Detect Faces Efficiently: A Survey and Evaluations

Yuantao Feng, Shiqi Yu, *Member, IEEE*, Hanyang Peng, Yan-Ran Li, and Jianguo Zhang, *Senior Member, IEEE*

***Abstract*—Face detection is to search all the possible regions for faces in images and locate the faces if there are any. Many applications including face recognition, facial expression recognition, face tracking and head-pose estimation assume that both the location and the size of faces are known in the image. In recent decades, researchers have created many typical and efficient face detectors from the Viola-Jones face detector to current CNN-based ones. However, with the tremendous increase in images and videos with variations in face scale, appearance, expression, occlusion and pose, traditional face detectors are challenged to detect various "in the wild" faces. The emergence of deep learning techniques brought remarkable breakthroughs to face detection along with the price of a considerable increase in computation. This paper introduces representative deep learning-based methods and presents a deep and thorough analysis in terms of accuracy and efficiency. We further compare and discuss the popular and challenging datasets and their evaluation metrics. A comprehensive comparison of several successful deep learning-based face detectors is conducted to uncover their efficiency using two metrics: FLOPs and latency. The paper can guide to choose appropriate face detectors for different applications and also to develop more efficient and accurate detectors.**

***Index Terms*—Face detection, computational performance, survey.**

I. Introduction

FACE detection, one of the most popular, fundamental and practical tasks in computer vision, is to detect human faces from images and return the spatial locations of faces via bounding boxes [1], as shown in Fig. 1. Starting with the Viola-Jones (V-J) detector [2] in 2001, the solution to face detection has been significantly improved from handcrafting features such as Haar-like features [2], to end-to-end convolutional neural networks (CNNs) for better feature extraction. Face detection is the first step for many face-related applications, such as face recognition, face tracking, facial expression recognition, facial landmarks detection and so on. Those technologies can achieve an overall better performance by faster and more accurate face detectors.

Fig. 1. Examples of face detection from WIDER Face [3]. A simple case (a) where there is only one clear frontal face. Common variations are in scale (b), pose (c), occlusion (d), expression (e), illumination (f). Red boxes are faces in extreme conditions.

Before deep learning was employed for face detection, the cascaded AdaBoost classifier was the dominant method for face detection. Some algorithms were specifically designed for face detection by using some kinds of features, such as Haar-like features [2], SURF [4] and Multi-Block LBP [5]. In recent years, deep learning has been proven to be more powerful for feature extraction and helps to achieve very impressive accuracy on object detection. Numerous object detection deep models have been designed for generic object detection which is much more challenging than face detection. Therefore, many models from face detection are adopted from or inspired by models for generic object detection. We can train a deep face detector directly using Faster R-CNN [6], YOLO [7] or SSD [8], and much better detection results can be obtained than traditional cascaded classifiers. Some similar works can be found, such as Face R-CNN [9] and Face R-FCN [10] which are modified and improved based on Faster R-CNN, R-FCN [11] respectively. Additionally, some other detectors, such as MTCNN [12], HR [13], SSH [14], are originally designed for face detection. Some techniques in generic object detection have also been adapted into face detection, such as the multi-scale mechanism from SSD, the feature enhancement from FPN [15], and the focal loss from RetineNet [16] according to the special pattern of human faces for face detection. These techniques lead to the proposal of various outstanding face detectors such as S^3FD [17], PyramidBox [18], SRN [19], DSFD [20], and RetinaFace [21].

Manuscript received November 7, 2020; revised February 28, 2021 and June 14, 2021; accepted September 15, 2021. Date of publication October 19, 2021; date of current version February 25, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61976144 and Grant 61806128; in part by the National Key Research and Development Program of China under Grant 2020AAA0140002; and in part by the Stable Support Plan Program of Shenzhen Natural Science Fund under Grant 20200925155017002. This article was recommended for publication by Associate Editor I. Kakadiaris upon evaluation of the reviewers' comments. *(Corresponding author: Shiqi Yu.)*

Yuantao Feng, Shiqi Yu, Hanyang Peng, and Jianguo Zhang are with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: yusq@sustech.edu.cn).

Yan-Ran Li is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China.

Digital Object Identifier 10.1109/TBIOM.2021.3120412

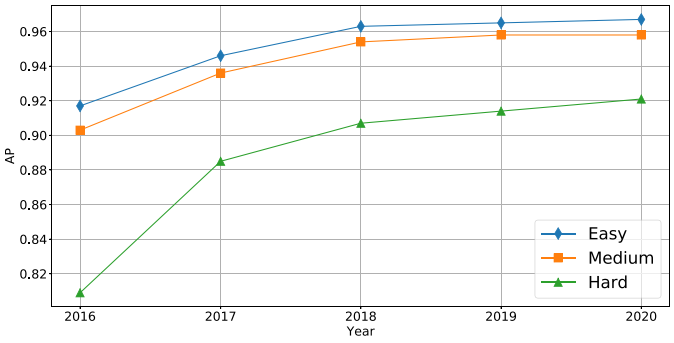


Fig. 2. The best AP on the easy, medium and hard subsets of WIDER Face [3] test set in the recent years.

TABLE I
DIFFERENT MODELS ADOPT DIFFERENT RANGES AND DIFFERENT PRESETS OF TEST SCALES. '0.25X' DENOTES SHRINKING THE WIDTH AND HEIGHT BY 0.25, AND OTHERS FOLLOW. SPECIFICALLY, 'SX' AND 'EX' ARE SHRINKING AND ENLARGING IMAGES ACCORDINGLY, WHILE 'FX' IS ENLARGING THE IMAGE INTO A FIXED SIZE. TEST IMAGE SIZES STAND FOR RE-SCALING THE SMALLER SIDE OF THE IMAGE TO THE GIVEN VALUE, AND THE OTHER SIDE FOLLOWS THE SAME RATIO

Model	Test image scales
HR,2017 [13]	0.25x, 0.5x, 1x, 2x
S3FD,2017 [17]	0.5x, 1x, Sx, Ex
SRN,2019 [19]	0.5x, 1x, 1.5x, 2.25x, Fx
DSFD,2019 [20]	0.5x, 1x, 1.25x, 1.75x 2.25x, Sx, Ex
CSP,2019 [25]	0.25x, 0.5x, 0.75x, 1x, 1.25x, 1.5x, 1.75x, 2x
Model	Test image sizes
SSH,2017 [14]	500, 800, 1200, 1600
SFA,2019 [26]	500, 600, 700, 800, 900, 1000, 1100, 1200, 1600
SHF,2020 [27]	100, 300, 600, 1000, 1400
RetinaFace, 2020 [21]	500, 800, 1100, 1400, 1700

Face detection is sometimes considered as a solved problem because the average precision (AP) on many face detection datasets such as PASCAL Face [22], AFW [23] and FDDB [24], has reached or exceeded 0.990 since 2017.[1] On the most popular and challenging WIDER Face dataset [3], the AP has reached 0.921 even on the hard test set.

But face detection is not a solved problem. If we observe the best results of each year in Fig. 2, we can find the AP is still improving but slowly in recent 3 years. Therefore, with such near-to-saturated performance improvement, one question would be asked: If a tiny improvement is achieved by a much heavier deep model with great computational cost, will we consider the model is a good one? If we look slightly deeper into the implementation of some recent models, we can find that multiple scaling is heavily used in the evalutions on WIDERFACE benchmark. If we resize the input image with many different scales, such as 1/4, 1/2, 1, 3/2, 2, 4 and more, and feed all those resized images into a detector, the combined results will have a better AP, which in another word is achieved by the assembling and suppressing (NMS) the multi-scale outputs, and is independent to the backbone of the underlying face detector. We listed the scales used by some models in Table I. None of them tested an image using only one scale. It is a trend that more scales are used recently. There is a risk that multiple scales with a heavy computational cost are employed, and outstanding accuracy is claimed, which overshadows the performance gain the by the detector itself and the computational cost by such a multi-scale operation is not known. It is also worth noting that most benchmarks do not evaluate the computational cost. Most often, it is difficult for users to know by which the improvement is achieved, a better backbone technology or the follow-up computational-intensive multi-scale ensemble strategy?

We do expect a perfect face detector which is robust and accurate even for some faces in extremely difficult conditions, while being extremely fast with low computational cost. However, we all know the *no free lunch theorem*. Therefore, in this survey, we investigate the recent deep learning based face detection methods and evaluate them in terms of accuracy and computational cost. The main contributions are as follows.

1) Different from previous face detection surveys [28], [29], [30], [31], [32] in which the content is mainly built on reviewing traditional methods, our survey focuses on deep learning-based face detectors. We have noted the existence of surveys [33], [34], [35] on deep learning; however, they focus on generic object detection, not specifically for face detection. In this paper, we provide a clear view of the path by which deep learning based face detection has evolved in recently years.
2) Accuracy and efficiency are both studied and analyzed in the paper. In addition to detailed introductions to deep learning based face detectors, some experiments are carried out to analyze different deep face detectors using different metrics. Some tricks to improve accuracy are also introduced. So the paper can help readers understand better how good accuracy and efficiency can be achieved.
3) With a focus on the efficiency of face detectors, comprehensive experiments are carried out to evaluate the accuracy and particularly efficiency of different face detectors. In addition to latency, we also propose an accurate metric for the computational cost of a CNN model. It is **FL**oating point **OP**eration**s** (FLOPs) under certain rules. FLOPs is more neutral than latency which heavily depends on hardware and deep network structure. The code to compute the FLOPs has been released in https://github.com/fengyuentau/PyTorch-FLOPs.git.

The rest of the paper is organized as follows. Some key challenges in face detection are summarized in Section II. In Section III, we provide a roadmap to describe the development of deep learning-based face detection with detailed reviews. In Section IV, we review several fundamental subproblems including backbones, context modeling, the handling of face scale variations and proposal generation. Popular datasets for face detection and state-of-the-art performances are presented in Section V. Section VI reveals the relation between computational cost and AP by conducting extensive experiments on several open-source one-stage face detectors. In addition, speed-focusing face detectors collected from Github are reviewed in Section VII. Finally, we conclude the paper with a discussion on future challenges in face detection in Section VIII.

[1] State-of-the-art AP can be found in the official result pages of the datasets, and https://paperswithcode.com/task/face-detection which also collects results from published papers.

II. Main Challenges

Most face-related applications need clear frontal faces. Detecting a clear frontal face is a relatively easy task. Some may argue that some faces are useless for the next step such as face recognition if the faces are tiny and with occlusion; but it is not. Effectively detecting any faces in extremely difficult conditions can greatly improve the perception capability of a computer but is still a challenging task. If a face is detected and evaluated as a bad quality sample, the subject can be suggested to be closer to the camera, or the camera can adjust automatically for a better image. Face detection is still a problem far from to be well solved. Many challenges do still exist.

Accuracy-related challenges are from face appearance and imaging conditions. In real-world scenes, there are many different kinds of face appearance, varying in different skin color, makeup, expression, wearing glasses or a mask and so on. In unconstrained environments, imaging a face can be impacted by various lighting, viewing angles and distances, backgrounds, and weather conditions. The face images will vary in illumination, pose, scale, occlusion, blur and distortion. The face samples in difficult conditions can be found in Fig. 1. There have been several datasets and competitions featuring face detection in unconstrained conditions, such as FDDB [24], WIDER Face [3] and WIDER Face Challenge 2019.[2] More than 45% of faces are smaller than 20× 20 pixels in WIDER. In most face-related applications, we seldom need small faces whose sizes are less than 20. However, if we can detect small or even tiny faces, we can resize the original large images to smaller ones and send them to a face detector. Then, the computational cost can be greatly reduced since we only need to detect faces in smaller images. Therefore, a better accuracy sometimes also means a higher efficiency.

Masked face detection is becoming more important since people are wearing and will continuously wear masks to prevent COVID-19 in the next few years. Face-related applications did not consider this situation in the past. Wearing masks will reduce the detection accuracy obviously. Some masks are even printed with some logos or cartoon figures. All those can disrupt face detection. If a face has a mask and sunglasses at the same time, face detection will be even more difficult. Therefore, in the next few years, masked face detection should be explored and studied.

Efficiency-related challenges are brought by the great demands on edge devices. Since the increasing demands on edge devices, such as smartphones and intelligent CCTV cameras, massive amount of data is generated per day. We frequently take selfies, photos of others, long video meetings, etc. Modern CCTV cameras record 1080P videos constantly at 30 FPS. These result in a great demand for facial data analysis, and the data is considerable. In contrast, edge devices have limited computational capability, storage and battery life to run advanced deep learning-based algorithms. In this case, efficient face detection is essential for face applications on edge devices.

