Machine Vision and Applications (2022) 33:3
https://doi.org/10.1007/s00138-021-01245-y

SPECIAL ISSUE PAPER

Semantic convolutional features for face detection

The-Anh Pham[1]

Received: 13 April 2021 / Revised: 13 July 2021 / Accepted: 10 August 2021 / Published online: 30 October 2021

Abstract
Convolutional neural networks have been extensively used as the key role to address many computer vision applications. Traditionally, learning convolutional features is performed in a hierarchical manner along the dimension of network depth to create multi-scale feature maps. As a result, strong semantic features are derived at the top-level layers only. This paper proposes a novel feature pyramid fashion to produce semantic features at all levels of the network for specially addressing the problem of face detection. Particularly, a Semantic Convolutional Box (SCBox) is presented by merging the features from different layers in a bottom-up fashion. The proposed lightweight detector is stacked of alternating SCBox and Inception residual modules to learn the visual features in both the dimensions of network depth and width. In addition, the newly introduced objective functions (e.g., focal and CIoU losses) are incorporated to effectively address the problem of unbalanced data, resulting in stable training. The proposed model has been validated on the standard benchmarks FDDB and WIDER FACES, in comparison with the state-of-the-art methods. Experiments showed promising results in terms of both processing time and detection accuracy. For instance, the proposed network achieves an average precision of 96.8% on FDDB, 82.4% on WIDER FACES, and gains an inference speed of 106 FPS on a moderate GPU configuration or 20 FPS on a CPU machine.

Keywords Face detection · Feature enhancement · Feature pyramid network

1 Introduction

Face detection is an interesting problem in the domain of computer vision due to its various helpful applications in real life such as face verification, face recognition, people tracking, and smart advertising. Face detectors can be reformulated as single class object detection, and thus, a number of general object detectors can be applied, including R-CNN [1], Fast R-CNN [2], Faster R-CNN [3], YOLO [4], SSD [5].

A modern object detector (i.e., using deep convolutional neural networks) is generally composed of the following components: a backbone to learn multi-scale features, a neck to enhance the learned features and a head to perform object prediction and classification. In recent years, a number of attempts have been presented to improve each of these components in different ways. High-performance detectors often favor very deep backbones such as ResNet-101 [6] and Darknet53 [7]. To further improve the feature discrimination, one can integrate additional feature enhancement blocks (e.g., RFB [8], Inception [9], Inception-ResNet [10]), and/or attach appropriate neck networks, typically FPN [11], PAN [12], and BiFPN [13]. Besides, the newly introduced loss functions (FocalLoss [14], IoU [15], GIoU [16], CIoU [17]) have been designed to achieve stable and fast training, as well as to improve the classification and regression abilities.

In the specific area of face detection, the state-of-the-art methods (e.g., [18–22]) are capable of producing high detection accuracy on the standard benchmarks FDDB [23] and WIDER FACES [24]. However, these methods are subjected to computationally intensive complexity and require powerful GPU computer systems for inference, making them difficult to time-critical application deployment. Recently, some attempts (e.g., [25–27]) have been conducted to design real-time face detectors. Nonetheless, the obtained results are still limited in the means of either detection accuracy or inference speed.

In the present work, we introduce and exploit a set of enhanced features to solve the problem of efficient yet accuracy face detection, targeting to real-time deployment on conventional CPU environments. Particularly, the Semantic Convolutional Box (SCBox), that combines visual information between the current layer and the upper layer of a convolutional neural network (CNN), is presented to enhance

✉ The-Anh Pham
phamtheanh@hdu.edu.vn

[1] Hong Duc University (HDU), Thanh Hoa city, Vietnam

the feature discrimination ability at each level of the network. This strategy is opposed to traditional techniques, such as Feature Pyramid Networks (FPN [11]) and PyramidBox [18], where feature enhancement is applied to the detection scales only. As a result, the feature maps learned at the lower layers are not sufficiently discriminated to detect small faces. To further enrich the visual features, the Inception residual structure is incorporated that expands the backbone in the dimension of network width. Besides, the newly introduced loss functions, FocalLoss [14] and CIoU [17] are integrated to strengthen the learning performance. The proposed detector has been validated on the standard datasets, FDDB [23] and WIDER FACES [24], showing promising performance in comparison with other methods. For clarity, the main contribution of the present work can be summarized as follows:

- We propose a novel feature merging method, namely Semantic Convolutional Box, to learn more discriminative features at each scale of the network.
- The network backbone is enhanced by integrating the Inception residual structure to learn useful features for object detection.
- Advanced loss functions (e.g., Smooth L1, CIoU, FocalLoss) are incorporated to achieve stable learning performance.
- We provide detailed analysis and evaluation of the proposed detector on the aspects of computational complexity, stability, parameter tuning, and empirical results.
- We achieve promising detection performance in terms of both accuracy and running times on the standard benchmarks.

The remaining of this paper is structured as follows. Section 2 reviews the state-of-the-art methods for face detection, specifically focusing on deep learning methods and evolutionary computation. Section 3 presents the proposed face detection algorithm. Section 4 is dedicated to experiments and discussion. Section 5 concludes the paper and identifies potential follow-up works.

2 Related work

In this section, we shall select and review the representative works for face detection. The revised methods are classified into three groups: heavy-weight CNN detectors, real-time CNN methods, and soft algorithms based on evolutionary computation. The main characteristics of each method are summarized in Table 3. To have a degree of detection performance, we shall mention the accuracy in terms of true positive rate (recall rate) measured at 1000 false positives on the standard benchmark FDDB [23]. Besides, the inference time shall be provided, if any, in terms of frames per second (FPS) for VGA-resolution images on CPU configuration, unless otherwise stated. For the purpose of presentation, the main abbreviations and variables used in the present work are described in Tables 1 and 2, respectively.

Table 1 The main abbreviations used in the paper

Abbreviation	Description
SCBox (or B)	Semantic Convolutional Box
Conv (or C)	A convolutional layer
R	An inception residual module
FLOPs	Floating point operations
MFLOPs	Million FLOPs
BFLOPs	Billion FLOPs
AP	Average precision
RF	Receptive field
IoU	Intersection over union (Jaccard score)
CNN	Convolutional neural network
NA	Not available

Table 2 The main variables used in the paper

Variable	Description
L_{loc}	Localization loss function
L_{cls}	Classification loss function
CE	Cross-entropy function
FL	Focal loss function
CIoU	Complete IoU loss function
GIoU	Generalized IoU loss function
$\gamma \geq 0$	A focusing parameter used in FL
$\alpha \in [0, 1]$	A balancing factor used in FL
C_{in}	The number of input channels
C_{out}	The number of output channels
W, H	The spatial size of an output feature map
F	The kernel width or height (symmetric)
$w \times h \times c$	A tensor shape of width, height, depth
s2	The stride of 2 (symmetric) applied to a filter
n_1, n_2	The weight parameters of CIoU loss

2.1 Heavy-weight CNN face detectors

We first review the category of detectors that yield high performance but incur intensively computational cost, and hence, a high-end GPU model is required for the inference phase. These methods [18–22] often conduct a significant investigation on feature enhancement task.

The authors in [19] introduced a Single Shot Scale-Invariant Face Detector (SFD) that is composed of a backbone derived from the VGG16 [37] network. To handle small face detection, the authors presented an equal-proportion

Table 3 Characterization of methods for face detection

Approach	Methods	Backbone	Additional features	Loss functions	Recall
Heavy-weight CNNs	SFD [19]	VGG16	Adapted scale + maxout	Multi-task loss	98.3%
	PyramidBox [18]	VGG16	FPN + CPM + Maxout	Multi-task loss	98.6%
	FANet [20]	VGG16	Feature Agglomeration	Hierarchical loss	98.3%
	DSFD [21]	VGG16	FPN + RFB	Progressive anchor loss	99.0%
	RefineFace [22]	ResNet	STC + STR + FSM + RFE	Scale-aware margin loss	99.1%
Real-time CNNs	CNN cascade [28]	AlexNet	Multi-resolution CNN cascade	Multinomial regression	85.7%
	MTCNN [29]	AlexNet	Image pyramid, CNN cascade	L2 + Cross-entropy	94.3%
	DDFD [30]	AlexNet	Sliding window	Softmax loss	66.9%
	Faceness [31]	AlexNet	Facial CNNs, Faceness scores	Cross-Entropy	91.0%
	FaceBoxes [25]	RDCL MSCL	CReLU, Anchor densification inception structure	Softmax + Smooth L1	95.9%
	UnitBox [15]	VGG16	Face confidence heatmap	IoU loss + Cross-entropy	95.1%
	YOLO-face [27]	Darknet	FPN + Deeper darknet	GIoU loss	95.8%
Soft computation	[32]	NA	GA, Face template	Similarity function	NA
	[33]	3 layers	GA, Gradient-based learning	L2 + Cross-Entropy	NA
	[34]	NA	Haar filters, DE+GA+PSO	AdaBoost loss	96.0%
	[35]	NA	Joint integral histogram, GA	AdaBoost, ϵ-constraint	NA
	[36]	YOLOv2	GA for data augmentation	Difference of confidence	NA

interval principle for anchor titling that determines the anchor scale and interval based on the stride size associated with a given prediction layer. The Maxout method is introduced to handle the class imbalance problem. Experimental results showed a good recall rate of 0.983 on FDDB at the cost of high inference time (e.g., 36 FPS on a single GPU Titan X). Similarly, PyramidBox [18] applies VGG16 as its backbone, Maxout, and presents a new fashion of feature enhancement. Specifically, FPN [11] is applied from the middle layers instead of the top layer to enrich the features learned from the backbone. Besides, a Context-sensitive Predict Module (CPM) has been introduced to predict the faces in accordance with the heads and bodies. PyramidBox achieves a recall rate of 0.986 on FDDB.

