Fast Deep Convolutional Face Detection in the Wild Exploiting Hard Sample Mining

Danai Triantafyllidou, Paraskevi Nousi*, Anastasios Tefas

Artificial Intelligence and Information Analysis Lab
Department of Informatics
Aristotle University of Thessaloniki

Abstract

Face detection constitutes a key visual information analysis task in Machine Learning. The rise of Big Data has resulted in the accumulation of a massive volume of visual data which requires proper and fast analysis. Deep Learning methods are powerful approaches towards this task as training with large amounts of data exhibiting high variability has been shown to significantly enhance their effectiveness, but often requires expensive computations and leads to models of high complexity. When the objective is to analyze visual content in massive datasets, the complexity of the model becomes crucial to the success of the model. In this paper, a lightweight deep Convolutional Neural Network (CNN) is introduced for the purpose of face detection, designed with a view to minimize training and testing time, and outperforms previously published deep convolutional networks in this task, in terms of both effectiveness and efficiency. To train this lightweight deep network without compromising its efficiency, a new training method of progressive positive and hard negative sample mining is introduced and shown to drastically improve training speed and accuracy. Additionally, a separate deep network was trained to detect individual facial features and a model that combines the outputs of the two networks was created and evaluated. Both methods are capable of detecting faces under severe occlusion and unconstrained pose variation and meet the difficulties of large scale real-world, real-time face detection, and are suitable for deployment even in mobile environments such as Unmanned Aerial Vehicles (UAVs).

Keywords: Deep Learning, Convolutional Neural Networks, Face Detection

*Corresponding author

Email addresses: danaitri22@gmail.gr (Danai Triantafyllidou), paranous@csd.auth.gr (Paraskevi Nousi), tefas@aiia.csd.auth.gr (Anastasios Tefas)

1. Introduction

The spread of social media and the rise of the Internet of Things have significantly boosted the amount of data readily available in all aspects of human life and led us into the era of Big Data. This vast volume of data has the potential to enhance our understanding of the state of the world and to allow for more accurate predictions of a future state [1]. Big Data has already been exploited in many fields for this reason, including biometrics systems [2, 3, 4], where face detection poses a critical auxiliary role towards improved face recognition, as facial features capture a large part of the individuality of a person. In fact, face detection has been an active research area in the computer vision field for more than three decades, mainly due to the countless number of applications that require face detection as a first step [5, 6, 7, 8].

Nowadays, commercial and professional robotic units (e.g., drones) and mobile devices such as smartphones and tablets provide users with various face-based applications related to the task of face detection, such as intelligent video shooting, privacy preserving navigation and control, security-enhancing applications, automatic annotation of visual content and affective computing, including face and facial expression recognition and tracking [9]. In Unmanned Aerial Vehicles (UAVs) especially, face detection may serve as a tool to help guide the on-board camera towards faces of people of interest. As an example, in sports events, face detection may be the first step towards recognizing important athletes, such as bicyclists in professional cycling events.

However, the limited availability of powerful Graphical Processing Units (GPU) on such devices asserts a limitation to the performance of the algorithms that can be used efficiently. The recently released mobile on-board GPUs for drones are approximately ten times slower than desktop ones, with only a fraction of RAM and this constraint renders most of the published deep learning algorithms inadequate for such applications. The challenges of face detection and recognition using drones have been studied in [10].

Many non neural network methods have been proposed and deployed in various commercial products like digital cameras or smartphones in the last decade. The influential work of Viola and Jones [11] made it possible to detect faces in real-time but with a limited efficiency and later on inspired many cascade-based methods. Since then, research in face detection has made remarkable progress as a result of the availability of data in unconstrained capture conditions, the development of publicly available benchmarks and the fast growth in computational and processing power of modern computers. The introduction of features extraction methodologies such as Histograms of Oriented Gradients (HoGs) [12], Speeded Up Robust Features (SURF) [13], and Integral Channel Features (ICF) [14] was also of great benefit to face detection algorithms. In another approach, a mixture of trees was utilized for unified face detection, pose estimation and landmark localization [15].

More recently, Deep Learning methods have been deployed for the task of face detection with impressive results [16, 17, 18]. The term Deep Learning refers to a set of Machine Learning algorithms that utilize complex architectures

consisting of multiple levels, to extract multiple abstractions of their input data. The recent resurgence of interest in deep neural networks owes a certain debt to the availability of powerful GPUs which routinely speed up common operations such as large matrix multiplications. Deep convolutional neural networks have wide applications in language processing [19, 20], object classification [21, 22, 23, 24] and recommendation systems [25].

Big Data and Deep Learning seem to be interdependent on one another and exhibit a mutually beneficial relationship: large amounts of data allow Deep Learning techniques to achieve better generalization thus yielding significantly more informative results in the field of big media analysis [26, 27]. In general, Deep Learning techniques inherently require a large amount of processing power to be trained efficiently, due to their complex architectures. It is also a significantly difficult task to parallelize the computations required for this task. However, as computing power increases, it facilitates the training of deeper models which are capable of learning complex abstractions by utilizing multiple layers, and generalizing to unseen data, by learning from vast amounts of training data.

In this paper, a novel lightweight CNN for face detection is presented that, to the best of our knowledge, is the first one that has minimum computational complexity and attains comparable or better performance than previously published complex CNNs [16]. Additionally, a second CNN has been trained in order to detect facial features and it has been combined with the first one in a single architecture. The second CNN was trained exclusively for the detection of facial features (e.g., eyes, nose, mouth) while the first CNN was trained for full face detection. Figure 1 shows examples of the proposed face and facial parts detection. It is shown that a properly trained CNN can be further extended to perform more complex and computationally expensive tasks (i.e., face detection and facial feature detection in one forward pass). The proposed models are thus excellent candidates for machine learning assisted face detection in UAVs. This paper makes the following contributions:

1. A novel, computationally minimal model is proposed, consisting of only 76,375 trainable parameters, and shown to provide formidable results despite its low complexity for large-scale visual information analysis, and to be suitable for real-time detection with standard processing power, as opposed to most neural network based detection techniques.
2. A new training methodology is presented according to which the CNN is gradually supplied with training examples of scaling difficulty. It is shown that this method can drastically improve training speed and significantly reduce the number of false positives.
3. The addition of a pooling layer to the output of the deep CNN to smoothen the produced heat map as well as the addition of a regression output neuron that tries to estimate the width of each detected face in the detection rectangle is proposed.

During training, our models learn from more than half a million samples of faces and non-faces — approximately 600K samples of three publicly available

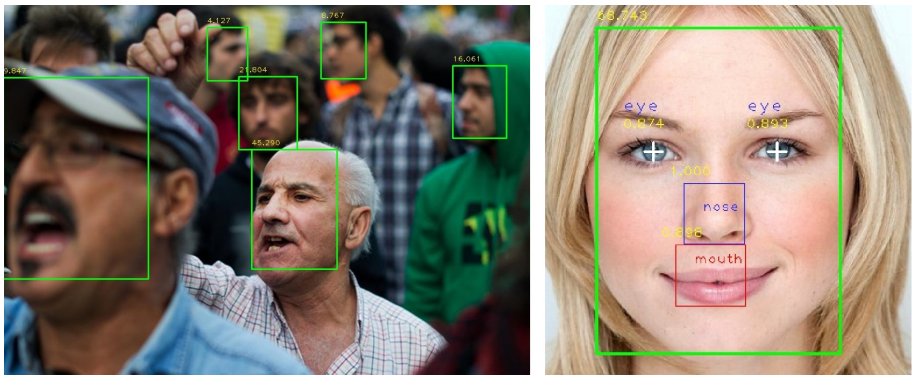


Figure 1: Left: An example of face detection in various poses and occlusions. Right: An example of facial parts detection. The bounding boxes and scores show output of the trained CNN.

datasets were used to properly train them, as discussed in more detail in Section 3.5. Our approach is then evaluated in the challenging FDDB [28] face detection dataset, depicting about 5K faces, where it achieves a recall rate of 92.6% and is compared against other recently published face detection methods. Our models are also evaluated on the WIDER FACE dataset [29], which depicts about 390K faces in total and exhibits large variability in the faces depicted in terms of scale, rotation and occlusion, thus making it a very challenging dataset. To the best of our knowledge, the used data corpus is one of the biggest for the given task.