[2]https://competitions.codalab.org/competitions/20146

III. Face Detection Frameworks

Before deep learning was used for face detection, cascaded AdaBoost-based classifiers were the most popular classifiers for face detection. The features used in AdaBoost were designed specifically for faces, not generic objects. For example, the Haar-like [2] feature can describe facial patterns of eyes, mouth and others. In recent years, facial features can be automatically learnt from data via deep learning techniques. Therefore, many deep learning-based face detectors are inspired by modern network architectures designed from object detection. Following the popular manner of organizing object detection frameworks, we organize deep learning-based face detectors into three main categories.

- Multi-stage face detection frameworks. It is inspired by cascaded classifiers in face detection and is an early exploration of applying deep learning techniques to face detection.
- Two-stage face detection frameworks. The first stage generates some proposals, and the proposals are confirmed in the second stage. The efficiency should be better than multi-stage ones.
- One-stage face detection frameworks. Feature extraction and proposal generation are performed in a single unified network. These frameworks can be further categorized into anchor-based methods and anchor-free methods.

To show how the deep learning-based face detection evolves, milestone face detectors and some important object detectors are plotted in Fig. 3. The two-stage and multi-stage face detectors are on the top branch, and the single-stage ones are on the bottom branch. The generic object detectors are in the middle branch and in blue. A More detailed introduction of those detectors is provided in the following subsections.

A. Multi-Stage and Two-Stage Face Detectors

In the early era when deep learning techniques entered face detection, face detectors were designed to have multiple stages, also known as the cascade structure which has been widely used in most early face detectors. With the remarkable breakthrough brought by Faster R-CNN [6], some researchers turned to improve Faster R-CNN based on face data.

In the cascade structure, features are usually extracted and refined one or multiple times before being fed into classifiers and regressors, so as to reject most of the sliding windows to improve efficiency. As shown on the result page[3] of FDDB [24], Li *et al.* made an early attempt and proposed their CNN-based face detector, named **CascadedCNN** [36]. CascadeCNN consists of 3 stages of CNNs, as shown in Fig. 4. Sliding windows are first resized to 12 × 12 pixels and fed into the shallow 12-net to reduce candidate windows by 90%. The remaining windows are then processed by the 12-calibration-net to refine the size for face localization. Retained windows are then resized to 24 × 24 as the input for the combination of 24-net and 24-calibration-net, and so on for the next CNNs combination. CascadeCNN achieved state-of-the-art performance on AFW [23] and FDDB, while reaching a

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

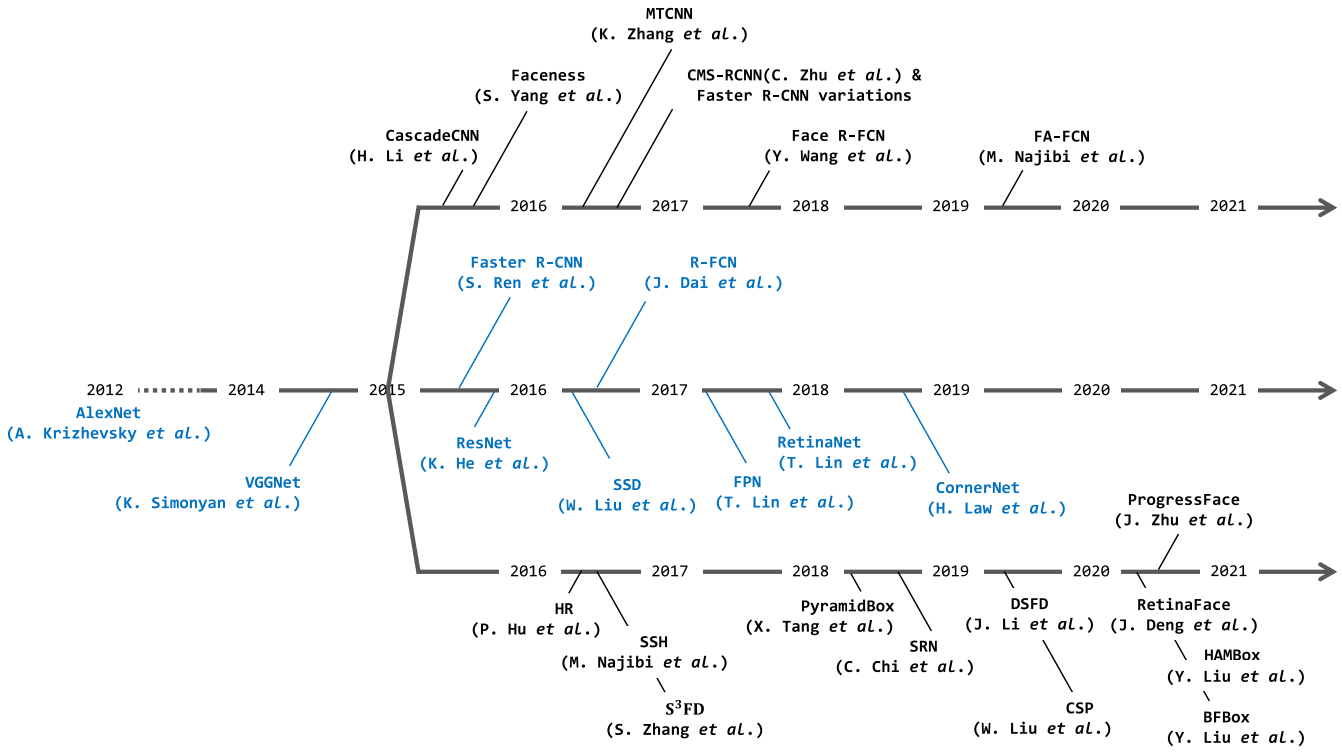


Fig. 3. Timeline of milestone face detectors [10], [12], [13], [14], [17], [18], [19], [20], [21], [25], [36], [37], [38], [39], [40], [41], [42], and remarkable works from object recognition [43], [44] and object detection [6], [8], [11], [15], [16], [45] (marked as blue, attached to the middle branch). Since the proposal of AlexNet [46], various face detection works inspired by deep learning techniques from object recognition and object detection were published in the 2012-post deep learning-based face detection era. The top branch is two/multi-stage face detectors, while the bottom branch is one-stage detectors, which has become the most popular network design adopted by researchers.

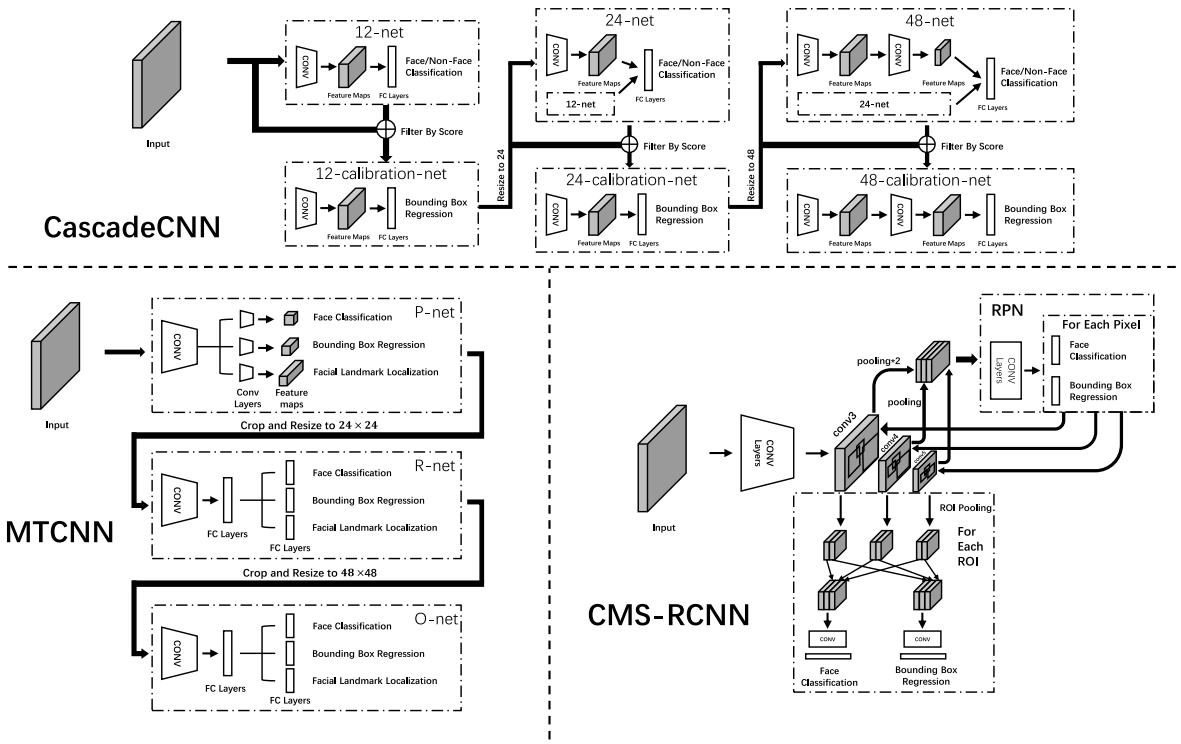


Fig. 4. Diagrams of milestone multi/two-stage face detectors [12], [36], [38]. Others share similar architectures as the three.

compelling speed of 14 FPS for the typical 640 × 480 VGA images on a 2.0 GHz CPU. Another attempt at cascaded CNNs for face detection is the well-known **MTCNN** [12] proposed by Zhang *et al.* MTCNN is composed of 3 subnetworks, which are P-Net for obtaining candidate facial windows, R-Net for rejecting false candidates and refining remaining candidates,

O-Net for producing the final output with both face bounding boxes and landmarks in the multi-task manner. P-Net is a shallow fully convolutional network with 6 `CONV` layers, which can take images of any sizes as input. MTCNN was a great success with large and state-of-the-art advantages on WIDER Face [3], FDDB and AFW, while reaching 16 fps on a 2.6 GHz CPU.

In the object-detection-fashion two-stage network architectures, a region proposal network (RPN) [6] is required to generate object proposals. RPN can be considered as a straightforward classification CNN, which generates proposals based on the preset anchors on CNN features, filters out non objects and refines object proposals. However, as the CNNs shrink the image to extract features, the corresponding output features for tiny faces can be less than 1 pixel, making it insufficient to encode rich information. To address this problem, Zhu *et al.* proposed **CMS-RCNN** [38], which is equiped with a contextual multi-scale design for both RPN and final detection. As shown in Fig. 4, multi-scale features from *conv3*, *conv4* and *conv5* are concatenated by shrinking them into the same shape with *conv5* as the input for RPN, so as to collect more information for tiny faces and also improve the localization capability from low-level layers. CMS-RCNN achieved an AP of 0.899, 0.874, 0.624 on the easy, medium and hard sets of the WIDER Face dataset respectively, outperforming MTCNN by 0.051(Easy), 0.049(Medium) and 0.016(Hard).

In addition to CMS-RCNN, there are others making improvements based on Faster R-CNN. **Bootstrapping Faster R-CNN** [47] builds a training dataset by iteratively adding false positives from a model's output to optimize Faster R-CNN. **Face R-CNN** [9] adopts the same architecture as Faster R-CNN with center loss, online hard example mining and multi-scale training strategy. **FDNet** [48] exploits multi-scale training and testing and a vote-based NMS strategy on top of Faster R-CNN with a light-head design. Position-sensitive average pooling was proposed in **Face R-FCN** [10] to assign different weights to different parts of the face based on R-FCN [11]. With the improvements considering the special patterns of face data, these methods achieved better performance than their original version on the same WIDER Face dataset.

Whether it is the cascaded multi-stage or two-stage network design, its computation is heavily dependent on the number of faces in the image, the increase in which also increases proposals passed to the next stage in the interior of the network. Notably, the multi-scale test metric, which usually enlarges the images multiple times to make tiny faces detectable, can dramatically increase the computational cost on this basis. Considering that the number of faces in the image from the actual scene varies from one face in a selfie to many faces in a large group photo, we consider the robustness of cascade or two-stage networks in terms of runtime.

B. One-Stage Face Detectors

In real-time face-related applications, face detection must be performed in real time. If the system is deployed on edge devices, the computing power is low. In those kinds of situations, one-stage face detectors are more suitable since their process time is stable regardless of how many faces there are in images. Different from the multi/two-stage detectors, the one-stage face detectors perform feature extraction, proposal generation and face detection in a single and unified convolutional neural network, whose runtime efficiency is independent of the number of faces. Dense anchors are designed to replace proposals in two-stage detectors [14]. Starting from CornerNet [45], an increasing number of works use the anchor-free mechanism in their frameworks.