FANet [20] (Feature Agglomeration Networks) describes an agglomeration connection (A-block) to enhance the feature maps at the prediction layers. The A-block consists of an Inception-like module [9] and a FPN-like module [11] for merging features of the current layer and the upper layer. An hierarchical agglomeration structure is presented to create three levels of feature enhancement, each of which is associated with a loss layer, yielding a hierarchical loss for the entire model. FANet obtains a recall rate of 0.983 at the speed of 35.6 FPS on a GTX 1080Ti. Similarly, DSFD [21] (Dual Shot Face Detector) introduces a Feature Enhance Module (FEM) that combines FPN and RFB [8] to make the features more discriminated and robust. Besides, to handle multi-scale face detection, a progressive anchor loss is presented that is principally similar to the hierarchical loss. DSFD yields a promising accuracy of 0.99 on FDDB with the speed of 22 FPS on a GPU P40.

RefineFace [22] builds a backbone based on ResNet model [6] with several improvements. First, the authors applied selective two-step classification (STC) to filter out most simple negative samples and employed selective two-step regression (STR) to yield better prediction of bounding boxes. Second, a Receptive Field Enhancement (RFE) module is designed to diversify the shape and size of receptive fields. Third, the authors employed the margin-based loss function that adds an extra margin to enhance the classification ability. At last, a feature supervision module (FSM) is added to the backbone that uses RoIAlign [38] for better fitting of target boxes. RefineFace achieves a recall rate of 0.991 around the speed of 28.5 FPS (with GTX 1080Ti).

2.2 Real-time CNN face detectors

This part is dedicated to the group of methods that have been specifically designed to work in a real-time speed on CPU environments. The authors in [28] presented a CNN cascade that consists of three classification CNNs for filtering out most of the detection windows and three calibration CNNs for adjusting the sizes and locations of the remaining bounding boxes. The detector can perform around 14 FPS on a CPU machine with the recall rate of 0.857 on FDDB. Another cascaded framework [29] (MTCNN), consisting of three CNNs (P-Net, R-Net, O-Net), has been presented to exploit the semantic correlation between face detection and alignment. An image pyramid is first created and fed into P-Net for generating the proposals. Next, R-Net refines the obtained candidates. At last, O-Net performs final verification to output face boxes and landmarks. MTCNN gives the recall rate of 0.9435 but is slower than cascade CNN [28] for inference time probably due to the use of image pyramid framework.

In [30] a Deep Dense Face Detector (DDFD) has been constructed by fine-tuning the AlexNet [39] accompanying with a sliding window strategy to perform face detection. The method is very efficient due to its simplicity but gives low recall rate (e.g., 0.669 on FDDB). In [31], the authors presented five CNNs for capturing different facial parts. The facial responses are combined to compute the faceness score for a given candidate window. Faceness can achieve a recall rate of 0.91 on FDDB, but the inference speed is relatively high.

Recently, a noticeable real-time face detector, FaceBoxes [25], has been presented. Basically, FaceBoxes consists of two parts: Rapidly Digested Convolutional Layers (RDCL) and Multiple Scale Convolutional Layers (MSCL). The key part of the RDCL is the inclusion of the CReLU activation function [40] to reduce the number of output channels, enabling real-time computation of the whole network. As for the MSCL, Inception modules [9] are incorporated to enrich the visual features in the dimension of network width. In addition, a novel anchor densification strategy is presented to improve the prediction of small faces. FaceBoxes is able to achieve real-time speed on CPU (e.g., 20 FPS on an Intel Xeon E5-2660v3@2.60GHz) with reasonable recall rate (e.g., 0.959 on FDDB). An improvement of this work has been presented in [26] where the FPN is incorporated as the neck of the detector and a Divide and Conquer Head (DCH) is presented to separately predict the different scales of the anchors. Despite the inclusion of more computational blocks compared with the work in [25], the method still achieves better inference speed (e.g., 26 FPS) while obtaining higher recall rate (e.g., 0.965).

Differing from the afore-mentioned methods, UnitBox [15] employs VGG16 [37] as its backbone and develops a new loss function, namely IoU loss, for accurate bounding box regression. The IoU loss function takes into account the correlation of four points of a bounding box representing a face. In addition, UnitBox builds a confidence heatmap inferring whether a pixel belongs to a face (positive) or not (negative) and applies bounding box regression at all positive pixels. UnitBox achieves a recall rate of 0.951 on FDDB with the inference speed of about 10 FPS. However, the IoU

loss is ineffective when the intersection of the target box and the prediction box is empty. To solve this problem, the work in [27] integrates GIoU loss [16] into YOLOv3 [7] network, that is a generic object detection algorithm, resulting in the YOLO-face method. The newly introduced face detector gains a recall rate of around 0.958 on FDDB at the speed of 45 FPS (on a GPU 1080Ti).

2.3 Soft methods for face detection

In this part, we review soft computing methods that employ evolutionary algorithms for face detection [32–36].

Early works using genetic algorithm (GA) [32,33] for face detection have included some heuristic techniques for image enhancement such as size normalization, noise filtering, histogram equalization, or gray-scale conversion. In [32], the authors applied GA to find the faces of a given image by using a face template. Each chromosome encodes the parameters of a searching window and is assigned with a fitness value calculated as the similarity between the face template and the sub-image at the searching location. The work in [33] describes a hybrid method for predicting the label (i.e., face or non-face) of a given image window. Given a set of chromosomes, each represented by a shallow network structure (i.e., 3-layer CNN), the goal is to optimize the networks by reducing the number of hidden nodes without loss of accuracy. The hybrid evolutionary process applies GA and gradient-based optimization to create new offspring. The evolved solution improves the classification speed up to 30% while achieving the same level of accuracy as the expert-designed architecture.

There are several works combining GA, Haar-based features [41], and a machine learning algorithm (i.e., AdaBoost) for face detection [34,35]. In [34], three evolutionary methods are employed for generating the novel Haar features, including: differential evolution (DE), genetic algorithm (GA), and particle swarm optimization (PSO). These methods employ the same fitness function that is computed as the error of the AdaBoost classifier on the generated Haar features. The experiments showed that the DE method yields the best results with the detection precision of 96.01% on FDDB given that the number of false positives is 152. The work in [35] proposes using joint integral histogram to extract advanced features and combining GA and AdaBoost, so-called GAdaBoost, to select the best features for face detection. Here the ϵ-constraint-based GA [42] is employed to guide the training of each strong classifier and the cascade as well. Experimental results showed that GAdaBoost is capable of reducing the number of features for each stage and improving the detection performance when comparing to the conventional AdaBoost.

Recently, the authors in [36] applied a genetic algorithm for data augmentation to improve the performance of a YOLOv2-based face detector [43]. The main goal is to create new face instances by using a crossover operator that exchanges facial parts (i.e., eyebrow, eye, nose, and mouth) from different faces. The evolutionary process is driven by a fitness function that employs a face detector to find new faces that have not been explored or detected so far. As a result, the GA-based augmentation process tends to enrich the training data and to exploits the vulnerabilities of the detector. A significant improvement in detection precision (i.e., up to 30%) has been reported for the FDDB dataset.