The rest of this paper is organized as follows: in Section 2 previous work related to face detection is presented, in Section 3, after an introduction to CNNs, the proposed model is presented and analyzed, and Section 4 shows the experimental setup used as well as the yielded results. Finally, conclusions are drawn and summarized in Section 5.

2. Related Work

The original Viola-Jones detector used Haar-like features and is fast to evaluate, yet fails in detecting faces from different angles, variable resolution, blurred image, etc. This issue was initially addressed either by using one classifier cascade for each specific facial view, [30, 31], or by using a decision tree for pose estimation and the corresponding cascade to verify the detection [32]. However, these approaches require pose/orientation annotations while complex cascade structures increase the computational cost. The main line of research in this direction was based on the combination of robust descriptors with classifiers [33, 34]. Among the variants, a method named Headhunter [35] improved the performance by deploying the integral channel features method along with 22 cascades. The last method was extended in [36] where sub-sampled channel features are used to learn a cascade of classifiers. Finally, a joint cascade-based

method proposed in [37] achieving state-of-the-art results by introducing an alignment step in the cascade structure.

Another common family of face detection algorithms learn and deploy a Deformable Parts-based Model (DPM) [38] to model the information between facial parts. While DPM detectors are more robust to occlusion than cascade based methods, they lack in computational efficiency and are prohibitive for real-time detection. In [38] a unified DPM framework for face detection, pose estimation, and landmark estimation was proposed. A general approach for making DPM based methods faster is to build a cascade of classifiers from DPMs [39]. In [39], a simple DPM provides excellent performance and outperforms more complex DPM variants. Finally, the detection accuracy was significantly improved by a face detector called Deep Pyramid DPM [40]. The last method, generates a deep feature pyramid and uses a linear SVM for classification. Later, the same authors proposed a CNN trained with a multi-task learning algorithm [41] for simultaneous face detection, landmark localization, pose estimation and gender recognition.

A deep network named Alexnet [21], which was trained on ILSVRC 2012 [42], rekindled interest in convolutional neural networks and outperformed all other methods used for large scale image classification. The R-CNN method proposed in [43] generates category-independent region proposals and uses a CNN to extract a feature vector from each region. Then it applies a set of class-specific linear SVMs to recognize the object category. In [17], a cascade of CNNs was proposed which consists of 6 CNNs and operates on multiple resolutions. In [18] a deep CNN with three output branches for face/non-face classification, face pose estimation and facial landmarks localization was proposed. The model consists of three convolutional layers each followed by a max pooling layer and the last pooling layer is followed by a fully connected layer, whose output comprises the input of the three aforementioned branches.

Recently, a face detector called DDFD [16], showed that a CNN can detect faces in a wide range of orientations using a single model. The model accepts input images of size 227×227, and scales images up or down to detect faces larger or smaller than this size respectively. Lately, the architecture of VGG16 [44] was utilized for the task of face detection [45], in conjunction with region-based CNN detection models [46]. Another recent approach, also used a deep CNN [47] for the task of face detection by combining information from facial parts proposals and responses. Finally, an extensive survey of face detection methods can be found in [48].

Amongst other deep learning strategies, Denoising Autoencoders [49] are comprised of fully connected layers and they are typically used for unsupervised pretraining as well as dimensionality reduction, leading to more compact and robust representations of data. Face detection is a supervised visual learning task where convolutional deep approaches dominate in terms of performance and speed. It's also worth noting that most recently proposed methods for object detection don't make use of fully connected layers for this very reason: they impose a large number of trainable parameters and heavy time constraints. This time delay imposed by fully connected layers, in combination with the fact

that convolutional layers accept inputs of arbitrary size are the main reasons behind the recent trend of deploying fully convolutional neural networks for multiple tasks.

Even so, all the above methods use very complex networks having more than 1M number of parameters that renders them inappropriate for large-scale visual information analysis or real-time face detection with constrained computational power. In contrast, our proposed method is trained on input images of size 32×32 and it requires the training of three orders of magnitude fewer free parameters than the aforementioned models. Moreover, our method exhibits a novel training methodology, involving the gradual introduction of samples of scaling difficulty and introduces the addition of a pooling layer to the output of the network, which smoothens the produced heatmap.

3. Proposed Method

3.1. Convolutional Neural Networks

Convolutional neural networks (CNNs) [50, 51] constitute a subclass of feed-forward neural networks (FNNs) [52, 53]. An FNN's objective is to learn a parametrized function which models the relationship between its input data $\mathbf{x}$ and a desired output $\mathbf{y}$. At minimum, an FNN consists of an input and an output layer, but adding more layers in between the two has been shown empirically to enhance the network's performance [54], thus reinvigorating scientific interest in Deep Learning techniques. Each layer consists of a number of neurons and connections are established between all neurons belonging to consecutive layers with weights assigned to each connection.

A non-linearity s, called an activation function, is applied to the output of each neuron, which is a linear combinations of its input with their respective weights in addition to a bias value, in order to model non-linear relationships between the representations. Most commonly, the sigmoid function $s(z) = 1/(1 + e^{-z})$, or the hyperbolic tangent function $s(z) = (e^z - e^{-z})/(e^z + e^{-z})$ are used for this purpose. Finally, an objective function is used to minimize the error between the network's output $\hat{\mathbf{y}}$ and the desired output $\mathbf{y}$, whether the task at hand involves classification or regression. Using a training algorithm, such as backpropagation [55], and an optimizer such as Stochastic Gradient Descent (SGD) [56] for example, the parameters of the model are adjusted in a way such that the objective function of the model is optimized. Formally, the parameters θ of a neural network are updated using SGD as:

$$\theta := \theta - \eta \nabla \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}; \theta) \tag{1}$$

where $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}; \theta)$ is the objective function of the network, i.e., the sum of errors between the network's predictions and the desired outputs for all samples. A *forward-pass* of an input sample through the network involves passing each successive layer's output to the next layer as input, up until the output layer, where the gradients of the objective function are computed and used for the optimization of the network's parameters.

CNNs model relationships between data in a similar manner, however they differ from FNNs in that the parameters learned correspond to filter coefficients instead of weights. A CNN typically consists of a number of convolutional layers, where each convolutional layer is comprised of a set of such filters. The input of each such layer is convolved with all of the filters comprising the layer, and a set of 2-dimensional features, or heatmaps, is produced, corresponding to different channels. The set of heatmaps has the same cardinality as the set of filters comprising the layer, and the size of the heatmaps themselves depends on the stride and the kernel size of the filters. The filters' coefficients are learned in a similar manner as the weights in an FNN, by optimizing an objective function which defines the network's task. SGD may also be applied to the objective function of a CNN to learn its parameters.

