FaceBoxes: A CPU Real-time Face Detector with High Accuracy

Shifeng Zhang Xiangyu Zhu Zhen Lei Hailin Shi Xiaobo Wang Stan Z. Li
CBSR & NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
{shifeng.zhang,xiangyu.zhu,zlei,hailin.shi,xiaobo.wang,szli}@nlpr.ia.ac.cn

Abstract

Although tremendous strides have been made in face detection, one of the remaining open challenges is to achieve real-time speed on the CPU as well as maintain high performance, since effective models for face detection tend to be computationally prohibitive. To address this challenge, we propose a novel face detector, named FaceBoxes, with superior performance on both speed and accuracy. Specifically, our method has a lightweight yet powerful network structure that consists of the Rapidly Digested Convolutional Layers (RDCL) and the Multiple Scale Convolutional Layers (MSCL). The RDCL is designed to enable FaceBoxes to achieve real-time speed on the CPU. The MSCL aims at enriching the receptive fields and discretizing anchors over different layers to handle faces of various scales. Besides, we propose a new anchor densification strategy to make different types of anchors have the same density on the image, which significantly improves the recall rate of small faces. As a consequence, the proposed detector runs at 20 *FPS on a single CPU core and* 125 *FPS using a GPU for VGA-resolution images. Moreover, the speed of FaceBoxes is invariant to the number of faces. We comprehensively evaluate this method and present state-of-the-art detection performance on several face detection benchmark datasets, including the AFW, PASCAL face, and FDDB. Code is available at* https://github.com/sfzhang15/FaceBoxes.

1. Introduction

Face detection is one of the fundamental problems in computer vision and pattern recognition. It plays an important role in many subsequent face-related applications, such as face alignment [47], face recognition [48] and face tracking [12]. With the great progress over the past few decades, especially the breakthrough of convolutional neural network, face detection has been successfully applied in our daily life under various scenarios.

However, there are still some tough challenges in uncontrolled face detection problem, especially for the CPU devices. The challenges mainly come from two requirements for face detectors: 1) The large visual variation of faces in the cluttered backgrounds requires face detectors to accurately address a complicated face and non-face classification problem; 2) The large search space of possible face positions and face sizes further imposes a time efficiency requirement. These two requirements are conflicting, since high-accuracy face detectors tend to be computationally expensive. Therefore, it is one of the remaining open issues for practical face detectors on the CPU devices to achieve real-time speed as well as maintain high performance.

In order to meet these two conflicting requirements, face detection has been intensely studied mainly in two ways. The early way is based on hand-craft features. Following the pioneering work of Viola-Jones face detector [37], most of the early works focus on designing robust features and training effective classifiers. Besides the cascade structure, the deformable part model (DPM) is introduced into face detection tasks and achieves remarkable performance. However, these methods highly depend on non-robust hand-craft features and optimize each component separately, making the face detection pipeline sub-optimal. In brief, they are efficient on the CPU but not accurate enough against the large visual variation of faces.

The other way is based on the convolutional neural network (CNN) which has achieved remarkable successes in recent years, ranging from image classification to object detection. Recently, CNN has been successfully introduced into the face detection task as feature extractor in the traditional face detection framewrok [23, 41, 42]. Moreover, some face detectors [4, 46] have inherited valid techniques from the generic object detection methods, such as Faster R-CNN [29]. These CNN based face detection methods are robust to the large variation of facial appearances and demonstrate state-of-the-art performance. But they are too time-consuming to achieve real-time speed, especially on the CPU devices.

These two ways have their own advantages. The former has fast speed while the latter owns high accuracy. To perform well on both speed and accuracy, one natural

idea is to combine the advantages of these two types of methods. Therefore, cascaded CNN based methods [16, 45] are proposed to put features learned by CNN into cascade framework in order to boost the performance and keep efficient. However, there are three problems in cascaded CNN based methods: 1) Their speed is negatively related to the number of faces on the image. The speed would dramatically degrade as the number of faces increases; 2) The cascade based detectors optimize each component separately, making the training process extremely complicated and the final model sub-optimal; 3) For the VGA-resolution images, their runtime efficiency on the CPU is about 14 FPS, which is not fast enough to reach the real-time speed.