HR [13] proposed by Hu and Ramanan is one of the first to perform anchor-based face detection in a unified convolutional neural network. The backbone of HR is ResNet-101 [44] with layers truncated after `conv4_5`. Early feature fusion on layers `conv3_4` and `conv4_5` is performed to encode context since high-resolution features are beneficial for small face detection. Through experiments on faces clustered into 25 scales, 25 anchors are defined for 2X, 1X and .5X inputs, to achieve the best performance of three input scales. HR outperformed CMS-RCNN [38] by 0.199 on the WIDER Face validation hard set, and more importantly, the run-time of HR is independent of the number of faces in the image, while CMS-RCNN's linearly scale up with the number of faces.

Different from HR, **SSH** [14] attempts to detect faces at different scales on different levels of features, as shown in Fig. 5. Taking VGG-16 [43] as the backbone, SSH detects faces on the enhanced features from `conv4_3`, `conv5_3` and `pool5` for small, medium and large faces respectively. SSH introduces a module (SSH module) that greatly enriches receptive fields to better model the context of faces. The SSH module is widely adopted by later works [18], [20], [21], [40], which turns out to be efficient for performance boosting.

Since **S^{3}FD** [17], many one-stage face detectors [18], [19], [20], [21], [25], [40], [41], [42] fully utilize multi-scale features attempting to achieve scale-invariant face detection. S^{3}FD extends the headless VGG-16 [43] with more convolutional layers, whose stride gradually doubles from 4 to 128 pixels, so as to cover a larger range of face scales. **PyramidBox** [18] adopts the same backbone as S^{3}FD, integrates FPN [15] to fuse adjacent-level features for semantic enhancement, and improves the SSH module with wider and deeper convolutional layers inspired by Inception-ResNet [49] and DSSD [50]. **DSFD** [20] also inherits the backbone from S^{3}FD, but enhances the multi-scale features by the Feature Enhance Module (FEM), so that detection can be made on two shots - one from non-enhanced multi-scale features, and the other from the enhanced features. The same scale features from the second shot have larger RFs than those from the first shot, but also have smaller RFs than the next-level features from the first shot, indicating that the face scales are split more refined across these multi-scale detection layers. Similarly, **SRN** [19] has a dual-shot networks but is trained differently on multi-scale features: low-level features need two-step classification to refine, since they have higher resolution and contribute the vast majority of anchors and also negative samples; additionally, high-level features have lower resolution which is worth two-step regression using the Cascade R-CNN [51] to have more accurate bounding boxes.

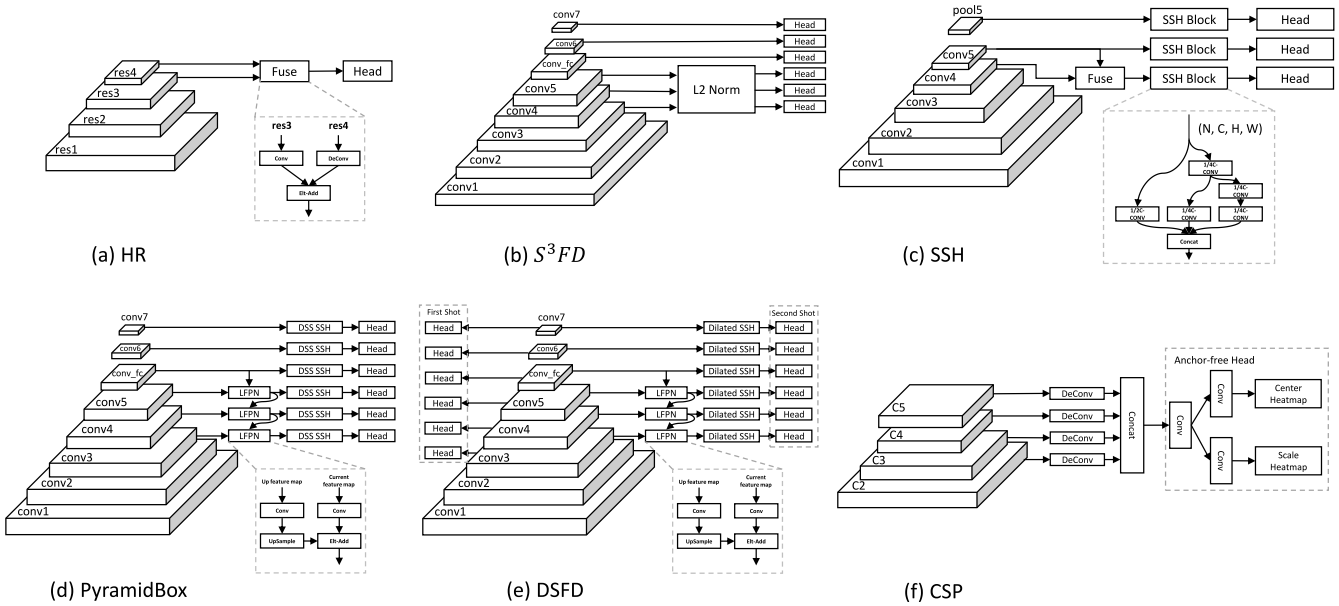


Fig. 5. Diagrams of milestone one-stage face detectors [13], [14], [17], [18], [20], [25].

There are also some significant anchor-based methods using the FPN [15] as the backbone. **RetinaFace** adds one more pyramid layer on top of the FPN and replaces CONV layers with the deformable convolution network (DCN) [52], [53] within FPN's lateral connections and context module. RetinaFace models a face in three ways: a 3D mesh (1k points), a 5-landmark mask (5 points), and a bounding box (2 points). Cascade regression [51] is employed with multi-task loss in RetinaFace to achieve better localization. Instead of using the handcrafting structures, Liu *et al.* proposed **BFBox**, which explores face-appropriate FPN architectures using the successful Neural Architecture Search (NAS). Liu decouples FPN as the backbone and FPN connections, the former of which can be replaced by VGG [43], ResNet [44] or the backbone from NAS, and the latter of which can be top-down, bottom-up or cross-level fusion from NAS.

Since the proposal of CornerNet [45] back in 2018, which directly predicts the top-left and bottom-right points of bounding boxes instead of relying on prior anchors, many explorations [54], [55], [56], [57] have been made to remodel object detection more semantically using the anchor-free design. **CSP** models a face bounding box as a center point and the scale of the box as shown in Fig. 5. CSP takes multi-scale features from the modified ResNet-50 [44], and concatenates them to take the advantage of rich global and local information for detection heads using transpose convolution layers. In particular, the anchor-free detection head can also be an enhancement module for anchor-based heads. **ProgressFace** [42] appends an anchor-free module to provide more positive anchors for the highest resolution feature maps in FPN, so as to reduce the imbalance of positive and negative samples for small faces.

One-stage frameworks are popular on face detection in recent years for the following three reasons. (a) The runtime of one-stage face detectors is independent of the number of faces in an image by design. Therefore, it enhances the robustness of runtime efficiency. (b) It is computationally efficient and straightforward for one-stage detectors to reach near scale invariance by contextual modeling and multi-scale feature sampling. (c) Face detection is a relatively less complex task than general object detection. This means that innovations and advanced network designs in object detection can be quickly adjusted to face detection by considering the special pattern of faces.

IV. Face Representation

The key idea of face detection has never changed whether it is in the traditional era or deep learning era. It finds the common patterns of all faces in the dataset. In the traditional era, many of handcrafted features, such as SIFT [58], Haar [2] and HOG [59], are employed to extract local features from the image, which are aggregated by approaches such as AdaBoost for the higher-level representation of faces.

Different from traditional methods, which require rich prior knowledge to design handcrafted features, deep convolutional neural networks can directly learn even more powerful features from face images. A deep learning-based face detection model can be considered as two parts: a CNN backbone and several detection branches. Starting from some popular CNN backbones, the feature extraction methods that can handle face scale invariance are introduced as well as several strategies to generate proposals for face detection.

A. Popular CNN Backbones

In most deep face detectors there is a CNN backbone for feature extraction. Some popular backbone networks are listed in Table II. They are VGG-16 from the VGGNet [43] series, ResNet-50/101/152 from the ResNet [44] series, and MobileNet [60]. The models are powerful and can achieve good accuracy on face detection, but they are a little heavy.

Early attempts on deep learning-based face detection were cascaded structures that did not take the above CNN

TABLE II
CNN BACKBONES COMMONLY USED BY MODERN DEEP LEARNING-BASED FACE DETECTORS. FC LAYERS OF THESE CNNS ARE IGNORED WHEN CALCULATING '#CONV LAYERS', '#PARAMS' AND 'FLOPS'. THE INPUT SIZE FOR CALCULATING 'FLOPS' IS 224 × 224. THE CALCULATION OF FLOPS IS DISCUSSED IN SECTION VI. 'TOP-1 ERROR' REFERS TO THE PERFORMANCE ON THE IMANGENET [61] VALIDATION SET. NOTE THAT 9 OF THE 20 CONV LAYERS IN MOBILENET [60] ARE DEPTH-WISE

CNN Backbones	#CONV Layers	#Params ($\times 10^6$)	FLOPs ($\times 10^9$)	Top-1 Error
VGG-16	13	14.36	30.72	28.07%
ResNet-50	52	23.45	8.25	22.85%
ResNet-101	136	42.39	15.72	21.75%
ResNet-152	188	56.87	23.19	21.43%
MobileNet	20	3.22	1.28	29.40%

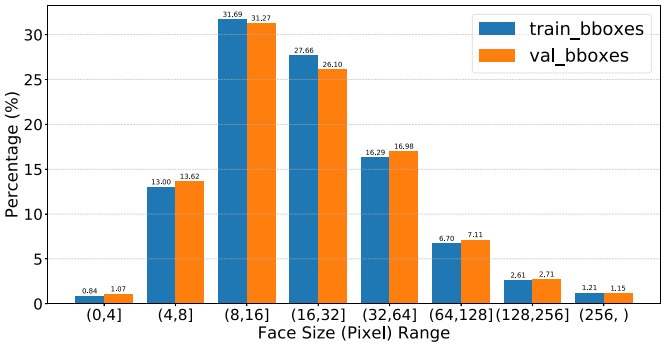


Fig. 6. The distribution of face scales on WIDER Face [3] dataset.

architectures. Even some simple structured CNN is much more computational heavy than AdaBoost, cascaded CNN is computational heavy also. With breakthroughs in object detection, some of the techniques have been borrowed and applied on face detection. VGG-16 [43] has 13 CONV layers, which is the first choice for the baseline backbones for many face detectors, such as SSH [14], S^3FD [17] and PyramidBox [18]. Performance improvements can easily be obtained by simply swapping the backbone from VGG-16 to ResNet-50/101/152 [44], as shown in [20]. Since state of the arts have achieved AP >0.900 even on WIDER Face hard sets, it is common for recent face detectors [20], [42], [62] to equipe with a deeper and wider backbone for higher AP, such as the ResNet-152 and ResNets with FPN [15] connections. Liu *et al.* employs Neural Architecture Search (NAS) to search face-appropriate backbones and FPN connections.

One of the most inexpensive choices is ResNet-50 which is listed in Table II, which has less parameters and less FLOPs, while achieving very similar performance compared to deeper nets. Another choice for state-of-the-art face detectors to reach real-time speed is to change the backbone to MobileNet [60], which has similar performance to VGG-16 but one order of magnitude less in '#Params' and FLOPs.

B. Towards Face Scale Invariance

One of the major challenges for face detection is the large span at face scales. As statistics shown in Fig. 6, there are 157,025 and 39,123 face bounding boxes in the train and validation set respectively, both of which have more than 45%

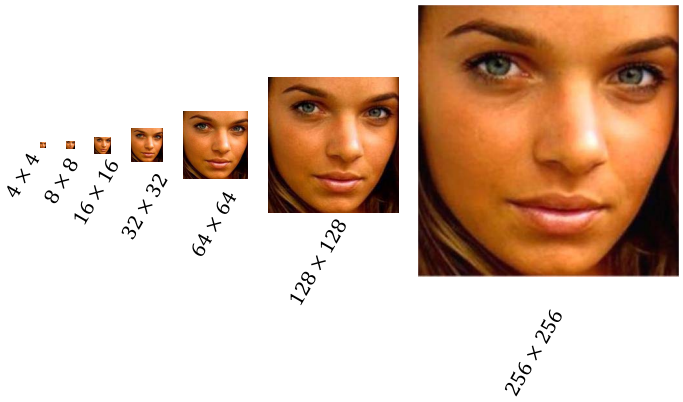


Fig. 7. A face in different scales. Could you tell the images of sizes 4 × 4, 8 × 8 contain a face?

of face bounding boxes are 16 × 16 and smaller, and a non-negligible 1% are 256 × 256 and larger. We choose these scales to perform clustering to match the strides of feature maps selected for detection; for example there is only 1 pixel in the feature maps of stride 4 for encoding a face of size equal to or less than 4 × 4. We also present the visual differences among scales in Fig. 7. It is challenging even for humans to tell whether the image of size 16 × 16 contains a face. In the following, we describe the mechanism of face detectors towards face scale invariance even with tiny faces.

