Fast and Accurate Face Detection using Feature Pyramid with Grid Anchors

1st Liguo Zhou
Institut für Informatik VI
Technical University of Munich
Garching, Germany
liguo.zhou@tum.de

2nd Guang Chen
School of Automotive Studies
Tongji University
Shanghai, China
guangchen@tongji.edu.cn

3rd Alois Knoll
Institut für Informatik VI
Technical University of Munich
Garching, Germany
knoll@in.tum.de

***Abstract*—CNN-based face detection methods have achieved significant progress in recent years. However, making a good balance between time cost and detection accuracy is still a challenging problem. Those methods which can reach a very high detection accuracy always have complicated networks and rely on expensive GPUs for inference, while those methods which have shallow networks and can run on common devices always lose detection accuracy to a large extent. In this paper, we propose an effective anchor generation and bounding-box regression method which can improve the detection accuracy by modifying the detection head of the popular detection networks. With this effectiveness, we can reduce the trainable weights of the network to speed up the inference while maintaining high accuracy. As a result, our method can get a better speed-accuracy balance. In our method, we divide the input image into grids according to the sizes of the pyramid-like feature maps produced by CNN. In training, those grids close to the center of the ground-truth bounding-boxes are selected as anchors. After training, the regression mapping from the anchors to the ground-truth bounding-boxes can be acquired by the exponential transformation we designed. Our method explicitly and strictly use feature maps in different levels to detect faces of different sizes. The higher-level feature maps and larger grid anchors are responsible for detecting larger faces, while the lower-level feature maps and smaller grid anchors are dedicated to detecting smaller faces. Therefore, our method is effective for detecting multi-scale faces. The experiments on both GPU and CPU demonstrate that our method is effective. Our source code is publicly available on https://github.com/zhouliguo/GAFace.**

***Index Terms*—CNN, Face Detection, Bounding-box Regression, Image Processing**

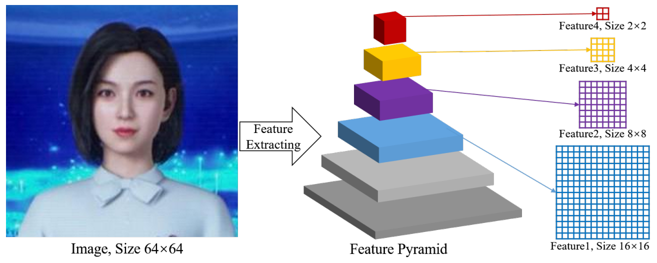
I. Introduction

Face detection is an important task in computer vision and has been widely studied in the past decades. Nowadays, many emerging applications, such as identity authentication and security surveillance, hinge on face detection. Since AlexNet [1] was proposed, Convolutional Neural Networks (CNN) have achieved significant progress in face detection. However, for high-performance face detection, there are still a series of challenging problems. The balance between detection speed and accuracy is an essential problem because it determines whether a face detection method can be applied in practical applications. To improve the accuracy of face detection, more and more complicated structures of CNN [2], [3] are proposed. These methods take a long time for inference, even on expensive GPUs. To make face detection run in real-time on common CPUs or mobile devices, some face detectors [4], [5] utilize light networks with fewer weights. However, the accuracy of these methods drops a lot. Instead of modifying the structure of CNN, in this paper, we propose an effective anchor generation and bounding-box regression method to achieve a good speed-accuracy balance.