In this paper, inspired by the RPN in Faster R-CNN [29] and the multi-scale mechanism in SSD [21], we develop a state-of-the-art face detector with real-time speed on the CPU. Specifically, we propose a novel face detector named FaceBoxes, which only contains a single fully convolutional neural network and can be trained end-to-end. The proposed method has a lightweight yet powerful network structure (as shown in Fig. 1) that consists of the Rapidly Digested Convolutional Layers (RDCL) and the Multiple Scale Convolutional Layers (MSCL). The RDCL is designed to enable FaceBoxes to achieve real-time speed on the CPU, and the MSCL aims at enriching the receptive fields and discretizing anchors over different layers to handle various scales of faces. Besides, we propose a new anchor densification strategy to make different types of anchors have the same density on the input image, which significantly improves the recall rate of small faces. Consequently, for VGA-resolution images, our face detector runs at 20 FPS on a single CPU core and 125 FPS using a GPU. More importantly, the speed of FaceBoxes is invariant to the number of faces on the image. We comprehensively evaluate this method and demonstrate state-of-the-art detection performance on several face detection benchmark datasets, including the AFW, PASCAL face, and FDDB.

For clarity, the main contributions of this work can be summarized as four-fold:

- We design the Rapidly Digested Convolutional Layers (RDCL) to enable face detection to achieve real-time speed on the CPU;
- We introduce the Multiple Scale Convolutional Layers (MSCL) to handle various scales of face via enriching receptive fields and discretizing anchors over layers.
- We present a new anchor densification strategy to improve the recall rate of small faces;
- We further improve the state-of-the-art performance on the AFW, PASCAL face, and FDDB datasets.

The rest of the paper is organized as follows. Section 2 reviews the related work. Analysis of the FaceBoxes is presented in section 3. Section 4 shows the experimental results and section 5 concludes the paper.

2. Related work

Modern face detection approaches can be roughly divided into two different categories. One is based on hand-craft features, and the other one is built on CNN. This section briefly reviews these two kinds of methods.

2.1. Hand-craft based methods

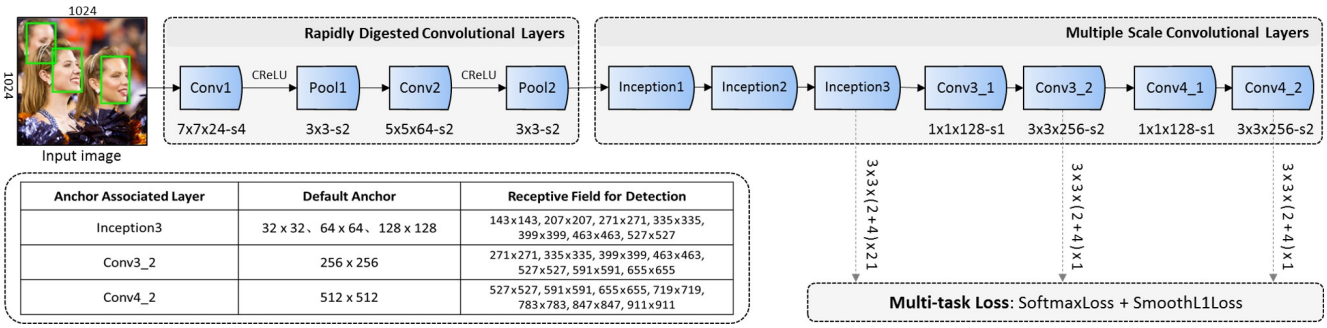
Previous face detection systems are mostly based on hand-craft features. Since the seminal Viola-Jones face detector [37] that proposes to combine Haar feature, Adaboost learning and cascade inference for face detection, many subsequent works are proposed for real-time face detection, such as new local features [20, 40], new boosting algorithms [3, 25] and new cascade structures [2, 18].

Besides the cascade framework, methods based on structural models progressively achieve better performance and become more and more efficient. Some researches [38, 39, 49] introduce the deformable part model (DPM) into face detection tasks. These works use supervised parts, more pose partition, better training or more efficient inference to achieve remarkable detection performance.

2.2. CNN based methods

The first use of CNN for face detection can be traced back to 1994. Vaillant et al. [36] use a trained CNN in a sliding windows manner to detect faces. Rowley et al. [30, 31] introduce a retinally connected neural network for upright frontal face detection, and a "router" network designed to estimate the orientation for rotation invariant face detection. Garcia et al. [7] develop a neural network to detect semi-frontal faces. Osadchy et al. [24] train a CNN for simultaneous face detection and pose estimation. These earlier methods can get relatively good performance only on easy dataset.