Most of the modern face detectors are anchor-based. Anchors are predefined boxes of different scales and aspect ratios attached to each pixel in the feature maps, which serve as the proposal to match with the ground truth faces. More details about anchors are provided in Section IV-C. As [17] noted, since the predefined anchor scales are discrete while the face scales in the wild change continuously, outer faces whose scales are distributed away from anchor scales cannot match enough anchors. It will result in a low recall rate. A simple solution for a trained face detector is to perform multi-scale test on an image pyramid, which is built by progressively resizing the original image. It is equal to re-scale faces and hopefully brings outer faces back into the detectable range of scales. This solution does not require retraining the detector, but it may come with a sharp increase in redundant computation, since there is no certain answer to how deep the pyramid we should build to match with the certain extent of scale invariance of a trained CNN.

Another better solution to face scale invariance is to make full use of the feature maps produced in CNNs. One can easily observe that the layers of standard CNN backbones gradually decrease in size. The subsampling of these layers naturally builds up a pyramid with different strides and receptive fields (RFs). It produces multi-scale feature maps. In general, high-level feature maps produced by later layers with large RFs are encoded with strong semantic information, and lead to its robustness to variations such as illumination, rotation and occlusion. Low-level feature maps produced by early layers with small RFs are less sensitive to semantics, but have high resolution and rich details, which are beneficial for localization. To take both the advantages, a number of methods are

proposed, which can be categorized into **modeling context**, **detecting on a feature pyramid**, and **predicting face scales**.

Modeling context: Additional context is essential for detecting faces, especially for detecting small ones. HR [13] shows that context modeling by fusing feature maps of different scales can dramatically improve the accuracy of detecting small faces. Following a similar fusion strategy as HR, [27] detects on three different dilated CONV branches, aiming to enlarge RF without too much increase in computation. Reference [38] downsamples feature maps of strides 4 and 8 to concatenate with those of stride 16, so as to improve the capability of the RPN to produce proposals for faces at different scales. SSH [14] exploits an approach similar to Inception [63], which concatenates the output from three CONV branches that have 3×3, 5×5 and 7×7 filters respectively. PyramidBox [18] first adopts an FPN [15] module to build up context and is further enhanced by deeper and wider SSH modules. Reference [20] improves the SSH module by replacing CONV layers with dilated CONV layers. Reference [25] upsamples feature maps of strides 8, 16 to concatenate with those of stride 4, which is fed to an FCN to produce center, scale and offset heatmaps. The fusion of feature maps encodes rich semantics from high-level feature maps with rich geometric information from low-level feature maps, based on which the detectors can improve their capability of localization and classification towards face scale invariance. Meanwhile, the fusion of feature maps also introduces more layers, such as CONV and POOL to adjust scales and channels, which creates additional computational overhead.

Detecting on a Feature Pyramid: Inspired by SSD [8], a majority of recent approaches, such as [14], [17], [18], [19], [20], [21], detect at multiple feature maps of different scales respectively, and combine detection results. It is considered to be an effective method for weighing between speed and accuracy. SSD [8] puts default boxes on each pixel of the feature maps from 6 detection layers that have strides of 8, 16, 32, 64 and 128. Sharing a similar CNN backbone with SSD, [17], [18] detect on a wider range of layers, which have strides gradually doubling from 4 to 128 pixels. SRN [19] and DSFD [20] introduce the two-stream mechanism, which detects on both the detection layers from the backbone and extra layers applied on the detection layers for feature enhancement. Different from subsampling on more layers, [14], [21], [26] detects only at the last three level feature maps, which are enhanced by their context modeling methods. By detecting on a feature pyramid, detection layers are implicitly trained to be sensitive to different scales, while it also leads to an increase in model size and redundant computation, since the dense sampling may cause some duplicate results from adjacent-level layers.

Predicting Face Scales: To eliminate the redundancy from pyramids, several approaches [64], [65], [66] predict the face scales before making a detection. Reference [64] first generates a global face scale histogram from the input image by the Scale Proposal Network (SPN), which is trained with image-level ground truth histogram vectors and without face location information. A sparse image pyramid is built according to the output histogram, so as to have faces rescaled to the detectable range of the later single-scale RPN. Similarly, [65] detects on a feature pyramid without unnecessary scales, which is built by using the scale histogram to a sequential ResNet [44] blocks that can downsample feature maps recursively. Reference [66] predicts not only face scales but also face locations by a shallow ResNet18 [44] with scale attention and spatial attention attached, named S^2AP. S^2AP generates a 60-channel feature map, meaning face scales are mapped to 60 bins, each of which is a spatial heatmap that has high response to its responsible face scale. With the 60-channel feature maps, it is possible to decrease the unnecessary computation with the low-response channels and the low-response spatial areas by a masked convolution.

C. Proposal Generation

Faces in the wild can be of any possible locations and scales in the image. The general pipeline for most of the early successful face detectors, is to first generate proposals in the sliding-window manner, extract features from the windows using handcrafted descriptors [2], [23], [67], [68] or CNNs [12], [36], and finally apply face classifiers. However, inspired by RPN [6] and SSD [8], modern anchor-based face detectors generate proposals by applying k anchor boxes on each pixel of the extracted CNN features. Specifically, 3 scales and 3 aspect ratios are used in Faster R-CNN [6], yielding $k = 9$ anchors on each pixel of the feature maps. Moreover, the detection layer takes the same feature maps as input, yielding $4k$ outputs encoding the coordinates for k anchor boxes from the regressor and $2k$ outputs for face scores from the classifier.

Considering that most of the face boxes are near square, modern face detectors tend to set the aspect ratio of anchors to 1, while the scales depends. HR [13] defines 25 scales so as to match the cluster results on the WIDER Face [3] training set. S^3FD assigns the anchor scale of 4 times the stride of the current layer to keep anchor sizes smaller than effective receptive fields [69] and ensure the same density of different scale anchors on the image. PyramidBox [18] introduces PyramidAnchors, which generates a group of anchors with larger regions corresponding to a face, such as head and body boxes, to have more context to help detect faces. In [70], extra shifted anchors are added to increase the anchor sample density, and significantly increased the average IoU between anchors and small faces. GroupSampling [71] assigns anchors of different scales only on the bottom pyramid layer of FPN [15], but it groups all training samples according to the anchor scales, and randomly samples from groups to ensure the positive and negative sample ratios between groups are the same.

V. Datasets and Evaluation

To evaluate different face detection algorithms, datasets are needed. There have been several public datasets, which are FDDB [24], AFW [23], PASCAL Face [22], MALF [74], WIDER Face [3], MAFA [75], 4K-Face [79], UFDD [80] and DARK Face [81]. These datasets all consist of colored images from real-life scenes. Different datasets may utilize different evaluation criterion. In Section V-A, we present overviews of

TABLE III
COMPARISON OF CURRENTLY ACCESSIBLE FACE DETECTION DATASETS, LISTED IN THE ORDER OF PUBLICATION OR STARTED YEAR. NOTE THAT UCCS [72] AND WILDEST FACE [73] ARE NOT INCLUDED BECAUSE THEIR DATA IS NOT CURRENTLY AVAILABLE. 'BLUR', 'APP.', 'ILL.', 'OCC.', 'POSE' IN THE 'VARIATIONS' COLUMNS DENOTE BLUR, APPEARANCE, ILLUMINATION, OCCLUSION AND POSE RESPECTIVELY

Dataset	#Images	#Faces	#Faces Per Image	AVG Resolution ($W \times H$)	Split			Variations				
					Train	Val	Test	Blur	App.	Ill.	Occ.	Pose
FDDB [24]	2,845	5,171	1.8	377×399	-	-	100%	✓			✓	✓
AFW [23]	205	468	2.3	1491×1235	-	-	100%		✓			✓
PASCAL Face [22]	851	1,335	1.5	-	-	-	100%					
MALF [74]	5,250	11,931	2.2	-	-	-	100%		✓		✓	✓
WIDER Face [3]	32,203	393,703	12.2	1024×888	40%	10%	50%	✓	✓	✓	✓	✓
MAFA [75]	30,811	39,485	1.2	516×512	85%	-	15%		✓		✓	
IJB-A [76]	48,378	497,819	10.2	1796×1474	50%	-	50%		✓	✓	✓	✓
IJB-B [77]	76,824	135,518	1.7	894×599	-	-	100%		✓	✓	✓	✓
IJB-C [78]	138,836	272,335	1.9	1010×671	-	-	100%		✓	✓	✓	✓
4K-Face [79]	5,102	35,217	6.9	3840×2160	-	-	100%					
UFDD [80]	6,425	10,897	1.6	1024×774	-	-	100%	✓		✓		
DARK Face [81]	6,000	43,849	7.3	1080×720	100%	-	-			✓		

different datasets covering some statistics such as the number of images and faces, the source of images, the rules of labeling and challenges brought by the dataset. A detailed analysis of the face detection evaluation criterion is also included in Section V-B. Detection results on the datasets are provided and analyzed in Section V-C.

A. Datasets

Some essential statistics of currently accessible datasets are summarized in Table III including the total number of images and faces, faces per image, how the data was splitted different sets, etc. More details are introduced in the following part.

FDDB[4] [24] is short for **F**ace **D**etection **D**ataset and **B**enchmark, which has been one of the most popular datasets for face detector evaluation since its publication in 2010. The images of FDDB were collected from Yahoo! News, 2,845 of which were selected after filtering out duplicate data. Faces were excluded with these factors, (a) height or width less than 20 pixels, (b) the two eyes being non-visible, (c) the angle between the nose and the ray from the camera to the head being less than 90 degrees, (d) failure estimation on position, size or orientation of faces by a human. This led to 5,171 faces left, which were annotated by drawing elliptical face regions covering from the forehead to the chin vertically, and the left cheek to the right cheek horizontally. FDDB helped advance uncontrained face detection in terms of the robustness of expression, pose, scale and occlusion. However, its images can be heavily biased toward celebrity faces since they were collected from the news. It is also worth noting that although the elliptical style of the face label adopted by FDDB is closer to human cognition, it is not adopted by later datasets and deep learning-based face detectors, which favor the bounding box style with a relatively easier method for defining positive/negative samples by calculating the Intersection over Union (IoU).

Zhu and Ramanan built an annotated faces in-the-wild (**AFW**)[5] dataset [23] by randomly sampling images with at least one large face from Flickr. 468 faces were annotated from 205 images, each of which is labeled with a bounding box and 6 landmarks. **PASCAL Face**[6] [22] was constructed by selecting 851 images from the PASCAL VOC [1] test set with 1,335 faces annotated. Since the two datasets were built to help evaluate the face detectors proposed by [23] and [1], they only contain a few hundred images, resulting in limited variations in face appearance and background.

Yang *et al.* created the **M**ulti-**A**ttribute **L**abelled **F**aces [74] (**MALF**)[7] dataset for fine-grained evaluation on face detection in the wild. The MALF dataset contains 5,250 images from Flickr and Baidu Search with 11,931 faces labeled, which is an evidently larger dataset than FDDB, AFW and PASCAL Face. The faces in MALF were annotated by drawing axis-aligned square bounding boxes, attempting to contain a complete face with the nose in the center of the bounding box. This may introduce noise for training face detectors since a square bounding box containing a 90-degree side faces can have over half of its content being cluttered background. In addition to labeling faces, some attributes were also annotated, such as gender, pose and occlusion.

In 2016, **WIDER Face**[8] [3] was released, which has been the most popular and widely used face detection benchmark. The images in WIDER Face were collected from popular search engines for predefined event categories following LSCOM [82] and examined manually to filter out similar images and images without faces, resulting in 32,203 images in total for 61 event categories, which were split into 3 subsets for training, validation testing set. To keep large variations in scale, occlusion and pose, the annotation was performed following two main policies: (a) a bounding box should tightly contain the forehead, chin and cheek and is drew for each recognizable face and (b) an estimated bounding box should be drawn for an occluded face, producing 393,703 annotated faces in total. The number of faces per image reaches 12.2 and 50% of the faces are of height between 10-50 pixels. WIDER Face outnumbers other datasets in Table III by a large margin. It means WIDER Face pays

[4]http://vis-www.cs.umass.edu/fddb/
[5]http://www.cs.cmu.edu/ deva/papers/face/index.html
[6]http://host.robots.ox.ac.uk/pascal/VOC/
[7]http://www.cbsr.ia.ac.cn/faceevaluation/
[8]http://shuoyang1213.me/WIDERFACE/

never-seen-before attention to small faces detection by providing a large number of images with the densest small faces for training, validation and testing. Furthermore, the authors of WIDER Face defined 'easy', 'medium' and 'hard' levels for the validation and test sets based on the detection rate of EdgeBox [83]. It offers a much more detailed and fine-grained evaluation for face detectors. Hence, the WIDER Face dataset greatly advances the researches of CNN based face detectors, especially the multi-scale CNN designs and utilization of context.