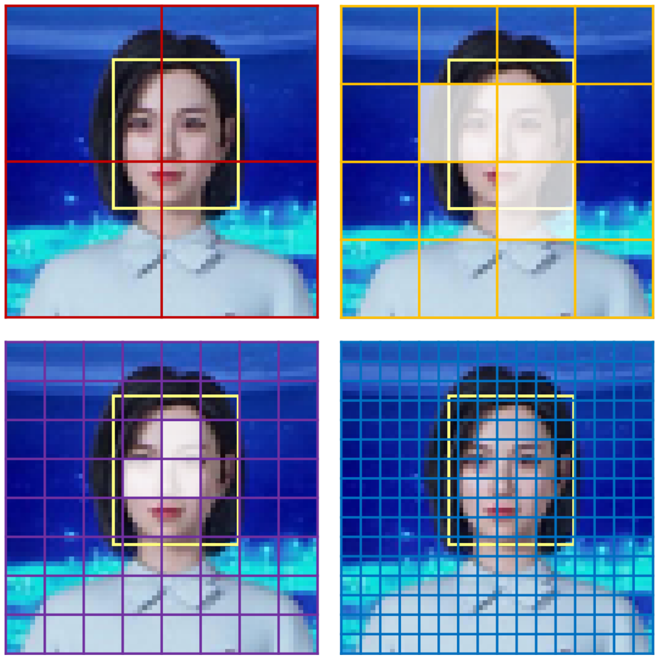
As a special kind of object detection, the pipeline of face detection is similar to that of general object detection. Regions with CNN features (R-CNN) [6], the first successful CNN-based object detection method, contains two stages. First, thousands of candidate regions of objects are proposed by selective search [7]. Second, each of the candidate regions is cropped from the image and input into CNN to classify what kind of object it contains. If a candidate region contains the needed object, its position will be refined by bounding-box regression to get the predicted bounding-box. Fast R-CNN [8] and Faster R-CNN [9] improve R-CNN and focus on selecting candidate regions of objects better and faster. Faster R-CNN designs a Region Proposal Network (RPN) to search the potential regions containing objects. In RPN, a series of rectangles with multi-scales are proposed and assumed to contain objects. These rectangles are called anchors. If an anchor is determined to contain an object, the coordinate of this anchor will be transformed by bounding-box regression to get the candidate region. Then the candidate region is further classified to get the class of the contained object and refined by bounding-box regression to get the predicted bounding-box. While the series methods of R-CNN only use a single scale of feature map to detect objects, Feature Pyramid Networks (FPN) [10] utilizes multi-scale feature maps to enhance the network's ability to detect multi-scale objects. To reduce the time consumption, SSD [11] and YOLO [12] combine the region proposal and the region classification and refinement to one stage by mapping the anchors and their content to bounding box coordinates and class probabilities directly. In general, except for the CNN's ability of feature extraction, both one-stage and two-stage detection methods rely on anchor/region proposal and bounding box regression.

By taking advantage of the characteristic of CNN, we propose a new method for anchor generation and bounding box regression to improve the performance of CNN in face detection. Although CNN has experienced considerable development, the most popular networks always have a similar structure with the very original LeNet [13] which contains several downscale operations and produces pyramid-like feature maps. Fig. 1(a) shows a face image and its feature pyramid produced by CNN. The correlation and downscale operations in CNN establish a connection between the points in the feature map and the pixels in the input image. In general, a point in the feature map has a relationship with a square area in the input image. This square area is called Receptive Field [14]. The point in the feature map of lower-level has a smaller Receptive Field, while the point in the feature map of higher-level has a larger Receptive Field. Inspired by the pyramid-like feature maps and Receptive Field, we assign each point in the feature map a square area in the image. This square area has the same center as its corresponding feature point's Receptive Field. If the corresponding square areas of all the points in one feature map cover the whole image without overlap and gap, the image will be divided into grids. As shown in Fig. 1, the four images in Fig. 1(b) are divided into grids according to the sizes of the four feature maps selected in Fig. 1(a). The grids located at or near the center of the faces are the nature anchors for bounding-box regression. For each ground-truth bounding-box, we select three grids closest to its center point as anchors. If a selected anchor does not overlap with the ground-truth bounding-box, we ignore it. We enhance the ability of our method to detect multi-scale faces by regressing smaller anchors to smaller faces and regressing larger anchors to larger faces. Therefore, if a selected anchor is too large or too small compared to the ground-truth bounding-box, we also ignore it. Table I shows the correspondence between the anchor size and the range of face size. In Fig. 1(b), the white grids are the selected anchors. In the first and last images, none of the grids is selected as an anchor because the grids are too small or too large compared to the ground-truth bounding-box. Each anchor corresponds to a point in the feature pyramid and this point is responsible for outputting the regression parameters and the score of the predicted bounding-box. In our method, the sizes of our anchors are too small compared to the sizes of many faces in normal images and the common bounding-box regression methods are not able to regress our anchors to bounding-box of the larger face properly. However, we can normalize the sizes of anchor and ground-truth bounding-box to similar scale by logarithmization. In inference, the network only needs to output exponents that can transform the size of the anchor into the size of bounding-box.

Compare with the state of the arts, the generation of anchors in our method is simpler and more well-founded. We use different levels of feature maps for detecting faces of different sizes more explicitly and strictly. The ranges of face size for the feature map of each level to detect are finer. These enhance the network's ability to detect multi-scale faces. The grid anchor is more suitable for predicting face bounding-box which is square-like. For one feature point, the single anchor can make better use of the representation ability of the network than the multi-anchors with different aspect ratios and multi-scales used in other methods.



(a) Select Feature Maps for Face Detection



(b) Divide Image into Grids and Label Each Grid

Fig. 1. (a) CNN processes a 64×64 face image and produces a feature pyramid. We select four features with resolutions of 2×2, 4×4, 8×8, and 16×16 for face detection. (b) The yellow box is the ground-truth. The face images are divided into grids according to the sizes of the feature maps. The grids with white shade are selected as anchors.

