[16] Review and testing of frontal face detectors

I.A. Kalinovskiy [1], *V.G. Spitsyn* [1]
[1] *Tomsk Polytechnic University*

Abstract

This paper presents comparison results for the proposed face detection algorithm based on a compact convolutional neural network cascade and modern frontal face detectors. Test results for 16 frontal view face detectors on two public benchmarks datasets are shown. A comparative assessment of the performance of face detection algorithms is made.

Keywords: face detection, cascade classifiers, convolutional neural networks, deep learning.
Citation: Kalinovskiy IA, Spitsyn VG. Review and testing of frontal face detectors. Computer Optics 2016; 40(1): 99-111. DOI: 10.18287/0134-2452-2016-40-1-99-111.

Introduction

Face detection is the first step in solving face analysis problems such as person identification, emotion, gender, and age recognition. The practical interest to these problems focuses on the fact that they are much needed in digital cameras, smartphones and other devices. They are also widely used in smart security systems and are highly demanded in photo control services through social media.

So far, efforts of researchers are aimed at developing face detection algorithms in the wild with considering to three degrees of head motion freedom. The complexity of the problem consists in a wide variety of face expressions, conditions, and postures, in which a person can be captured, as well as in a smaller facial area compared to the total picture surface.

However, we can offer quite a lot of scenarios for whom detection of persons for any angles of filming is an excessive requirement. For example, when designing biometrical access control systems and interactive advertisement billboards, it is assumed that a person looks at a camera frontally or at a small angle thereto.

In case of searching objects through video streams, apart from high precision-and-recall factors, a detector should run in real-time on relatively cheap computer equipment. If we do not take into account the detectors based on empirical models (for example, parametric models of the distribution of skin tones) that do not work well in real conditions, then, as a rule, such algorithms have high complexity and occupy most of the time of frame processing. In this case, their operating speed depends on the video stream resolution, a minimum size of target objects, the frame scaling factor, and a number of objects presented in scene. Variation of these parameter values can result in fast performance degradation. Therefore, improvement of computational efficiency of these algorithms is currently very important.

The authors have developed a new frontal view face detection approach based on a compact convolutional neural network cascade with a minimum number of parameters [1]. In this paper, it is benchmarked with 15 face detectors that have their source codes or demo versions publicly available.

1. Face detection algorithms

Ideas proposed by Viola and Jones in the early 2000s are the basis for many modern object detection algorithms [2]. A detector construction scheme developed thereof allows its computational complexity to be significantly reduced when its generalization performance remains the same. It is based on the following idea:

a) simple Haar functions (primitives), which can be efficiently calculated via an integral image;

b) using the AdaBoost algorithm to build up a composition (a strong classifier) consisting of simple threshold decision rules (weak classifiers) that use Haar functions to detect target objects;

c) construct detectors in the form of cascades containing several strong classifiers (stages) with different complexity for quick selection of image background areas at early stages.

The use of cascading structures is now a standard procedure when building-up real-time detectors. However, only simple features extracted by Haar functions are not enough for reliable detection of complex objects in the wild (inhomogeneous backgrounds, insufficient lighting, overlapping, and perspective distortions).

There are many papers dedicated to improvement of the Viola-Jones classical approach (Table 1). A key point of them is to expand primitive Haar-like features

[3] or to use other functions in order to extract features, as well as to modify weak classifiers.

Table 1. Cascade face detectors

Algorithm	Feature	Classifier	Type*
Lienhart R. [3]	Haar	Boosting over decision stumps	frontal
Jain V. [4]	Haar		
Subburaman [5]	MCT	Boosting over specific decision rule	
Marku N. [6]	Binary test	Boosting over decision trees	
Li J. [7]	SURF	Boosting over logistic regression	Frontal/ profiled
Barr J. [8]	Haar	Boosting over decision stumps	
Yang B. [9]	Set of channels		
Mathias M. [10]			

* *Different models are given for each head orientation.*

Grayscale images should be enough for the operation of most face detectors [2-8]. In papers [9, 10], an alternative approach is offered when classifiers are trained by a combination of different color channels (grayscale, RGB, HSV, LUV) with addition of HOG descriptor maps and a gradient magnitude. In other words, both color and geometric information about an object is apparently taken into account. Thus, in paper [9], a subsampling operation is preliminary applied to obtained maps followed by forming a vector, whereas in paper [10] their integral expression is used for fast computation of features.

The disadvantage of a boosted cascade classifier by Viola and Jones and some other similar classifiers is the dependence of image processing time on its content, because it is impossible to predict in advance at what stage of the cascade a background area will be rejected. Besides, other problems emerge in classifying objects that possess large intraclass dispersion. For example, when solving the face detection problem, it is general practice that different models are trained for different angles of head rotation with relative to camera ($0°\pm\psi$ – frontal, $45°\pm\psi$ – half-frontal, $90°\pm\psi$ – profiled).

In addition to face detection task, of interest is also the determination of head tilts and arrangement of key points (positions of eyes, nose, lips, etc.). The fact that these additional problems can be directly solved at the stage of detection allows significantly reducing the number of false alarms. This approach is discussed in papers [11, 12]. In paper [11], a two-level detector is considered. The first level is presented by a standard cascade face detector, and the second one – by a multitasking convolutional neural network that additionally inspects detections, identifies face orientations, and detect facial landmarks. In paper [12], a cascade model is offered that simultaneously solves face detection and face alignment problems. It helps improve classifier precision while retaining an acceptable operation speed.

Another class includes methods, in which the search is performed by comparing each image region with a target pattern or with a deformable object model that allows us to simulate a wide range of variations in its shape. The latest advances in these areas are presented in papers [13, 14]. In paper [13], the mixture of deformable models is studied. Its characteristic feature is the ability to detect faces, to determine their postures, and to predict facial landmarks within the framework of a single procedure. In paper [14], the efficient search method is proposed using a pattern matching technique, in which negative images are additionally used to suppress false detections. For fast calculation of a response map, the authors used the generalized Hough transform. Algorithms mentioned in papers [13, 14] provide high precision-and-recall scores for classification on standard tests. However, they are not suitable for real-time tasks since they have a very low execution speed (they are hugely slower than cascade classifiers).

The most advanced objects detection systems are currently built based on deep convolutional neural networks (CNNs) [15, 16]. In contrast to other machine learning methods, which demand preliminary extraction of informative features to perform classification, the convolutional networks solve both of these problems in process of learning, using directly source image data. The first attempts to build CNN-based face detectors were made in the mid 2000s [17, 18], though they didn't get widespread use and they are significantly inferior in their quality and operation speed vs state-of-the-art cascade detectors. However, the most advanced CNNs have also been recently applied to solve the face detection problem in the wild [19-21], and they have been of better quality compared to the above algorithms on standard benchmark datasets.