Recent years have witnessed the advance of CNN based face detectors. CCF [41] uses boosting on top of CNN features for face detection. Farfade et al. [6] fine-tune CNN model trained on 1k ImageNet classification task for face and non-face classification task. Faceness [42] trains a series of CNNs for facial attribute recognition to detect partially occluded faces. CascadeCNN [16] develops a cascade architecture built on CNNs with powerful discriminative capability and high performance. Qin et al. [26] propose to jointly train CascadeCNN to realize end-to-end optimization. Similar to [5], MTCNN [45] proposes a multi-task cascaded CNNs based framework for joint face detection and alignment. UnitBox [44] introduces a new intersection-over-union loss function. CMS-RCNN [46] uses Faster R-CNN in face detection with body contextual information. Convnet [19] integrates CNN with 3D face model in an end-to-end multi-task learning framework. STN [4] proposes a new supervised transformer network and a ROI convolution for face detection.



Anchor Associated Layer	Default Anchor	Receptive Field for Detection
Inception3	32 x 32、64 x 64、128 x 128	143x143, 207x207, 271x271, 335x335, 399x399, 463x463, 527x527
Conv3_2	256 x 256	271x271, 335x335, 399x399, 463x463, 527x527, 591x591, 655x655
Conv4_2	512 x 512	527x527, 591x591, 655x655, 719x719, 783x783, 847x847, 911x911

Figure 1. Architecture of the FaceBoxes and the detailed information table about our anchor designs.

3. FaceBoxes

This section presents our three contributions that make the FaceBoxes accurate and efficient on the CPU devices: the Rapidly Digested Convolutional Layers (RDCL), the Multiple Scale Convolutional Layers (MSCL) and the anchor densification strategy. Finally, we introduce the associated training methodology.

3.1. Rapidly Digested Convolutional Layers

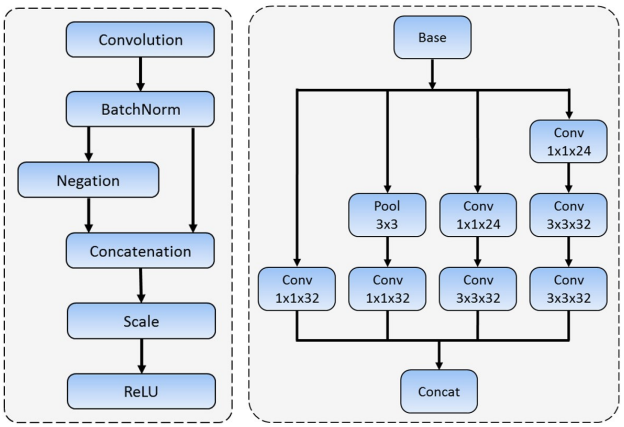
Most of the CNN based face detection methods are usually limited by the heavy cost of time, especially on the CPU devices. More precisely, the convolution operation for CPU is extremely time-consuming when the size of input, kernel and output are large. Our RDCL is designed to fast shrink the input spatial size by suitable kernel size with reducing the number of output channels, enabling the FaceBoxes to reach real-time speed on the CPU devices as follows:

- **Shrinking the spatial size of input**: To rapidly shrink the spatial size of input, our RDCL sets a series of large stride sizes for its convolution and pooling layers. As illustrated in Fig. 1, the stride size of Conv1, Pool1, Conv2 and Pool2 are 4, 2, 2 and 2, respectively. The total stride size of RDCL is 32, which means the input spatial size is reduced by 32 times quickly.
- **Choosing suitable kernel size**: The kernel size of the first few layers in one network should be small so as to speed up, while it is also supposed to be large enough to alleviate the information loss brought by the spatial size reducing. As shown in Fig. 1, to keep efficient as well as effective, we choose 7×7, 5×5 and 3×3 kernel size for Conv1, Conv2 and all Pool layers, respectively.
- **Reducing the number of output channels**: We utilize the C.ReLU activation function (illustrated in Fig. 2(a)) to reduce the number of output channels. C.ReLU [32] is motivated from the observation in CNN that the filters in the lower layers form pairs (*i.e.*, filters with opposite phase). From this observation, C.ReLU can double the number of output channels by simply concatenating negated outputs before applying ReLU. Using C.ReLU significantly increases speed with negligible decline in accuracy.