TABLE I
FEATURES OF DIFFERENT LEVELS RESPONSIBLE FOR DETECTING FACES OF DIFFERENT SIZES

Feature Level	**Anchor/Grid Size**	**The Range of the Side Length of Ground-Truth Bounding-Box**
Feature4 (Red)	32×32	$32 \sim 32^2$
Feature3 (Orange)	16×16	$16 \sim 16^2$
Feature2 (Purple)	8×8	$8 \sim 8^2$
Feature1 (Blue)	4×4	$4 \sim 4^2$

II. Related Works

A. Object Detection

As a special kind of object detection task, the progress of face detection benefits from the development of object detection. Since R-CNN [6] was proposed, various excellent object detection algorithms have emerged one after another. Object detection algorithms can be broadly divided into two categories: one-stage and two-stage. R-CNN and its improved versions, Fast R-CNN [8] and Faster R-CNN [9], all belong to the two-stage. They firstly search the region proposals and then classify the contents in the proposals. SSD [11] and YOLO [12] are two representative one-stage methods. They are based on global regression/classification in which the image pixels are mapped directly to bounding box coordinates and class probabilities. In the above methods, bounding-box regression is an essential part. In Faster R-CNN, YOLO, and SSD, a series of anchors, which are assumed to contain objects, are proposed to assist the bounding-box regression. For detecting objects of different sizes and shapes, they design complicated anchor generation methods.

B. Face Detection

Many CNN-based object detection methods are used in face detection. To improve the accuracy on the popular face benchmarks, FDDB [15] and WIDER FACE [16], loads of complicated components are added in the networks, which result in over-fitting in these benchmarks and loss of speed. PyramidBox [3] proposes a context-assisted single shot face detector. DSFD [2] introduces a feature enhance module to extend the single shot detector to the dual shot detector. EXTD [17] generates the feature maps by iteratively reusing a shared lightweight and shallow backbone network instead of a single backbone network. RetinaFace [18] unifies face detection, 2D face alignment, and 3D face reconstruction in single-shot inference. These SSD-based face detectors all get high accuracy on the WIDER FACE benchmark. Based on the YOLOv5 object detector, YOLO5Face [19] achieves a better performance on the WIDER FACE benchmark by adding a five-point landmark regression head and using the Wing loss function [20]. Hambox [21] proposes an online strategy to mine high-quality anchors for detecting outer faces.

III. Method

A. Network

Our network is shown in Fig. 2. The feature extracting part includes a backbone, FPN [10] and PAN [22]. Our backbone is similar to the CSPDarkNet which is used in YOLOv5 [23]. CSP [24] and SPP [25] are used for better feature extraction. The Conv with S=2 and Upscale layers downscale the feature map to 1/4 and enlarge the feature map to 4 times, respectively. The operation layers with the same color output feature maps with the same size.

At the end of the network, there are four branches for predicting face bounding-boxes. The widths and heights of Feature1-4 are 1/4, 1/8, 1/16, and 1/32 of the width and height of the input image. In Feature1-4, the channels is 5 and a feature point can be denoted as a vector $Z(z_0,\, z_1,\, z_2,\, z_3,\, z_4)$. $(z_0,\, z_1,\, z_2,\, z_3)$ is transformed for representing the predicted bounding-box and z_4 is used for representing the score of the predicted bounding-box.

B. Anchor Generation

By dividing the image into grids according to the sizes of Feature1-4, we get four grid images as shown in Fig. 1(b). Then we select grids as anchors for training.

For a ground-truth bounding-box on a grid image, we select anchors in three steps. First, we select the three grids closest to the center point of the ground-truth as candidate anchors. Second, if a candidate anchor does not overlap with the ground-truth, we filter out this candidate anchor. Third, according to the correspondence in Table I, if a candidate anchor is too large or too small compared to the ground-truth, it should also be filtered out. As shown in Fig. 1(b), the white grids are selected as anchors.

C. Bounding-box Regression

We redesign the method of bounding-box regression to support our anchor generation method.

In training, each grid anchor G corresponds to a ground-truth bounding-box T and a feature point Z in Feature1-4. The goal of training is minimizing the difference between $f_1(G,T)$ and $f_2(G,Z)$. In testing, we can get the predicted bounding-box T' by $f_1^{-1}(G, f_2(G,Z))$. f_1, f_2 and f_1^{-1} are the transformations we will design below.

$$\begin{aligned} t_{x\_c} &= \frac{t_x - g_x}{d}, \\ t_{y\_c} &= \frac{t_y - g_y}{d}, \\ t_{w\_c} &= \frac{t_w}{d}, \\ t_{h\_c} &= \frac{t_h}{d}. \end{aligned} \tag{1}$$

f_1 is described in (1). In image domain, each grid anchor and its corresponding ground-truth bounding-box can be denoted as $G(g_x, g_y, d, d)$ and $T(t_x, t_y, t_w, t_h)$. (g_x, g_y) and (t_x, t_y) are the center points. (d, d) and (t_w, t_h) are the widths and heights. The transform result is $T_c(t_{x\_c}, t_{y\_c}, t_{w\_c}, t_{h\_c})$. Since G is one of the three grids closest to (t_x, t_y), t_{x_c} and $t_{y\_c} \in [-1, 1]$. (t_{w_c}, t_{h_c}) is the size of the ground-truth in the feature domain.