The last four datasets listed in Table III are less generic than those reviewed above, and focus on face detection in specified and different aspects. The **MAFA**[9] [75] dataset focuses on masked face detection, containing 30,811 images with 39,485 masked faces labeled. In addition to the location of eyes and masks, the orientation of the face, the occlusion degree and the mask type were also annotated for each face. The IJB series[10] [76], [77], [78] were collected for multiple tasks, including face detection, verification, identification, and identity clustering. The IJB-C is the combination of IJB-A and IJB-B with some new face data. **4K-Face**[11] [79] was built for the evaluation of large face detection, and contains 5,102 4K-resolution images with 35,217 large faces (>512 pixels). **UFDD**[12] [80] provides a test set with 6,425 images and 10,897 faces in the variation of different weather conditions and degradtion such as lens impediments. **DARK Face**[13] [81] concentrates on face detection in low light conditions, and provides 6,000 low-light images for training dark face detector. Since the images are captured in real-world nighttime scenes such as streets, each image in DARK Face contains 7.3 faces on average which is relatively dense.

B. Accuracy Evaluation Criterion

There are mainly two accuracy evaluation criteria adopted by the datasets reviewed above, one of which is the receiver operating characteristic (ROC) curve obtained by plotting the true positive rate (TPR) against false positives such as those adopted by FDDB [24], MALF [74], UCCS [72] and IJB [78], the other of which is the most popular evaluation criterion from PASCAL VOC [1] by plotting the precision against recall while calculating average precision (AP), such as those adopted by AFW [23], PASCAL Face [22], WIDER Face [3], MAFA [75], 4K-Face [79], UFDD [80], DARK Face [81] and Wildest Face [73]. Since these two kinds of evaluation criterion are two different methods for revealing the performance of detectors under the same calculation of the confusion matrix,[14] we choose the most popular evaluation criteria AP calculated from the precision-again-recall curve in the paper.

To get a precision-again-recall curve, the confusion matrix, which is to define the true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) from the detection and ground truths, should be firstly calculated. A true positive is a detection result matched with a ground truth; otherwise, it is a false positive. The unmatched ground truths are defined as the false negatives. True negatives are not applied here since the background can be a large part of the image. To define whether two regions are matched or not, the commonly used intersection over union (IoU), also known as the Jaccard overlap, is applied:

$$IoU = \frac{area(P) \cap area(GT)}{area(P) \cup area(GT)} \tag{1}$$

where P is the predicted region, and GT is the ground truth region. In a widely used setting, the IoU threshold is set to 0.5, meaning if the IoU of a predicted region and a ground truth region is greater than or equal to 0.5, the predicted region is marked as matched and thus a true positive, otherwise it is a false positive.

After determining true or false positives for each detection, the next step is to calculate the precision and recall from the detection result list sorted by score in descending order to plot the precision-against-recall curve. A granular confidence gap can be defined to sample more precision and recall, but for a simple explanation, we define the gap as a detection result. In nth sampling, we calculate the precision and recall from the top-n detection results:

$$Precision_n = \frac{TP_n}{TP_n + FP_n} \tag{2}$$

$$Recall_n = \frac{TP_n}{TP_n + FN_n} \tag{3}$$

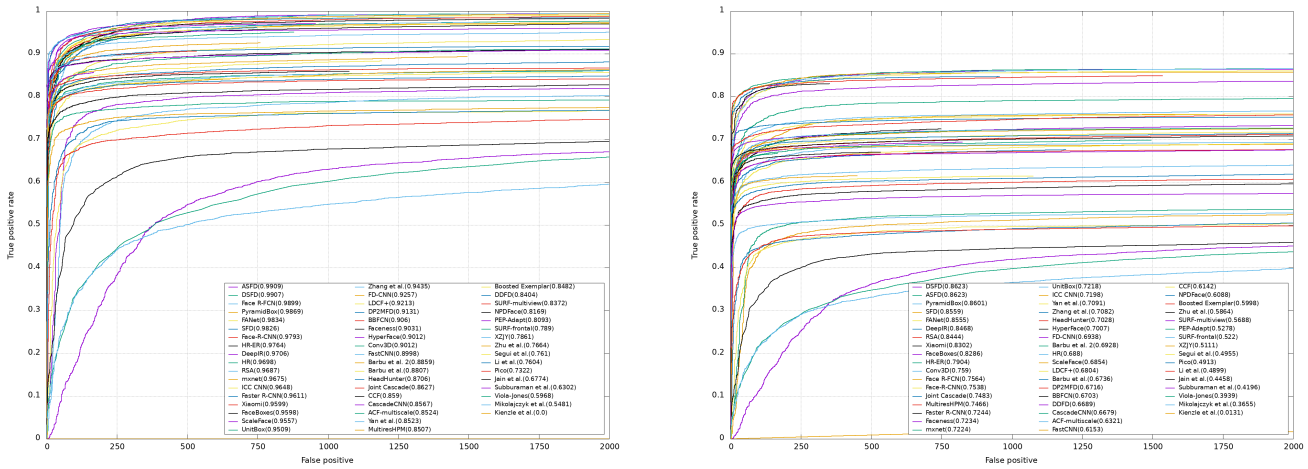
where TP_n, FP_n and FN_n are true positives, false positives and false negatives from the top-n results respectively. Let us say we have 1,000 detection results; then, we have 1,000 pairs of $(recall_i, precision_i)$ which are enough for plotting the curve.

We can compute the area under the precision-against-recall curve, which is AP, to represent the overall performance of a face detector. Under the single IoU threshold setting of 0.5 in WIDER Face evaluation, the top AP for the hard test subset of WIDER reached 0.924. In the WIDER Face Challenge 2019 which uses the same data as the WIDER Face dataset but evaluates face detectors in 10 IoU thresholds of 0.50:0.05:0.95, the top average AP reaches 0.5756.

C. Results on Accuracy

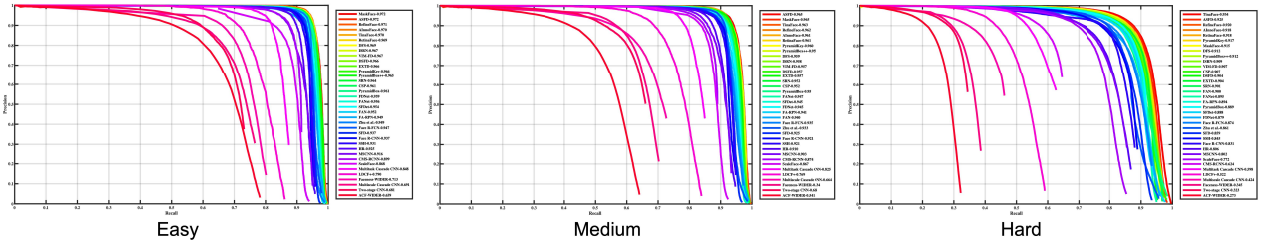
To understand the progress in recent years on face detection, the results of different datasets are collected from their official homepages. Because of space limitations, only the results from the two most popular datasets are listed. They are Fig. 8 for FDDB [24] and Fig. 9 for WIDER Face [3]. The FDDB results since 2004 are listed. The current ROC curves are much better than those in the past. This means that the detection accuracy is much higher than in the past. The true positive rate is reaching 1.0. If you look into the samples in FDDB, you can find there are some tiny and blur faces in the ground truth data. Sometimes it is hard to decide whether they should be faces, even by humans. Therefore, we can say that the current

[9] http://www.escience.cn/people/geshiming/mafa.html
[10] https://www.nist.gov/programs-projects/face-challenges
[11] https://github.com/Megvii-BaseDetection/4K-Face
[12] https://ufdd.info
[13] https://flyywh.github.io/CVPRW2019LowLight/
[14] https://en.wikipedia.org/wiki/Confusion_matrix

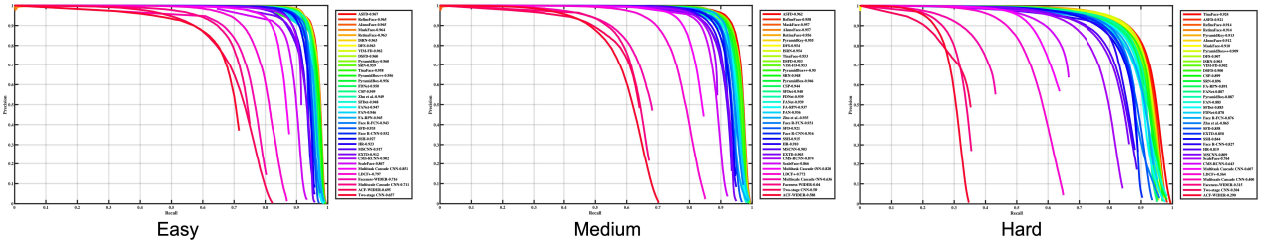


(a) Discontinuous ROC curves (b) Continuous ROC curves

Fig. 8. The results on the FDDB dataset, which are from the result page of FDDB http://vis-www.cs.umass.edu/fddb/results.html.



(a) WIDER Face Validation Set



(b) WIDER Face Test Set

Fig. 9. The results on the WIDER Face validation and test sets. The figures are from WIDER face homepage http://shuoyang1213.me/WIDERFACE/.

detectors achieve perfect accuracy on FDDB, and almost all faces can be detected.

The WIDER face is newer, larger and more challenging than FDDB. Most recent face detectors have been tested with it. From Fig. 9, it can be found that the accuracy is also very high even on the hard set. The improvement on mAP is not so obvious now. The mAP is almost saturated similar to FDDB.

We must note that the current benchmarks, regardless of FDDB, WIDER or others, only evaluate the accuracy of detection and do not evaluate efficiency. If two detectors achieve similar mAP, but the computational cost of one is just half of another, surely we will think the detector with half computational cost is better than another. Since the accuracy metric is almost saturated, it is time to include efficiency in the evaluation.

VI. Evaluation of Computational Cost

Deep learning techniques have brought momentous improvement to face detection, and can detect faces more robustly in unconstrained environments. Most of the recent works train and test their models on WIDER Face [3]. As shown in Fig. 2, we can find a large AP leap from 2016 to 2017. However, the line has been flat since 2017. If we look deep into the official releasing code of recent works, it can be easily found that newer models tend to use larger scales and a wider range of scales as shown in Table V. These test scales are usually not mentioned in the papers, but can lead to a non-negligibly great increase in computational cost just for slightly boosting the AP. We may even question: Is the AP improved by a better algorithm or the usage of a wider range of test scales?

TABLE IV
EQUATIONS OF FLOPS CALCULATION OF DIFFERENT LAYERS

NN layers	FLOPs	Explanation
Conv	$C_{out}H_{out}W_{out}(2C_{in}K^2-1)$	For each element in the output tensor, there are C_inK^2 multiplications between the kernels and sliding windows, and C_inK^2-1 additions to sum up. If bias is used, 1 FLOPs should be added to the FLOPs calculation of each element. [84]
Max Pool	$K^2C_{out}H_{out}W_{out}$	For each element in the output tensor, we consider the worst situation where every element in the kernel requires a comparison with each other.
ReLU	$2C_{out}H_{out}W_{out}$	ReLU is usually implemented as $x*(x>0)$, which is much faster than directly comparing x with 0. We consider a comparison as 1 FLOPs for simplicity.
Batch Norm	$6C_{out}H_{out}W_{out}$	As [85] stated, the variances and means are fixed during inference. Therefore, 6 FLOPs is accounted for applying the linear transform to each element.
L2-Norm	$3C_{out}H_{out}W_{out}$	The L2-norm layer was proposed by [86] to help features of late fusion work well, which is defined as $L_2-norm(x)=\frac{x}{\|\|x\|\|_2}=\frac{x}{\sqrt{\sum_{i=1}^{d}\|x_i\|^2}}$, where d usually stands for channels. It takes approximately $2CHW$ FLOPs to calculate the L_2 norm channel-wisely and CHW FLOPs to perform L_2 norm element-wisely.
Bilinear Upsample	$19C_{out}H_{out}W_{out}$	The definition of bilinear upsampling [15] contains 9 non-duplicate additions and subtractions and 10 multiplications/divisions for calculating one element in the output.
Sigmoid	$3C_{out}H_{out}W_{out}$	The definition of sigmoid [16] contains 1 exponentiation, 1 addition and 1 division to calculate one element in the output.
Softmax	$3E$	E denotes the total number of elements in the output tensor. It takes approximately $2E$ FLOPs to calculate the sum of the exponentiation of each element in different channels, and E FLOPs to calculate the final result.