The authors [19] have taught the well-known network AlexNet using a collection of large scale pictures of the AFLW benchmark [22] containing a great variety of naturally captured postures and facial expressions. The training data were expanded by due to samples rotation on arbitrary angle. As a result, the authors have obtained a unified model, which helps consider to tilting and orientation thereof, and has low probability of false alarms, too. However, the AlexNet CNN

network contains $61 \cdot 10^6$ parameters and, because of the modern development of computer devices, it can't process live HD video streams using cost-effective hardware. Though, proper attempts are being made to optimize computation of CNNs [23].

In paper [20], as well as in our papers [1, 21], a proper attempt was made to improve the performance of deep convolutional networks solving object detection problems through building a cascade-structured detector in accordance with ideas by Viola and Jones. The cascade proposed by the authors and consisting of 6 CNNs is able to detect faces over a wide range of head positions, but it still has high computational complexity. All detector performance data presented herein indicate that the detector can process live VGA video streams using only high-end graphics cards such as Nvidia GeForce GTX TITAN Black GPU. It is evident in this case that a search time depends heavily on a number of persons presented in scene, since very "slow" networks with lots of convolution kernels are used at the last cascade stages.

2. Cascade of compact convolutional neural networks

In recent years, the convolutional neural networks have achieved great success in many computer vision tasks. They now help identify thousands of different classes of objects [24], thus a recognition rate for separate classes, e.g., such as house numbers [25], is comparable to medium human capabilities. One of the reasons for this success is the increasing number of neurons and connections. Sound or image analysis via CNNs containing billions of parameters does not seem to be a large problem due to increasing volumes of computing resources of cloud-based platforms and, most significantly, the advent of GPU virtualization technologies (e.g., Nvidia GRID). In analysis problems for video streams generated by mega-pixel CCTV systems, volumes of data increase significantly. Though VSaaS technologies (Video Surveillance as a Service) have been flourishing, these services have usually limited opportunities for video analysis reduced to simple functions (e.g., motion detection). The best solution to this problem is to place compute nodes directly in digital cameras and to transfer data mining functions thereto. This would solve scaling problems, but should require adaptation of algorithms in accordance with limited computational capabilities of embedded systems.

The frontal face detection problem is a relatively simple classification task, since it can be solved even with the use of elementary features such as, for example, MCT [5] or binary testing [6]. The main difficulty is the reduction of false detection because it is impossible to consider all conditions at training (e.g., background, lighting conditions) under which a real-time algorithm will perform. Therefore, first, the use of complex models is not feasible to solve this private problem, especially in circumstances when computational resources are limited. Second, to build a high-end and efficient classifier, it is needed to select features keeping balance between the informational value of object description and the complexity of extraction.

Convolutional neural networks possess high flexibility and they can preset the complexity of models by changing a number of layers, maps, and sizes of convolutional kernels. The capability to fine-tune features extracted at each layer, when getting training in detection of objects of one particular class, allows CNNs to achieve high precision levels in searching objects against strongly inhomogeneous backgrounds. It should be considered that capabilities of the neural networks to generalize the object images are reduced with decreasing number of parameters, whereby the frequency of type I errors (false detection) has been growing. However, this problem can be solved by additional check of detections using more complicated networks (i.e., those that are capable to provide a greater classification accuracy) similarly to the structure of the cascade classifier by Viola and Jones.

Cascade structure

This proposed cascade face detector consists of 3 convolutional neural networks whose architectures are shown in Figure 1. Each CNN solves the problem of a background/face binary classification and contains 797 (CNN stage 1), 1,819 (CNN stage 2) and 2,923 (CNN stage 3) parameters. Rational approximation of a hyperbolic tangent is used as an activation function:

$$f(x) = 1.7159 \cdot \tanh\left(\frac{2}{3}x\right),$$

$$\tanh(y) \approx \operatorname{sgn}(y)\left(1 - \frac{1}{1 + |y| + y^2 + 1.41645 \cdot y^4}\right).$$

Neurons of subsampling layers have additionally one weight and bias. The convolution stride is 1 pixel, and the pooling stride is 2 pixels. In CNN_1 and CNN_2, instead of traditional fully connected layers, sparse layers are used (similarly to [17]). This gives a 50% increase in the speed of the forward pass.

Training process

When designing the detector, we focused on video processing. For the CNN training, aligned face images were taken from YouTube Faces Database [26].

Background images were sampled from random YouTube videos in several stages during the preparation of models. Face areas (such as eyes, nose, etc.) were also added to the negative samples.

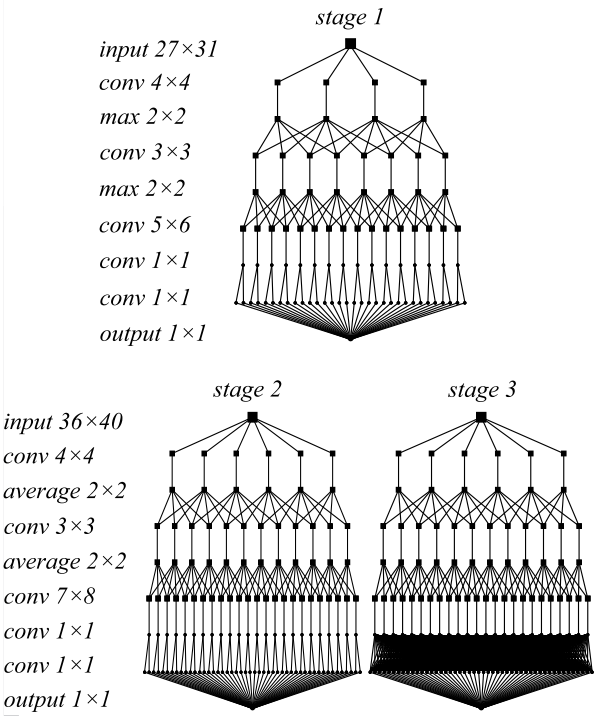

Fig. 1. Compact convolutional neural networks cascade

The authors have developed a framework for training convolutional neural networks supporting multithreaded implementations on CPU and integration capability for Intel IPP and Intel MKL libraries to speed up linear algebraic operations. About 1,000 experiments were carried out with models in respect of selection of the best CNN architecture and training parameters. In our experiments, we aimed to find the minimum configuration of a CNN which would be able to classify the validation test set with an error below 0.5%. The training time varied from several hours to 2-3 days depending on the CNN size (for 10^6 training examples on Intel Core i7 CPUs).

We used grayscale images for the training set. The training set was normalized preliminarily in some experiments using a procedure combining gamma correction and DoG filter [27]. This transformation allowed faces to be detected even in faint light. However, these models have not been included in the final detector option due to the computational complexity of normalization procedures.