$$z' = \frac{1}{1 + e^{-z}}, \tag{2}$$

f_2 contains two steps. First, Z is normalized to 0$\sim$1 by (2) and get $Z'(z'_0,\, z'_1,\, z'_2,\, z'_3,\, z'_4)$. Second, $(z'_0,\, z'_1,\, z'_2,\, z'_3)$ is processed by

$$\begin{aligned} p_{x\_c} &= 2z'_0 - 1, \\ p_{y\_c} &= 2z'_1 - 1, \\ p_{w\_c} &= \frac{d^{z'_2+1}}{d}, \\ p_{h\_c} &= \frac{d^{z'_3+1}}{d}, \end{aligned} \tag{3}$$

and get $P_c(p_{x\_c}, p_{y\_c}, p_{w\_c}, p_{h\_c})$. Since $z' \in (0,1)$, p_{x_c} and $p_{y\_c} \in (-1,1)$ and $d^{z'+1} \in (d, d^2)$. P_c can be used to represent T_c.

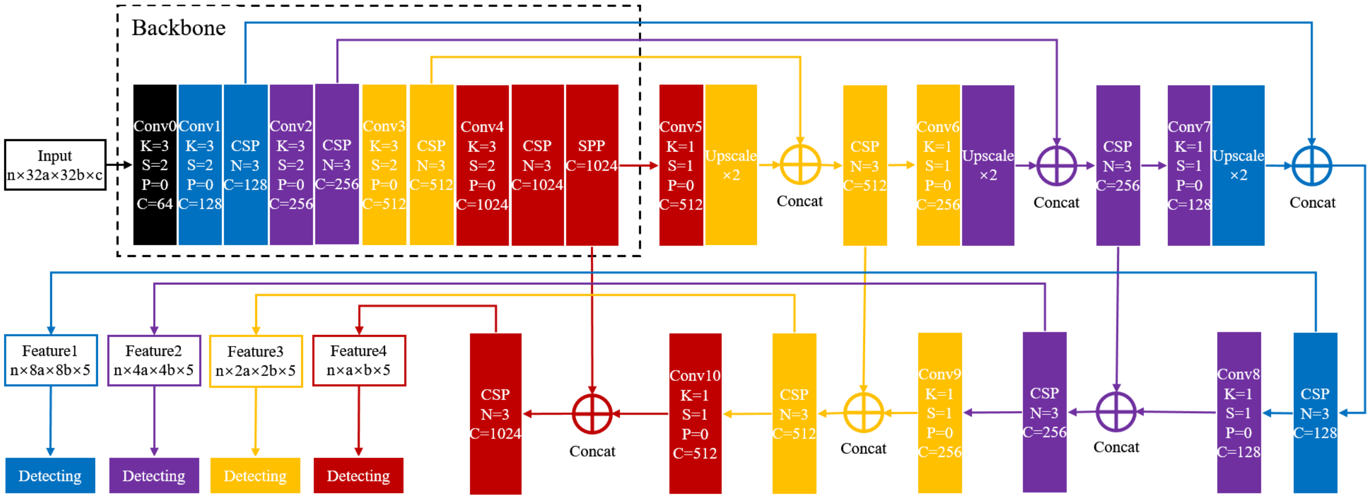


Fig. 2. The network consists of a backbone and a detection head with four branches. The input size is $n \times 32a \times 32b \times c$ where n is batch-size, $32a$, $32b$ and c are height, width and channel. K, S, P, C and N denote the kernel size, stride, padding size, output channel and number of repeated module, respectively.

We use CIoU [26] to calculate the similarity between P_c and T_c. The regression loss of each branch is defined in

$$L_{reg}^{(d)} = \frac{1}{N^{(d)}} \sum_{n=1}^{N^{(d)}} [1 - CIoU(P_c^{(n)}, T_c^{(n)})], \tag{4}$$

where $N^{(d)}$ is the number of grids that are selected as anchors in one batch in one branch. The total regression loss is

$$L_{reg} = L_{reg}^{(4)} + L_{reg}^{(8)} + L_{reg}^{(16)} + L_{reg}^{(32)}. \tag{5}$$