Typically, the activation functions of convolutional layers involve Rectified Linear Units (ReLU) [57, 58], that is $s(z) = \max(0, z)$. More recently, Parametric Rectified Linear Units (PReLUs) [59], which learn the slope of the negative part of ReLUs, were proposed and shown to improve results on large image datasets. Formally, a PReLU function can be described as $s(z_i) = \max(0, z_i) + a_i \min(0, z_i)$, where z_i is the input of the activation of the i-th channel and a_i is the corresponding parameter. Thus, the use of these units allows for the training of a set of parameters different for all the channels of a layer of the network, although a channel-shared, or layer-wise, variant was also proposed. It was also shown that the set of parameters for these activation layers can be trained through backpropagation-based techniques, including SGD.

Pooling layers may also be added to CNNs [60, 61], which have been shown to enhance performance by handling small distortions. These layers typically compute an average or max value over a set of neighboring values in a produced heatmap. A CNN may exhibit some fully connected layers in an FNN fashion, and these have been typically used in classification tasks as the last layers of a model, as in [21]. A fully convolutional CNN (FCN) is one where all the learnable layers are convolutional, so it implements the last fully connected layer as convolutional with filter size equal to the input size. A fully convolutional network is therefore able to output a heatmap of its input.

It's worth mentioning that fully connected layers impose expensive space constraints, as they contain a large number of parameters [62, 63]. The number of trainable parameters for a fully connected layer is $s_{input} \times s_{output}$, where s_{input} is the size of the input and s_{output} is the size of the output of the layer. For a convolutional layer, the number of trainable parameters is $N \times d^2 \times c_{input}$, where N is the number of filters of the layer, d is the size of the filters and c_{input} is the number of channels of the layer's input, e.g., 3 for RGB images, or equal to the number of filters of the previous convolutional layer. This gives a clear advantage to fully convolutional networks, even with the computational complexity of fully connected layers being lower. Thus, recent research has been steered towards the utilization of fully convolutional networks, to make use of the multiple advantages of convolutional layers over fully connected components.

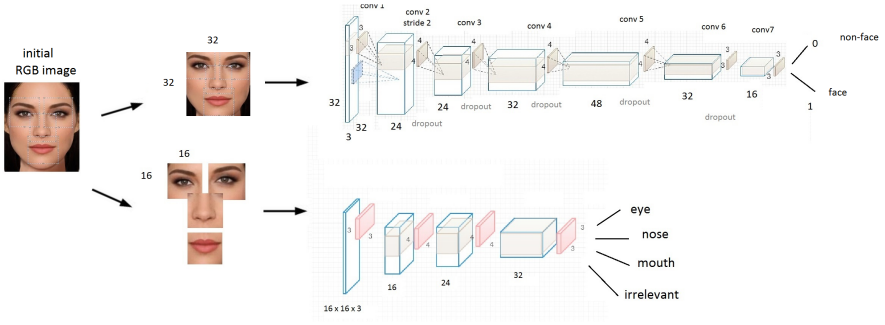


Figure 2: Top: The CNN trained for the task of full face detection. Bottom: The CNN trained for the task of facial parts detection.

3.2. CNN Architecture

Firstly, a fully-convolutional CNN comprised of seven convolutional layers was trained with RGB images of size 32×32, the architecture of which corresponds to the top part of the network shown in Figure 2. Table 1 displays the trained architecture of this network as well as the number of trainable parameters, where *conv* accompanied by an index denotes a convolutional layer and *prelu* accompanied by the same index denotes the respective activation layer. In between successive activation and convolutional layers, dropout layers are added, which have been shown to improve the performance of FNNs by allowing for better generalization [64]. A softmax function is applied to the output of the last convolutional layer, which produces probabilities for the two different detection scores, each one corresponding to the two classes of the detection task (e.g., face, non-face). Channel-wise parameters were learned for each PReLU layer. In total, the training of 76,376 free parameters is required for this network. The architecture of the network has been designed having in mind the constraint of a very fast face detector that will require minimum GPU memory and will be appropriate for large-scale visual information analysis and real-time face detection in the wild. Decisions regarding the number of the filters, the type of activations, the training procedures have been made by studying all the relevant bibliography and heavy experimentation with the large dataset that has been used for training.

Additionally, a second network consisting of four convolutional layers interspersed by three dropout layers for the task of facial parts detection was trained with RGB images of size 16×16. The output of the network is comprised of four detection scores, each one corresponding to the four classes of the facial parts (e.g., mouth, nose, eyes, irrelevant). The architecture of this CNN is also summarized in Figure 2 and Table 2. A total of 20,088 free parameters require training for this network.

The first three layers of the facial parts CNN were connected in a parallel manner to the first three layers of the second CNN as shown in Figure 3 forming

a two stream-CNN that is able in one forward pass to detect faces and facial features. The architecture and layer-wise learnable parameters are shown in Table 3. The output of the layers of the facial parts CNN, $11 \times 11 \times 24$ is concatenated with the output of the layers of the face detection CNN, $11 \times 11 \times 32$ to produce a volume of $11 \times 11 \times 56$ which is then entered as input to the conv4 layer as shown in Figure 3. The combined model is trained end-to-end with RGB images of size 32×32 and the gradients are propagated to both streams of the combined network. The 16×16 images of facial parts occupy $1/4$ of the 32×32 training face images. The intuition behind this architecture is that the information provided by the detection of local face parts is crucial for detecting face regions and should be part of the initial stages of the detection process. Training this network requires the optimization of 104,280 free parameters.

Table 1: Face detection CNN architecture.

layer	**kernel**	**filters**	**input**	**output**	**parameters**
conv1	3×3	24	$32 \times 32 \times 3$	$30 \times 30 \times 24$	648
prelu1			$30 \times 30 \times 24$	$30 \times 30 \times 24$	24
conv2	4×4	24	$30 \times 30 \times 24$	$14 \times 14 \times 24$	9216
prelu2			$14 \times 14 \times 24$	$14 \times 14 \times 24$	24
conv3	4×4	32	$14 \times 14 \times 24$	$11 \times 11 \times 32$	12288
prelu3			$11 \times 11 \times 32$	$11 \times 11 \times 32$	32
conv4	4×4	48	$11 \times 11 \times 32$	$8 \times 8 \times 48$	24576
prelu4			$8 \times 8 \times 48$	$8 \times 8 \times 48$	48
conv5	4×4	32	$8 \times 8 \times 48$	$5 \times 5 \times 32$	24576
prelu5			$5 \times 5 \times 32$	$5 \times 5 \times 32$	32
conv6	3×3	16	$5 \times 5 \times 32$	$3 \times 3 \times 16$	4608
prelu6			$3 \times 3 \times 16$	$3 \times 3 \times 16$	16
conv7	3×3	2	$3 \times 3 \times 16$	$1 \times 1 \times 2$	288

Table 2: Facial parts detection CNN architecture.

layer	**kernel**	**filters**	**input**	**output**	**parameters**
conv1	3×3	16	$16 \times 16 \times 3$	$14 \times 14 \times 16$	432
prelu1			$14 \times 14 \times 16$	$14 \times 14 \times 16$	16
conv2	4×4	24	$14 \times 14 \times 16$	$6 \times 6 \times 24$	6144
prelu2			$6 \times 6 \times 24$	$6 \times 6 \times 24$	24
conv3	4×4	32	$6 \times 6 \times 24$	$3 \times 3 \times 32$	12288
prelu3			$3 \times 3 \times 32$	$3 \times 3 \times 32$	32
conv4	3×3	4	$3 \times 3 \times 32$	$1 \times 1 \times 4$	1152

Our work follows the pipeline presented in [16], in the sense that it does not require any extra module (e.g., SVM) for classification as the CNN's output can be utilized directly for the task of face detection.