3.2. Multiple Scale Convolutional Layers

The proposed method is based on RPN which is developed as a class-agnostic proposer in the scenario of multi-category object detection. For the single-category detection task (*e.g.*, face detection), RPN is naturally a detector for the only category concerned. However, as a stand-alone face detector, RPN is not able to obtain competitive performances. We argue that such unsatisfactory performance comes from two aspects. Firstly, the anchors in the RPN are only associated with the last convolutional layer whose feature and resolution are too weak to handle faces of various sizes. Secondly, an anchor-associated layer is responsible for detecting faces within a corresponding range of scales, but it only has a single receptive field that can not match different scales of faces. To solve the above two problems, our MSCL is designed along the following two dimensions:

- **Multi-scale design along the dimension of network depth.** As shown in Fig. 1, our designed MSCL consists of several layers. These layers decrease in size progressively and form the multi-scale feature maps. Similar to [21], our default anchors are associated with multi-scale feature maps (*i.e.*, Inception3, Conv3_2 and Conv4_2). These layers, as a multi-scale design along the dimension of network depth, discretize anchors over multiple layers with different resolutions to naturally handle faces of various sizes.
- **Multi-scale design along the dimension of network width.** To learn visual patterns for different scales of faces, output features of the anchor-associated layers should correspond to various sizes of receptive fields, which can be easily fulfilled via Inception modules [34]. The Inception module consists of multiple convolution branches with different kernels. These branches, as a multi-scale design along the dimension of network width, is able to enrich the receptive fields. As shown in Fig. 1, the first three layers in MSCL are based on the Inception module. Fig. 2(b) illustrates our Inception implementation, which is a cost-effective module to capture different scales of faces.



(a) (b)

Figure 2. (a) The C.ReLU modules where Negation simply multiplies -1 to the output of Convolution. (b) The Inception modules.

3.3. Anchor densification strategy

As illustrated in Fig. 1, we impose 1:1 aspect ratio for the default anchors (i.e., square anchor), because the face box is approximately square. The scale of anchor for the Inception3 layer is 32, 64 and 128 pixels, for the Conv3_2 layer and Conv4_2 layer are 256 and 512 pixels, respectively.

The tiling interval of anchor on the image is equal to the stride size of the corresponding anchor-associated layer. For example, the stride size of Conv3_2 is 64 pixels and its anchor is 256×256, indicating that there is a 256×256 anchor for every 64 pixels on the input image. We define the tiling density of anchor (*i.e.*, $A_{density}$) as follows:

$$A_{density} = A_{scale}/A_{interval} \quad (1)$$

Here, A_{scale} is the scale of anchor and $A_{interval}$ is the tiling interval of anchor. The tiling intervals for our default anchors are 32, 32, 32, 64 and 128, respectively. According to Equ. (1), the corresponding densities are **1**, **2**, 4, 4 and 4, where it is obviously that there is a tiling density imbalance problem between anchors of different scales. Comparing with large anchors (*i.e.*, 128×128, 256×256 and 512×512), small anchors (*i.e.*, 32×32 and 64×64) are too sparse, which results in low recall rate of small faces.

To eliminate this imbalance, we propose a new anchor densification strategy. Specifically, to densify one type of anchors n times, we uniformly tile $A_{number} = n^2$ anchors around the center of one receptive field instead of only tiling one at the center of this receptive field to predict. Some examples are shown in Fig. 4. In our paper, to improve the tiling density of the small anchor, our strategy is used to densify the 32×32 anchor 4 times and the 64×64 anchor 2 times, which guarantees that different scales of anchor have the same density (*i.e.*, 4) on the image, so that various scales of faces can match almost the same number of anchors.

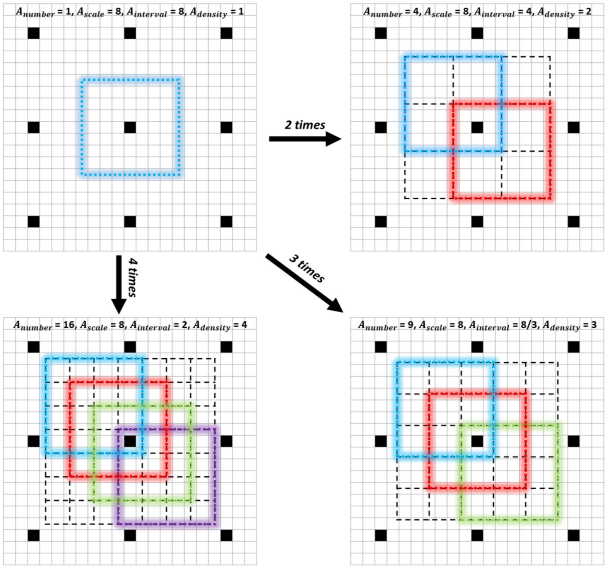


Figure 3. Examples of anchor densification. For clarity, we only densify anchors at one receptive field centre (*i.e.*, the central black cell), and only color the diagonal anchors.