In forwarding, each feature point in Feature1-4 will be transformed into Z'. In testing, z_4' represents the score at which the predicted bounding-box correctly contains a face. In training, z_4' is given a ground-truth value to indicate whether this point corresponds to an anchor. If this point corresponds to an anchor, the ground-truth value is 1, otherwise 0. The score loss of each branch is defined in

$$\begin{aligned} L_{score}^{(d)} = -\frac{1}{M^{(d)}} \sum_{m=1}^{M^{(d)}} [(1 - z_t^{(m)}) \log(1 - z_4'^{(m)}) \\ + z_t^{(m)} \log(z_4'^{(m)})], \end{aligned} \tag{6}$$

where $M^{(d)}$ is the number of feature points in one batch and z_t is the ground-truth value of z_4'.The total score loss is

$$L_{score} = L_{score}^{(4)} + L_{score}^{(8)} + L_{score}^{(16)} + L_{score}^{(32)}. \tag{7}$$

The loss of the whole network is

$$L = L_{reg} + L_{score} + \lambda ||W||^2, \tag{8}$$

where W are the weights in the network and $\lambda ||W||^2$ is added to avoid over-fitting.

In testing, a predicted bounding-box $P(p_x, p_y, p_w, p_h)$ can be obtained from a transformed feature point Z' which has a higher score z_4' by f_1^{-1} as describe in

$$\begin{aligned} p_x &= (2z_0' - 1)d + g_x, \\ p_y &= (2z_1' - 1)d + g_y, \\ p_w &= d^{z_2'+1}, \\ p_h &= d^{z_3'+1}. \end{aligned} \tag{9}$$

D. Post Processing in Testing

In testing, if input an image into the network, each point in Feature1-4 will be transformed into a vector of regression parameters and a score. Fig. 3 shows how to generate a predicted bounding-box using these regression parameters and scores in the third branch of the network. As shown in the left image of Fig. 3, the scores are labeled in the feature points' corresponding grids. Then the green grids, whose scores are higher than a threshold (such as 0.5), are regressed to bounding-boxes using their regression parameters. We fuse these bounding-boxes by Non-Maximum Suppression (NMS) and get the best predicted bounding-boxes of the third branch. The other branches can also generate predicted bounding-boxes, then we fuse the results of the four branches by NMS to obtain the final predicted bounding-boxes.

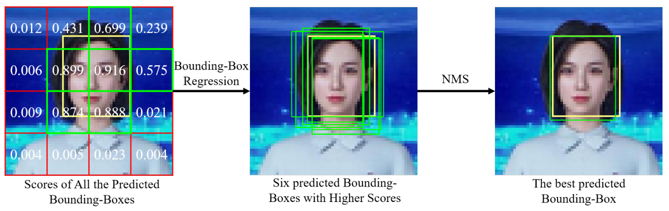

0.012
0.431
0.699
0.239
0.006
0.899
0.916
0.575
0.009
0.874
0.888
0.021
0.004
0.005
0.023
0.004
Bounding-Box Regression
NMS
Scores of All the Predicted Bounding-Boxes
Six predicted Bounding-Boxes with Higher Scores
The best predicted Bounding-Box

Fig. 3. The post processing pipeline on Feature3 (output of the Third Branch).

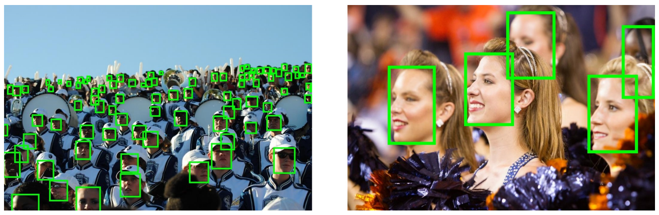
IV. Experiment

Our experiments mainly consist of two parts. One is training large models using the network in Fig. 2 to compare the accuracy and speed with the state of the arts on GPU. The other is reducing the parameters of the network in Fig. 2 and training it to compare the accuracy and speed on CPU.

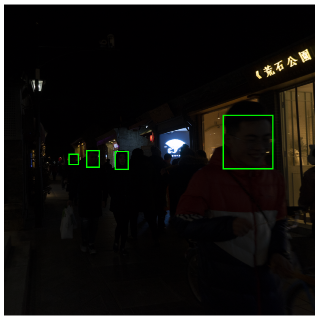
Our models are trained on NVIDIA GPU V100 with the deep learning framework PyTorch [27]. Our data augmentation includes mosaic augmentation, resizing the image with a random scale from 0.5 to 1.5, and flipping the image randomly and horizontally. We use the maximum batch size that the memory can handle and train 300 epochs. We set the initial learning rate to 0.01 and adjust the learning rate every epoch by cosine learning rate decay [28] until 0.001. The optimizer used in the training is stochastic gradient descent [13].