3.3. Progressive positive and hard negative example mining

As previously stated, the lightweight architecture of the proposed model establishes the need for an effective training methodology, to allow the model to accurately been trained using more than half a million training samples in an efficient manner. Intuitively, the model should learn easier positive examples

Table 3: Combined face and parts-based detection CNN architecture.

layer	kernel	filters	input	output	parameters
conv1	3×3	24	$32 \times 32 \times 3$	$30 \times 30 \times 24$	648
prelu1			$30 \times 30 \times 24$	$30 \times 30 \times 24$	24
conv2	4×4	24	$30 \times 30 \times 24$	$14 \times 14 \times 24$	9216
prelu2			$14 \times 14 \times 24$	$14 \times 14 \times 24$	24
conv3	4×4	32	$14 \times 14 \times 24$	$11 \times 11 \times 32$	12288
prelu3			$11 \times 11 \times 32$	$11 \times 11 \times 32$	32
conv1-pb	3×3	16	$32 \times 32 \times 3$	$30 \times 30 \times 16$	768
prelu1-pb			$30 \times 30 \times 16$	$14 \times 14 \times 16$	16
conv2-pb	4×4	24	$30 \times 30 \times 16$	$22 \times 22 \times 24$	3456
prelu2-pb			$22 \times 22 \times 24$	$22 \times 22 \times 24$	24
conv3-pb	4×4	32	$22 \times 22 \times 24$	$11 \times 11 \times 32$	5184
prelu3-pb			$11 \times 11 \times 32$	$11 \times 11 \times 32$	32
conv4	4×4	48	$11 \times 11 \times 56$	$8 \times 8 \times 48$	43008
prelu4			$8 \times 8 \times 48$	$8 \times 8 \times 48$	48
conv5	4×4	32	$8 \times 8 \times 48$	$5 \times 5 \times 32$	24576
prelu5			$5 \times 5 \times 32$	$5 \times 5 \times 32$	32
conv6	3×3	16	$5 \times 5 \times 32$	$3 \times 3 \times 16$	4608
prelu6			$3 \times 3 \times 16$	$3 \times 3 \times 16$	16
conv7	3×3	2	$3 \times 3 \times 16$	$1 \times 1 \times 2$	288

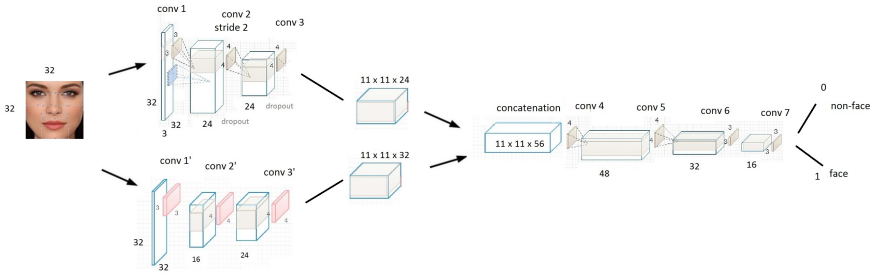


Figure 3: The combined model used for face detection during training and deployment.

first (i.e., clear frontal faces), followed by progressively harder positive examples as the training process proceeds and converges. Thus, easier examples are preferred for the training of the network at the beginning of training, followed by progressively harder ones. Frontal images of faces without any occlusions constitute easy examples in the case of face detection. Then, as the network learns to accurately detect easy examples, slightly harder ones are added to its training dataset. We call this method of adding positive examples to the training set of the network *progressive positive example mining*. The progressive learning approach that increases the difficulty of the training set is intuitively valid since this is also what humans do when they try to learn a difficult task (e.g., progressively going from elementary school teachers to university professors). This approach has been also used in other learning tasks in computer games with success [65].

We determine a positive sample's difficulty level by examining the score produced by the network for it. As the training proceeds, feeding the network with unseen face images will produce a (pseudo)probability for the existence of a face in these images. Images producing a high probability are considered as easy positive examples for the network and can be added to its training set, and iteratively repeating this process positive examples of progressing difficulty are added to the training dataset of the model.

The process of mining negative samples follows a similar intuition, but in reverse: hard negative examples must be collected in conjunction to the positive ones to avoid false positives and force the training process to differentiate between faces and examples mistaken for faces. Given some images which serve as negative examples, the network produces scores that represent the probability that these images depict faces. The higher the score, the harder the example is for the network to distinguish. Thus, such examples must be added to the training set of the network first to guide the training process. This process simulates the *hard negative sample mining* procedure.

Let $\mathcal{N}_t$ be a collection of images that will serve as a pool of negative examples, and $\mathcal{P}_t$ be a pool of positive examples, where the subscript t denotes that samples can be collected at multiple time steps during the training process. Let $\mathcal{D}_0$ be the original training set consisting of the original set of positive examples $\mathcal{P}_0$ and the set of negative examples $\mathcal{N}_0$:

$$\mathcal{D}_0 = \mathcal{P}_0 \cup \mathcal{N}_0 \tag{2}$$

Once the training process is complete, we run the network to the set of images $\mathcal{N}$ from which we collect a subset of false positives $\mathcal{F}_1$ which is added to the original set of negative examples $\mathcal{N}_0$:

$$\mathcal{N}_1 = \mathcal{N}_0 \cup \mathcal{F}_1 \tag{3}$$

The set $\mathcal{F}_1$ is selected according to the network's output. During each training round, we sort the false positives according to their score and we select a predefined number of samples. In order to maintain the same ratio of positive to negative samples after each training round we increase the number of positive examples respectively. A new set of images containing faces $\mathcal{T}_1$ is added to the original set of positive examples $\mathcal{P}_0$

$$\mathcal{P}_1 = \mathcal{P}_0 \cup \mathcal{T}_1 \tag{4}$$

The aforementioned process of training and increase of training examples is repeated after completion of training in the set D_t:

$$\mathcal{N}_{t+1} = \mathcal{N}_t \cup \mathcal{F}_{t+1} \tag{5}$$

$$\mathcal{P}_{t+1} = \mathcal{P}_t \cup \mathcal{T}_{t+1} \tag{6}$$

$$\mathcal{D}_{t+1} = \mathcal{P}_{t+1} \cup \mathcal{N}_{t+1} \tag{7}$$

Hence, the sets $\mathcal{N}_{t+1}$, $\mathcal{P}_{t+1}$ contain a larger number of negative and positive examples than the sets $\mathcal{N}_t$, $\mathcal{P}_t$.

Hard negative mining techniques have been deployed in the past [66, 67], and heuristically find hard negative examples which can be used during training to improve the classification performance. However, our methodology as described above also progressively finds and adds positive examples to the dataset, to balance the classification task and improve performance on difficult positive samples (e.g., occlusions, heavy pose variation, etc.) as well as on difficult negative samples which highly resemble faces and produce false positives.

The increasing difficulty of the described process as well as the adaptation of the training set in the neural network's errors improved the network's performance in unknown data. The process of gradual training in t stages, as described, resolves a significantly important issue which was indeed validated in practice: in the event of a training set being unequally distributed between the two classes, a training batch may contain little to no actual samples of one of the classes. As a result, the network may be deprived of the presence of samples of said class and, by extension, the ability to identify between the two classes may be negatively impacted.

3.4. Training and Deployment

The proposed training algorithm is summarized in Algorithm 1 and Figure 4. The *dataset augmentation* step corresponds to the dataset collection, for the first iteration of the proposed training scheme, and its augmentation by hard negative and progressive positive sample mining for later steps. During the *training process*, all samples are passed through the network, typically in minibatches. The network produces a probability for the face/no-face case for each sample, and finally the network's parameters are updated using SGD by comparing the produced probability to the ground truth. When the training process finishes, the filters accompanying each layer of the network have been updated so that the final layer is able to detect the presence or absence of a face in the input sample. The updated CNN parameters affect the dataset augmentation process that takes place in the next iteration.