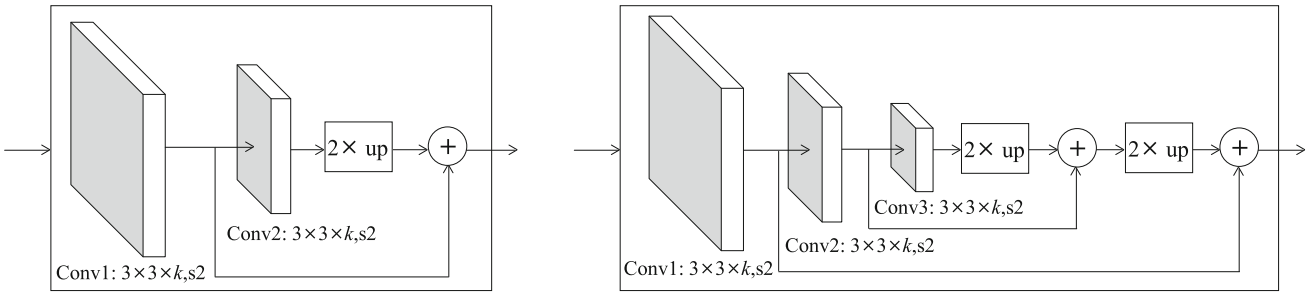
3 The proposed approach

In this section, we describe the proposed approach for handling the problem of face detection. We firstly introduce an advanced feature enhancement module, namely Semantic Convolutional Box, then present our network architecture. Finally, we provide detailed analysis of the loss functions, computational evaluation of the model, and training settings.

3.1 Semantic Convolutional Box

Traditionally, the backbone of a convolutional neural network (CNN) is composed of alternating convolutional layers and max-pooling layers. When designing such a backbone, a number of issues have been identified. As for the max-pooling, it is commonly agreed that this structure causes the visual content lost and hence may be a source of degradation in recognition performance [44]. Besides, the feature maps derived at the upper layers are semantically stronger yet coarser in the means of spatial resolution. On the other hand, the features learned at the lower layers are mainly concerned with facial cues (e.g., low-level information) and thus are subjected to poorly visual discrimination. To handle these issues, some advanced models (e.g., FPN-based structures [11,18–20]) propose combining the features created at the upper layers and those at the lower ones. However, such methods have been designed to improve the feature maps at the detection scales only.

In the present work, we propose to enrich the features at every layer of the network by substituting the conventionally convolutional layer for a more distinctive structure, namely Semantic Convolutional Box (SCBox). Conceptually, a SCBox combines the visual content at the current layer and the upper layers that is opposed to FPN [11] where the features at the top layers are embedded into the lower layers. Figure 1 presents several SCBox structures, each of which is parameterized by the number of fusion layers. A SCBox consists of a sequence of convolutional layers and produces the output by merging the feature maps derived at these layers in a bottom-up manner.



(a) SCBox(k, 2) **(b)** SCBox(k, 3)

Fig. 1 SCBox structures: applying feature fusion from 2 layers (**a**) and 3 layers (**b**). The convolutional layers of each SCBox have the same number of kernels (e.g., k) with filter size of 3 × 3 and stride of 2. The 2× upsampling operator is done with bilinear, and feature fusion is processed in the element-wise addition fashion. More convolutional layers may be added to generate more contextual features, if necessary

It is worth noting that we have imposed the same number of kernels on all the convolutional layers inside a specific SCBox to facilitate the fusion process. For instance, Fig. 1a describes SCBox(k,2) consisting of two convolutional layers having k kernels. The feature map at the upper layer is first undergone a 2× bilinear operation, then added to the feature map at the lower layer to produce the final output. Figure 1b creates a SCBox(k,3) in the same manner but with a deeper network. As such, SCBox can adaptively control the level of feature fusion, avoiding the problem of spatial size misalignment between the receptive field and the anchor box. In other words, SCBox offers a better fashion to combine the semantic features and the facial ones. As a result, the feature map at every layer is contextually enhanced while being more accurate in the means of spatial resolution, especially helpful for small face detection.

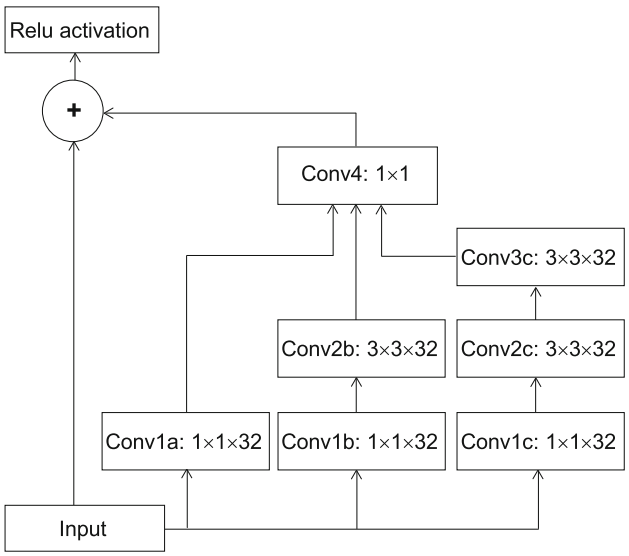


Fig. 2 The structure of Inception-ResNet-A block [10]. All the convolutional layers use the stride of 1

3.2 Network architecture

In this section, we describe the proposed convolutional neural network to address the problem of face detection. The main goal is to work out an efficient detection model such that it can work in a near real-time fashion even on CPU machines while providing a promising accuracy. In our face detection network, conventional max-pooling layers are replaced by Inception residual blocks to avoid the loss of visual information [44]. Specifically, the Inception-ResNet-A module [10], as shown in Fig. 2, is adopted in the present work to enhance the learned features for object detection [25]. This structure employs three convolutional pathways with a fixed number of kernels of 32. In addition, each kernel applies the same stride of 1 on an input. As a result, the output shape of this structure equals to the shape of the input feature map.

The proposed model is presented in Fig. 3 that consists of alternating SCBox layers, Inception residual blocks, and convolutional layers. The detailed descriptions of the number of kernels and the strides are showed next to each layer. As can be seen in Fig. 3, we use SCBox(k, 3) module for the lower layers and SCBox(k, 2) for the deeper layers. The reason is based on the observation that small faces should be detected at the lower layers and thus the corresponding feature maps are expected to be sufficiently discriminated for achieving high detection accuracy. While many high-performance face detectors take as input the images of large size (e.g., 640×640 as for PyramidBox [18], DSFD [21], and SFD [19]), the proposed detector requires an input shape of 384 × 384 × 3 to balance speed and accuracy. In addition, the network size must be multiple of 128 as the model uses six SCBox layers (i.e., $B_1 - B_6$) and the layer B_6 consists of two convolutional layers as indicated in Fig. 1a. To maintain good performance of small face detection, one can choose larger model sizes: for instance, 512 × 512 or 640 × 640.

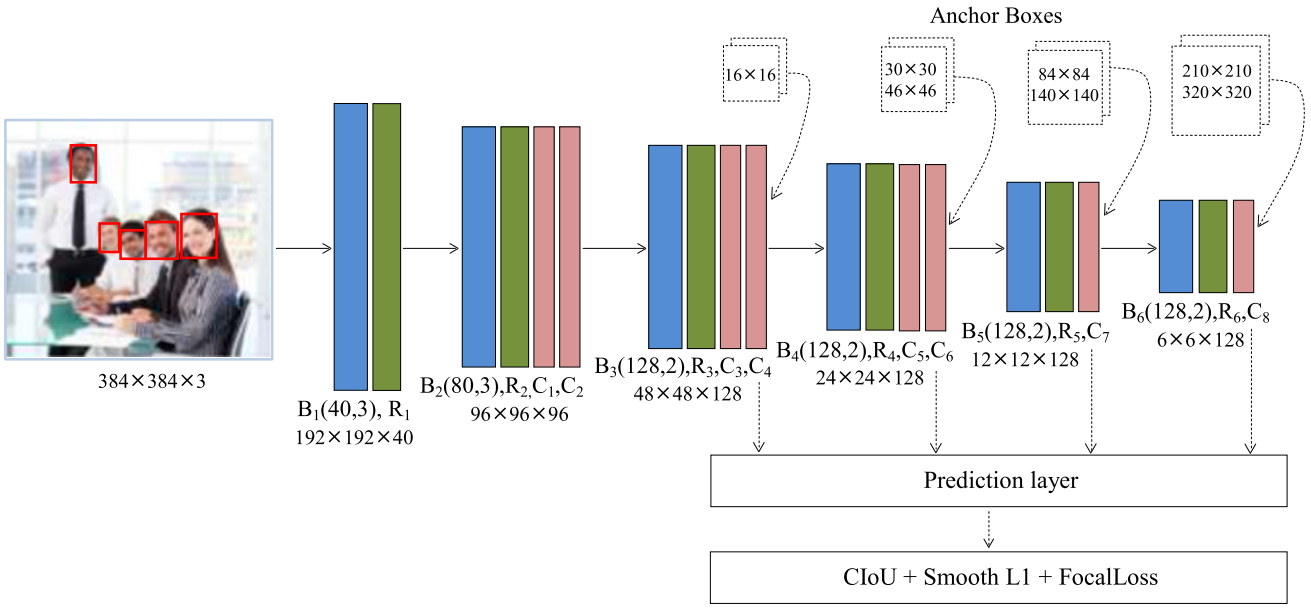


Fig. 3 Network architecture: the proposed detector is composed of alternating SCBox layers (i.e., $B_1 - B_6$), Inception residual blocks (i.e., $R_1 - R_6$), and small convolutional layers (i.e., $C_1 - C_8$ having the stride of 1 and kernel size of 3×3). Note that the Inception residual block does not change the dimensions of the previous layer. C_1 and C_2 have the same number of kernels (i.e., 96), while $C_3 - C_8$ have 128 kernels. Bounding boxes of faces are learned in a multi-scale manner with the anchor boxes attached to the feature maps at the C_4, C_6, C_7, C_8 layers