3.4. Training

This subsection introduces the training dataset, data augmentation, matching strategy, loss function, hard negative mining, and other implementation details.

Training dataset. Our model is trained on 12,880 images of the WIDER FACE [43] training subset.

Data augmentation. Each training image is sequentially processed by the following data augmentation strategies:

- Color distortion: Applying some photo-metric distortions similar to [9].
- Random cropping: We randomly crop five square patches from the original image: one is the biggest square patch, and the size of the others range between [0.3, 1] of the short size of the original image. Then we arbitrarily select one patch for subsequent operations.
- Scale transformation: After random cropping, the selected square patch is resized to 1024×1024.
- Horizontal flipping: The resized image is horizontally flipped with probability of 0.5.
- Face-box filter: We keep the overlapped part of the face box if its center is in the above processed image, then filter out these face boxes whose height or width is less than 20 pixels.

Matching strategy. During training, we need to determine which anchors correspond to a face bounding box. We first match each face to the anchor with the best jaccard overlap, and then match anchors to any face with jaccard overlap higher than a threshold (*i.e.*, 0.35).

Loss function. Our loss function is the same as RPN in Faster R-CNN [29]. We adopt a 2-class softmax loss for classification and the smooth L1 loss for regression.

Hard negative mining. After the anchor matching step, most of the anchors are found to be negative, which introduces a significant imbalance between the positive and negative examples. For faster optimization and stable training, we sort them by the loss values and pick the top ones so that the ratio between the negatives and positives is at most 3:1.

Other implementation details. All the parameters are randomly initialized with the "xavier" method. We fine-tune the resulting model using SGD with 0.9 momentum, 0.0005 weight decay and batch size 32. The maximum number of iterations is $120k$ and we use 10^{-3} learning rate for the first $80k$ iterations, then continue training for $20k$ iterations with 10^{-4} and 10^{-5}, respectively. Our method is implemented in the Caffe library.

4. Experiments

In this section, we firstly introduce the runtime efficiency of FaceBoxes, then analyze our model in an ablative way, finally evaluate it on the common face detection benchmarks.

4.1. Runtime efficiency

CNN based methods have always been accused of its runtime efficiency. Although the existing CNN face detectors can be accelerated via high-end GPUs, they are not fast enough in most practical applications, especially CPU based applications. As described below, our FaceBoxes is efficient enough to meet practical requirements.

During inference, our method outputs a large number of boxes (*e.g.*, $8,525$ boxes for a VGA-resolution image). We first filter out most boxes by a confidence threshold of 0.05 and keep the top 400 boxes before applying NMS, then we perform NMS with jaccard overlap of 0.3 and keep the top 200 boxes. We measure the speed using Titan X (Pascal) and cuDNN v5.1 with Intel Xeon E5-2660v3@2.60GHz. As listed in Tab. 1, comparing with recent CNN-based methods, our FaceBoxes can run at 20 FPS on the CPU with state-of-the-art accuracy. Besides, our method can run at 125 FPS using a single GPU and has only 4.1MB in size.

Approach	CPU-model	mAP(%)	FPS
ACF [40]	i7-3770@3.40	85.2	20
CasCNN [16]	E5-2620@2.00	85.7	14
FaceCraft [26]	N/A	90.8	10
STN [4]	i7-4770K@3.50	91.5	10
MTCNN [45]	N/A@2.60	94.4	16
Ours	E5-2660v3@2.60	**96.0**	**20**

Table 1. Overall CPU inference time and mAP compared on different methods. The **FPS** is for VGA-resolution images on CPU and the **mAP** means the true positive rate at 1000 false positives on FDDB. Notably, for STN [4], its mAP is the true positive rate at 179 false positives and with ROI convolution, its FPS can be accelerated to 30 with 0.6% recall rate drop.

4.2. Model analysis

We carried out extensive ablation experiments on the FDDB dataset to analyze our model. Comparing with AFW and PASCAL face, FDDB is much more difficult so that analyzing our model on FDDB is convincing. For all the experiments, we use the same settings, except for specified changes to the components.