Algorithm 1 Training process for the proposed FD-CNN

Input: Dataset of images with annotated faces $\mathcal{I}$, number of hard negative and progressive positive example mining steps T

Output: A CNN capable of face detection

Construct initial training dataset $\mathcal{D}_0$ by collecting positive and negative examples from the annotated images (Equation (2))

for $t = 0 \dots T$ **do**

Update the parameters θ of the network by forward-passing the samples in $\mathcal{D}_t$ (Equation (1))

Update $\mathcal{N}_{t+1}$ and $\mathcal{P}_{t+1}$ by collecting hard false positive and easy true positive examples respectively, and use them to construct $\mathcal{D}_{t+1}$ (Equation (7))

end for

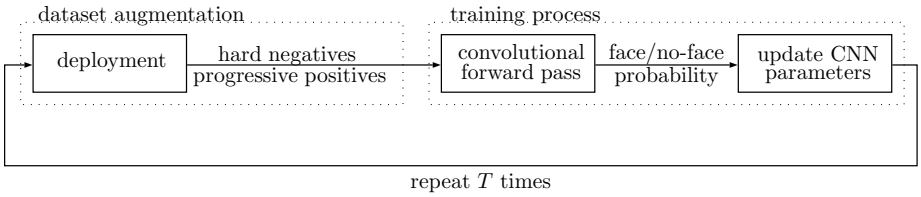


Figure 4: The training process of the proposed face detection CNNs.

During the *deployment* process, an image pyramid is produced and fed to the network, which produces a face/no-face probability distribution for each pyramid scale. For each scale, the result of the forward pass is the production a heatmap of similar size to the original input image, which indicates the presence or absence of faces for all parts of the image. Figure 5 summarizes the process of face detection during the deployment phase of the proposed CNNs. First, a spatial pyramid of the input image is created and all of the produced images are passed through the trained network. Then, a heatmap indicating the presence or absence of faces in all parts of the input image is created, for each scale of the produced pyramid. Finally, the heatmaps at all scales are combined using Non Maximum Suppression (NMS) to produce the final bounding boxes containing faces of various sizes.

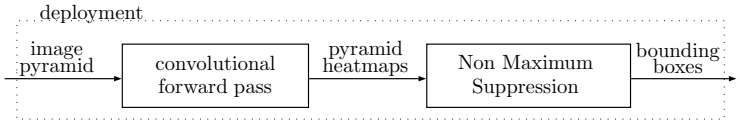


Figure 5: The deployment process of the proposed face detection CNNs.

In all our experiments we trained the CNNs using Stochastic Gradient Descent (SGD). We start with a learning rate of 0.001 for the first 200,000 iterations and then we lower to 0.0001. The parallel layers of the combined model shown in Figure 3 were initially locked as they had a fixed learning rate of zero value. After training the layers conv4 to conv7, the locked layers were unlocked to finetune and finalize the model. The weights of the network were initialized using the Xavier method [68]. Figure 6 shows the results of the described procedure for the FDDB dataset where it can be seen that the proposed progressive addition of training samples boosts the training accuracy avoiding local minima.

3.5. Training dataset

The CNN was trained with positive samples extracted from the AFLW [69], MTFL [70] and WIDER FACE [29] datasets. The first consists of 21K images with 24K face annotations while MTFL consists of 12K face annotations, and WIDER FACE contains 32K images with about 390K face annotations, with half of these intended for training. All three datasets include real world images with expression, pose, gender, age and ethnicity variations.

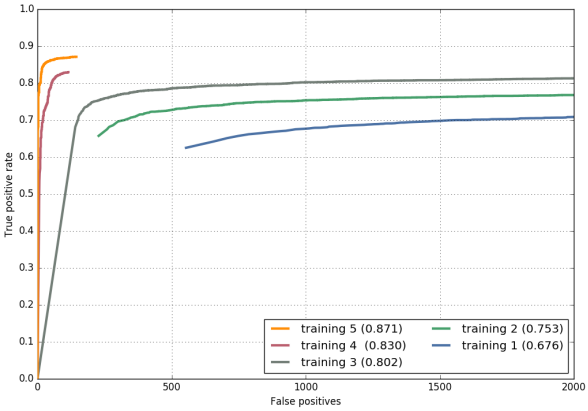
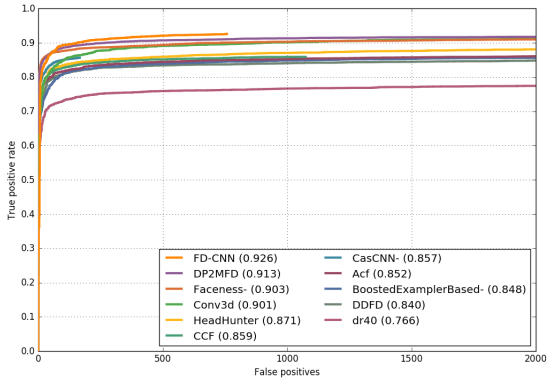


Figure 6: The results of the proposed training methodology on the FDDB dataset for the face detection CNN. A similar approach was used for the part-based CNN and the combined CNN.

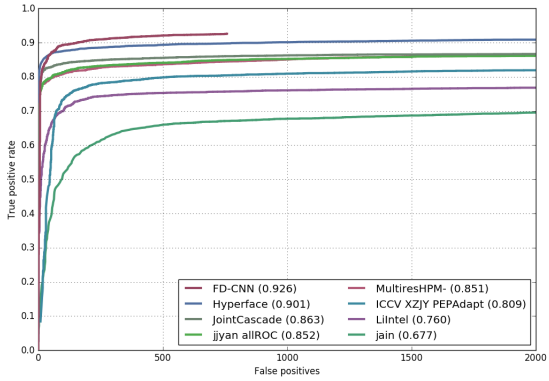
In total, 168K faces were extracted from all three datasets, as well as 156K negative examples. As both positive and negative examples are mirrored with a probability of 0.5 by our models, a total of 648K images were passed through the network and contributed to its training and convergence. The mirroring transformation proved to be quite effective, as it increased the number of training samples and also led to a large number of combinations of images to be entered in each training batch, thus driving the CNN to achieve better generalization.

The WIDER FACE database provides manually created rectangle annotations with a wide range of height to width ratio. Our CNN is designed to detect square face regions as our training examples were squares. In order to maximize the Intersection over Union (IoU) metric between the predicted bounding boxes and the ground truth boxes of the WIDER dataset a regression model was trained to predict the height to width ratio of the positive samples. This technique produced better matching results.

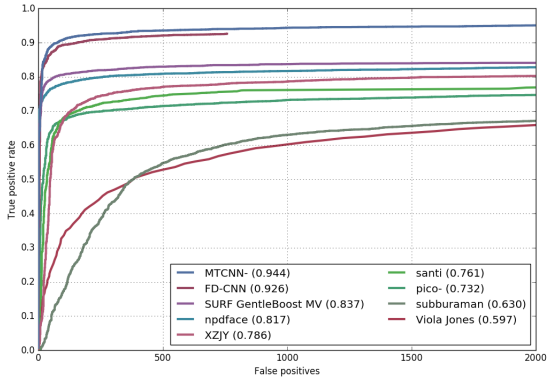
For the AFLW dataset, the provided face rectangle annotations were used. For the MTFL dataset, the given facial landmark annotations were used to produce face rectangles in a similar manner with AFLW samples regarding the positioning of faces. The final training images of faces were resized to 32×32. This is a relatively small image size compared to image sizes typically used by AlexNet and other deep networks (e.g., in [16] a fully-convolutional version of AlexNet is trained with images of size 227×227). However, it has been shown that images of this size contain enough information to train the CNN [71]. The relatively small image size allowed for the reduction of the output image down to 1×1 (a face/no-face result) without the use of pooling layers. We used only convolutional, dropout and PReLU layers, as there was no need to further decrease the training input.