TABLE V
TEST SCALES USED BY OPEN-SOURCE ONE-STAGE FACE DETECTORS [13], [14], [17], [18], [19], [20], [25]. NOTE THAT THE DOUBLE CHECK MARKS DENOTE IMAGE FLIPPING VERTICALLY IN ADDITION TO THE IMAGE AT THE CURRENT SCALE. SSH SHRINKS AND ENLARGES IMAGES TO SEVERAL PRESET FIXED SIZES. SINCE S^{3}FD, TWO ADAPTIVE TEST SCALES ARE USED TO SAVE GPU MEMORY, ONE OF WHICH IS "S" FOR ADAPTIVE SHRINKING, THE OTHER OF WHICH IS "E" FOR RECURSIVELY ADAPTIVE ENLARGING. SCALE "F" DENOTES ENLARGING THE IMAGE TO THE PRESET LARGEST SIZE

Model	Publication	test Scales (ratio)											
		0.25	0.5	0.75	1	1.25	1.5	1.75	2.0	2.25	S	E	F
HR	CVPR'17	✓	✓		✓								
S^{3}FD	ICCV'17		✓		✓						✓	✓	
PyramidBox	ECCV'18	✓		✓	✓✓	✓	✓	✓			✓	✓	
SRN	AAAI'19		✓		✓✓		✓			✓			✓
DSFD	CVPR'19		✓		✓✓	✓		✓		✓	✓	✓	
CSP	CVPR'19	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓			
		test Scales (resize longer side)											
		100	300	500	600	700	800	900	1000	1100	1200	1400	1600
SSH	ICCV'17			✓			✓				✓		✓
SHF	WACV'20	✓	✓		✓				✓			✓	
RetinaFace	CVPR'20			✓			✓			✓		✓	✓

A. Rules of FLOPs Calculation

What kind of models are we going to re-evaluate? First, the models must be open-source at least with the release of its test code and a trained model. We do not re-implement the methods since we want to ensure that the accuracy should be 100% the same as the original authors claimed. Additionally, it is essential for us to choose one-stage models, as their FLOPs are independent of the number of faces in the images, and they have been the most studied frameworks in recent years. Third, we mainly choose the models from the WIDER Face result page for fair comparisons.

How do we calculate the FLOPs of different models? We first validate whether the officially released trained models can perform as well as the authors state in their papers. It should be noted that we do not calculate the pre-processing and post-processing stages from a model's pipeline. In other words, only FLOPs of neural network layers such as convolution, activation, normalization, pooling and other layers are calculated.

Given a 4D input tensor of size $N \times C_{in} \times H_{in} \times W_{in}$ as input, a neural network layer produces a 4D output tensor of size $N \times C_{out} \times H_{out} \times W_{out}$, where N is the batch size which is dismissed for simplicity in the following since it is usually set to 1 during test, C, H and W are the channels, height and width of the tensor respectively. Additionally, K is introduced to represent the kernel size for layers utilizing kernels such as convolution and pooling layers. Specifically, we treat floating point operations, such as addition, subtraction, multiplication, division and exponentiation the same, which should be 1 FLOPs for simplicity. With these assumptions, we are able to derive the equations for calculating FLOPs for different layers as listed in Table IV.

We implement our FLOPs calculator based on PyTorch regarding all the rules and equations we discussed above, which accelerates the calculation of FLOPs by dismissing any calculation related to the value of tensors, while only computing the sizes of tensors and FLOPs. This calculator can also allow us to use the code of defining models from authors with minor changes, which reduces

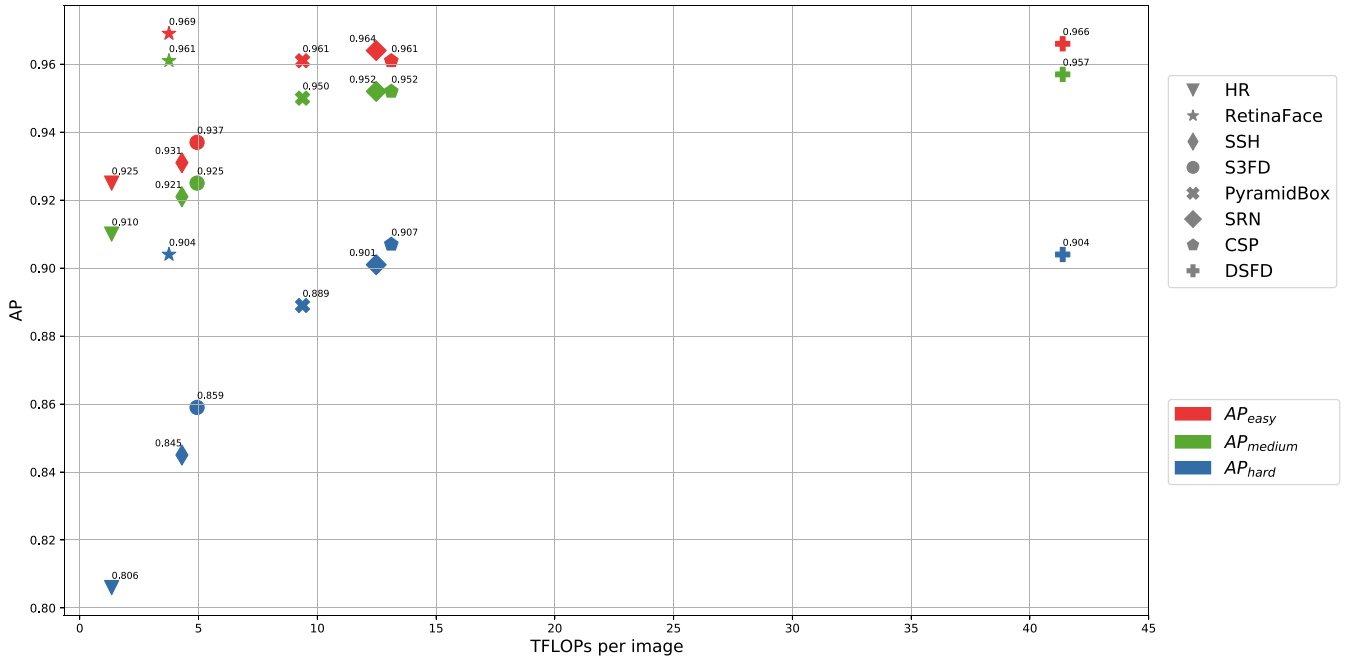


Fig. 10. The FLOPs vs. multi-scale AP of WIDER Face validation set. 7 models from the WIDER Face result page are listed, which are HR [13], SSH [14], S^{3}FD [17], PyramidBox [18], SRN [19], DSFD [20], CSP [25]. (*The TFLOPs for some speed-focusing face detectors are listed in Table X because the TFLOPs are in a much smaller scale and cannot fit in this figure.*).

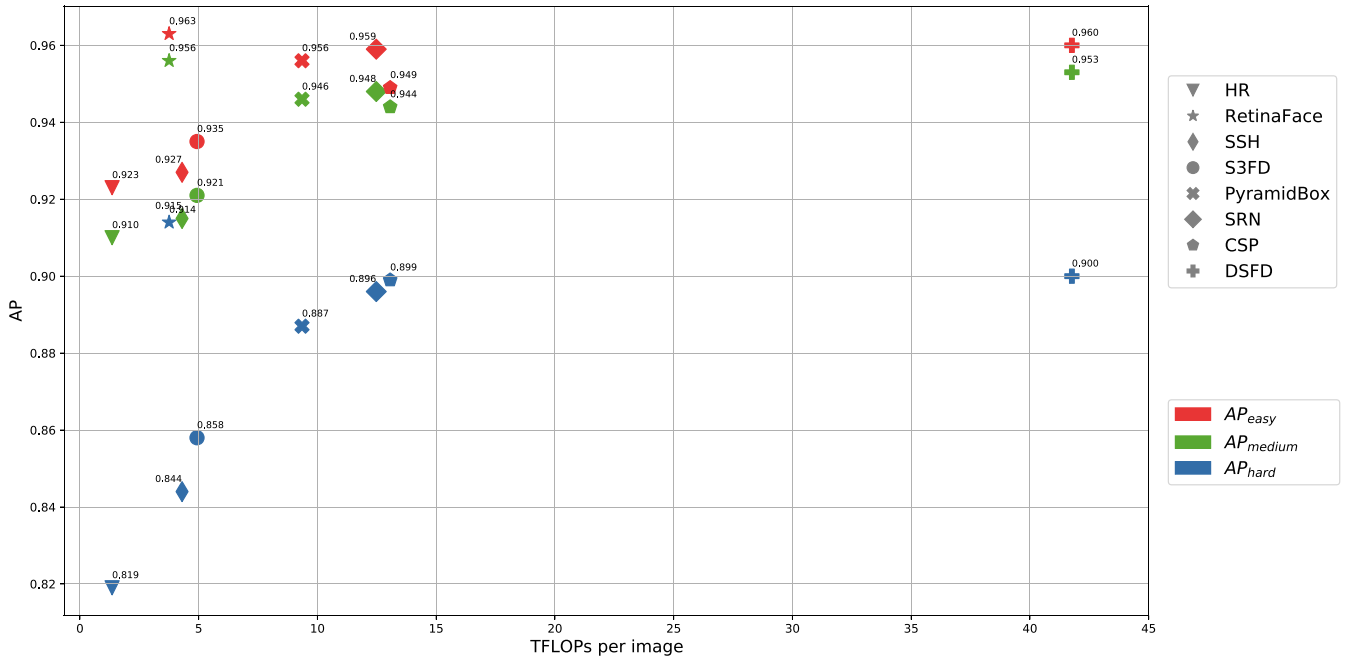


Fig. 11. The FLOPs vs. multi-scale test AP of WIDER Face test set. 7 models from the WIDER Face result page are listed, which are HR [13], SSH [14], S^{3}FD [17], PyramidBox [18], SRN [19], DSFD [20], CSP [25].

the statistics workload. We released our source code at https://github.com/fengyuentau/PyTorch-FLOPs.

B. FLOPs vs. AP in Multi-Scale Test

The multi-scale test metric is to test a model with a set derived from an image at original and different scales (with aspect ratio fixed). The detection results of different scales are then merged and applied with the non-maximum suppression (NMS), so as to suppress the overlapped bounding boxes and reduce false positives. Based on the training data and scheme, a *comfort zone* of a model is determined, which is a range of scales of faces that can be detected. The multi-scale test metric can improve a model's AP by re-scaling out-of-zone faces back in the comfort zone. However, since we cannot determine which of the faces in the test set are out-of-zone, we have to apply re-scaling to every image in the set. It leads to the multiplied increase in FLOPs per image.