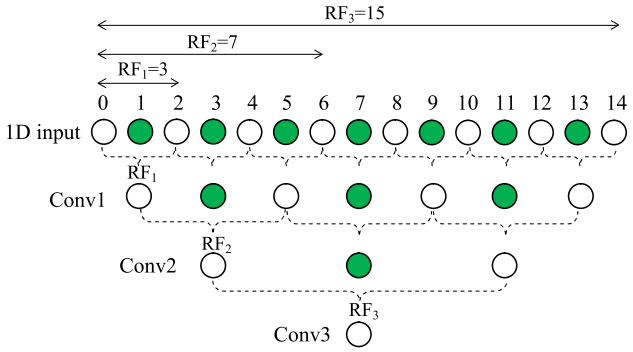


Fig. 4 Illustration of computing the receptive field sizes of an 1D input signal. All the convolution operators (i.e., Conv1, Conv2, and Conv3) have the filter length of 3 and the stride of 2. The green samples are the origins or centers of filters at corresponding locations. The Conv1 layer has the RF size of 3 (i.e., $RF_1 = 3$) with respect to the filter length. The RF size of the current convolutional layer (RF_i) is derived from that of the previous layer (RF_{i-1}) by: $RF_i = 2RF_{i-1} + 1$

Table 4 Receptive field (RF) sizes at different layers of the proposed network

Layer	Small RF size	Large RF size
R_1	3	15
C_2	7	31
C_4	15	63
C_5	31	63
C_7	63	127
C_8	127	255

Fig. 5 Effect of applying a SCBox(k,2) structure: the cell is assigned two receptive fields. The outer one helps describe the contextual information around the face, while the inner one puts more focus of facial features

To deal with multi-scale face detection, the anchor boxes are attached to several deeper layers. In particular, we use four feature maps (i.e., C_4, C_6, C_7, C_8) whose the corresponding shapes are of $48 \times 48 \times 128$, $24 \times 24 \times 128$, $12 \times 12 \times 128$, and $6 \times 6 \times 128$. Each cell of a feature map is assigned with one or several anchor boxes. The number of boxes and sizes is given in Fig. 3. When designing the sizes of anchors at each scale, it is important to take into consideration the information of corresponding receptive fields. In our case, because a SCBox

structure merges information from different layers, the receptive field sizes at each level are also varying. Generally, the RF size of the current convolutional layer (RF_i) is derived from that of the previous layer (RF_{i-1}) by: $\text{RF}_i = 2\text{RF}_{i-1}+1$ provided that all the convolutional layers have the filter size of 3×3 and the stride of 2. As for the B_1 component, because it consists of three convolutional layers (i.e., Conv1, Conv2, and Conv3 as illustrated in Fig. 4) with the size of 3×3 and the stride of 2, the RF sizes are 3, 7, and 15, respectively. On the other hand, as the R_1 block employs convolutional layers with the stride of 1, the RF sizes can be considered as equivalent to those of the B_1 component. For clarity, we provide the RF sizes at different layers as shown in Table 4. Figure 5 illustrates the case that two receptive fields are imposed at the cell of a given feature map. This structure brings an appealing benefit: it collects more contextual information around the face while enhancing the confidence by focusing on facial details.

The anchor boxes along with feature vectors learned at each cell are used by the model to predict the locations (in the mean of bounding boxes) and labels of interested objects (i.e., face or non-face). This task is driven by the prediction layer that employs a small kernel (i.e., $3 \times 3 \times 2$) to predict the label and another kernel (i.e., $3 \times 3 \times 4$) to perform regression of the locations (i.e., four coordinates) of each object. It is worth noting that the predicted coordinates of each object are relatively transformed to the specific anchor box's size.

One important aspect of building deep CNN models for object detection concerns the preparation of target training samples. Usually, the anchor boxes are matched against the groundtruth information (i.e., labels and box coordinates) to create two sorts of training targets, namely positive and negative samples. The matching procedure is basically presented as follows [3,5]:

- Each groundtruth box is matched to the anchor that has the largest Jaccard overlap.
- Assign the matched anchors as positive samples and remove them from the list.
- For each remaining anchor, find the groundtruth box with largest Jaccard overlap and consider it as a positive sample if and only if the obtained overlap is higher than a given threshold (i.e., 0.35 in our experiments).

When done, the anchors that are not matched to any groundtruth box are regarded as negative samples. As a side effect of the matching process, the number of negative samples is relatively higher than the number of positives. This fact tends to create a bias to the negative samples, thus degrading the detection accuracy. One common method to alleviate this issue is known as online hard negative mining (OHEM) [3,5,25]. Its basic idea is to sort the negative samples in the decreasing order of classification loss. The top-list samples are then selected until the ratio between negatives and positives meets a given threshold (typically 3:1). In the present work, we will employ the idea of OHEM in combination with the use of advanced loss functions to address the problem of class imbalance.

3.3 Loss functions

To drive the training process, two sorts of loss functions are used by the model, namely classification loss (L_{cls}) and localization loss (L_{loc}). Usually, a cross-entropy (CE) function is a common choice to compute L_{cls}:

$$\text{CE} = -\sum_{i \in X} \Big(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \Big) \tag{1}$$

where $y_i \in \{1, 0\}$ is the class label of the i^{th} anchor, p_i is the predicted probability that the i^{th} anchor is a face (i.e., $y_i = 1$), and X denotes the anchor set.

Denoting $\text{CE}_i = y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$, the CE loss can be rewritten as follows:

$$\text{CE} = -\sum_{i \in X} \text{CE}_i . \tag{2}$$

As discussed in [14], a less attracting property of CE loss is that it does not distinguish between well-classified examples and hard examples. As a result, the loss function applied to the easy samples (e.g., p close to 1) can even have high magnitude, degrading the training performance. To handle this issue, a new loss function, FocalLoss, has been introduced in [14] whose main idea is to put more weight on hard examples while giving less priority to the well-classified ones. Specifically, FocalLoss (FL) is created by adding a modulating factor to the CE loss:

$$\text{FL} = -\sum_{i \in X} \text{FL}_i \tag{3}$$

where $\text{FL}_i = \Big(y_i p_i + (1 - y_i)(1 - p_i) \Big)^{\gamma} \text{CE}_i$, and $\gamma \geq 0$ is a focusing parameter.

In practice, the full form of FL loss employs a balancing factor $\alpha \in [0, 1]$ for class $y = 1$ and $(1 - \alpha)$ for class $y = 0$. Its purpose is to further reduce the impact of class imbalance matter. Putting all together, the classification loss is defined as follows:

$$L_{\text{cls}} = -\sum_{i \in X} \Big(y_i \alpha + (1 - y_i)(1 - \alpha) \Big) \text{FL}_i . \tag{4}$$

On the other hand, the localization loss (L_{loc}) is designed to perform the regression of the bounding box's coordinates. Traditional CNN models often employ the Smooth L1 loss as

done in [3,5]. This loss function treats 4 points of a box independently and does not take into consideration the inherent correlation between the four points of a given face. To solve this problem, different novel loss functions have been introduced including IoU [15], GIoU [16], CIoU [17]. A common idea of these losses is the use of overlapping areas of the predicted box and the target box to make better the optimization process. As its name, IoU takes the intersection over union of the two boxes to drive the learning. When the two boxes are not overlapped, the loss is set to zero and thus is less meaningful. GIoU [16] (Generalized IoU) develops a more appealing idea by combining the shape properties and region property, yielding a better metric for comparing two objects having arbitrary shapes. Nonetheless, if the target box is completely covered by the predicted box, GIoU is identical to IoU loss. CIoU [17] (Complete IoU) solves this problem by taking into consideration three geometric factors: the overlapping area, the distance between two centers of the boxes, and the aspect ratio between the spatial dimensions of the boxes. In this paper, we incorporate the use of CIoU loss in addition to the common Smooth L1 loss to ensure a stable training.

$$L_{\text{loc}} = \frac{n_1 \,\text{CIoU} + n_2 \,\text{Smooth}_{L1}}{n_1 + n_2} \tag{5}$$

where we set $n_1 = 5, n_2 = 1$ to favor the weight of CIoU loss. Finally, the two loss functions, L_{cls} and L_{loc}, are linearly combined to drive the training process.