(a)



(b)



(c)

Figure 7: Comparison of different face detectors on FDDB dataset.

4. Experiments

4.1. System analysis

We implemented the proposed face detector using the Caffe Deep Learning framework [72]. The output of the network corresponds to the scores of the CNN for every 32×32 window with a stride of 2 pixels in the original image. In order to detect faces smaller or larger than 32×32 we scale the original image up or down respectively. We apply the non maximum suppression strategy according to which all bounding-boxes with a possibility lower than the score of the maximum window multiplied by a constant factor are removed. The system was able to detect faces in the FDDB dataset that were not included in the annotated samples. These detections were removed as they would count for false positives and lead to a deteriorated performance.

During deployment of the CNN, we add an extra average pooling layer to the final output of the network. The addition of this layer reduces the number of false positives. The heatmap produced by the CNN is smoothened and only the pixel coordinates having values greater than a specified threshold are stored. Additionally, the heatmap pixel coordinates having neighbouring coordinates with similar values are stored resulting in reduced false positives and improved performance.

4.2. Evaluation

The proposed detector was evaluated on the challenging dataset Face Detection Data Set and Benchmark (FFDB) [16]. Some of the recently published methods compared in this Section include: DP2MFD [40], DDFD, Faceness [47], Headhunter, JointCascade [37], SURF [33], ACF [36], CCF [73] and MT-CNN [74]. For evaluation, the toolbox provided by [35] which includes corrected annotations for the aforementioned benchmark was used. FDDB dataset is one of the most commonly used benchmarks for face detection and consists of 2,845 images with 5,171 face annotations collected from journalistic articles. It is a really challenging dataset mainly due to the fact that it is rich in occluded and out-of-focus cases. FDDB faces are annotated with elliptic regions. As stated in [35] changing the output format of detections to ellipses increases the overlap region between the detections and ground truth boxes. However, our detector achieves a high recall rate without this conversion. Figure 7 shows the results on the FDDB dataset. The original face detection CNN achieves a recall rate of 88.9% on this dataset, while the combined face and parts-based detection CNN achieves a recall rate of 92.6%, outperforming many recently published face detection methods. Figure 8 shows samples of face detection in this dataset. Recently several CNNs that are based on VGG or on ResNet50 with a corresponding number of parameters exceeding 1M have been proposed to deal with small resolution faces using image pyramids. The resulting models are very slow and they cannot perform real-time face detection even when they use state-of-the-art desktop GPUs. They are usually able to perform detection at 1-3 frames per second.

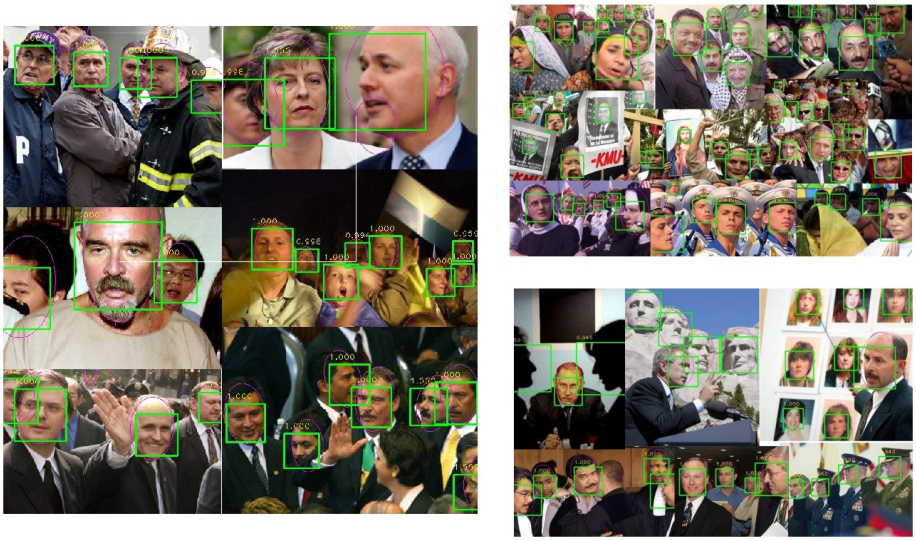


Figure 8: Face detection examples in the FDDB dataset using the proposed CNN.

The proposed detector was also evaluated on the WIDER Face Dataset [29]. The images are split into 61 event categories containing 32K images in total which depict about 393K faces. The dataset is split into training, validation and testing sets, each containing 50%, 10% and 40% respectively of the images corresponding to each event category, and half of the faces were indeed used for training as discussed in Section 3.5. The faces depicted in this dataset exhibit a very wide variety in terms of scale, pose, occlusions, facial expressions and ethnicity and include many blurred and out-of-focus faces. Figure 9 shows the precision recall curves for this dataset and Figure 10 shows an example of face detection in an aerial shot from this dataset, while Figure 11 shows examples of face detection in professional cycling events, on images obtained by the VOC2012 dataset [75]. Both Figures illustrate that the proposed face detector is capable of detecting faces of intense pose and occlusion variations.

4.3. Computational Complexity

The complexity of the compared competitive algorithms is very large in comparison to the proposed networks. Indeed, the proposed face detection and facial parts detection CNNs have 76,375 and 104,280 free parameters respectively, whereas the previously proposed deep CNN [16] had 60 million parameters. This issue is very important during training as well as during testing and deployment. The proposed lightweight model can be easily deployed to smart devices (e.g., smartphones, notepads, etc.) or robotic systems (e.g., drones) that do not have expensive and energy consuming multiple GPUs installed. Additionally, the proposed approach proves that when we have to deal with a specific task (i.e. face detection), even if it is very complex, we can design and train