The CNN configuration was carried out as follows. The number of convolutional layers varied from 1 to 4; the number of feature maps on each subsequent layer was redoubled. Pooling layers retrieved the training set from a 2×2 domain with the stride of 2 pixels along each axis that decreased the maps area by a factor of 4. The number of pooling layer maps interconnected with convolutional layer maps varied from 2 to 5. The number of neurons in the next-to-last layer was a multiple of the number of feature maps at the last convolutional layer.

The CNN (stage 1) architecture (Fig. 1) is the best obtained solution possessing simultaneously absolute compactness and reasonable generalization perfomanse. CNNs with 2 and 3 maps at the first layer (399 and 598 parameters, respectively) did not overcome a 0.5% error level. Also, the network with a less input size, for example 20×20 (461 parameters), hasn't been trained. Note that this configuration contains the least number of convolution kernels from all CNN architectures formerly suggested to solve the face detection problem [17-20]. In this case, the probability of deviations of a background-containing window is 100 times higher for CNN stage 1 compared to the cascade described in paper [20].

All experiments were conducted mainly with CNNs with small numbers of parameters. Therefore, for the CNN training a Levenberg-Marquardt algorithm was used as a basic training algorithm with a faster convergence (but a higher computational cost of iteration) compared to widespread methods for solving similar optimization problems, such as GD, CG, L-BFGS techniques [28]. The training results of detector-based CNNs are shown in Table 2.

A test database set used to evaluate the quality of different models contained several video clips. Table 3 shows the comparison of quality for every cascade stage and their cooperation on test data.

Table 2. Error levels for detector-based CNNs

Database set	Number of images, thou	Classification error, %		
		stage 1	stage 2	stage 3
Train	faces – 433 background – 585	0.142	0.059	0.047
Validation	faces – 239 background – 233	0.484	0.481	0.353

Table 3. Evaluation of cascade stages on test data

Metrics	Stage 1	Stage 2	Stage 3	Cascade
Recall	0.891	0.931	0.945	0.844
Precision	0.179	0.219	0.104	0.990

3. Test protocol

This section describes characteristics of 15 modern frontal face detection algorithms, which were compared to this detector proposed by the authors. It also presents test database sets and a proper estimating method.

Algorithms and libraries involved in testing

1) A compact convolutional neural networks cascade described in paper [1]. Denoted by: CompactCNN.
2) OpenCV 3.0.0 is a popular open source computer vision library that provides image processing algorithms.
3) It contains an object detectors framework based on the modified Viola-Jones algorithm [3]. It is supplied with five frontal face detectors. Four of them use Haar-like features and one – local binary patterns (LBP). Denoted by: OpenCV-default, OpenCV-alt, OpenCV-alt2, OpenCV-alt-tree, OpenCV-lbp..
4) MathWorks MatLab 2013b, Computer Vision Toolbox is a package of computer vision algorithms for frontal face detectors, including a Haar cascade and an LBP cascade. Denoted by: Matlab-CART, Matlab-LBP.
5) The algorithm in [7] is based on SURF-descriptors and uses a logistic regression in the capacity of a weak classifier. It is supplied as a dynamic library containing two models. Denoted by: SURF-24, SURF-32.
6) The algorithm in [6] uses, as one of its features, pixel intensity comparison-based object detection (a binary test). A source code is provided. Denoted by: PICO.
7) The algorithm in [29] is an LBP-based cascade trained using OpenCV features based on the AFLW benchmark [22]. It is supplied in model format for OpenCV object detectors. Denoted by: OpenCV-Kostinger.
8) The algorithm in [30] is a Haar-based cascade trained using OpenCV features. It is supplied in model format for OpenCV object detectors. Denoted by: OpenCV-Pham.
9) The algorithm in [13] is based on the mixture of deformable models and includes models for face profile orientations. The source code in the Matlab language and two trained detectors are provided. Denoted by: FDPL-small, FDPL-large.
10) The algorithm in [31] is a cascade based on support vector machines (SVM). It is supplied in the form of a dynamic link library. Denoted by: FDLIB.
Table 4 gives some characteristics for the above mentioned algorithms.

Test database sets

There are many database sets specially designed for evaluating face detectors such as FDDB [32], AFW [13], MALF [33], IJB-A [34] benchmarks, etc. For some sets (e.g., FDDB, MALF, IJB-A) a standardized assessment algorithm is provided. Detectors were tested on the most popular and, at the same time, more complicated benchmarks such as FDDB and AFW, which are able to evaluate the efficiency of algorithms for the problem of searching faces in the wild.
The Face Detection Data Set and Benchmark (FDDB) benchmark is a collection consisting of 2,845 pictures (no more than 0.25 megapixels). It contains annotations for 5,171 faces with a dimension of 20x20 pixels each. For evaluating algorithms, the cross-validation is used for 10 image subsets followed by results averaging.

Table 4. Summary table of face detector characteristics

Algorithm	Number of stages	Input size	Search stride, pixel
CompactCNN (our)	3	23×27	4
OpenCV-default	25	24×24	1
OpenCV-alt	22	20×20	1
OpenCV-alt2	20	20×20	1
OpenCV-alt-tree	47	20×20	1
OpenCV-lbp	20	24×24	1
Matlab-CART	-	20×20	-
Matlab-LBP	-	24×24	-
SURF-24	5	24×24	Variable
SURF-32	5	32×32	Variable
PICO	24	24×24	0.1· *minSize**
OpenCV-Kostinger	24	24×24	1
OpenCV-Pham	31	20×20	1
FDPL-small	no	80×80	-
FDPL-large	no	150×150	-
FDLIB	-	19×19	-

** minSize is the minimum face size*

The Annotated Faces in the Wild (AFW) benchmark consists of 205 large-scale (0.5-5 megapixels) images and contains annotations for 468 faces. This benchmark has been developed relatively recently and is mainly used for multi-view detectors evaluation.
The quality of the binary classifier may be evaluated via ROC and PR (Precision-Recall) curves [35]. In this paper, PR curves reflecting the dependence of algorithm precision (precision = TP/(TP+FP)) and recall (recall = TP/(TP+FN)) are used when varying the threshold of the decision rule. However, in practice, especially in video processing, readjusting of the threshold performed

to obtain the best precision-and-recall ratio under specific detector operating conditions is a complicated problem. Therefore, a fundamentally different approach is often used (see OpenCV Object detection framework: http://docs.opencv.org/3.0-beta/index.html), in which an image region is classified as an object-containing area if the number of neighbor detections (*minNeighbors*) inside this area exceeds the given threshold (Fig. 2). This method ensures strong regulation of the precision-and-recall ratio. Besides, it is convenient in setting up because of the parameter is integer.

When estimating the quality of the object detection algorithm, some difficulties emerge which are associated with ambiguous matching of localization areas found by the detector (detection) and pointed out by an expert (annotation). The following assessment criterion proposed for the PASCAL Visual Object Classes contest [36] is widely distributed now:

$$\sigma = \frac{S(A \cap D)}{S(A \cup D)}, \quad \delta(\sigma) = \begin{cases} \sigma \geq 0{,}5, & 1 \\ other, & 0 \end{cases},$$

where σ is the area overlapping factor; S is the area; A, D are respectively annotated and found by the detector object localization areas; δ is the assessment of true or false detections (each annotation can be matched with only one detection; the rest of them are considered to be false).