3.4 Computational analysis

In this part, the computational complexity of the proposed model is evaluated and compared with other methods by using the metric of floating point operations (FLOPs). Specifically, we compute FLOPs for the convolutional layers only as they have been arguably considered as the most intensively computational components of modern neural networks. Other additional layers (e.g., max-pooling, batch normalization, activation) are often ignored because their computational cost is relatively small when comparing to that of convolutional layers. Mathematically, we have adopted the following calculation formula to compute FLOPs for a given convolutional layer:

$$\text{FLOPs} = C_{\text{in}} F^2 C_{\text{out}} \text{WH} \tag{6}$$

where F indicates the kernel width or height (assumed to be symmetric), $C_{\text{in}}, C_{\text{out}}$ represent the input and output channels, and W, H denote the output feature map size (i.e., width and height, respectively).

Because the proposed model consists of the SCBox modules, Inception residual blocks, and the convolutional layers, we shall first provide the detailed justification of FLOPs for each of these components in the following. Specifically, Table 5 shows the FLOPs of the Inception-ResNet-A block given an input feature map having the shape of $w \times h \times c$ (i.e., width, height, and depth, respectively). Similarly, Table 6 presents the FLOPs of the SCBox(k,2) and SCBox(k,3) modules provided an input having the shape of $w \times h \times c$. It is noted that the kernel width is fixed (i.e., $F = 3$) for all convolutional layers in the SCBox modules. Hence, this parameter is not presented in Table 6.

Table 5 FLOPs of inception-ResNet-A module

Layer	Input shape	F	Output shape	FLOPs
Conv1a	$w \times h \times c$	1	$w \times h \times 32$	$32whc$
Conv1b	$w \times h \times c$	1	$w \times h \times 32$	$32whc$
Conv1c	$w \times h \times c$	1	$w \times h \times 32$	$32whc$
Conv2b	$w \times h \times 32$	3	$w \times h \times 32$	96^2wh
Conv2c	$w \times h \times 32$	3	$w \times h \times 32$	96^2wh
Conv3c	$w \times h \times 32$	3	$w \times h \times 32$	96^2wh
Conv4	$w \times h \times 96$	1	$w \times h \times c$	$96whc$
Total FLOPs: $192wh(c + 144)$				

Table 6 FLOPs of SCBox(k,2) and SCBox(k,3) modules

Layer	Input shape	Output shape	FLOPs
Conv1	$w \times h \times c$	$w/2 \times h/2 \times k$	$9kwhc/4$
Conv2	$w/2 \times h/2 \times k$	$w/4 \times h/4 \times k$	$9k^2wh/16$
Total FLOPs of SCBox(k,2): $9kwh(4c + k)/16$			
Conv3	$w/4 \times h/4 \times k$	$w/8 \times h/8 \times k$	$9k^2wh/64$
Total FLOPs of SCBox(k,3): $9kwh(16c + 5k)/64$			

Table 7 presents the complete analysis of FLOPs of the proposed network for an input image having the shape of $384 \times 384 \times 3$. Here, the FLOPs of each layer is computed and rounded to the nearest integer. Overall, the proposed face detection model requires around 5.25 BFLOPs. For comparative justification, the proposed model is compared against other modern face detectors, including EXTD-Mobilenet [45] (10.62 BFLOPs), PyramidBox-Res50 (111 BFLOPs), and ASFD-D6 [46] (183.11 BFLOPs). These figures demonstrate the theoretical efficiency of the proposed face detector. In addition, the empirical results on running times shall be provided in the subsequent section.

3.5 Training settings

The detection model is trained using the same protocol and dataset as described in [25]. Specifically, the WIDER FACE [24] is used for the training (12,880 images) and validation tasks (3,220). We also apply the data augmentation strategy as described in [25] including random color distortion, random image cropping, scale transformation, horizontal flipping,

Table 7 FLOPs of the proposed face detector

Layer	Input shape	Output shape	MFLOPs
B_1	384 × 384 × 3	192 × 192 × 40	206
R_1	192 × 192 × 40	192 × 192 × 40	1302
B_2	192 × 192 × 40	96 × 96 × 80	431
R_2	96 × 96 × 80	96 × 96 × 80	396
C_1	96 × 96 × 80	96 × 96 × 96	637
C_2	96 × 96 × 96	96 × 96 × 96	764
B_3	96 × 96 × 96	48 × 48 × 128	340
R_3	48 × 48 × 128	48 × 48 × 128	120
C_3	48 × 48 × 128	48 × 48 × 128	340
C_4	48 × 48 × 128	48 × 48 × 128	340
B_4	48 × 48 × 128	24 × 24 × 128	106
R_4	24 × 24 × 128	24 × 24 × 128	30
C_5	24 × 24 × 128	24 × 24 × 128	85
C_6	24 × 24 × 128	24 × 24 × 128	85
B_5	24 × 24 × 128	12 × 12 × 128	27
R_5	12 × 12 × 128	12 × 12 × 128	8
C_7	12 × 12 × 128	12 × 12 × 128	21
B_6	12 × 12 × 128	6 × 6 × 128	7
R_6	6 × 6 × 128	6 × 6 × 128	2
C_8	6 × 6 × 128	6 × 6 × 128	5
Total: 5252 MFLOPs			

Table 8 Parameter settings for training phase

Parameter	Value
num_training_step	250,000
batch_size	32
balancing_weight (α)	0.25
modulating_weight (γ)	2.0
neg_to_pos_rate	5:1

and face-box filtering. Other training parameters are detailed in Table 8.

4 Experiments

Two standard benchmarks, FDDB [23] and WIDER FACE [24], have been selected for performance evaluation of the proposed detector. As for the state-of-art face detection methods, we have reproduced their results from the public sources linked to the two benchmarks.[1],[2] The training process has been performed on the following GPU configuration: GeForce RTX 2080 Ti, 11 GB memory. All testing experiments have been run on a moderate GPU machine (GeForce GTX 1070 8Gb, CPU Core i7-7700). Our face detector is implemented using TensorFlow framework (v1.15) based on this project.[3]