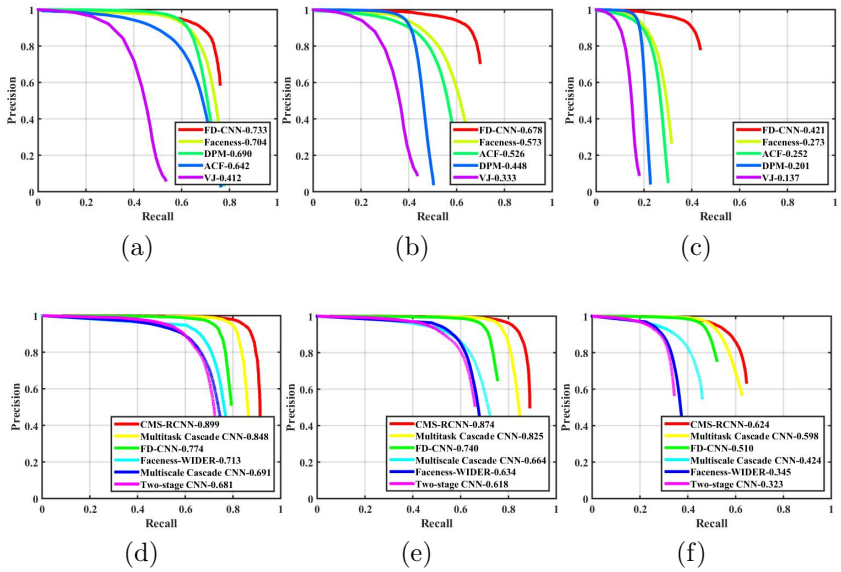
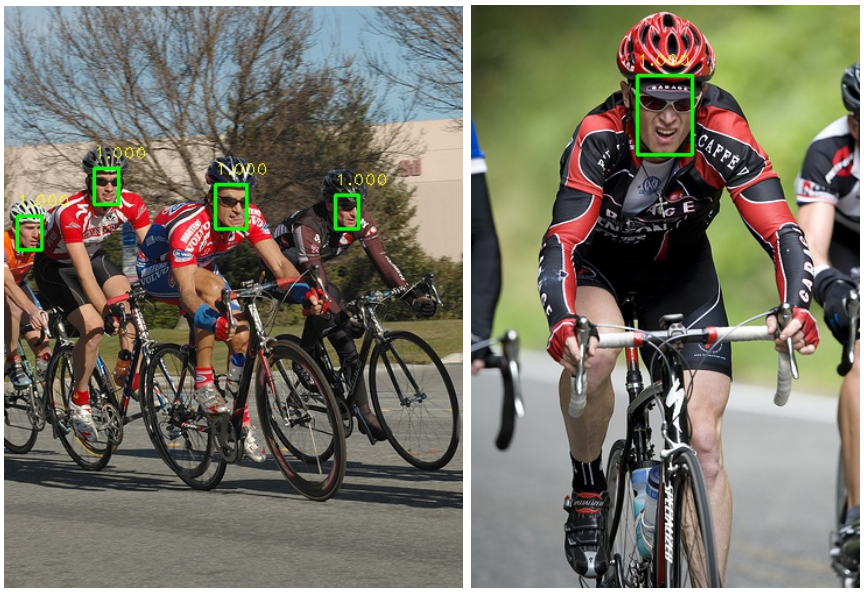


Figure 9: Comparison of different face detectors on WIDER dataset. The first row shows results achieved by using MTLF and ALFW training data, while the second row shows results achieved using the WIDER FACE dataset training partition for the training process.

Figure 10: Face detection example of an aerial shot from the WIDER FACE dataset with scores produced by the proposed CNN.



(a) (b)

Figure 11: Examples of detection on professional cyclists' faces, on images from the VOC2012 dataset [75].

smaller and efficient architectures that outperform deeper and larger networks in performance and in execution time.

Formally, the computational complexity of a convolution with N filters of size $d \times d$ on a single channel input is $\mathcal{O}(d^2 N h w)$, where h and w are the height and width of the produced heatmap. It's worth noting that the size of the produced heatmap is affected by both the filter size and the length of the strides taken during the convolution. Moreover, the complexity of a convolutional layer is multiplied by the number of channels of its input. For a forward pass through a convolutional neural network, this complexity adds up for each convolutional layer with the heatmap dimensions decreasing.

This gives the proposed models a direct computational advantage over more complex networks with deeper architectures, i.e., a larger number of layers. Furthermore, as the proposed models are trained using small $32 \times 32 \times 3$ input images, the produced heatmaps' sizes are kept to a minimum, in contrast to related methods which use larger inputs. During deployment, the relatively small number of channels per convolutional layer in combination with the fewer layers in comparison to related methods, are what give the proposed models a computational advantage, making them very fast, and suitable for deployment on mobile applications.

4.4. Time Performance

Table 4 summarizes floating point operations and time measurements for our models (denoted by FD-CNN and PBD-CNN for the face detection and the parts-based detection networks respectively) as well as for the architectures proposed in [16] (denoted by DDFD) and in [45] (denoted by VGG16 R-CNN). More specifically, the floating point operations and execution time (in seconds) required for the forward pass of an RGB image of size 227×227 are compared. The last column shows the time required for a forward pass of the original image. Our models are trained to detect faces of size 32×32. In order to detect faces in different sizes the original image is scaled up and down. However, it should also be noted that the DDFD detector is able to detect faces of size 227×227, and would also require multiple passes of resized images in order to detect all faces in the original image. The time required for a forward pass for our models is about 6 to 7 times less than the time required for one forward pass of the original image in DDFD even when the input image size that is used is the one that is more appropriate for DDFD. When the face size we want to detect is smaller than 227×227 the proposed model is one to two orders of magnitude faster than the competitive ones. Finally, the last row corresponding to the VGG16 architecture serves to show that larger models are impractical when it comes to real time applications.

Table 4: Execution times and FLOPs comparison between the proposed CNNs and other neural network face detectors.

Network	**FLOPs**	**Time**
FD-CNN	865M	0.007575
PBD-CNN	1.1B	0.010429
DDFD	224B	0.049671
VGG16 R-CNN	634B	0.138781

Our model was also tested in an application where images from a video camera were given as input to detect faces from. Using an NVIDIA GTX 1080 graphics card, our model was able to detect faces at a rate of 31 frames per second, for an input image size of 320×240, making it suitable for use in real-time applications.

4.5. Discussion

We have shown both in theory and experimentally the time-wise advantage of the proposed methods, which makes the task of face detection very *fast*. The low number of floating point operations required, along with the speedup provided by even low-end GPUs, allow for the deployment of the models on UAVs with limited computational capabilities.

Moreover, the proposed models achieve results comparable to those achieved by more complex and computationally expensive models, making very *accurate* face detections. The pyramid scheme used enables the proposed models to accurately detect faces of various sizes. The proposed training scheme, consisting of

multiple steps of hard negative and progressive positive sample mining, allows the models to accurately detect faces under severe pose, occlusion and other variations. In combination with the readily available vast volume of annotated datasets, we have shown that a lightweight architecture may achieve competitive results, given an appropriate training procedure.

In conclusion, the proposed methods are at once lightweight enough to be deployed on mobile applications and accurate enough to be comparable to state-of-the-art face detection methods, thus offering fast and accurate face detection making them suitable for deployment for drone-assisted intelligent video shooting.

5. Conclusion

In this paper, a novel fast deep convolutional neural network architecture was presented for the task of large scale and real time face detection in the wild. The experiments on publicly available benchmarks show the success of the proposed method. The presented detector is able to recognize faces in a wide range of orientations and expressions. It does not require any extra modules usually used in deep learning methods such as SVM or bounding-box regression. Our work, extends previously published detectors by using a lightweight model that improves run time and training speed. Additionally, the proposed model combines outputs of two different networks trained for face detection and local facial parts. It also outperforms the DDFD detector in the challenging FDDB dataset by a magnitude of 8%. We show that a properly trained smaller model is efficient and outperforms a more complex and large network used for the same task. Last but not least, our method is suitable for real-time face detection and can be deployed to smart devices and robotic systems, e.g. for intelligent video shooting purposes using UAVs.

Acknowledgments

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] X. Jin, B. W. Wah, X. Cheng, Y. Wang, Significance and challenges of big data research, Big Data Research 2 (2) (2015) 59–64.

[2] E. Kohlwey, A. Sussman, J. Trost, A. Maurer, Leveraging the cloud for big data biometrics: Meeting the performance requirements of the next generation biometric systems, in: 2011 IEEE World Congress on Services, IEEE, 2011, pp. 597–601.

[3] N. Ratha, J. Connell, S. Pankanti, Big data approach to biometric-based identity analytics, IBM Journal of Research and Development 59 (2/3) (2015) 4–1.

[4] G. Goudelis, A. Tefas, I. Pitas, Emerging biometric modalities: a survey, Journal on Multimodal User Interfaces 2 (3) (2008) 217–235.

[5] D. Triantafyllidou, A. Tefas, A fast deep convolutional neural network for face detection in big visual data, in: INNS Conference on Big Data, Springer, 2016, pp. 61–70.