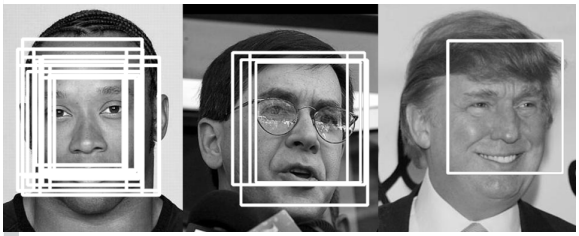

Fig. 2. Classifier-generated detections cluster in multiresolution image analysis

In case of face detection, a problem of estimation is additionally complicated by the following reasons (Fig. 3):

a) it is difficult to note an exact face boundary, especially for nonfrontal head postures;

b) detectors usually classify rectangular areas; this doesn't fit an oval face shape;

c) each detector defines various localization areas, depending on the training set and the detection clustering algorithm.

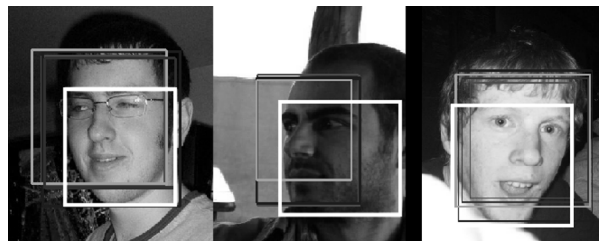

Fig. 3. Face annotations from the AFW benchmark (a white frame) and localization areas found by different detectors

This can cause errors in the annotation-detection comparison when the assessment of their mutual arrangement σ turns out to be less than 0.5, whereas the face is visually placed within a region localized by the algorithm [19]. This problem is usually solved by expanding detections boundaries [20]. In this paper, within a view to correcting the assessment of algorithms, annotation boundaries are expanded and narrowed according to rules described by algorithm 1 (Fig. 4). This allows us to use a single assessment procedure for all face detectors on different database sets.

Fig. 4. Annotations positions in the FDDB benchmark (at the top) and the AFW benchmark (at the bottom) generated by algorithm 1

Algorithm 1. Detections assessment

Input: X_a, Y_a, W_a, H_a, X_d, Y_d, W_d, H_d are the coordinates of the height, width, and coordinates of annotation and detection areas, respectively
Output: bool means the truth or false detections

```
1   for j = -1 to 1 do
2   for i = -1 to 1 do
3       for s = -2 to 2 do
4           if s < 0
5               scale = 1.1^-s
6           else
7               scale = 0.95^s
8           end if
9           X1 = Xa − 0.5·(scale − 1)·Wa
10          Y1 = Ya − 0.5·(scale − 1)·Ha
11          X2 = X1 + scale·Wa
12          Y2 = Y1 + scale·Ha
13          if i < 0
14              X1 = X1 + i·0.2·(X2 − X1)
15          else
16              X2 = X2 + i·0.2·(X2 − X1)
17          end if
18          Y1 = Y1 + j·0.2·(Y2 − Y1)
19          X3 = Xb
20          Y3 = Yb
21          X4 = X3 + Wb
22          Y4 = Y3 + Hb
23          if ¬ (X1 ≥ X4 || X2 ≤ X3 || Y1 ≥ Y4 || Y2 ≤ Y3)
24              if σ(X1, ..., X4, Y1, ..., Y4) ≥ 0.5
25                  return true
26              end if
27          end if
28      end for
29  end for
30  end for
31  return false
```

Annotations modification in the FDDB benchmark

Annotations in the FDDB benchmark are represented by ellipses; this allows us to describe face boundaries more precisely compared to rectangular areas. However, because of the above reasons, the comparison of elliptic shape annotations with rectangular detections results in faulty assessment of the latter ones (Fig. 5).

Fig. 5. Test results by CompactCNN on the FDDB benchmark. Detections marked in white are considered to be true, detections marked in black are considered to be false, annotations are marked with oval curves, and σ values are marked with numbers

In this context, to possibly use the proposed estimating method (algorithm 1), the ellipses have been replaced by the limiting rectangles as follows:

$$W = 2\sqrt{a^2 + \left(b^2 - a^2\right)\cdot \sin^2(\omega)},$$

$$H = 2\sqrt{a^2 + \left(b^2 - a^2\right)\cdot \cos^2(\omega)},$$

$$X = x - 0.5\,W, \quad Y = y - 0.5\,H,$$

where X, Y, W, H are the coordinates of the height, width, and coordinates of the rectangle, respectively; a, b, ω, x, y are the major and minor semi-axes, the angle of rotation, and the coordinates of the ellipse center, respectively.

Benchmark problems and algorithm parameters

We have considered three tasks: detection of small (from 20×20 pixels), medium (from 40×40 pixels) and big (from 80×80 pixels) faces. We calculated proper PR curves for each detector, depending on the *minNeighbors* parameter. The algorithms were totally benchmarked with 9 parameter values sets as follows:

1) minNeighbors – {1, 2, 3} (for the case 1, the detection algorithm in terms of recall prevails; for the case 3 – the detection algorithm in terms of precision prevails);

2) the minimum size of face (*minSize*) and the scaling factor connected with them (*scaleFactor*) used in constructing an image pyramid– {(20; 1.05); (40; 1.1), (80; 1.1)}.

Note 1: During testing, around every image makes a zero border with size 50 pixels.

Note 2: For the Matlab, PICO and FDLIB detectors, no *minNeighbors* parameter is provided. However, it is possible to control only the threshold of the decision rule (*threshold*). The threshold for these detectors was set as follows:

a) Matlab, PICO: *threshold* = 2 + *minNeighbors*;
b) FDLIB: *threshold* = 2 · *minNeighbors*.

Note 3: The FDPL detector provides no interface to control the *minNeighbors, minSize* and *scaleFactor* parameters, so it was tested only with default settings.

Note 4: The FDLIB detector provides no interface to control the *inSize* and *scaleFactor* parameters, so testing was performed only for 3 values of the *threshold*.

Note 5: The CompactCNN, SURF and OpenCV-Kostinger detectors were taught on face patterns cut precisely across the width of eyes and in line of eyes and chin. When training other models, a wider facial area was obviously used, including a forehead. As a result, detectors of the second group recognize smaller faces at a fixed *minSize* value. In this regard, with a view of meaningful comparison, a proportional factor for *minSize* was empirically selected to be used during initializing the

SURF and OpenCV-Kostinger detectors:
$minSize' = 0.75 \cdot minSize$.
In this case, face localization areas detected by these algorithms were expanded:
$X' = X - 0.2W$, $Y' = Y - 0.3H$, $W' = 1.4W$, $H' = 1.6H$,
where X, Y, W, H are the coordinates of the height, width, and coordinates of the upper left corner of a limiting rectangular, respectively.
Note 6. Values of additional parameters specific for each individual algorithm are shown in Table 5.