Ablative Setting. To better understand FaceBoxes, we ablate each component one after another to examine how each proposed component affects the final performance. 1) Firstly, we ablate the anchor densification strategy. 2) Then, we replace MSCL with three convolutional layers, which all have 3×3 kernel size and whose output number is the same as the first three Inception modules of MSCL. Meantime, we only associate the anchors with the last convolutional layer. 3) Finally, we take the place of C.ReLU with ReLU in RDCL. The ablative results are listed in Tab. 2 and some promising conclusions can be summed up as follows:

Contribution	FaceBoxes			
RDCL				×
MSCL			×	×
Strategy		×	×	×
Accuracy (mAP)	96.0	94.9	93.9	94.0
Speed (ms)	50.98	48.27	48.23	67.48

Table 2. Ablative results of the FaceBoxes on FDDB dataset. Accuracy (mAP) means the true positive rate at 1000 false positives. Speed (ms) is for the VGA-resolution images on the CPU.

Anchor densification strategy is crucial. Our anchor densification strategy is used to increase the density of small anchors (*i.e.*, 32×32 and 64×64) in order to improve the recall rate of small faces. From the results listed in Tab. 2, we can see that the mAP on FDDB is reduced from 96.0% to 94.9% after ablating the anchor densification strategy. The sharp decline (*i.e.*, 1.1%) demonstrates the effectiveness of the proposed anchor densification strategy.

MSCL is better. The comparison between the second and third columns in Tab. 2 indicates that MSCL effectively increases the mAP by 1.0%, owning to the diverse receptive fields and the multi-scale anchor tiling mechanism.

RDCL is efficient and accuracy-preserving. The design of RDCL enables our FaceBoxes to achieve real-time speed on the CPU. As reported in Tab. 2, RDCL leads to a negligible decline on accuracy but a significant improvement on speed. Specifically, the FDDB mAP decreases by 0.1% in return for the about 19.3ms speed improvement.

4.3. Evaluation on benchmark

We evaluate the FaceBoxes on the common face detection benchmark datasets, including Annotated Faces in the Wild (AFW), PASCAL Face, and Face Detection Data Set and Benchmark (FDDB).

AFW dataset [49]. It has 205 images with 473 faces. We evaluate FaceBoxes against the well-known works [4, 20, 22, 33, 39, 42, 49] and commercial face detectors (*e.g.*, Face.com, Face++ and Picasa). As illustrated in Fig.4, our FaceBoxes outperforms all others by a large margin. Fig.7(a) shows some qualitative results on the AFW dataset.

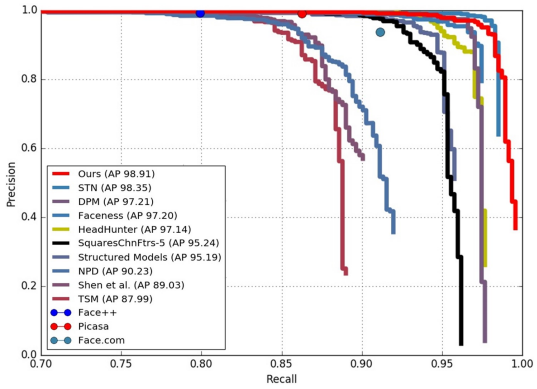


Figure 4. Precision-recall curves on AFW dataset.

PASCAL face dataset [39]. It is collected from the test set of PASCAL person layout dataset, consisting of 1335 faces with large face appearance and pose variations from 851 images. Fig.5 shows the precision-recall curves on this dataset. Our method significantly outperforms all other methods [4, 11, 22, 39, 42, 49] and commercial face detectors (*e.g.*, SkyBiometry, Face++ and Picasa). Fig.7(b) shows some qualitative results on the PASCAL face dataset.