[6] E. Marami, A. Tefas, Using particle swarm optimization for scaling and rotation invariant face detection, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–7.

[7] E. Marami, A. Tefas, Face detection using particle swarm optimization and support vector machines, in: Hellenic Conference on Artificial Intelligence, Springer, 2010, pp. 369–374.

[8] C. Kotropoulos, A. Tefas, I. Pitas, Frontal face authentication using variants of dynamic link matching based on mathematical morphology, in: Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on, Vol. 1, IEEE, 1998, pp. 122–126.

[9] J. Ren, X. Jiang, J. Yuan, A complete and fully automated face verification system on mobile devices, Pattern Recognition 46 (1) (2013) 45–56.

[10] H.-J. Hsu, K.-T. Chen, Face recognition on drones: Issues and limitations, in: Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use, ACM, 2015, pp. 39–44.

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

[12] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, IEEE, 2005, pp. 886–893.

[13] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Computer vision and image understanding 110 (3) (2008) 346–359.

[14] P. Dollár, Z. Tu, P. Perona, S. Belongie, Integral channel features.

[15] X. Zhu, D. Ramanan, Face detection, pose estimation, and landmark localization in the wild, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 2879–2886.

[16] S. S. Farfade, M. J. Saberian, L.-J. Li, Multi-view face detection using deep convolutional neural networks, in: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, ACM, 2015, pp. 643–650.

[17] H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, A convolutional neural network cascade for face detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5325–5334.

[18] C. Zhang, Z. Zhang, Improving multiview face detection with multi-task deep convolutional neural networks, in: IEEE Winter Conference on Applications of Computer Vision, IEEE, 2014, pp. 1036–1041.

[19] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 160–167.

[20] L. Deng, G. Hinton, B. Kingsbury, New types of deep neural network learning for speech recognition and related applications: An overview, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 8599–8603.

[21] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[22] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 3642–3649.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[24] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556.

[25] A. Van den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, in: Advances in Neural Information Processing Systems, 2013, pp. 2643–2651.

[26] X.-W. Chen, X. Lin, Big data deep learning: challenges and perspectives, IEEE Access 2 (2014) 514–525.

[27] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic, Deep learning applications and challenges in big data analytics, Journal of Big Data 2 (1) (2015) 1.

[28] V. Jain, E. Learned-Miller, Fddb: A benchmark for face detection in unconstrained settings, Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst (2010).

[29] S. Yang, P. Luo, C. C. Loy, X. Tang, Wider face: A face detection benchmark, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[30] B. Wu, H. Ai, C. Huang, S. Lao, Fast rotation invariant multi-view face detection based on real adaboost, in: Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on, IEEE, 2004, pp. 79–84.

[31] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, H. Shum, Statistical learning of multi-view face detection, in: European Conference on Computer Vision, Springer, 2002, pp. 67–81.

[32] M. Jones, P. Viola, Fast multi-view face detection, Mitsubishi Electric Research Lab TR-20003-96 3 (2003) 14.

[33] J. Li, Y. Zhang, Learning surf cascade for fast and accurate object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 3468–3475.

[34] B. Jun, I. Choi, D. Kim, Local transform features and hybridization for accurate face and human detection, IEEE transactions on pattern analysis and machine intelligence 35 (6) (2013) 1423–1436.

[35] M. Mathias, R. Benenson, M. Pedersoli, L. Van Gool, Face detection without bells and whistles, in: European Conference on Computer Vision, Springer, 2014, pp. 720–735.

[36] B. Yang, J. Yan, Z. Lei, S. Z. Li, Aggregate channel features for multi-view face detection, in: Biometrics (IJCB), 2014 IEEE International Joint Conference on, IEEE, 2014, pp. 1–8.

[37] D. Chen, S. Ren, Y. Wei, X. Cao, J. Sun, Joint cascade face detection and alignment, in: European Conference on Computer Vision, Springer, 2014, pp. 109–122.

[38] P. Felzenszwalb, D. McAllester, D. Ramanan, A discriminatively trained, multiscale, deformable part model, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[39] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, Cascade object detection with deformable part models, in: Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, IEEE, 2010, pp. 2241–2248.

[40] R. Ranjan, V. M. Patel, R. Chellappa, A deep pyramid deformable part model for face detection, in: Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on, IEEE, 2015, pp. 1–8.

[41] R. Ranjan, V. M. Patel, R. Chellappa, Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition, arXiv preprint arXiv:1603.01249.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision (IJCV) 115 (3) (2015) 211–252. doi:10.1007/s11263-015-0816-y.

[43] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[44] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[45] H. Jiang, E. Learned-Miller, Face detection with the faster r-cnn, arXiv preprint arXiv:1606.03473.

[46] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, 2015, pp. 91–99.

[47] S. Yang, P. Luo, C.-C. Loy, X. Tang, From facial parts responses to face detection: A deep learning approach, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3676–3684.

[48] S. Zafeiriou, C. Zhang, Z. Zhang, A survey on face detection in the wild: past, present and future, Computer Vision and Image Understanding 138 (2015) 1–24.

[49] P. Nousi, A. Tefas, Deep learning algorithms for discriminant autoencoding, Neurocomputing.

[50] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Handwritten digit recognition with a back-propagation network, in: Advances in neural information processing systems, Citeseer, 1990.

[51] S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back, Face recognition: A convolutional neural-network approach, IEEE transactions on neural networks 8 (1) (1997) 98–113.

[52] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (5) (1989) 359–366.

[53] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural networks 2 (3) (1989) 183–192.

[54] Y. Bengio, Learning deep architectures for ai, Foundations and trends® in Machine Learning 2 (1) (2009) 1–127.

[55] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551.

[56] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[57] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807–814.

[58] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks., in: Aistats, Vol. 15, 2011, p. 275.

[59] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[60] K. Jarrett, K. Kavukcuoglu, Y. Lecun, et al., What is the best multi-stage architecture for object recognition?, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 2146–2153.

[61] Y. LeCun, K. Kavukcuoglu, C. Farabet, et al., Convolutional networks and applications in vision., in: ISCAS, 2010, pp. 253–256.

[62] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, S.-F. Chang, An exploration of parameter redundancy in deep networks with circulant projections, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2857–2865.

[63] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[64] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[65] C. Athanasiadis, D. Galanopoulos, A. Tefas, Progressive neural network training for the open racing car simulator, in: 2012 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, 2012, pp. 116–123.

[66] A. Shrivastava, A. Gupta, R. Girshick, Training region-based object detectors with online hard example mining, arXiv preprint arXiv:1604.03540.

[67] J. F. Henriques, J. Carreira, R. Caseiro, J. Batista, Beyond hard negative mining: Efficient detector learning via block-circulant decomposition, in: proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2760–2767.

[68] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks., in: Aistats, Vol. 9, 2010, pp. 249–256.

[69] M. Köstinger, P. Wohlhart, P. M. Roth, H. Bischof, Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization, in: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 2144–2151.

[70] Z. Zhang, P. Luo, C. C. Loy, X. Tang, Facial landmark detection by deep multi-task learning, in: European Conference on Computer Vision, Springer, 2014, pp. 94–108.

[71] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net, arXiv preprint arXiv:1412.6806.

[72] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM international conference on Multimedia, ACM, 2014, pp. 675–678.

[73] B. Yang, J. Yan, Z. Lei, S. Z. Li, Convolutional channel features, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 82–90.

[74] Z. L. Kaipeng Zhang, Zhanpeng Zhang, Joint face detection and alignment using multi-task cascaded convolutional networks, IEEE Signal Processing Letters 23 (10) (2016) 1499–1503.

[75] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.