Table 5. Specific detector parameters

Algorithm	Parameter	Value
CompactCNN	T_1, T_2, T_3	-1.5, -1.25, -1.0
OpenCV	*useOptimized*	true
	useOpenCL	false
SURF	*step*	1
	fast	true
PICO	*stridefactor*	0.1

4. Results and benchmarks

The test results for 16 frontal face detectors on the AFW and DDB benchmarks (all data resulting from benchmarks are available at https://github.com/Bkmz21/FD-Evaluation) are given below. The comparison of all algorithms is performed at equal parameter sets, and the used test protocol is as close to real-time operating conditions as possible. This distinguishes these performed tests from existing performance evaluations of some algorithms (SURF, PICO, OpenCV Kstinger, FDPL) performed by ROC analysis, which reflects the ratio of true-to-false detection levels. However, it does not give an indication of algorithm behavior when choosing the best value of the threshold of the decision rule.
Figure 6 shows PR curves obtained when solving the face detection problem for faces with a dimension of 40×40 pixels. On the FDDB benchmark, the best results were shown by the OpenCV-Kstinger detector. The CompactCNN detector has a similar recall level but less precision. On the AFW benchmark, followed far behind thereof by recall ratio (R = 0.9), the leading role is played by the FDPL-small detector. However, the FDDB benchmark detects only 73% of the total number of objects in photos, because it is not designed for detecting small-size and blurred images. The CompactCNN cascade and the OpenCV-Kstinger cascade on the AFW benchmark are comparable with each other in their operation quality. Moreover, the CompactCNN detector on these benchmarks is of better precision compared to all Viola-Jones cascade detectors added to the OpenCV 3.0.0 library.

Quality assessment of face detectors

It is easier to assess the quality of binary classifiers using the F-measure that is defined as a harmonic mean of recall (R) and precision (P):

$$F = \left(\alpha \frac{1}{P} + (1-\alpha) \frac{1}{R} \right)^{-1}, \alpha \in [0,1],$$

where α is a weighting factor.

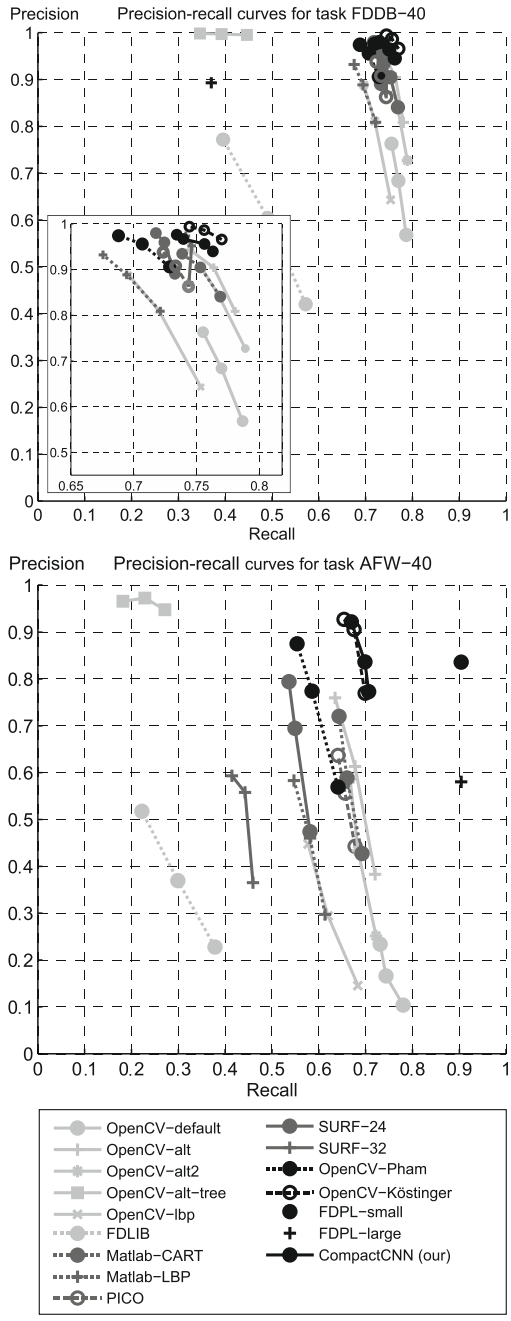

Fig. 6. The PR curve graphics for the face detection problem for minimum face size of 40×40 pixels on the FDDB benchmark and the AFW benchmark

The mean value of *F*-measures calculated along the *PR* curve is accepted as the single detector assessment for each benchmark (Table 6). Besides, an individual level α is set up for all *PR* curve points (Table 7), since precision and recall are not always of equal value for different scenarios of using detectors (for example, in biometric systems, precision is of higher priority).

Table 6. Evaluation of face detectors on benchmarks

	FDDB-20	FDDB-40	FDDB-80	AFW-20	AFW-40	AFW-80	$\overline{F}$
Compact CNN (our)	0.854	0.850	0.814	0.717	0.778	0.789	0.800
OpenCV-default	0.630	0.739	0.761	0.143	0.294	0.494	0.510
OpenCV-alt	0.806	0.837	0.803	0.462	0.662	0.754	0.721
OpenCV-alt2	0.751	0.816	0.798	0.319	0.552	0.712	0.658
OpenCV-alt-tree	0.665	0.597	0.531	0.475	0.402	0.398	0.511
OpenCV-lbp	0.697	0.778	0.763	0.225	0.420	0.578	0.577
Matlab-CART	0.795	0.831	0.794	0.441	0.647	0.752	0.710
Matlab-LBP	0.756	0.794	0.756	0.327	0.532	0.634	0.633
SURF-24	0.751	0.833	0.814	0.324	0.631	0.671	0.671
SURF-32	0.738	0.840	0.824	0.269	0.492	0.548	0.619
PICO	0.810	0.821	0.798	0.514	0.617	0.686	0.708
OpenCV-Koestinger	**0.873**	**0.863**	0.814	0.723	0.780	0.797	0.808
OpenCV-Pham	0.826	0.823	0.774	0.586	0.692	0.735	0.739
FDPL-small	0.842	0.842	**0.842**	**0.869**	**0.869**	**0.869**	**0.856**
FDPL-large	0.547	0.547	0.547	0.715	0.715	0.715	0.631
FDLIB	0.574	0.574	0.574	0.358	0.358	0.358	0.466

Table 7. Values of parameter α when calculating the F-measure

min Neighbors	α	Explanation
1	0.2	Detectors with high recall levels are prioritized
2	0.5	Precision and recall are of equal value
3	0.8	Detectors with high precision levels are prioritized

For each task, Table 8 shows space allocation between the detectors in relation to descending their *F*-measures and a relative delay behind the winning side (percentage difference) – ΔF. The mean value ΔF is offered as a relative general assessment of algorithm quality on all benchmarks. The assessment $\overline{\Delta F}$ shows how the *F*-measure level of the algorithm is comparable to the best solution. According to the above criterion, the ranking result for face detectors is given in Figure 7. The best detector is the FDPL-small detector (though not for all benchmarks). This is reflected by the assessment $\overline{\Delta F} = -1\%$. The detector is able to search either frontal or profiled faces that ensure recall criterion leadership on the AFW benchmark. It is followed by the OpenCV-Kostinger detector that is slower by 5.44% of percentage points. The CompactCNN detector ranks third, being slower than the FDPL-small detector by 6.36% and the OpenCV-Kostinger detector – by 0.92% of percentage points.