Fig. 10 and Fig. 11 show the multi-scale test AP and FLOPs of different models on the validation and test sets of the

TABLE VI
HOW DIFFERENT SCALES IMPACT THE AP OF PYRAMIDBOX [18]. WE USE SCALE = 1 AS THE BASELINE, AND THEN TRY ADDING DIFFERENT SCALES ONE BY ONE TO TEST HOW AP IS IMPACTED BY DIFFERENT SCALES

Test Scales						AP_{easy}	AP_{medium}	AP_{hard}	TFLOPs
0.25	0.75	1	1.25	1.5	1.75				
		✓				0.947	0.936	0.875	1.37
✓		✓				0.954(+0.007)	0.939(+0.003)	0.872(-0.003)	1.45(+0.008)
	✓	✓				0.952(+0.005)	0.940(+0.004)	0.874(-0.001)	2.14(+0.77)
		✓	✓			0.948(+0.001)	0.938(+0.002)	0.884(+0.009)	2.72(+1.35)
		✓		✓		0.947(+0.000)	0.937(+0.001)	0.881(+0.006)	2.46(+1.09)
		✓			✓	0.946(-0.001)	0.936(+0.000)	0.874(-0.001)	1.63(+0.26)

TABLE VII
HOW MUCH WILL AP AND FLOPS DECREASE IF A SCALE IS REMOVED? THE DETECTOR PYRAMIDBOX IS EMPLOYED

Test Scales						AP_{easy}	AP_{medium}	AP_{hard}	TFLOPs
0.25	0.75	1	1.25	1.5	1.75				
✓	✓	✓	✓	✓	✓	0.957	0.945	0.886	4.94
	✓	✓	✓	✓	✓	0.949(-0.008)	0.940(-0.005)	0.884(-0.002)	4.85(-0.009)
✓		✓	✓	✓	✓	0.954(-0.003)	0.942(-0.003)	0.885(-0.001)	4.16(-0.780)
✓	✓		✓	✓	✓	0.955(-0.002)	0.940(-0.005)	0.850(-0.013)	3.58(-1.360)
✓	✓	✓		✓	✓	0.957(+0.000)	0.944(-0.001)	0.880(-0.006)	3.58(-1.360)
✓	✓	✓	✓		✓	0.958(+0.001)	0.945(+0.000)	0.884(-0.002)	3.84(-1.100)
✓	✓	✓	✓	✓		0.957(+0.000)	0.945(+0.000)	0.886(+0.000)	4.67(-0.270)

WIDER Face dataset, respectively. We can find a clear trend in the two figures. The FLOPs are increasing and the AP is improving in the sequence of methods HR [13], SSH [14], S^3FD [17], PyramidBox [18], SRN [19] and CSP [25]. There are two methods do not follow the trend. The first one is DSFD [20] which has more than 3 times of FLOPs than SRN and CSP, but the AP is similar with those of SRN and CSP. It means DSFD has unreasonable high computational cost. Then second detector is RetinaFace [21] which gained the best AP but the computational cost is much lower than most other methods.

The two figures (Fig. 10 and Fig. 11) give us a clear view of different face detection models and can guide us understand different models deeper.

C. FLOPs vs. AP in Single-Scale Test

FLOPs can sharply increase in two ways: fundamentally increasing through introducing more complex modules to the network, and through multi-scale testing. As Table V shows, these models are all tested on various scales. However, why models are tested on these various scales is seldom discussed. How much contribution on AP can one scale bring? Are any scales not necessary?

Single-scale test on a single model: Table VI shows the AP contribution of different scales. The easy subset in WIDER Face [3] contains a large margin of faces of regular size and some large faces, as a result of which shrinking images can help improve the AP. We can observe that AP_{hard} gains the most from scales 1, 1.25 and 1, 1.5, but not for scale 1, 1.75. Together with FLOPs, we can also observe an increase to the peak at scale 1, 1.25 and then a sharp drop for larger scales. The reason is that a threshold for the largest size of images is set to avoid exceeding the GPU memory. This means that not all 1.75x resized images were sent to a detector in the experiments.

TABLE VIII
AP AND FLOPS OF DIFFERENT MODELS ON SCALE 1

Model	AP_{easy}	AP_{medium}	AP_{hard}	TFLOPs
RetinaFace	0.952	0.942	0.776	0.198
S3FD	0.924	0.906	0.816	0.571
CSP	0.948	0.942	0.774	0.571
SSH	0.925	0.909	0.731	0.587
PyramidBox	0.947	0.936	0.875	1.387
DSFD	0.949	0.936	0.845	1.532

Table VII shows how much the AP and FLOPs will decrease if a model tested without a scale. As the missing scale becomes larger, the decrease of AP_{easy} decreases. However, this pattern does not apply to AP_{medium} and AP_{hard}. The reason is that the enlarged images will be skipped if their size goes beyond the preset limit, so as to avoid exceeding GPU memory. The larger the scale is, the fewer images will be re-scaled and tested. The drop of FLOPs greatly decreases on scale 1.75. This is because the PyramidBox pretrained model is mainly trained on scale 1.

The two Tables VI and VII imply that AP_{easy} is the most sensitive to scales 0.25, AP_{medium} is the most sensitive to scale 0.25 and 1, and AP_{hard} is the most sensitive to scale 1. Note that this is highly related to the training scale. If the model is trained differently, the conclusion may change accordingly.

Single-scale test on multiple models: Table VIII shows the AP and FLOPs of different models on scale 1. The large overall leap is brought by PyramidBox [18], which mainly introduces the FPN [15] module to fuse features from two adjacent scales and the context enhancing module from SSH [14]. The computational cost of PyramidBox is 2X compared with SSH but less than 1/2 of DSFD. However, the AP achieved by PyramidBox and DSFD are comparable.

If some benchmarks can evaluate FLOPs or some other similar efficiency measurements, different face detectors can compare more fairly. It will also promote face detection research to a better stage.

TABLE IX
STATE-OF-THE-ART OPEN-SOURCE MODELS TESTED WITH A 720P IMAGE CONTAINING SEVERAL FACES AT SCALE = 1.0 ONLY. WE AVERAGE THE FLOPS (AVG TFLOPS) AND LATENCY (AVG LATENCY) BY RUNNING THE TEST FOR EACH MODEL 100 TIMES. NOTE THAT 'POST-PROC' DENOTES POST-PROCESSING STAGES, SUCH AS DECODING FROM ANCHORS, NMS AND SO ON. FOR THIS STAGE, WE ADOPT THE ORIGINAL PROCESSING CODE OF EACH MODEL

Model	AVG TFLOPs	AVG Latency (ms)		
		Forward (GPU)	Forward (CPU)	Post-Proc
RetinaFace	0.201	131.60	809.24	8.74 (GPU)
CSP	0.579	154.55	1955.20	27.74 (CPU)
SRN	1.138	204.77	2933.16	8.71 (GPU)
DSFD	1.559	219.63	3671.46	76.32 (CPU)

D. FLOPs vs Latency

To compare the two measurements, we convert existing models to the Open Neural Network Exchange (ONNX) format and run them using the ONNXRUNTIME[15] in this comparison for fair comparison. Note that due to the different supports to ONNX converting of different DL frameworks, we managed to convert RetinaFace [21], SRN [19], DSFD [20] and CSP [25] to ONNX format. The results are in Table IX. These models are evaluated using an NVIDIA QUADRO RTX 6000 with CUDA 10.2, and an INTEL Xeon Gold 6132 CPU @ 2.60 GHz. The powerful GPU contains 4,609 CUDA parallel-processing cores and 24GB memory.

We can observe that both FLOPs and forward latency increase from RetinaFace [21] to DSFD [20]. Note that although the average FLOPs of RetinaFace are just one-fifth of SRN's, the forward latency of RetinaFace is almost near half of SRN's, implying that FLOPs are not linearly correlated to latency due to the differences in implementation, hardware settings, memory efficiency and so on. The reason why the post-processing latency of DSFD and CSP sharply increase is that they do not use GPU-accelerated NMS as others do.

VII. SPEED-FOCUSING FACE DETECTORS

For the face detectors introduced in the previous sections, the main target is to reach a better AP. Their computational costs are heavy and normally in magnitude of TFLOPs. It is unrealistic to deploy those heavy models to a face-related system. There are some other open source face detectors whose target is to make face detection run in real time for practical applications. Their computational costs are in the magnitude of those of GFLOPs or 10 GFLOPs and are much less than the previous costs. Here we group them as speed-focusing face detectors. We collect the most-popular ones from github.com, and review them in terms of network architectures, AP, FLOPs and efficiency.

FaceBoxes [87] is one of the first one-stage deep learning-based models to achieve real-time face detection. FaceBoxes rapidly downsamples feature maps to a stride 32 with two convolution layers with large kernels. Inception blocks [63] are introduced to enhanced feature maps at stride of 32. Following the multi-scale mechanism from SSD [8], FaceBoes detects on layers `inception3`, `conv3_2` and `conv4_2` for faces at different scales, resulting in an AP of 0.960 on FDDB [24] and 20 FPS on an INTEL E5-2660v3 CPU at 2.60 GHz.

YuFaceDetectNet [89] adopts a light MobileNet [60] as the backbone. Compared to FaceBoxes, YuFaceDetectNet has more convolution layers on each stride to have fine-grained features, and detects on the extra layer of stride 16, which improves the recall of small faces. The evaluation results of the model on the WIDER Face [3] validation set are 0.856 (Easy), 0.842 (Medium) and 0.727 (Hard). The main and well-known repository, libfacedetection [91], takes YuFaceDetectNet as the detection model and offers pure C++ implementation without dependence on DL frameworks, resulting from 77.34 FPS for 640 $\times$ 480 images to 2,027.74 FPS for 128 $\times$ 96 images on an INTEL i7-1065G7 CPU at 1.3 GHz.

LFFD [90] introduces residual blocks for feature extraction, and proposes receptive fields as the natural anchors. Its faster version LFFD-v2 managed to achieve 0.875 (Easy), 0.863 (Medium) and 0.754 (Hard) on the WIDER Face validation set, while running at 472 FPS using CUDA 10.0 and an NVIDIA RTX 2080Ti GPU. **ULFG** [88] adds even more convolution layers on each stride, taking the advantage of depth-wise convolution, which is friendly to edge devices in terms of FLOPs and forward latency. As reported, the slim version of ULFG has an AP of 0.770 (Easy), 0.671 (Medium) and 0.395 (Hard) on the WIDER Face validation set, and can run at 105 FPS with an input resolution of 320 $\times$ 240 on an ARM A72 at 1.5 GHz.

These light-weight models are developed using various frameworks and tested on different hardware. For fair comparison, we export these models from their original frameworks to ONNX and test using ONNXRUNTIME on a INTEL i7-5930K CPU at 3.50GHz. Results are shown in Table X. We can observe that more `CONV` layers do not lead to more parameters (FacesBoxes and ULFG series) and more FLOPs (YuFaceDetectNet and ULFG series). This is mainly because of the extensive usage of depth-wise convolution in ULFG. Additionally, note that more FLOPs do not lead to more forward latency due to depth-wise convolution. The post-processing latency across different face detectors seems inconsistent with the forward latency, and we verified that this is caused by different numbers of bounding boxes sent to NMS and the different implementations of NMS (Python-based or Cython-based).

VIII. CONCLUSION AND DISCUSSIONS

Face detection is one of the most important and popular topics yet still challenging in computer vision. Deep learning has brought remarkable breakthroughs for face detectors. Face detection is more robust and accurate even in unconstrained real-world environments. In this paper, recent deep learning-based face detectors and benchmarks are introduced. From the evaluations of accuracy and efficiency on different deep face detectors, we can find that we can reach a very high accuracy if we do not consider the computational cost. However, there should be a simple and beautiful solution for face detection since it is simpler than generic object detection. The research

[15]https://github.com/microsoft/onnxruntime

TABLE X
POPULAR AND ACTIVE OPEN-SOURCE FACE DETECTORS AT GITHUB. NOTE THAT 'AVG GFLOPS' ARE COMPUTED ON WIDER FACE VALIDATION SET IN SINGLE-SCALE TEST WHERE ONLY SCALE=1.0. ALSO NOTE THAT LATENCY IS MEASURED ON CPU

Model	#CONV Layers	#Params ($\times 10^6$)	AVG GFLOPs	WIDER Face Val Set			Latency (ms)	
				AP_{easy}	AP_{medium}	AP_{hard}	Forward	Post-Proc
FaceBoxes [87]	33	1.013	1.541	0.845	0.777	0.404	16.52	7.16
ULFG-slim-320 [88]	42	0.390	2.000	0.652	0.646	0.520	19.03	2.37
ULFG-slim-640 [88]				0.810	0.794	0.630		
ULFG-RFB-320 [88]	52	0.401	2.426	0.683	0.678	0.571	21.27	1.90
ULFG-RFB-640 [88]				0.816	0.802	0.663		
YuFaceDetectNet [89]	43	0.085	2.549	0.856	0.842	0.727	23.47	32.81
LFFD-v2 [90]	45	1.520	37.805	0.875	0.863	0.752	178.47	6.70
LFFD-v1 [90]	65	2.282	55.555	0.910	0.880	0.778	229.35	10.08

on face detection can focus on the topics introduced in the following topics in the future.

Superfast Face Detection. There is no definition for superfast face detection. Ideally, superfast face detector should be able to run in real time on low-cost edge devices even when the input image is 1080P. Empirically speaking, we would like to expect it to be less than 100M FLOPs with a 1080P image as input. For real-world applications, efficiency is one of the key issues. Efficient face detectors can help to save both energy, the cost of hardware and improve the responsiveness for edge devices, such as CCTV cameras and mobile phones.

Detecting Faces in the Long-tailed Distribution. Face samples can be regarded as a long-tailed distribution. Most face detectors are trained for the dominant part of the distribution. We have already had enough samples for faces with variances in illumination, pose, scale, occlusion, blur, distortion in the WIDER Face dataset. But what about other faces like the old and damaged ones? As people getting old, there are many wrinkles on their faces; and people who suffer from illnesses or accidents may have damaged faces, such as burn scars on the faces. Face detection is not only a technical problem but also a humanitarian problem, meaning that this technology should serve all the people, not only the dominant part of the population. Ideally, face detectors should be able to detect all kinds of faces. However, in most face datasets and benchmarks, most faces are from young people.