A. Datasets

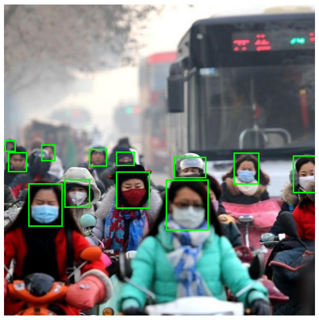
WIDER FACE [16], Dark Face [29] and MAFA [30] are used to demonstrate the effectiveness and robustness of our method. WIDER FACE contains 32,203 images and 393,703 labeled faces with a high degree of variability in scale, pose, and occlusion. The whole dataset is divided into training, validation, and test sets. The Dark Face releases 6000 images captured in dark environments. We randomly select 5000 images for training and validation. The rest 1000 images are used for the test. MAFA contains 30,811 Internet images and 35,806 masked faces. Faces in MAFA have various orientations and occlusion degrees, while at least one part of each face is occluded by a mask. Fig. 4 shows the examples of these datasets.



(a) WIDER FACE



(b) Dark Face



(c) MAFA

Fig. 4. Examples in WIDER FACE, Dark Face and MAFA.

B. Effectiveness Analysis

To demonstrate the effectiveness of our method, we replace the detection heads of the state-of-the-art object detection methods with our method and compare the performances between the original detection heads and ours. As shown in Table II, the performances of the detection methods with our detection head are better.

TABLE II
Comparison of Accuracy with Different Detection Heads (WIDER FACE Val Set)

Methods	**Average Precision**			**FLOPs** ($\times 10^9$)
	Easy	Medium	Hard	
EfficientDet-D4 [31]	0.938	0.922	0.823	40.4
EfficientDet-D4-Ours	0.946	0.941	0.899	45.1
YOLOv3 [12]	0.966	0.959	0.897	154.9
YOLOv3-Ours	0.969	0.962	0.915	144.8
YOLOv5x [23]	0.969	0.960	0.901	204.2
YOLOv5x-Ours	**0.972**	**0.964**	**0.921**	203.3

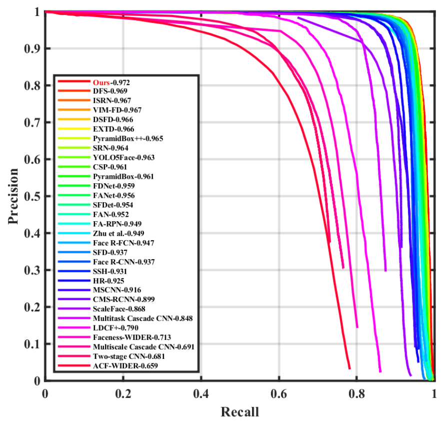
To demonstrate the ability of our method to detect multi-scale faces, We count the detection accuracy of each method at different scales. Table III shows that our method outperforms the other methods at each scale, especially the tiny faces.

TABLE III
Comparison of Accuracy on Faces with Different Sizes (WIDER FACE Val Set)

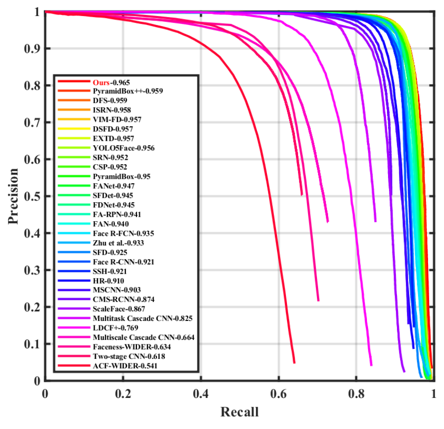
	Average Precision			
Longer Side of GT BBox	≤16	(16, 64]	(64, 256]	>256
Number of GT BBox	16844	17793	4482	586
PyramidBox [3]	0.566	0.898	0.954	0.934
EXTD [17]	0.507	0.862	0.917	0.917
DSFD [2]	0.534	0.917	0.968	0.951
YOLOv3 [12]	0.569	0.919	0.967	0.874
EfficientDet-D4 [31]	0.356	0.846	0.936	0.941
YOLOv5x [23]	0.579	0.922	0.969	0.895
Ours	**0.675**	**0.926**	**0.972**	**0.958**