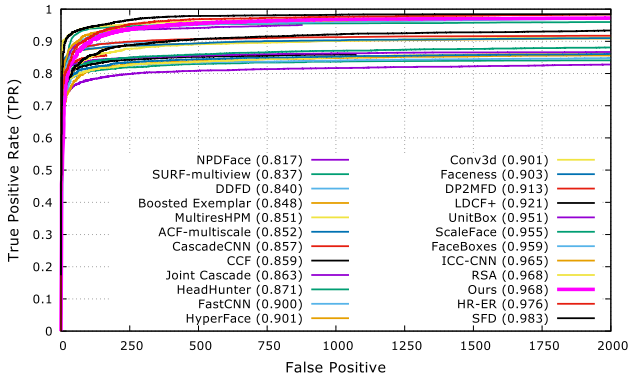


Fig. 6 Comparative evaluation on the FDDB dataset

4.1 Evaluation on FDDB benchmark

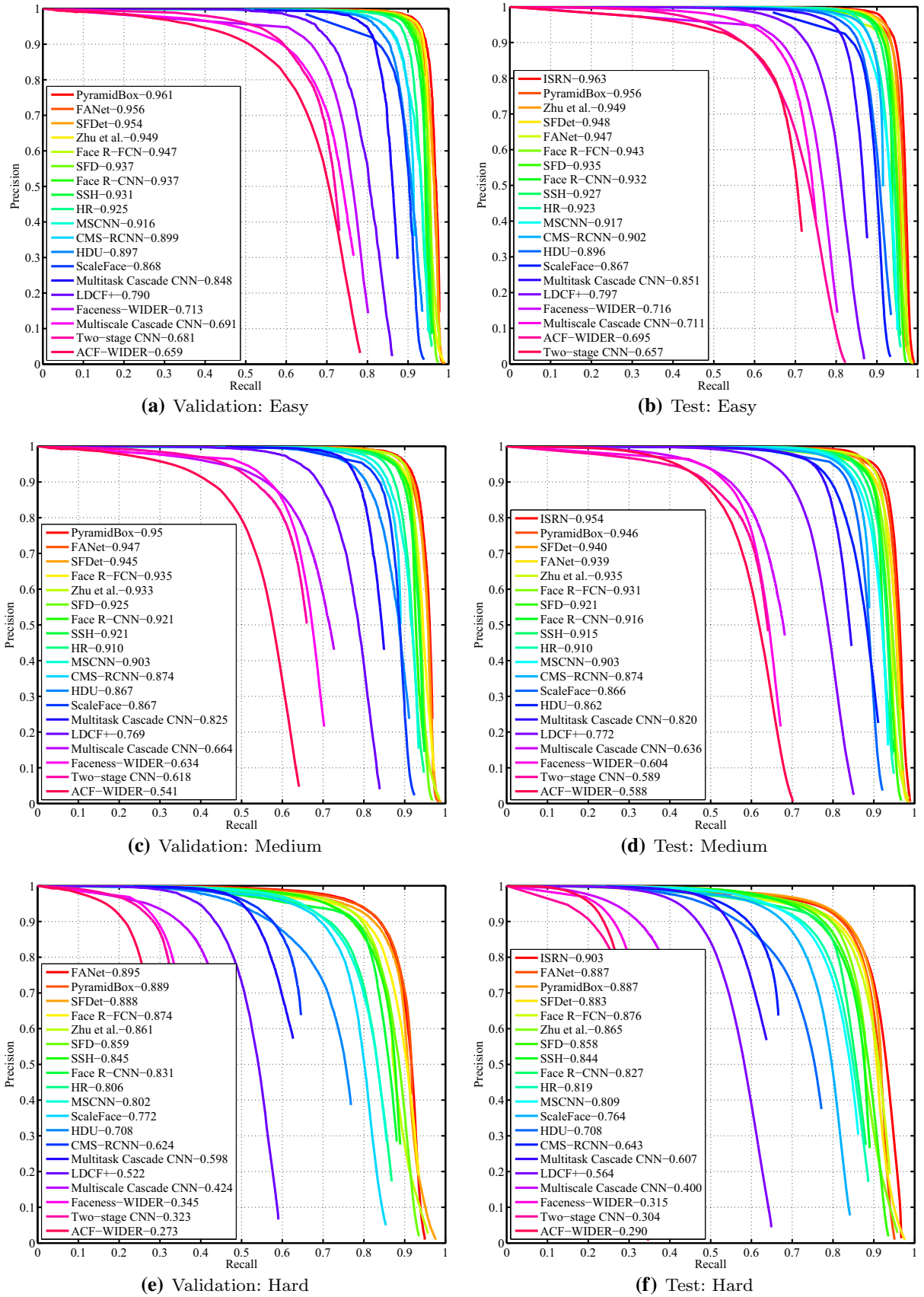
FDDB consists of 5171 faces from a set of 2845 images. The dataset covers a wide range of challenges including occlusions, difficult poses, and low resolution and out-of-focus faces. In addition, the images are provided in both gray-scale and color formats. Noticeably, the groundtruth face information is provided in the mean of bounding ellipses. Therefore, to make the evaluation comparable with the state-of-the-art methods, we follow the protocol described in [25] that a elliptical regressor is created to transform the bounding boxes to elliptical regions. Concerning evaluation metric, true positive rate versus the number of false positives is employed because it is commonly used by previous works. To obtain different operating points of score, the threshold for predicted probability is set to a relatively low value (i.e., 0.1 in our experiment).

Figure 6 shows the detection accuracy of the proposed model in comparison with other methods ([15,19,25,28,30,31,47–63]). As can be observed, our model performs competitively with the heavy-weight face detectors (e.g., HR-ER [63], SFD [19]). It is worth mentioning that these two methods have been trained from the pre-trained ImageNet models before fine-tuning on WIDER FACE. In contrast, our detector is trained from scratch using only 12K images in the training set of WIDER FACE. Impressively, the proposed detector outperforms many other systems with a significant gap in accuracy. Considering the number of false positives at 1000, the proposed model gives a true positive rate of 0.968 which is superior to many other near real-time CPU detectors, for instance FaceBoxes [25] (0.959), UnitBox [15] (0.951),

[1] http://vis-www.cs.umass.edu/fddb/results.html.

[2] http://shuoyang1213.me/WIDERFACE/.

[3] https://github.com/TropComplique/FaceBoxes-tensorflow.



(a) Validation: Easy **(b)** Test: Easy

(c) Validation: Medium **(d)** Test: Medium

(e) Validation: Hard **(f)** Test: Hard

Fig. 7 Comparative results on validation and test subsets of WIDER FACE dataset

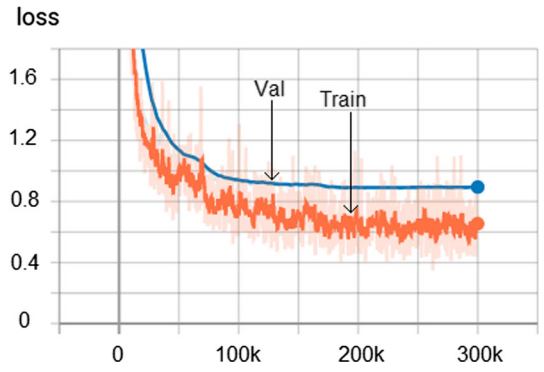


Fig. 8 Training (pink) and validation (green) loss functions for a wide range of parameter: *num_training_step*

Faceness [31] (0.910), Conv3d [47] (0.901), Boosted Exemplar [48] (0.848). Figure 10 shows some visual results on FDDB.

4.2 Evaluation on WIDER FACE benchmark

This dataset contains 393,703 faces from a collection of 32,203 images having a high degree of variability in scale, pose, facial expression, lighting environment, and occlusion. In addition, it covers 60 common events in daily life and is randomly structured into three sets: 40%, 10%, and 50% with respect to the training, validation, and testing. As mentioned previously, our model is trained only on the training set and is evaluated on both the validation and testing sets. Besides, to have more insight of detection performance, the evaluation results are divided into three difficulty levels (i.e., Easy, Medium, and Hard) based on the detection rate of a baseline detector (i.e., EdgeBox [64]).

Figure 7 shows the precision–recall curves of our detector and other methods ([18–20,24,29,31,59,60,62,63,65–71]). The proposed system consistently outperforms other near real-time detectors. Specifically, the AP rates of our detector on the validation set are following: 89.7% (Easy), 86.7% (Medium), and 70.8% (Hard), while on the test set the results are 89.6% (Easy), 86.2% (Medium), and 70.8% (Hard). The obtaining results are encouraging when considering the fact that the detector has been designed from a lightweight backbone. For this merit, we will justify the real-time aspect of the detector in the following part. A few visual results on this benchmark are shown in Fig. 11. As can be visually examined, the proposed detector can handle impressively the detection of small faces with accurate bounding box coordinates. Additional results are presented in Fig. 12 where similar images are fed into three face detectors: our method, FaceBoxes [26], and SFD [19]. The proposed model produces better detection accuracy than FaceBoxes and is competitive with the high-performance detector SFD.

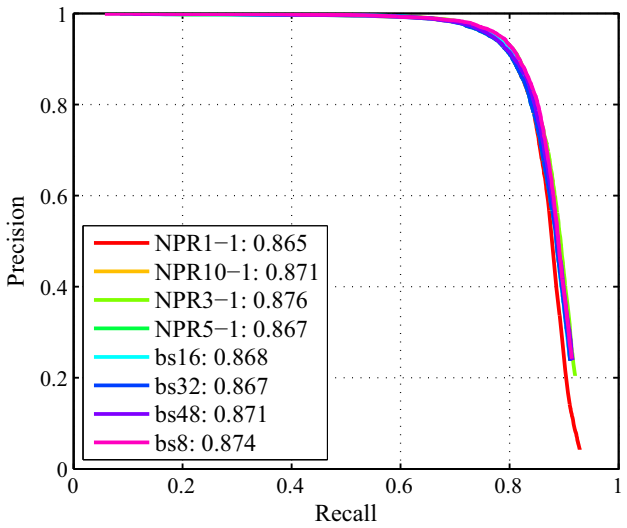


Fig. 9 Effect of the parameters *neg_to_pos_rate* and *batch_size* on the validation:medium set of WIDER FACE

4.3 Inference time

In this part, we investigate the efficiency, in terms of frames per second (FPS), of the proposed detector on both CPU and GPU configurations. Table 9 provides a comparative report of running times of all the studied methods. Besides, the average precision (%) on the FDDB benchmark is also provided (by setting the number of false positives to 1000). As for the methods running on CPU configuration, our detector performs impressively with a significant improvement of AP while still maintaining a good speed of 19.5 FPS. When comparing with heavy-weight detectors (e.g., SFD, FANet), the proposed detector gains a 3× speed with a small drop in accuracy (e.g., 1.5%). For instance, it achieves a notable speed of 106.3 FPS on a fairly moderate GPU. In addition, the model is also remarkable in terms of memory with about 11.1 MB of memory fingerprint.