Table 8. Rating of face detectors by scores

	FDDB-20	FDDB-40	FDDB-80	AFW-20	AFW-40	AFW-80	$\overline{\Delta F}$,%
Compact CNN (our)	2 (-2.18)	2 (-1.51)	3 (-3.33)	3 (-17.5)	3 (-10.5)	3 (-9.21)	-7.36
OpenCV-default	13 (-27.8)	13 (-14.4)	9 (-9.62)	16 (-83.5)	16 (-66.2)	14 (-43.2)	-40.78
OpenCV-alt	6 (-7.67)	5 (-3.01)	4 (-4.63)	8 (-46.8)	6 (-23.8)	4 (-13.2)	-16.53
OpenCV-alt2	9 (-14)	10 (-5.45)	5 (-5.23)	13 (-63.3)	10 (-36.5)	8 (-18.1)	-23.75
OpenCV-alt-tree	12 (-23.8)	14 (-30.8)	13 (-36.9)	7 (-45.3)	14 (-53.7)	15 (-54.2)	-40.81
OpenCV-lbp	11 (-20.2)	12 (-9.85)	8 (-9.38)	15 (-74.1)	13 (-51.7)	12 (-33.5)	-33.11
Matlab-CART	7 (-8.94)	7 (-3.71)	6 (-5.7)	9 (-49.3)	7 (-25.5)	5 (-13.5)	-17.77
Matlab-LBP	8 (-13.4)	11 (-8)	10 (-10.2)	11 (-62.4)	11 (-38.8)	11 (-27)	-26.63
SURF-24	9 (-14)	6 (-3.48)	3 (-3.33)	12 (-62.7)	8 (-27.4)	10 (-22.8)	-22.28
SURF-32	10 (-15.5)	4 (-2.67)	2 (-2.14)	14 (-69)	12 (-43.4)	13 (-36.9)	-28.27
PICO	5 (-7.22)	9 (-4.87)	5 (-5.23)	6 (-40.9)	9 (-29)	9 (-21.1)	-18.04
OpenCV-Koestinger	**1 (0)**	**1 (0)**	3 (-3.33)	2 (-16.8)	2 (-10.2)	2 (-8.29)	-6.44
OpenCV-Pham	4 (-5.38)	8 (-4.63)	7 (-8.08)	5 (-32.6)	5 (-20.4)	6 (-15.4)	-14.41
FDPL-small	3 (-3.55)	3 (-2.43)	**1 (0)**	**1 (0)**	**1 (0)**	**1 (0)**	**-1.00**
FDPL-large	15 (-37.3)	16 (-36.6)	12 (-35)	4 (-17.7)	4 (-17.7)	7 (-17.7)	-27.03
FDLIB	14 (-34.3)	15 (-33.5)	11 (-31.8)	10 (-58.8)	15 (-58.8)	16 (-58.8)	-46.00

* *in parentheses, the percentage difference ΔF is given with the algorithm having the maximal F-measure for each of the tasks.*

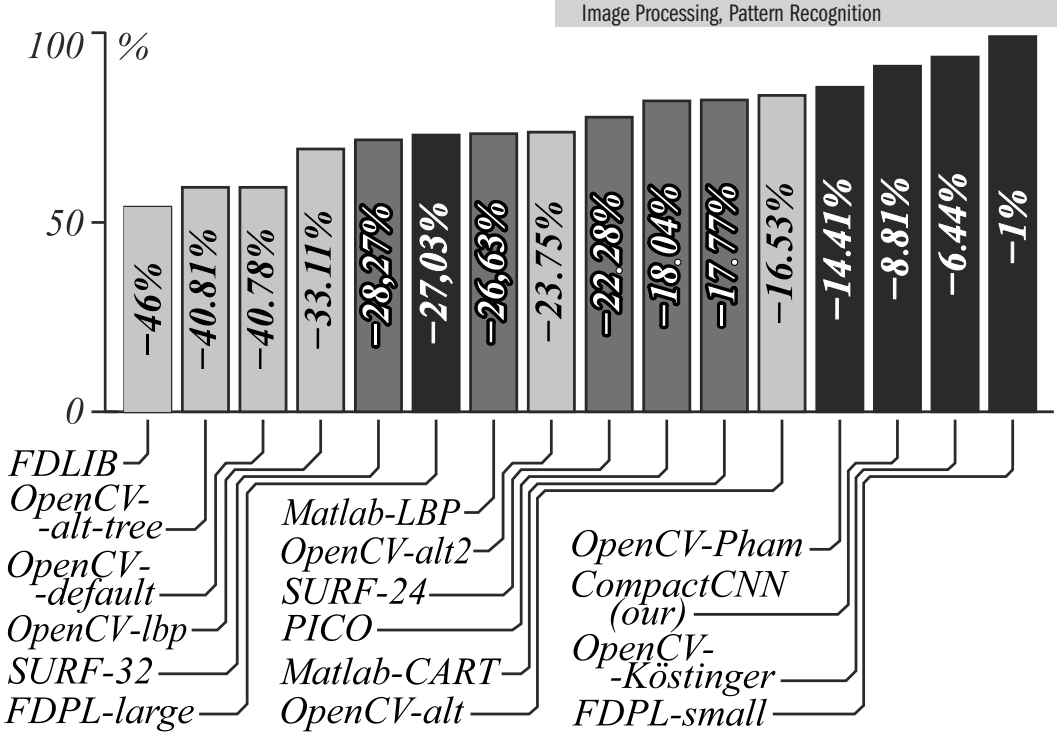

Fig. 7. Detectors ranking according to assessment of $\overline{\Delta F}$

Besides, the classifiers were evaluated using the Friedman test that showed lack of statistically significant differences in performance quality between the CompactCNN, OpenCV Köstinger and FDPL-small detectors at significance level of 0.05. Note that the FDPL-small and OpenCV Köstinger algorithms were trained on megapixel pictures from the AFLW benchmark, whereas for training the CompactCNN detector we used frames from random YouTube videos. Which is why, the designed detector can gain the advantage in problems of face search within videos, because in this case the processed data are comparable, to a far greater degree, to statistical distribution of the training set.