C. Comparison of Models Inferring on GPU

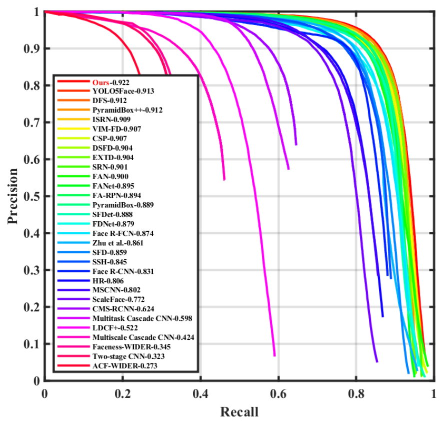
WIDER FACE collects and releases the detection accuracy of the state of the arts. As shown in Fig. 5, the performance of our method can reach the state-of-the-art on WIDER FACE. To compare the accuracy on Dark Face and MAFA, the open-source face detection methods DSFD [2], EXTD [17] and PyramidBox [3] as well as the state-of-the-art object detection methods YOLOv3 [12], EfficientDet [31] and YOLOv5x [23] are selected. Table IV shows our method outperforms the others and demonstrates that our method is robust in adverse conditions. The above methods are also used for comparing the detection speed. We resize the 3,226 images in WIDER FACE Val set to specific resolutions and input them into the networks running on NVIDIA V100 GPU one by one to calculate the average FPS. Table V shows our method is faster than the other methods at each resolution.



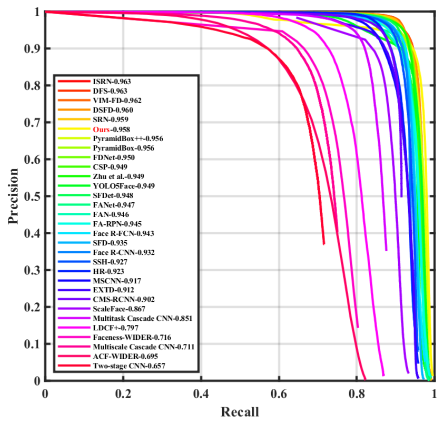
(a) Val-Easy



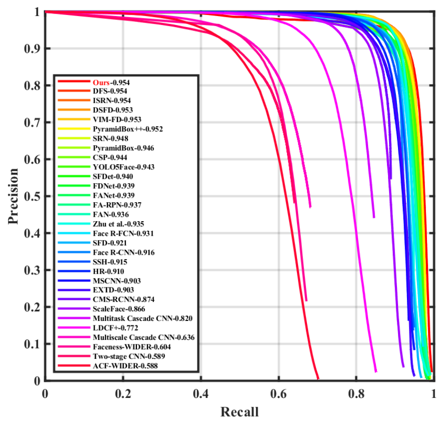
(b) Val-Medium



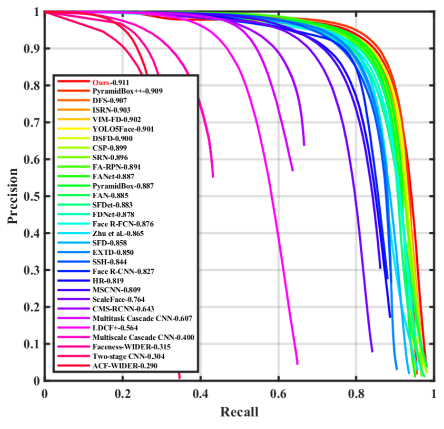
(c) Val-Hard



(d) Test-Easy



(e) Test-Medium



(f) Test-Hard

Fig. 5. Comparison of PR Curves and Average Precision on Val and Test Sets of WIDER FACE.

TABLE IV
COMPARISON OF ACCURACY ON DARK FACE AND MAFA

Methods	Average Precision	
	Dark Face	MAFA
PyramidBox [3]	0.796	0.748
EXTD [17]	0.689	0.661
DSFD [2]	0.634	0.772
EfficientDet-D4 [31]	0.755	0.753
YOLOv3 [12]	0.801	0.773
YOLOv5x [23]	0.824	0.788
Ours	**0.850**	**0.796**

TABLE V
COMPARISON OF DETECTION SPEED ON NVIDIA V100 GPU

Methods	Layers	**Params**	**FLOPs**	**Speed** (FPS)		
		$(\times 10^6)$	$(\times 10^9)$	640×480	1280×720	1920×1080
PyramidBox	234	67.269	296.2	2.75	1.99	1.40
EXTD	93	0.162	27.9	6.07	4.24	2.45
DSFD	544	120.058	691.6	3.48	3.36	2.37
YOLOv3	333	61.524	154.9	25.27	22.94	22.39
EfficientDet-D4	1111	20.543	40.4	14.50	8.53	3.15
YOLOv5x	567	86.218	204.2	77.09	75.02	74.13
Ours	679	57.054	155.7	**80.87**	**76.56**	**75.18**

D. Comparison of Light Models Inferring on CPU

By reducing the channels of each Conv layer in Fig. 2 to 1/4 and setting the module number of each CSP block to 1, we can get a light network that can work on Non-GPU devices. We convert the existing light face detection models, retrained YOLOv5n [23] model and ours to ONNX [32] format and run them on Intel i7-5930K CPU for comparison. The images in the WIDER FACE Val set are resized to 640×480 for the test. Table VI shows our method can run on a Non-GPU device with real-time speed while keeping a high detection accuracy.

