
27. Subburaman V., Marcel S. Fast bounding box estimation based face detection // In European Conference on Computer Vision. Workshop on Face Detection. 2010. P. 1-14.
28. Tek S.C., Gokmen M. GPU accelerated real-time object detection on high resolution videos using modified census transform // In VISAPP. 2012. P. 685-688.
29. Trefny J., Matas J. Extended set of local binary patterns for rapid object detection // In Computer Vision Winter Workshop. 2010. P. 1-7.
30. Viola P.A., Jones M.J. Robust real-time face detection // In International Journal of Computer Vision, 2004. P. 137-154.
31. Wolf L., Hassner T., Maoz I. Face recognition in unconstrained videos with matched background similarity // In Computer Vision and Pattern Recognition. 2011. P. 529-534.
32. Wu B., Ai H., Huang C., et al. Fast rotation invariant multi-view face detection based on real adaboost // In IEEE International Conference on Automatic Face and Gesture Recognition. 2004. P. 79-84.
33. Yan J., Lei Z., Wen L., et al. The fastest deformable part model for object detection // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014. P. 2497-2504.
34. Yang S., Luo P., Loy C.C., et al. From facial parts responses to face detection: A Deep Learning Approach. arXiv preprint. 2015. URL: http://arxiv.org/pdf/1509.06451.pdf (accessed: 14.02.2016).
35. Yang B., Yan J., Lei Z., et al. Aggregate channel features for multi-view face detection. arXiv preprint. 2014. URL: http://arxiv.org/pdf/1407.4023.pdf (accessed: 14.02.2016).
36. Zayani R., Bouallegue R., Raviras D. Levenberg-Marquardt learning neural network for adaptive pre-distortion for time varying HPA with memory in OFDM systems // In European Signal Processing Conference. 2008. P. 1-5.
37. Zhu X., Ramanan D. Face detection, pose estimation, and landmark localization in the wild // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2012. P. 1-8.