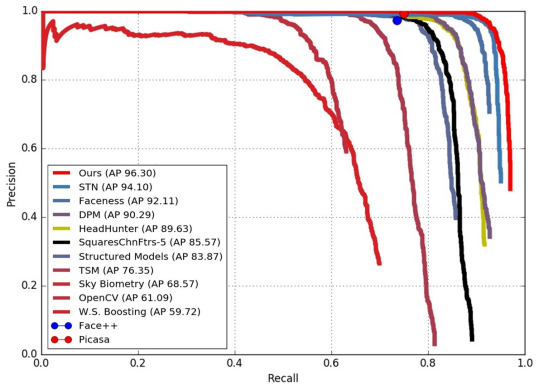
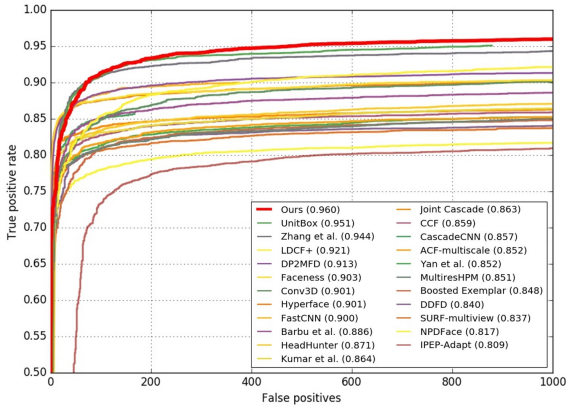
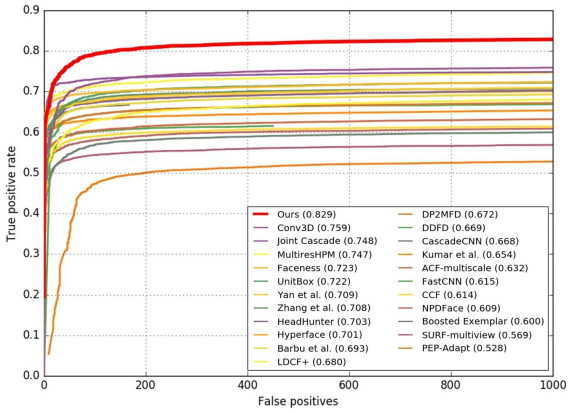


Figure 5. Precision-recall curves on PASCAL face dataset.

FDDB dataset [10]. It has 5,171 faces in 2,845 images taken from news articles on Yahoo websites. FDDB adopts the bounding ellipse, while our FaceBoxes outputs rectangle bounding box. This inconsistency has a great impact to the continuous score. For a more fair comparison under the continuous score evaluation, we train an elliptical regressor to transform our predicted bounding boxes to bounding ellipses. We evaluate our face detector on FDDB against the other methods [1, 6, 8, 13, 14, 15, 17, 19, 20, 23, 27, 28, 35, 42, 44, 45]. The results are shown in Fig. 6(a) and Fig.6(b). Our FaceBoxes achieves the state-of-the-art performance and outperforms all others by a large margin on discontinuous and continuous ROC curves. These results indicate that our FaceBoxes can robustly detect unconstrained faces. Fig.7(c) shows some qualitative results on the FDDB.



(a) Discontinuous ROC curves



(b) Continuous ROC curves

Figure 6. Evaluation on the FDDB dataset.

5. Conclusion

Since effective models for the face detection task tend to be computationally prohibitive, it is challenging for the CPU devices to achieve real-time speed as well as maintain high performance. In this work, we present a novel face detector with superior performance on both speed and accuracy. The proposed method has a lightweight yet powerful network structure, which consists of RDCL and MSCL. The former enables FaceBoxes to achieve real-time speed, and the latter aims at enriching receptive fields and discretizing anchors over different layers to handle faces of various scales. Besides, a new anchor densification strategy is proposed to improve the recall rate of small faces. The experiments demonstrate that our contributions lead FaceBoxes to the state-of-the-art performance on the common face detection benchmarks. The proposed detector is very fast, achieving 20 FPS for VGA-resolution images on CPU and can be accelerated to 125 FPS on GPU.

(a) AFW

(b) PASCAL face

(c) FDDB

Figure 7. Qualitative results on face detection benchmark datasets.

Acknowledgments

This work was supported by the National Key Research and Development Plan (Grant No.2016YFC0801002), the Chinese National Natural Science Foundation Projects #61473291, #61502491, #61572501, #61572536, #61672521 and AuthenMetric R&D Funds.