V. CONCLUSION

In this paper, we propose a new method of anchor generation and bounding-box regression for face detection, which can achieve a good balance between accuracy and speed on both GPU and CPU, and also perform well in adverse conditions, such as faces in low light and masked faces. Our method takes advantage of the characteristic of classic CNNs and uses the pyramidal feature maps to enhance the ability to detect multi-scale faces. Our method is more effective for detecting faces with small sizes.

TABLE VI
COMPARISON OF LIGHT MODELS ON INTEL I7-5930K CPU
(WIDER FACE VAL SET)

Methods	**Layers**	**Params** $(\times 10^6)$	**FLOPs** $(\times 10^9)$	**Average Precision**			**Latency** (ms)	
				Easy	Medium	Hard	Forward	Post-Proc
FaceBoxes [33]	33	1.013	1.541	0.845	0.777	0.404	16.52	7.16
ULFG-slim-640 [34]	42	0.401	2.000	0.810	0.794	0.630	19.03	2.37
ULFG-RFB-640 [34]	52	0.085	2.426	0.816	0.802	0.663	21.27	1.90
YuFaceDetectNet [5]	43	0.085	2.549	0.856	0.842	0.727	23.47	32.81
LFFD-v2 [4]	45	1.520	37.805	0.875	0.863	0.752	178.47	6.70
LFFD-v1 [4]	65	2.282	55.555	0.910	0.880	0.778	229.35	10.08
YOLOv5n [23]	270	1.872	4.520	0.942	**0.933**	0.856	29.21	0.80
Ours	301	1.746	4.536	**0.943**	**0.933**	**0.870**	26.15	0.69

REFERENCES

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

[2] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang, “Dsfd: dual shot face detector,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5060–5069.

[3] X. Tang, D. K. Du, Z. He, and J. Liu, “Pyramidbox: A context-assisted single shot face detector,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 797–813.

[4] Y. He, D. Xu, L. Wu, M. Jian, S. Xiang, and C. Pan, “Lffd: A light and fast face detector for edge devices,” *arXiv preprint arXiv:1904.10633*, 2019.

[5] S. Yu, “libfacedetection,” https://github.com/ShiqiYu/libfacedetection, 2021.

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

[7] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[8] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *arXiv preprint arXiv:1506.01497*, 2015.

[10] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[12] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] J.-M. Alonso and Y. Chen, “Receptive field,” *Scholarpedia*, vol. 4, no. 1, p. 5393, 2009.

[15] V. Jain and E. Learned-Miller, “Fddb: A benchmark for face detection in unconstrained settings,” UMass Amherst technical report, Tech. Rep., 2010.

[16] S. Yang, P. Luo, C.-C. Loy, and X. Tang, “Wider face: A face detection benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5525–5533.

[17] Y. Yoo, D. Han, and S. Yun, “Extd: Extremely tiny face detector via iterative filter reuse,” *arXiv preprint arXiv:1906.06579*, 2019.

[18] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, “Retinaface: Single-shot multi-level face localisation in the wild,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[19] D. Qi, W. Tan, Q. Yao, and J. Liu, “Yolo5face: Why reinventing a face detector,” *arXiv preprint arXiv:2105.12931*, 2021.

[20] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, “Wing loss for robust facial landmark localisation with convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2235–2245.

[21] Y. Liu, X. Tang, X. Wu, J. Han, J. Liu, and E. Ding, “Hambox: Delving into online high-quality anchors mining for detecting outer faces,” *arXiv preprint arXiv:1912.09231*, 2019.

[22] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[23] G. Jocher, “ultralytics/yolov5,” https://github.com/ultralytics/yolov5, Oct. 2020.

[24] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, “Cspnet: A new backbone that can enhance learning capability of cnn,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[26] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-iou loss: Faster and better learning for bounding box regression,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 12 993–13 000.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *arXiv preprint arXiv:1912.01703*, 2019.

[28] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of tricks for image classification with convolutional neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558–567.

[29] W. Yang, Y. Yuan, W. Ren, J. Liu, W. J. Scheirer, Z. Wang, T. Zhang, Q. Zhong, D. Xie, S. Pu *et al.*, “Advancing image understanding in poor visibility environments: A collective benchmark study,” *IEEE Transactions on Image Processing*, vol. 29, pp. 5737–5752, 2020.

[30] S. Ge, J. Li, Q. Ye, and Z. Luo, “Detecting masked faces in the wild with lle-cnns,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2682–2690.

[31] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[32] “Open neural network exchange,” https://github.com/onnx/onnx.

[33] 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 *2017 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 2017, pp. 1–9.

[34] Linzaer, "1mb lightweight face detection model," https://github.com/Linzaer/Ultra-Light-Fast-Generic-Face-Detector-1MB, 2020.