Performance evaluation of face detectors

The computational efficiency is an important characteristic of object detectors especially when solving real-time tasks and processing big data. Under such conditions, cascade structured detectors have the advantage, since they are able to quickly eliminate most of the image that doesn't contain any target objects. For example, the first stage of the Haar-like OpenCV-alt cascade consisting of 3 weak classifiers requires 26 arithmetic operations to perform classification (when using an integral matrix and rectangular parameters). The algorithmic complexity of CNNs is much higher because of necessity for computing output signals for several thousands of neurons. For example, to obtain the response map for CNN stage 1 (3,905 neurons, Fig. 1), we need $\approx 340 \cdot 10^6$ operations (with a window stride of 4 pixels) on the image with resolution of 1,280×720 pixels, while requiring only $\approx 23 \cdot 10^6$ for the OpenCV-alt cascade (with the window stride of 1 pixel).

In addition to the computational complexity, precision and the number of strong classifiers have a very significant effect on performance. In worst cases, the detector performance depends on the number of cascade stage when the image is completely filled with target objects. The total number of weak classifiers in the 22-staged OpenCV-alt detector is 2,135. Thus, in the worst case it will require $\approx 16 \cdot 10^9$ operations, while the CompactCNN detector will require only $\approx 1.5 \cdot 10^9$ operations.

Probability distribution of type I errors by cascade stages determines the detector performance speed in normal operation conditions (i.e. a level between processing of an absolutely black image and a fully face-filled image). The Haar-like cascades from the OpenCV detector reject 60-70% of positions of a sliding window at the first stage, using from 3 to 9 features. The detector [7] based on more complex features – SURF-descriptors – rejectes 95% of windows, however at the same time, it produces an increasingly larger number of computations. The CompactCNN detector extracting high-level features optimized for face detection is able to reject, already at the first stage, more than 99.99% of all window positions at a zero threshold of the decision rule [1]. Thus, its operation speed weakly depends on the

image content.

The microarchitecture of modern processors is superscalar. It contains blocks of reordering instructions and renaming registers, and the control devices are able to perform many various scalar and vector data operations. In this regard, the algorithm performance capacity depends not only on its computational complexity, but also on its structure (conditional constructs, order of memory assess, etc.), data type and instructions sets, as well as on the ability to be efficiently vectored and paralleled.

The decision trees are base elements for most detectors built with the help of a boosting procedure (see Table 1). Objects are searched by the sliding window where features are computed at different fixed points that is not preferable from the point of view of loading values in cache. A tree traversal procedure is poorly vectored due to the relationship between results and the transition sequence. In this regard, it is quite difficult to build an efficient algorithm to compute detectors of this type that takes advantages of SIMD extensions of modern CPUs and massively parallel GPU architectures. Many of papers are dedicated to the solution of this problem [37, 38].

From the point of view of the computational architecture, CNNs are much more efficient than Viola-Jones algorithms and their modern modifications:

1) The CNN response map can be computed without using the sliding window by applying a linear operation of correlation with filter sets, pooling operations, and non-linear by-pixel transformation [39] to the total image. Due to this property, data are read by continuous data memory units. This allows efficiently use a CPU cache.

2) There are several dozens of data operations (without branches inside iterations) for each reading operation.

3) By its definition, a neural network is a massively parallel algorithm, so it can be easily vectored and paralleled.

The above mentioned specific features of CNNs are essentially important when computations are transferred to GPUs. The Graphic processing units (GPUs) allow us to fully disclose all advantages of natural parallelism of CNNs and they are supported in hardware for two-dimensional database sets.

The main disadvantage of CNNs is the fact that it is necessary to store large volumes of intermediate computations (features maps). This increases the required memory space and reduces the efficiency of multithreaded implementation under insufficient size of CPU's memory cache.

The CompactCNN detector has been implemented using 3 technologies: SIMD extension of x86 processor family (for each of 3 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.

Figure 8 gives the performance comparison for the face detectors involved in testing. Measurements were made for a single execution mode on Intel Core i7-3610QM CPU (3.1 GHz) while processing the first image subset from the FDDB benchmark (*minSize* – 40×40, *scaleFactor* – 1.1, *minNeighbors* – 2, with no frame added). Program files and libraries for all detectors, except SURF and FDLIB, are 64-bit ones. We used the C++ compiler incorporated with IDE Microsoft Visual Studio Community 2013, OC Microsoft Windows 8.1 (64-bit).

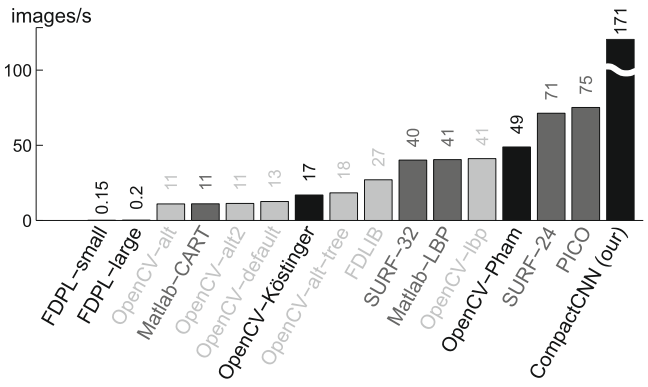

Fig. 8. Detectors ranking by operation speed for the FDDB-40 benchmark

Due to the original algorithm used for computing CNNs and optimizing the software code with the help of the vector intrinsic functions, and with due consideration of limited number of logical registers and specific features of Intel CPU microarchitectures, the CompactCNN detector demonstrates the outstanding data processing speed (during testing, we used the implementation optimized by AVX intrinsic functions).

Conclusion

The test results for 16 frontal view face detectors show that the proposed detector based on the compact CNN cascades ranks third by its *F*-measure level being averagely 7.4% behind the best solution. However, it takes the leading position in its data processing speed, thus improving the previous record by 2.3 times (PICO) and being 1140 times (FDPL-small) and 10 times (OpenCV-Kstinger) as greater as other detectors of better quality.

The designed cascade detector consists of only 3 stages. In addition, 99.99% of background-containing windows are rejected already at the first stage. This factor substantially reduces the dependence of the detector speed on image content [1]. Because of availability of implementations for 3 computing technologies, including OpenCL, the CompactCNN detector can be launched basically on any device. For this purpose, the implementation for Intel CPU and Nvidia GPU has been highly optimized with regard to specific features of processors microarchitectures. The cascade of small CNNs proved to be a very effective solution for the frontal face detection problem and made it possible, for the first time, to process a 4K Ultra HD high-resolution video stream even on low-power computing devices [1].