References

[1] A. Barbu, N. Lay, and G. Gramajo. Face detection with a 3d model. *arXiv:1404.3596*, 2014. 6

[2] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, 2005. 2

[3] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg. On the design of cascades of boosted ensembles for face detection. *IJCV*, 2008. 2

[4] D. Chen, G. Hua, F. Wen, and J. Sun. Supervised transformer network for efficient face detection. In *ECCV*, 2016. 1, 2, 5, 6

[5] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In *ECCV*, 2014. 2

[6] S. S. Farfade, M. J. Saberian, and L.-J. Li. Multi-view face detection using deep convolutional neural networks. In *ICMR*, 2015. 2, 6

[7] C. Garcia and M. Delakis. A neural architecture for fast and robust face detection. In *ICPR*, 2002. 2

[8] G. Ghiasi and C. Fowlkes. Occlusion coherence: Detecting and localizing occluded faces. *arXiv:1506.08347*, 2015. 6

[9] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 4

[10] V. Jain and E. G. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. *UMass Amherst Technical Report*, 2010. 6

[11] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, 2008. 6

[12] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley. Face tracking and recognition with visual constraints in real-world videos. In *CVPR*, 2008. 1

[13] V. Kumar, A. Namboodiri, and C. Jawahar. Visual phrases for exemplar face detection. In *ICCV*, 2015. 6

[14] H. Li, G. Hua, Z. Lin, J. Brandt, and J. Yang. Probabilistic elastic part model for unsupervised face detector adaptation. In *ICCV*, 2013. 6

[15] H. Li, Z. Lin, J. Brandt, X. Shen, and G. Hua. Efficient boosted exemplar-based face detection. In *CVPR*, 2014. 6

[16] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015. 2, 5

[17] J. Li and Y. Zhang. Learning surf cascade for fast and accurate object detection. In *CVPR*, 2013. 6

[18] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *ECCV*, 2002. 2

[19] Y. Li, B. Sun, T. Wu, and Y. Wang. Face detection with end-to-end integration of a convnet and a 3d model. In *ECCV*, 2016. 2, 6

[20] S. Liao, A. K. Jain, and S. Z. Li. A fast and accurate unconstrained face detector. *PAMI*, 2016. 2, 6

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2, 3

[22] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *ECCV*, 2014. 6

[23] E. Ohn-Bar and M. M. Trivedi. To boost or not to boost? on the limits of boosted trees for object detection. In *ICPR*, 2016. 1, 6

[24] M. Osadchy, Y. L. Cun, and M. L. Miller. Synergistic face detection and pose estimation with energy-based models. *JMLR*, 2007. 2

[25] M.-T. Pham and T.-J. Cham. Fast training and selection of haar features using statistics in boosting-based face detection. In *ICCV*, 2007. 2

[26] H. Qin, J. Yan, X. Li, and X. Hu. Joint training of cascaded cnn for face detection. In *CVPR*, 2016. 2, 5

[27] R. Ranjan, V. M. Patel, and R. Chellappa. A deep pyramid deformable part model for face detection. In *BTAS*, 2015. 6

[28] R. Ranjan, V. M. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *arXiv:1603.01249*, 2016. 6

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 5

[30] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *PAMI*, 1998. 2

[31] H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *CVPR*, 1998. 2

[32] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2016. 3

[33] X. Shen, Z. Lin, J. Brandt, and Y. Wu. Detecting and aligning faces by image retrieval. In *CVPR*, 2013. 6

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 3

[35] D. Triantafyllidou and A. Tefas. A fast deep convolutional neural network for face detection in big visual data. In *INNS Conference on Big Data*, 2016. 6

[36] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 1994. 2

[37] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 2004. 1, 2

[38] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. In *CVPR*, 2014. 2

[39] J. Yan, X. Zhang, Z. Lei, and S. Z. Li. Face detection by structural models. *Image and Vision Computing*, 2014. 2, 6

[40] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Aggregate channel features for multi-view face detection. In *IJCB*, 2014. 2, 5

[41] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Convolutional channel features. In *ICCV*, 2015. 1, 2

[42] S. Yang, P. Luo, C.-C. Loy, and X. Tang. From facial parts responses to face detection: A deep learning approach. In *ICCV*, 2015. 1, 2, 6

[43] S. Yang, P. Luo, C. C. Loy, and X. Tang. Wider face: A face detection benchmark. In *CVPR*, 2016. 4

[44] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. In *ACMMM*, 2016. 2, 6

[45] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *SPL*, 2016. 2, 5, 6

[46] C. Zhu, Y. Zheng, K. Luu, and M. Savvides. Cms-rcnn: contextual multi-scale region-based cnn for unconstrained face detection. *arXiv:1606.05413*, 2016. 1, 2

[47] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li. Face alignment across large poses: A 3d solution. In *CVPR*, 2016. 1

[48] X. Zhu, Z. Lei, J. Yan, D. Yi, and S. Z. Li. High-fidelity pose and expression normalization for face recognition in the wild. In *CVPR*, 2015. 1

[49] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2012. 2, 6