4.4 Impact of parameter settings

For the training, we have used a number of control parameters as described in Table 8. The two parameters, α and γ, are used in the FocalLoss function and have been initialized by default values as suggested in the original paper [14] (i.e., $\alpha = 0.25$, $\gamma = 2.0$).

For each of the remaining parameters, we vary its value while keeping the others unchanged. Figure 8 shows the behavior of training and validation losses when varying the parameter num_training_step in the range of [0, 300 k]. As can be seen, the two loss functions behave similar to each other and tend to converge to stability after 200 k steps. As a result, one can set this parameter to any value in the range of [200 k, 300 k].

Fig. 10 Visual results (red-color bounding boxes) of the proposed face detector on FDDB dataset

Fig. 11 Visual results of the proposed face detector on WIDER FACE dataset

Fig. 12 Qualitative results of three face detectors for similar input images: our method (top row), FaceBoxes [26] (middle row), and SFD [19] (bottom row). The proposed model is competitive with the high-performance detector SFD

Figure 9 demonstrates the effect of two parameters: batch_size and neg_to_pos_rate. The former is varied in the set of {8, 16, 32, 48} (i.e., with respect to the variables of bs8, bs16, bs32, and bs48 in Fig. 9). The latter is often set to 3:1 in the literature [25], but in our study this ratio is set to the following values: 1:1, 3:1, 5:1, 10:1 (i.e., with respect to NPR1-1, NPR3-1, NPR5-1, NPR10-1). The results in Fig. 9 have been reported on the validation set (i.e., Medium subset) of WIDER FACE. One can notice that all the precision–recall curves are very close to each other and the performance gap is neglectable for different settings of the two parameters. All of these results consistently confirm the robustness of our approach for varying values of parameters.

5 Conclusion

In this work, we have presented an effective face detector by designing a lightweight CNN model. The proposed model exploits several advanced ideas for improving both detec-

Table 9 A report of running times (FPS) and average precision (AP) on FDDB of different detectors

Method	Machine model	FPS	AP
Ours (HDU)	CPU i7-7700K	19.6	**96.8**
FaceBoxes [25]	CPU E5-2660v3	20.1	95.6
ACF [67]	CPU i7-3770	20	85.2
JointCascade [54]	CPU-	34.9	86.3
Cascade-CNN [28]	CPU-	14	85.7
Cascade-CNN [28]	GPU Titan Black	100	85.7
Faceness [31]	GPU Titan Black	20	87.0
MTCNN [29]	GPU Titan Black	99	94.3
SFD [19]	GPU Titan X	36	98.3
FANet [20]	GPU GTX 1080	35.6	98.3
Ours (HDU)	GPU GTX 1070	**106.3**	96.8

tion accuracy and inference time. By introducing a Semantic Convolutional Box as a core block of each layer, the feature maps are enhanced at all the levels of the network. Besides, we have chosen to substitute all the max-pooling modules by a more discriminated structure (i.e., Inception residual) to learn the features in both the wider and deeper dimensions of the model. In addition, by incorporating recently introduced loss functions (i.e., FocalLoss, CIoU), the class imbalance problem has been nicely solved while yielding stable training and convergence. The proposed model has been validated on standard benchmarks and showed interesting results. Its performance is competitive with state-of-the-art methods while being efficient in terms of inference time.

Although the proposed detector achieves promising performance, there are several points that could be further improved. Firstly, it can be derived from Table 7 that the proposed SCBox modules are indeed very efficient with around 1.1 BFLOPs (21%) in total. The Inception residual components (i.e., $R_1 - R_6$) and convolutional layers (i.e., $C_1 - C_8$) consume a high rate of computational complexity (i.e., 4.15 BFLOPS or 79%). Therefore, further studies on designing lightweight and discriminative structures for substitution of these components could be helpful.

Secondly, data augmentation plays an important role in training an object detection model. In the present work, we have just applied basic methods for data augmentation. Many advanced methods have been recently presented to increase the variability of training images, for instance, CutOut [72], MixUp [73], CutMix [74], and Mosaic [75]. One could probably gain a significant improvement of performance by incorporating these advanced augmentation strategies.

Finally, we also intend to conduct future works on the problems of face landmark detection and face embedding as they are closely correlated to face detection in practical applications.

Acknowledgements This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant No. 102.05-2020.02.

References

1. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
2. Girshick, R.: Fast r-cnn. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015)
3. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(06), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
4. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016). https://doi.org/10.1109/CVPR.2016.91
5. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd, : Single shot multibox detector. In: European Conference on Computer Vision (ECCV), pp. 21–37 (2016)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
7. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. arXiv:1804.02767 [cs.CV] (2018)
8. Liu, S., Huang, D., Wang, Y.: Receptive field block net for accurate and fast object detection. In: European Conference on Computer Vision (ECCV), pp. 404–419 (2018)
9. Szegedy, C., Liu, Wei, Jia, Yangqing, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015)
10. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17, pp. 4278–4284. AAAI Press (2017)
11. Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944 (2017). https://doi.org/10.1109/CVPR.2017.106
12. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8759–8768 (2018). https://doi.org/10.1109/CVPR.2018.00913
13. Tan, M., Pang, R., Le, Q.V.: Efficientdet, : Scalable and efficient object detection. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10, 778–10, 787 (2020). https://doi.org/10.1109/CVPR42600.2020.01079
14. Lin, T., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2999–3007 (2017). https://doi.org/10.1109/ICCV.2017.324
15. Yu, J., Jiang, Y., Wang, Z., Cao, Z., Huang, T.: Unitbox: an advanced object detection network. In: Proceedings of the 24th ACM International Conference on Multimedia, MM '16, pp. 516–520. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2964284.2967274
16. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: a metric and a loss for bounding box regression. In: 2019 IEEE/CVF Confer-