The research is carried out at Tomsk Polytechnic University within the framework of Tomsk Polytechnic University Competitiveness Enhancement Program.

References

1. Kalinovskiy IA, Spitsyn VG. Compact Convolutional Neural Network Cascade for Face Detection. Source: http://arxiv.org/abs/1508.01292.pdf.

2. Viola P, Jones MJ. Rapid object detection using a boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition; 2001; 1: 511-518.

3. Lienhart R, Maydt J. An extended set of Haar-like features for rapid object detection. IEEE International Conference on Image Processing; 2002: 1: 900–903.

4. Jain V, Learned-Miller E. Online domain adaptation of a pre-trained cascade of classifiers. IEEE Conference on Computer Vision and Pattern Recognition; 2011; 577–584.

5. Subburaman V, Marcel S. Fast bounding box estimation based face detection. European Conference on Computer Vision, Workshop on Face Detection; 2010; 1–14.

6. Marku N, Frljak M, Pandi IS, Ahlberg J, Forchheimer R. A method for object detection based on pixel intensity comparisons organized in decision trees. Source: http://arxiv.org/abs/1305.4537.pdf.

7. Li J, Zhang Y. Learning SURF cascade for fast and accurate object detection. IEEE Conference on Computer Vision and Pattern Recognition; 2013; 3468–3475.

8. Barr JR, Bowyer KW, Flynn PJ. The effectiveness of face detection algorithms in unconstrained crowd scenes. IEEE Winter Conference on Applications of Computer Vision; 2014; 1020–1027.

9. Yang B, Yan J, Lei Z, Li SZ. Aggregate channel features for multi-view face detection. IEEE International Joint Conference on Biometrics; 2014; 1-8.

10. Mathias M, Benenson R, Pedersoli M, Van Gool L. Face detection without bells and whistles. European Conference on Computer Vision; 2014; 720-735.

11. Zhang C, Zhang Z. Improving multiview face detection with multi-task deep convolutional neural networks. IEEE Winter Conference on Applications of Computer Vision; 2014; 1036-1041.

12. Chen D, Ren S, Wei Y, Cao X, Sun J. Joint cascade face detection and alignment. European Conference on Computer Vision; 2014; 109-122.

13. Zhu X, Ramanan D. Face detection, pose estimation, and landmark localization in the wild. IEEE Conference on Computer Vision and Pattern Recognition; 2012; 2879-2886.

14. Li H, Lin Z, Brandt J, Shen X, Hua G. Efficient boosted exemplar-based face detection. IEEE Conference on Computer Vision and Pattern Recognition; 2014; 1843-1850.

15. Zeiler M, Fergus R. Visualizing and understanding convolutional networks. European Conference on Computer Vision; 2014; 818-833.

16. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. Source: http://arxiv.org/abs/1409.4842.pdf.

17. Garcia C, Delakis M. Convolutional face finder: A neural architecture for fast and robust face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence; 2004; 1408-1423.

18. Osadchy M, LeCun Y, Miller M. Synergistic face detection and pose estimation with energy-based models. Journal of Machine Learning Research; 2007; 1197-1215.

19. Farfade SS, Saberian M, Li L-J. Multi-view face detection using deep convolutional neural networks. International Conference on Multimedia Retrieval; 2015.

20. Li H, Lin Z, Shen X, Brandt J, Hua G. A Convolutional neural network cascade for face detection. IEEE Conference on Computer Vision and Pattern Recognition; 2015; 5325-5334.

21. Kalinovskiy IA, Spitsyn VG. Algorithm for face detection on Ultra HD video [In Russian]. Conference on technical vision in control systems; 2015; 95-96.

22. Kostinger M, Wohlhart P, Roth PM, Bischof H. Annotated Facial Landmarks in the Wild: A Large-scale, real-world database for facial landmark localization. IEEE International Conference on Computer Vision Workshops; 2011; 2144-2151.

23. Vasilache N, Johnson J, Mathieu M, Chintala S, Piantino S, LeCun Y. Fast convolutional nets with fbfft: A GPU performance evaluation. Source: http://arxiv.org/abs/ 1412.7580.pdf.

24. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. Source: http://arxiv.org/abs/1502.03167.pdf.

25. Lee C-Y, Xie S, Gallagher P, Zhang Z, Tu Z. Deeply-supervised nets. Source: http://arxiv.org/abs/ 1409.5185.pdf.

26. Wolf L, Hassner T, Maoz I. Face recognition in unconstrained videos with matched background similarity. IEEE Conference on Computer Vision and Pattern Recognition; 2014; 529-534.

27. Kalinovskiy IA, Spitsyn VG. Face detection algorithm based on the convolutional neural network [In Russian]. Neurocomputers: Development and Applications; 2013; 10: 48-53.

28. Le QV, Coates A, Prochnow B, Ng AY. On Optimization Methods for Deep Learning. International Conference on Machine Learning; 2011; 265-272.

29. Kostinger M. Efficient metric learning for real-world face recognition. Graz University of Technology. PhD thesis; 2013.

30. Pham MT, Cham TJ. Fast training and selection and Haar features using statistics in boosting-based face detection. IEEE International Conference on Computer Vision; 2007; 1-7.

31. Kienzle W, Bakir G, Franz M, Scholkopf B. Face detection: efficient and rank deficient. Advances in Neural Information Processing Systems; 2005; 673-680.

32. Jain V, Learned-Miller E. FDDB: A Benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009. University of Massachusetts; 2010.

33. Yang B, Yan J, Lei Z, Li SZ. Fine-grained evaluation on face detection in the wild. IEEE International Conference on Automatic Face and Gesture Recognition; 2015.

34. Klare BF, Klein B, Taborsky E, Blanton A, Cheney J, Allen K, Grother P, Mah A, Burge M, Jain AK. Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark A. IEEE Conference on Computer Vision and Pattern Recognition; 2015; 1931-1939.

35. Davis J, Goadrich M. The relationship between Precision-Recall and ROC curves // International Conference on Machine Learning; 2006; 233-240.

36. Everingham M, Gool LV, Williams C, Winn J, Zisserman A. The PASCAL visual object classes (VOC) challenge. International Journal of Computer Vision; 2010; 88(2): 303-338.

37. Oro D, Fernandez C, Saeta JR, Martorell X, Hernando J. Real-time GPU-based face detection in HD video sequences. IEEE International Conference Computer Vision Workshops; 2011; 530–537.

38. Nguyen T, Hefenbrock D, Oberg J, Kastner R, Baden S. A software-based dynamic-warp scheduling approach for load-balancing the Viola–Jones face detection algorithm on GPUs. Journal of Parallel and Distributed Computing; 2013; 73(5): 677–685.

39. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y. OverFeat: Integrated recognition, localization and detection using convolutional networks. Source: http://arxiv.org/abs/1312.6229.pdf.