The final goal of face detection is to detect faces with very high accuracy and high efficiency. Therefore, the algorithms can be deployed to many kinds of edge devices and centralized servers to improve the perception capability of computers; currently, there still is a considerable gap. Face detectors can achieve good accuracy but still require considerable computations. Improving the efficiency should be the next step.

REFERENCES

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2001, pp. 511–518.

[3] S. Yang, P. Luo, C.-C. Loy, and X. Tang, "WIDER FACE: A face detection benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 5525–5533.

[4] J. Li and Y. Zhang, "Learning SURF cascade for fast and accurate object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 3468–3475.

[5] L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Z. Li, "Face detection based on multi-block LBP representation," in *Proc. Int. Conf. Biometrics*, 2007, pp. 11–18.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*. Montreal, QC, Canada: Curran Ass., Inc., 2015.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.

[8] W. Liu *et al.*, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 21–37.

[9] H. Wang, Z. Li, X. Ji, and Y. Wang, "Face R-CNN," 2017. [Online]. Available: arXiv:1706.01061.

[10] Y. Wang, X. Ji, Z. Zhou, H. Wang, and Z. Li, "Detecting faces using region-based fully convolutional networks," 2017. [Online]. Available: arXiv:1709.05256.

[11] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems (NIPS)*. Red Hook, NY, USA: Curran, 2016.

[12] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[13] P. Hu and D. Ramanan, "Finding tiny faces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1522–1530.

[14] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis, "SSH: Single stage headless face detector," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 4885–4894.

[15] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 936–944.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2999–3007.

[17] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S^3FD: Single shot scale-invariant face detector," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 192–201.

[18] X. Tang, D. K. Du, Z. He, and J. Liu, "Pyramidbox: A context-assisted single shot face detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 812–828.

[19] C. Chi, S. Zhang, J. Xing, Z. Lei, S. Z. Li, and X. Zou, "Selective refinement network for high performance face detection," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2019, pp. 8231–8238.

[20] J. Li *et al.*, "DSFD: Dual shot face detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 5060–5069.

[21] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "RetinaFace: Single-shot multi-level face localisation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 5202–5211.

[22] J. Yan, X. Zhang, Z. Lei, and S. Z. Li, "Face detection by structural models," *Image Vis. Comput.*, vol. 32, no. 10, pp. 790–799, 2014.

[23] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2012, pp. 2879–2886.

[24] V. Jain and E. Learned-Miller, "FDDB: A benchmark for face detection in unconstrained settings," Dept. Comput. Sci., Univ. Massachusetts, Amherst, MA, USA, Rep. UM-CS-2010-009, 2010.

[25] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, "High-level semantic feature detection: A new perspective for pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 5187–5196.

[26] S. Luo, X. Li, R. Zhu, and X. Zhang, "SFA: Small faces attention face detector," *IEEE Access*, vol. 7, pp. 171609–171620, 2019.

[27] Z. Zhang, W. Shen, S. Qiao, Y. Wang, B. Wang, and A. Yuille, "Robust face detection via learning small faces on hard images," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2020, pp. 1350–1359.

[28] A. Kumar, A. Kaur, and M. Kumar, "Face detection techniques: A review," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 927–948, 2019.

[29] S. Zafeiriou, C. Zhang, and Z. Zhang, "A survey on face detection in the wild: Past, present and future," *Comput. Vis. Image Understand.*, vol. 138, pp. 1–24, Sep. 2015.

[30] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," Microsoft Res., Redmond, WA, USA, Rep. MSR-TR-2010-66, 2010.

[31] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, Jan. 2002.

[32] E. Hjelmås and B. K. Low, "Face detection: A survey," *Comput. Vis. Image Understand.*, vol. 83, no. 3, pp. 236–274, 2001.

[33] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[34] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," 2019. [Online]. Available: arXiv:1905.05055.

[35] X. Wu, D. Sahoo, and S. C. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, Jul. 2020.

[36] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 5325–5334.

[37] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Aggregate channel features for multi-view face detection," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, 2014, pp. 1–8.

[38] C. Zhu, Y. Zheng, K. Luu, and M. Savvides, "CMS-RCNN: Contextual multi-scale region-based CNN for unconstrained face detection," in *Deep Learning for Biometrics*. Cham, Switzerland: Springer, 2017, pp. 57–79.

[39] M. Najibi, B. Singh, and L. S. Davis, "FA-RPN: Floating region proposals for face detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 7723–7732.

[40] Y. Liu, X. Tang, J. Han, J. Liu, D. Rui, and X. Wu, "HAMBox: Delving into mining high-quality anchors on face detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13043–13051.

[41] Y. Liu and X. Tang, "BFBox: Searching face-appropriate backbone and feature pyramid network for face detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 13565–13574.

[42] J. Zhu, D. Li, T. Han, L. Tian, and Y. Shan, "ProgressFace: Scale-aware progressive learning for face detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 344–360.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[45] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 765–781.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*. Red Hook, NY, USA: Curran, 2012.

[47] S. Wan, Z. Chen, T. Zhang, B. Zhang, and K.-K. Wong, "Bootstrapping face detection with hard negative examples," 2016. [Online]. Available: arXiv:1608.02236.

[48] C. Zhang, X. Xu, and D. Tu, "Face detection using improved faster RCNN," 2018. [Online]. Available: arXiv:1802.02142.

[49] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2017, pp. 4278–4284.

[50] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," 2017. [Online]. Available: arXiv:1701.06659.

[51] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 6154–6162.

[52] J. Dai *et al.*, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 764–773.

[53] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets V2: More deformable, better results," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 9308–9316.

[54] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (CVPR)*, 2019, pp. 9626–9635.

[55] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019. [Online]. Available: arXiv:1904.07850.

[56] X. Zhou, J. Zhuo, and P. Krähenbühl, "Bottom-up object detection by grouping extreme and center points," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 850–859.

[57] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "RepPoints: Point set representation for object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (CVPR)*, 2019, pp. 9656–9665.

[58] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 1999, pp. 1150–1157.

[59] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2005, pp. 886–893.

[60] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: arXiv:1704.04861.

[61] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2009, pp. 248–255.

[62] Y. Zhu, H. Cai, S. Zhang, C. Wang, and Y. Xiong, "TinaFace: Strong but simple baseline for face detection," 2020. [Online]. Available: arXiv:2011.13183.

[63] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–9.

[64] Z. Hao, Y. Liu, H. Qin, J. Yan, X. Li, and X. Hu, "Scale-aware face detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1913–1922.

[65] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, and X. Tang, "Recurrent scale approximation for object detection in CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 571–579.

[66] G. Song, Y. Liu, M. Jiang, Y. Wang, J. Yan, and B. Leng, "Beyond trade-off: Accelerate FCN-based face detector with higher accuracy," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7756–7764.

[67] J. Li and Y. Zhang, "Learning SURF cascade for fast and accurate object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 3468–3475.

[68] H. Jin, Q. Liu, H. Lu, and X. Tong, "Face detection using improved LBP under Bayesian framework," in *Proc. Int. Conf. Image Graph. (ICIG)*, 2004, pp. 306–309.

[69] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*. Red Hook, NY, USA: Curran, 2016.

[70] C. Zhu, R. Tao, K. Luu, and M. Savvides, "Seeing small faces from robust anchor's perspective," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 5127–5136.

[71] X. Ming, F. Wei, T. Zhang, D. Chen, and F. Wen, "Group sampling for scale invariant face detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 3446–3456.

[72] E. T. Boult, M. Gunther, and R. A. Dhamija. "2nd unconstrained face detection and open set recognition challenge." [Online]. Available: https://vast.uccs.edu/Opensetface/ (accessed Oct. 31, 2019).

[73] M. K. Yucel, Y. C. Bilge, O. Oguz, N. Ikizler-Cinbis, P. Duygulu, and R. G. Cinbis, "Wildest faces: Face detection and recognition in violent settings," 2018. [Online]. Available: arXiv:1805.07566.

[74] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Fine-grained evaluation on face detection in the wild," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, 2015, pp. 1–7.

[75] S. Ge, J. Li, Q. Ye, and Z. Luo, "Detecting masked faces in the wild with LLE-CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 426–434.

[76] B. F. Klare *et al.*, "Pushing the frontiers of unconstrained face detection and recognition: IARPA janus benchmark A," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1931–1939.

[77] C. Whitelam *et al.*, "IARPA janus benchmark-B face dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2017, pp. 592–600.

[78] B. Maze *et al.*, "IARPA janus benchmark—C: Face dataset and protocol," in *Proc. Int. Conf. Biometrics (ICB)*, 2018, pp. 158–165.

[79] J. Wang, Y. Yuan, B. Li, G. Yu, and S. Jian, "SFace: An efficient network for face detection in large scale variations," 2018. [Online]. Available: arXiv:1804.06559.

[80] H. Nada, V. A. Sindagi, H. Zhang, and V. M. Patel, "Pushing the limits of unconstrained face detection: A challenge dataset and baseline results," in *Proc. IEEE Int. Conf. Biometrics Theory Appl. Syst. (BTAS)*, 2018, pp. 1–10.
[81] C. Wei, W. Wang, W. Yang, and J. Liu, "Deep retinex decomposition for low-light enhancement," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2018, p. 155.
[82] M. Naphade *et al.*, "Large-scale concept ontology for multimedia," *IEEE Multimedia*, vol. 13, no. 3, pp. 86–91, Jul.–Sep. 2006.
[83] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 391–405.
[84] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–17.
[85] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 448–456.
[86] W. Liu, A. Rabinovich, and A. C. Berg, "ParseNet: Looking wider to see better," 2015. [Online]. Available: arXiv:1506.04579.
[87] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "FaceBoxes: A cpu real-time face detector with high accuracy," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, 2017, pp. 1–9.
[88] Linzaer. "Ultra-light-fast-generic-face-detector-1MB." 2020. [Online]. Available: https://github.com/Linzaer/Ultra-Light-Fast-Generic-Face-Detector-1MB
[89] S. Yu. "Libfacedetection.train." 2021. [Online]. Available: https://github.com/ShiqiYu/libfacedetection.train
[90] Y. He, D. Xu, L. Wu, M. Jian, S. Xiang, and C. Pan, "LFFD: A light and fast face detector for edge devices," 2019. [Online]. Available: arXiv:1904.10633.
[91] S. Yu. "Libfacedetection." 2021. [Online]. Available: https://github.com/ShiqiYu/libfacedetection

Yuantao Feng received the B.E. and M.E. degrees in computer science and technology from the College of Computer and Software Engineering, Shenzhen University in 2018 and 2021 respectively. He is currently a Research Assistant with the Department of Computer Science and Engineering, Southern University of Science and Technology, China. His research interests include object detection and computer vision.

Shiqi Yu (Member, IEEE) received the B.E. degree in computer science and engineering from Chu Kochen Honors College, Zhejiang University in 2002, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences in 2007. He is currently an Associate Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, China. He worked as an Assistant Professor and an Associate Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences from 2007 to 2010, and as an Associate Professor with Shenzhen University from 2010 to 2019. His research interests include gait recognition, face detection, and computer vision.

Hanyang Peng received the B.S. degree in measurement and control technology from the Northeast University of China, Shenyang, China, in 2008, the M.E. degree in detection technology and automatic equipment from the Tianjin University of China, Tianjin, China, in 2010, and the Ph.D. degree in pattern recognition and intelligence systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2017. He is currently with the Southern University of Science and Technology, Shenzhen, China. His current research interests include computer vision, machine learning, deep learning, and optimization.

Yan-Ran Li received the Ph.D. degree in communications and information systems from the Electronic Department, School of Information Science and Technology, Sun Yat-sen (Zhongshan) University, China, in 2009. He is currently an Associate Professor with the College of Computer Science and Software Engineering, Shenzhen University, China. He was a Visiting Scholar with the National University of Singapore, Singapore, in 2008, and the Chinese University of Hong Kong, Hong Kong, in 2011 and 2012, and Syracuse University, USA, in 2014. His current research interests include image processing and medical data processing.

Jianguo Zhang (Senior Member, IEEE) received the Ph.D. degree from the National Lab of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, 2002. He is currently a Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology. Previously, he was a Reader of Computing with the School of Science and Engineering, University of Dundee, U.K. His research interests include object recognition, medical image analysis, machine learning and computer vision. He serves as an Associate Editor of IEEE Transactions on Multimedia.