ence on Computer Vision and Pattern Recognition (CVPR), pp. 658–666 (2019)
17. Zheng, Z., Wang, P., Ren, D., Liu, W., Ye, R., Hu, Q., Zuo, W.: Enhancing geometric factors in model learning and inference for object detection and instance segmentation. arXiv:2005.03572 [cs.CV] (2020)
18. Tang, X., Du, D.K., He, Z., Liu, J.: Pyramidbox: a context-assisted single shot face detector. In: European Conference on Computer Vision (ECCV), pp. 812–828 (2018)
19. Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., Li, S.Z.: S^3fd: single shot scale-invariant face detector. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 192–201 (2017). https://doi.org/10.1109/ICCV.2017.30
20. Zhang, J., Wu, X., Zhu, J., Hoi, S.C.H.: Feature agglomeration networks for single stage face detection. arXiv:1712.00721 [cs.CV] (2018)
21. Li, J., Wang, Y., Wang, C., Tai, Y., Qian, J., Yang, J., Wang, C., Li, J., Huang, F.: Dsfd: Dual shot face detector. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5055–5064 (2019). https://doi.org/10.1109/CVPR.2019.00520
22. Zhang, S., Chi, C., Lei, Z., Li, S.Z.: Refineface: refinement neural network for high performance face detection. IEEE Trans. Pattern Anal. Mach. Intell. (2020). https://doi.org/10.1109/TPAMI.2020.2997456
23. Jain, V., Learned-Miller, E.: Fddb: A benchmark for face detection in unconstrained settings. Technical Report. UM-CS-2010-009, University of Massachusetts, Amherst (2010)
24. Yang, S., Luo, P., Loy, C.C., Tang, : X.: Wider face: a face detection benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5525–5533 (2016)
25. Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., Li, S.Z.: Faceboxes: a cpu real-time face detector with high accuracy. In: 2017 IEEE International Joint Conference on Biometrics (IJCB), pp. 1–9 (2017). https://doi.org/10.1109/BTAS.2017.8272675
26. Zhang, S., Wang, X., Lei, Z., Li, S.Z.: Faceboxes: a cpu real-time and accurate unconstrained face detector. Neurocomputing **364**, 297–309 (2019)
27. Chen, W., Huang, H., Peng, S., Zhou, C., Zhang, C.: Yolo-face: a real-time face detector. Vis. Comput. (2020). https://doi.org/10.1007/s00371-020-01831-7
28. Li, H., Lin, Z., Shen, X., Brandt, J., Hua, G.: A convolutional neural network cascade for face detection. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition pp. 5325–5334 (2015). https://doi.org/10.1109/CVPR.2015.7299170
29. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Process. Lett. **23**(10), 1499–1503 (2016). https://doi.org/10.1109/LSP.2016.2603342
30. Farfade, S.S., Saberian, M.J., Li, L.J.: Multi-view face detection using deep convolutional neural networks. In: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, pp. 643–650 (2015)
31. Yang, S., Luo, P., Loy, C.C., Tang, X.: From facial parts responses to face detection: a deep learning approach. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), pp. 3676–3684 (2015)
32. Tareeq, S., Parveen, R., Rozario, L., Bhuiyan, M.: Robust face detection using genetic algorithm. Inf. Technol. J. (2007). https://doi.org/10.3923/itj.2007.142.147
33. Wiegand, S., Igel, C., Handmann, U.: Evolutionary optimization of neural networks for face detection. In: 12th European Symposium on Artificial Neural Networks (ESANN 2004), pp. 139–144 (2004)
34. Besnassi, M., Neggaz, N., Benyettou, A.: Face detection based on evolutionary haar filter. Pattern Anal. Appl. **23**(1), 309–330 (2020)
35. Jammoussi, A.Y., Ghribi, S.F., Masmoudi, D.S.: Adaboost face detector based on joint integral histogram and genetic algorithms for feature extraction process. Springerplus **3**, 1–9 (2014)
36. Correia, J.A., Martins, T., Machado, P.: Evolutionary data augmentation in deep face detection. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO'19, pp. 163–164 (2019). https://doi.org/10.1145/3319619.3322053
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 [cs.CV] (2014)
38. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. arXiv:1703.06870 [cs.CV] (2018)
39. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25. Curran Associates Inc, New York (2012)
40. Shang, W., Sohn, K., Almeida, D., Lee, H.: Understanding and improving convolutional neural networks via concatenated rectified linear units. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning-volume 48, ICML'16, pp. 2217–2225. JMLR.org (2016)
41. Viola, P., Jones, M.: Robust real-time face detection. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, vol. 2, pp. 747–747 (2001). https://doi.org/10.1109/ICCV.2001.937709
42. Coello Coello, C.A., Christiansen, A.D.: An empirical study of evolutionary techniques for multiobjective optimization in engineering design. Ph.D. thesis, USA (1996)
43. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. arXiv:1612.08242 [cs.CV] (2016)
44. Zhang, X., Chen, F., Yu, T., An, J., Huang, Z., Liu, J., Hu, W., Wang, L., Duan, H., Si, J.: Real-time gastric polyp detection using convolutional neural networks. PLoS ONE **14**(3), 1–16 (2019). https://doi.org/10.1371/journal.pone.0214133
45. Yoo, Y., Han, D., Yun, S.: Extd: extremely tiny face detector via iterative filter reuse. arXiv:1906.06579 [cs.CV] (2019)
46. Zhang, B., Li, J., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Xia, Y., Pei, W., Ji, R.: Asfd: Automatic and scalable face detector. arXiv:2003.11228 [cs.CV] (2020)
47. Li, Y., Sun, B., Wu, T., Wang, Y.: Face detection with end-to-end integration of a convnet and a 3d model. In: European Conference on Computer Vision (ECCV), pp. 420–436 (2016)
48. Li, H., Lin, Z., Brandt, J., Shen, X., Hua, G.: Efficient boosted exemplar-based face detection. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1843–1850 (2014). https://doi.org/10.1109/CVPR.2014.238
49. Liao, S., Jain, A.K., Li, S.Z.: A fast and accurate unconstrained face detector. IEEE Trans. Pattern Anal. Mach. Intell. **38**(2), 211–223 (2016). https://doi.org/10.1109/TPAMI.2015.2448075
50. Li, J., Zhang, Y.: Learning surf cascade for fast and accurate object detection. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3468–3475 (2013). https://doi.org/10.1109/CVPR.2013.445
51. Ghiasi, G., Fowlkes, C.C.: Occlusion coherence: detecting and localizing occluded faces. arXiv:1506.08347 [cs.CV] (2016)
52. Yang, B., Yan, J., Lei, Z., Li, S.Z.: Aggregate channel features for multi-view face detection. In: IEEE International Joint Conference on Biometrics, pp. 1–8 (2014). https://doi.org/10.1109/BTAS.2014.6996284
53. Yang, B., Yan, J., Lei, Z., Li, S.Z.: Convolutional channel features. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 82–90 (2015). https://doi.org/10.1109/ICCV.2015.18
54. Chen, D., Ren, S., Wei, Y., Cao, X., Sun, J.: Joint cascade face detection and alignment. In: European Conference on Computer Vision (ECCV), pp. 109–122 (2014)

55. Mathias, M., Benenson, R., Pedersoli, M., Van Gool, L.: Face detection without bells and whistles. In: Computer Vision—ECCV 2014, pp. 720–735. Springer (2014)
56. Triantafyllidou, D., Tefas, A.: A fast deep convolutional neural network for face detection in big visual data. In: Advances in Big Data, pp. 61–70 (2016)
57. Ranjan, R., Patel, V.M., Chellappa, R.: Hyperface, : A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. arXiv:1603.01249 [cs.CV] (2017)
58. Ranjan, R., Patel, V.M., Chellappa, R.: A deep pyramid deformable part model for face detection. arXiv:1508.04389 [cs.CV] (2015)
59. Ohn-Bar, E., Trivedi, M.M.: To boost or not to boost? on the limits of boosted trees for object detection. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 3350–3355 (2016). https://doi.org/10.1109/ICPR.2016.7900151
60. Yang, S., Xiong, Y., Loy, C.C., Tang, : X.: Face detection through scale-friendly deep convolutional networks. arXiv:1706.02863 (2017)
61. Zhang, K., Zhang, Z., Wang, H., Li, Z., Qiao, Y., Liu, W.: Detecting faces using inside cascaded contextual cnn. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 3190–3198 (2017). https://doi.org/10.1109/ICCV.2017.344
62. Wang, Y., Ji, X., Zhou, Z., Wang, H., Li, Z.: Detecting faces using region-based fully convolutional networks. arXiv:1709.05256 [cs.CV] (2017)
63. Hu, P., Ramanan, D.: Finding tiny faces. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1522–1530 (2017). https://doi.org/10.1109/CVPR.2017.166
64. Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: Computer Vision–ECCV 2014, pp. 391–405. Springer (2014)
65. Zhu, C., Zheng, Y., Luu, K., Savvides, M.: CMS-RCNN: Contextual Multi-Scale Region-Based CNN for Unconstrained Face Detection, pp. 57–79 (2017)
66. Wang, H., Li, Z., Ji, X., Wang, Y.: Face r-cnn. arXiv:1706.01061 [cs.CV] (2017)
67. Yang, B., Yan, J., Lei, Z., Li, S.Z.: Aggregate channel features for multi-view face detection. arXiv:1407.4023 [cs.CV] (2014)
68. Najibi, M., Samangouei, P., Chellappa, R., Davis, L.: SSH: Single stage headless face detector. In: The IEEE International Conference on Computer Vision (ICCV) (2017)
69. Zhang, S., Wen, L., Shi, H., Lei, Z., Lyu, S., Li, S.Z.: Single-shot scale-aware network for real-time face detection. Int. J. Comput. Vis. (IJCV) **127**(6–7), 537–559 (2019)
70. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection. In: Computer Vision-ECCV 2016, pp. 354–370 (2016)
71. Zhu, C., Tao, R., Luu, K., Savvides, M.: Seeing small faces from robust anchor's perspective. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5127–5136 (2018). https://doi.org/10.1109/CVPR.2018.00538
72. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv:1708.04552 [cs.CV] (2017)
73. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: Mixup: Beyond empirical risk minimization. arXiv:1710.09412 [cs.LG] (2018)
74. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: regularization strategy to train strong classifiers with localizable features. arXiv:1905.04899 [cs.CV] (2019)
75. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: optimal speed and accuracy of object detection. arXiv:2004.10934v1 [cs.CV] (2020)

Anh Pham has been working at Hong Duc University as a permanent researcher since 2004. He received his PhD Thesis in 2013 from Francois Rabelais university in France. Starting from June 2014 to November 2015, he has worked as a full research fellow position at Polytech's Tours, France. He has then returned to Hong Duc University since 2016 and received the title of associate professor in 2019. His research interests include document image analysis, image compression, feature extraction and indexing, shape analysis and representation, and deep learning networks.